Vysoká škola báňská – Technická univerzita Ostrava

LOGICKÉ SYSTÉMY ŘÍZENÍ

učební text / příklady pro cvičení

Ing. Jiří Koziorek, Ph.D.
Bc. Libor Chromčák

Ostrava 2007

Recenze: Ing. Jaromír Škuta, Ph.D.

Název: Logické systémy řízení
Autor: Ing. Jiří Koziorek, Ph.D., Bc. Libor Chromčák
Vydání: první, 2007
Počet stran: 370
Vydavatel a tisk: Ediční středisko VŠB – TUO.

Studijní materiály pro studijní obor Elektrotechnika, Fakulty elektrotechniky a informatiky.
Jazyková korektura: nebyla provedena.

Určeno pro projekt:
Operační program Rozvoj lidských zdrojů.
Název: E-learningové prvky pro podporu výuky odborných a technických předmětů.
Číslo: CZ.O4.01.3/3.2.15.2/0326
Realizace: VŠB – Technická univerzita Ostrava.
Projekt je spolufinancován z prostředků ESF a státního rozpočtu ČR.

ISBN 978-80-248-1490-2

POKYNY KE STUDIU

Logické systémy řízení

Pro předmět Logické systémy řízení 5. semestru oboru Řídicí a informační systémy jste obdrželi studijní balík obsahující

- integrované skriptum pro distanční studium obsahující i pokyny ke studiu
- CD-ROM se vzorovými projekty a doplňkovými animacemi vybraných částí kapitol
- harmonogram průběhu semestru a rozvrh prezenční části
- rozdělení studentů do skupin k jednotlivým tutorům a kontakty na tutory
- kontakt na studijní oddělení

Cílem předmětu

je seznámení se základními pojmy z oblasti řídicích systémů na bázi programovatelných automatů. Po prostudování modulu by měl student být schopen zvolit pro daný řízený systém vhodný programovatelný automat, navrhnout hardwarovou konfiguraci systému, navrhnout a realizovat řídicí aplikaci.

Pro koho je předmět určen

Modul je zařazen do bakalářského studia oboru Řídicí a informační systémy studijního programu B2645 Elektrotechnika, sdělovací a výpočetní technika, ale může jej studovat i zájemce z kteréhokoliv jiného oboru.

Skriptum se dělí na části, kapitoly, které odpovídají logickému dělení studované látky, ale nejsou stejně obsáhlé. Předpokládaná doba ke studiu kapitoly se může výrazně lišit, proto jsou velké kapitoly děleny dále na číslované podkapitoly a těm odpovídá níže popsaná struktura.

Při studiu každé kapitoly doporučujeme následující postup:

Čas ke studiu: xx hodin

Na úvod kapitoly je uveden **čas** potřebný k prostudování látky. Čas je orientační a může vám sloužit jako hrubé vodítko pro rozvržení studia celého předmětu či kapitoly. Někomu se čas může zdát příliš dlouhý, někomu naopak. Jsou studenti, kteří se s touto problematikou ještě nikdy nesetkali a naopak takoví, kteří již v tomto oboru mají bohaté zkušenosti.

Cíl: Po prostudování tohoto odstavce budete umět

popsat ...

definovat ...

vyřešit ...

Ihned potom jsou uvedeny cíle, kterých máte dosáhnout po prostudování této kapitoly – konkrétní dovednosti, znalosti.

Výklad

Následuje vlastní výklad studované látky, zavedení nových pojmů, jejich vysvětlení, vše doprovázeno obrázky, tabulkami, řešenými příklady, odkazy na animace.

Shrnutí pojmů

Na závěr kapitoly jsou zopakovány hlavní pojmy, které si v ní máte osvojit. Pokud některému z nich ještě nerozumíte, vraťte se k nim ještě jednou.

Kontrolní otázky

Pro ověření, že jste dobře a úplně látku kapitoly zvládli, máte k dispozici několik teoretických otázek.

Řešená úloha 1.1

Zadání: Napište program, který provede logickou funkci zadanou následující tabulkou.

Protože většina teoretických pojmů tohoto předmětu má bezprostřední význam a využití v praxi, jsou Vám nakonec předkládány i praktické úlohy k řešení. V nich je hlavním cílem aplikovat čerstvě nabyté znalosti při řešení reálných situací.

Další zdroje

Na konci každé kapitoly naleznou zájemci o dobrovolné rozšíření znalostí popisované problematiky seznam další literatury, www odkazů a podobně.

DVD-ROM

V některých částech kapitol naleznete odkazy na přiložené DVD-ROM, kde naleznete projekty řešených příkladů a animace postupů při práci s použitými programovacími prostředími.

Technické prostředky nutné pro práci s připravenými programy a animacemi

Vytvořené softwarové aplikace – řešení příkladů z jednotlivých kapitol – je možné otevřít v programovacích nástrojích: Siemens Simatic Step7 v5.3 + SP3 , který lze zdarma stáhnout jako Lite verzi na http://support.automation.siemens.com (najdete jej pod ID: 21952802). Dále B&R

Automation Studio v2.5.2.21. Připravené animace lze prohlížet v běžném přehrávači .avi souborů, např. Windows Media Player.

Úspěšné a příjemné studium s touto učebnicí Vám přejí autoři výukového materiálu

Jiří Koziorek, Libor Chromčák

OBSAH

1. LOGICKÉ ŘÍZENÍ A TECHNOLOGICKÝ PROCES, PROGRAMOVATELNÉ AUTOMATY

Čas ke studiu: 1 hodina

Cíl

V této kapitole se seznámíte s prostředky na bázi mikroprocesoru, které se v dnešní **průmyslové automatizaci** používají. Jsou zde rovněž uvedeny trendy, které jsou v této oblasti v současnosti patrné.

V další části kapitoly je popsán pojem **programovatelný automat**. Čtenář se dozví, jaké zařízení si pod tímto pojmem představit, kde se využívá a jaké jsou jeho vlastnosti. Je zde rovněž uvedeno rozdělení programovatelných automatů z hlediska konstrukce a výkonu.

Výklad

1.1 Trendy v automatizační technice

Dnes není automatizace něčím unikátním, co je výsadou drahého komfortu rozsáhlých výrobních linek a náročných technologických procesů. Kvalitní a inteligentní řízení je dostupné i pro obyčejné stroje, pomocné mechanismy a technologická zařízení ve všech oborech. S inteligentní automatizační technikou se běžně setkáváme v "nevýrobní automatizaci", zejména v "malé energetice" a v technice budov (kde přináší značné úspory). Patrně nejrozšířenějšími řídicími systémy v průmyslové praxi jsou programovatelné automaty [1-1].

Počítače v automatizaci

V automatizaci se v současné době používá celá řada zařízení na bázi procesoru. Mezi ty základní patří:

- osobní počítače - slouží obvykle v automatizovaných systémech jako standardně vybavení velínů a dispečerských pracovišť, ale i jako pracoviště pro servis a seřizování, pro monitorování technologického procesu a dokumentování jeho průběhu, pro sledování kvality, spotřeby energie a surovin, pro dokumentování přítomnosti a zásahů obsluhujících. Někdy se setkáváme s přímým řízením technologických procesů standardním PC, mnohdy umístěným přímo v technologii. V drsných průmyslových podmínkách však mnohdy selhává (bývá málo spolehlivý, je citlivý na rušení, a přepětí, nemá potřebnou životnost).
- Průmyslové počítače (IPC, IC) – modifikovaná konstrukce osobního počítače, která je přizpůsobena průmyslovým požadavkům. Někdy se používají při přímém řízených strojů a technologií, někdy jen v roli inteligentního operátorského panelu nebo komunikačního adaptéru. Problémem při jejich nasazování vysoká cena a také to, že některé ze zásadních nedostatků osobních počítačů platí i pro průmyslová PC.
- Mikroprocesorové systémy – do této kategorie lze zařadit mikrokontrolery, jednodeskové počítače a podobné systémy. Jsou velice vhodné pro využití v určitých aplikacích, zejména jako vestavěné systémy při řízení přístrojů, strojů a celé řady elektronických výrobků.

- Programovatelné automaty – jsou velice vhodné pro řízení klasických průmyslových i ostatních úloh. Využití naleznou zejména tam, kde jsou vysoké nároky na spolehlivost řízení. V běžných automatizačních úlohách se tyto systémy v současnosti používají nejčastěji.
- Speciální procesorové systémy – do této skupiny lze zařadit systémy, které nezapadají do předchozích kategorií. Mohou to být například vysoce výkonné víceprocesorové systémy pro paralelní zpracování řídicích úloh apod.

Komunikace, integrace a distribuovanost

Od automatizace je neoddělitelná i komunikační technika. Komunikace je dnes důležitá i pro spojení řídicích systémů a jejich periferních prvků. Existují dva zdánlivě protikladné trendy: integrace a distribuovanost.

Integrované řídicí systémy vznikají sdružováním řídicích systémů, které dosud pracovaly samostatně. Na nejvyšší úrovni vznikají integrované systémy tak, že do informačních počítačových sítí bývají připojovány i počítače, sloužící dosud jen pro potřeby řízení, dispečerská pracoviště, velíny a monitorovací systémy. Sdružují (integrují) se tak řídicích a informačních systémů. Do sítě, zprostředkované průmyslovou sběrnicí (např. Profibus, Ethernet, ASI, CAN), bývají zapojovány řídicí systémy nižší úrovně, které dosud pracovaly nezávisle. Spojení bývá víceúrovňové, hierarchické.

Na komunikacích jsou založeny i distribuované systémy. Funkce, které tradičně provádí jediný řídicí systém (např. modulární PLC se stovkami vstupů a výstupů) realizuje v distribuovaném systému soubor podsystémů (např. desítky malých kompaktních PLC s několika vstupy a výstupy – typicky od 8, 12, 16 do 32 nebo 64). Každý z podsystémů má svou lokální inteligenci, lokální kompetence a řeší své lokální problémy. Informace globálního charakteru, týkající se společného fungování celého systému jsou předávány komunikační linkou ostatním účastníkům (podsystémům). Souboru podsystémů může (ale nemusí) být nadřazen další systém nebo počítač. Stále častěji se v aplikacích využívá nejnižší komunikační úroveň na kterou se připojují prvky dosud považované za pasivní: inteligentní ("smart") senzory, akční členy a pohony. Pro jejich připojení se někdy využívají průmyslové sběrnice pro spojení systémů (např. Profibus, CAN), běžné jsou ale i sběrnice specializované pro tuto nejnižší úroveň - ASI, Device Net, M-bus apod.). Inteligentní senzory a akční členy již jsou vyrobeny se schopností komunikovat na zvolené sběrnici. Standardní a starší prvky se obvykle připojují prostřednictvím komunikačních modulů. Analogicky je řešena komunikace mezi moduly distribuovaného systému.

Sdružování funkcí

Programovatelnost a variabilnost výstavby poskytuje PLC jejich univerzálnost a přizpůsobivost. Již neplatí, že PLC řešil jen logické úlohy, zatímco ke zpracování analogových veličin se používaly specializované regulátory. PLC dnes zvládne oba typy úloh (a mnoho dalších). Programem PLC lze realizovat vazby a ošetřit logické souvislosti, které jsou při použití specializovaných (uzavřených) přístrojů nedostupné - třeba regulaci teploty, vlhkosti, teploty a kvality spalování, teploty a dodržení sjednané spotřeby nebo optimalizovat proces a adaptovat jej podle měnících se podmínek, minimalizovat spotřebu, náklady nebo ztráty.

Umělá inteligence na dosah

Donedávna byla umělá inteligence spojována spíše se "sci-fi" nebo se "supersystémy" pro rozpoznávání obrazů a robotického vidění, pro komunikaci v přirozeném jazyce, s expertními systémy apod. Mnohé prostředky umělé inteligence jsou dnes dostupné pro běžné automatizační prostředky, někdy i pro spotřební produkty. PLC od významnějších výrobců disponují např. aparátem pro operace ve fuzzy logice. Použití fuzzy logiky může být daleko širší a prostší, než se všeobecně uvádí. Velmi perspektivním aplikačním oborem je například diagnostika a zabezpečovací technika, např. rozpoznávání chybových stavů a rizik, nebezpečných trendů.

Bezpečnost, spolehlivost a diagnostika

Automatizační technika je používána především proto, aby sloužila - předpokládá se, že spolehlivě. PLC, jako systémy pro průmyslové aplikace, jsou konstruovány s ohledem na maximální spolehlivost a odolnost proti rušení. Jejich poruchovost bývá zanedbatelná, obvykle pod úrovní poruchovosti běžných periferních prvků. Přesto je třeba při projektování a provozu dodržovat určité zásady a doporučení. Nejčastějšími zdroji poruch bývá změna vlastností technologického objektu: uvolněné spoje, vydření, zadření, přehřátí, ucpání, změna parametrů (např. přirozené stárnutí a opotřebení).

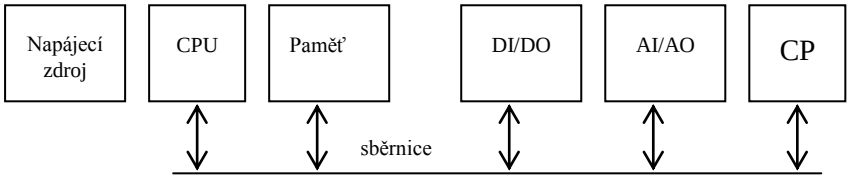
Mnohdy je příčinou poruch i selhání "lidského faktoru". Stále častěji je žádán bezobslužný provoz. To je zdrojem nového problému: řídicí systém musí rozpoznat i ty chybové stavy, které obsluhující rozeznával svými smysly (bez problémů pozná zamrzáni nebo zaplavení, vidí mechanické poškození, cítí hrozící požár, přehřátí motoru a únik plynu, sluchem obvykle pozná unikající vodu nebo páru, chybně seřízený hořák), intuitivně rozpozná řadu dalších poruch nebo rizik. Technická diagnostika, která je pomocí programovatelných automatů zvládnutelná, se proto stává neoddělitelnou součástí automatizační techniky.

1.2 Programovatelný automat – technické vybavení

Základní pojmy

Programovatelný automat je uživatelsky programovatelný řídicí systém přizpůsobený pro řízení průmyslových a technologických procesů nebo strojů, mnohdy specializovaný na úlohy převážně logického typu (obzvláště u starších typů nebo u nejmenších systémů). Nejčastěji se označuje zkratkou PLC (Programmable Logic Controller), v německé literatuře se lze setkat s označením SPS (SpeicherprogTammierbare Steuerung). Občas najdeme i označení PC (Programmable Controller). Česká zkratka, která se teprve začíná používat, je PA (Programovatelný automat).

Původně byly programovatelné automaty navrženy k řešení úloh logického řízení, často jako přímá náhrada pevné reléové logiky. V současných aplikacích se však zvyšuje podíl úloh regulačního typu, úloh monitorování řízeného procesu i úloh analogových měření. Každý programovatelný automat se v podstatě skládá z centrální procesorové jednotky, systémové paměti, uživatelské paměti, souboru vstupních a výstupních jednotek pro připojení řízeného systému (technologického procesu, výrobního stroje, výrobního zařízení, výrobku) a souboru komunikačních jednotek pro komunikaci se souřadnými i nadřazenými řídicími systémy. Jednotky programovatelného automatu jsou navzájem propojeny systémovou sběrnicí. Základní schéma programovatelného automatu je znázorněno na následujícím obrázku.



Obrázek 1.1: Blokové schéma programovatelného automatu.

Řídicí algoritmy jsou realizovány uživatelským programem, který může být zapsán v různých programovacích jazycích a po přeložení je uložen v uživatelské paměti programovatelného automatu. Program obsahuje posloupnost instrukcí, kterou procesor vykonává cyklicky. Chování programovatelného automatu je tedy dáno v podstatě zaměnitelným programem, zatímco u reléových systémů bylo chování určeno strukturou zapojení jednotlivých komponent, která byla po realizaci reléového systému téměř nezměnitelná.

Úloha programovatelného automatu v systémech řízení

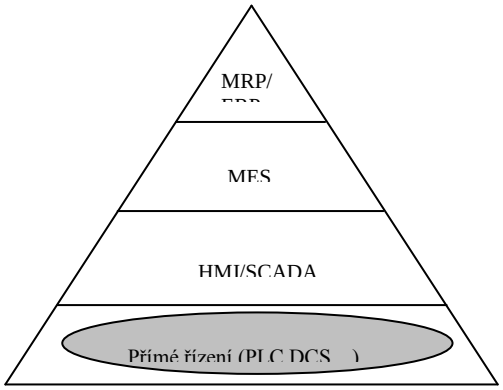
Programovatelný automat může být začleněn do systému řízení nejrůznějším způsobem. Při dopředném řízení působí programovatelný automat na řízený objekt jednosměrně, jen jej ovládá a nekontroluje dosažený stav. Mezi řídicím systémem a řízeným objektem jsou zařazeny jen akční členy. Povely pro programovatelný automat může zadávat při tzv. ručním řízení člověk- operátor. Programovatelný automat zde může figurovat jen jako prostředník mezi povely operátora a mezi jednotlivými akcemi pro řízení stroje. Mnohdy je totiž při řízeni stroje nutné zajistit složité posloupnosti dílčích akcí, zajistit koordinaci povelů pro pohony s jinými akčními zásahy, kontrolu zadaných akčních zásahů apod..

Programovatelný automat může plnit úlohy dopředného řízení i v automatizovaném či plně automatickém systému řízení, kdy povely pro automat poskytuje nadřazený řídicí systém ve spolupráci s člověkem nebo jen nadřazený řídicí systém.

Při zpětnovazebním řízená získává řídicí systém zpětnou informaci o stavu řízeného objektu (realizuje zpětnou vazbu, uzavírá zpětnovazební smyčku). Porovnává požadovaný stav se skutečným a podle zjištěné odchylky upravuje své akční zásahy tak, aby dosáhl požadovaného stavu (nebo se mu alespoň co možná nejvíce přiblížil). Zpětnovazební řízeni je typické pro regulační úlohy. Řízený objekt je proto třeba doplnit o potřebné snímače pro měření stavu sledovaných veličin (např. teploty, hladiny, polohy nebo tlaku).

Za zpětnovazební řízení ale můžeme považovat i logické řízení, při kterém na objekt působíme jen dvouhodnotovými povely typu "vypni - zapni" a zpracováváme i zpětnovazební informace dvouhodnotového charakteru ve významu hlášení o vykonání povelu nebo překročení povolených hodnot (např. informace typu: "hladina nízká", "hladina dosažena", "Hladina překročena", "nádrž prázdná", "nádrž přeplněna" apod.). I zpětnovazební řízení může být ruční, automatizované nebo plně automatické. Pro případy automatizovaného nebo automatického řízení je navíc naznačena i komunikační vazba řídicího systému k nadřízenému počítačovému systému (např. pro monitorováni procesu). V automatizovaných systémech je ponechána i účast člověka na řízení procesu, protože i v automatizovaných procesech bývá jeho přítomnost (alespoň občasná, pro kontrolu a seřízení) nezbytná. V praxi je běžná kombinace všech uvedených způsobů řízení.

Pokud sledujeme postavení programovatelných automatů v pyramidě řízení podniku, je jejich hlavní oblast uplatnění na úrovni jedna – to znamená oblast přímého řízení.



MRP/ERP – Manufacturing/Enterprice Resource Planning

MES - Manufacturing Execution System

SCADA - Supervisory Control And Data Acquisition

HMI – Human Machine Interface

PLC - Programmable Logic Controller

DCS - Distributed Control System

Obrázek 1.2: Počítačově řízená výroba.

Zařazení programovatelného automatu mezi řídicí systémy

Relé a kontakty versus programovatelné automaty

Relé, stykače a tlačítka jsou v některých případech nenahraditelné a nemá smysl se bránit jejich použití i v případech, kdy je k řízení použit programovatelný automat. Z bezpečnostních důvodů se takto realizují záložní bezpečnostní okruhy, např. obvod CENTRAL STOP. Nemá smysl bránit se příležitostnému vytváření logických funkcí pomocí propojení kontaktů, obzvláště, pokud tím ušetříme

počty vstupů a výstupů programovatelného automatu a snížíme tak cenu systému. Rozsáhlejší funkce se kontaktní a reléovou technologií již nerealizují a svěříme je důsledně programu programovatelného automatu.

Regulátory versus programovatelné automaty

Rozdíl mezi těmito dvěma tradičními kategoriemi výrobků - mezi regulátory a programovatelnými automaty - se postupně stírá. Obě dvě skupiny těchto dnes vyráběných výrobků pracují číslicově. Hranice je tak neostrá, že je mnohdy obtížné rozhodnout, zda produkt je ještě regulátorem nebo již programovatelným automatem. Výrobci programovatelných automatů postupně expandují do aplikačních oblastí dosud patřících regulátorům a to jak aplikacemi tradičních PLC, tak i vývojem nových výrobků, které se mnohdy odklánějí od původní koncepce PLC. Nejvíce je tento trend viditelný v technice budov, kde dochází k prolínání úloh řízené technologie, elektroenergetiky, tepelné techniky a vzduchotechniky s logistikou a monitorováním, nebo též při kompletním řízení energovodů a produktovodů. Na druhé straně tradiční výrobci regulátorů naopak své programovatelné regulátory uzpůsobují i pro logické řízení. Vyrábějí např. inteligentní regulátory, které mají přidány logické funkce a jsou schopny pracovat v různých režimech podle stavu řízeného systému, a pronikají s nimi do oblastí typických pro nasazení programovatelných automatů. Často je tedy rozlišení mezi programovatelným automatem a regulátorem v dnešní době spíše formální a je záležitostí tradice a firemní obchodní politiky.

Personální počítače versus programovatelné automaty

Někdy se setkáváme s přímým řízením technologických procesů standardním PC, mnohdy umístěným přímo v technologii. Toto řešení je přinejmenším riskantní a diskutabilní. Běžný počítač kategorie PC je produkt spotřební elektroniky a je konstruován pro povoz v prostředí domácností, laboratoří a kanceláří, kde obvykle funguje s vyhovující spolehlivostí. V drsných průmyslových podmínkách mnohdy selhává (bývá málo spolehlivý, je citlivý na rušení, a přepětí, nemá potřebnou životnost). Problémy vznikají už s pouhým připojením většího počtu vstupních a výstupních vodičů a s jejich odrušením. Průmyslové počítače (IPC, IC) se někdy používají při přímém řízení strojů a technologií, někdy jen v roli inteligentního operátorského panelu nebo komunikačního adaptéru. Problémem při jejich nasazování je vysoká cena. Jsou tedy účelné jen tam, kde je zdůvodněna, zejména při archivaci a zpracování velkých objemů dat, při využití obrazovky a standardního počítačového ovládání, při využívání standardních programových produktů, při využívání výkonných komunikací, při řešení geometrických a jiných výpočetně náročných úloh. Přímé řízení počítačem je dnes účelné jen v laboratorních podmínkách, pro potřeby výuky a řízení laboratorních a modelových úloh. Je-li v průmyslových podmínkách použití PC nezbytné, pak je nutno použít průmyslový typ. Standardním řešením je použití distribuovaného systému, kdy osobní počítač je použit ve velínu nebo na dispečerském pracovišti (nebo prostě v kanceláři mistra, energetika nebo technologa) a do drsného průmyslového prostředí jsou předsunuty programovatelné automaty.

Někteří výrobci nabízejí v sortimentu modulů svého programovatelného automatu i počítačový modul kompatibilní s PC. V něm lze odpovídajícími prostředky řešit úlohy příslušející počítači (složité a rychlé výpočetní algoritmy, grafické a geometrické úlohy, zpracovávání a archivace velkého množství dat, databázové úlohy, výkonné komunikace, napojení do počítačové sítě).

1.3 Hlavní charakteristiky programovatelných automatů

Výhody:

Rychlá realizace

Hlavní předností programovatelných automatů je možnost rychlé realizace systému. Technické vybavení nemusí uživatel vyvíjet. Stačí navrhnout a včas objednat vhodnou sestavu modulů

programovatelného automatu (konfiguraci) pro danou aplikaci, vytvořit projekt, napsat a odladit uživatelský program – a pak to vše realizovat a uvést do chodu.

Spolehlivost, odolnost, diagnostika

Technické vybavení programovatelných automatů je navrženo tak, že jsou extrémně spolehlivé i v drsných průmyslových podmínkách, jsou odolné proti rušení i poruchám, vyznačují se robustností a spolehlivostí. Programovatelné automaty bývají vybaveny i vnitřními diagnostickými funkcemi, které průběžně kontrolují činnost systému a včas zjistí případnou závadu, lokalizují ji, bezpečně ji ošetří a usnadní její odstranění.

Snadná přizpůsobitelnost řešení (nekončící změny v zadání)

Jen výjimečně se podaří, že první varianta řešení zůstane tou poslední a konečnou. Představy zadavatele a koncového uživatele, ale i projektanta a programátora postupně zrají, požadavky se průběžně vyvíjejí a rozšiřují. Při uvádění do provozu je třeba všechny funkce důkladně prověřit a odstranit mnohé chyby a slabá místa (každý tvůrce je chybující). Mnohé nedostatky zadání a zvoleného způsobu řešení se projeví právě ve fázi finalizace zakázky – v této fázi přicházejí s řešením poprvé do styku i noví lidé, kteří dříve neměli k zadání a k projektu přístup nebo nebyli schopni domyslet detaily a souvislosti (technolog, energetik, dispečer, operátor a obslužný personál, správce počítačové sítě, šéfové různých úrovní, bezpečnostní technik, ekonom, apod., ale i samotní tvůrci nedokonalého zadání se stávají "po bitvě generály"). Dodatečné požadavky a zadání nových funkcí vznikají i po mnohých měsících a letech rutinního provozu.

U řídicího systému s pevnou logikou (např. s relé) je každá změna zdrojem problémů (mnohdy nepřekonatelných). Při použití programovatelného automatu stačí mnohdy jen opravit, změnit nebo rozšířit uživatelský program. Pokud požadavky vyžadují použití nových vstupů a výstupů, můžeme někdy vystačit s využitím existujících rezerv v konfiguraci (nechávat si 5 až 15 % rezervu je prozíravé). V opačném případě stačí doplnit potřebné moduly (případně další PLC jako podsystém), doplnit projekt a program - a samozřejmě všechno opět důkladně odladit, ověřit, otestovat a zdokumentovat, seznámit operátory a všechny zúčastněné se změnami a doplňky.

Schopnost komunikace

K neopomenutelným výhodám programovatelných automatů patří jejich schopnost komunikace s nejrůznějšími systémy a zařízeními jak v podřízené úrovni, v takzvaném poli (anglicky Field, německy Feld), což je oblast senzorů, měřicích zařízení a akčních členů, tak i v souřadné úrovni s ostatními programovatelnými automaty či jinými řídicími systémy, a v neposlední řadě i směrem k systémům nadřízeným. Právě tato schopnost komunikace umožňuje stavbu distribuovaných nebo i hierarchických systémů řízení z nejrůznějších komponent a od různých výrobců.

Nevýhody:

Prodloužení odezvy

Řídicí systémy s pevnou logikou se od systémů s PLC se liší v době odezvy, tj. v době, za kterou zareagují výstupy na změnu na vstupech systému. V pevné logice jsou všechny logické členy trvale aktivní, algoritmus systému se realizuje paralelně a ve spojitém čase. Odezva na změnu vstupů je dána jen celkovýma zpožděním logických členů v nejdelší větvi. U integrovaných obvodů to bývají řádově nanosekundy až mikrosekundy, u reléových systémů jednotky, desítky, někdy i stovky milisekund.

Odezva PLC může být delší a je dána dobou průchodu programu. Závisí na rychlosti procesoru a na délce aktivní větve programu. Typicky nabývá hodnot v řádu jednotek až desítek milisekund (někdy stovek ms), což pro běžné aplikace postačuje. Je však třeba s touto skutečností počítat, aby v některých případech nebyla příčinou nečekaných "překvapení".

Nespojitost v čase

Dalším důležitým znakem programovatelných systémů je časová nespojitost zpracování. Algoritmus je vykonáván cyklicky, vždy jen v určitých okamžicích. Uvnitř intervalu mezi okamžiky aktivace systém nereaguje na změny vstupních hodnot. Tuto skutečnost je třeba respektovat při návrhu a programování systému, jinak může být příčinou hazardů a chyb, ztráty krátkého vstupního impulsu, nevyhodnocení hrany signálu apod.

Postupnost zpracování

Program PLC je vykonáván v pořadí, v jakém je zapsán, nikoliv v pořadí "toku signálů" v odpovídajícím logickém schématu. Je-li možné zapsat PLC program sousledně s tokem signálů (aby pořadí instrukcí sledovalo tok signálů směrem od vstupů k výstupům), nebývají problémy. U složitých a nepřehledných logických funkcí (a zejména v případech mnoha změn a vsuvek do programu) se to vždy nepodaří. V lepším případě je následkem prodloužení doby odezvy systému (k ustálení hodnoty výstupu je zapotřebí několika cyklů PLC programu). V případě nesystematického návrhu sekvenčních funkcí (se zpětnými vazbami) může být následkem i chybná funkce programu nebo jeho zdánlivě nahodilé chyby (hazardy).

Rozdělení programovatelných automatů podle konstrukčního hlediska

Programovatelné automaty je možno třídit dle různých hledisek. Menší systémy bývají řešeny jako kompaktní, větší jako modulární. Princip činnosti kompaktních i modulárních programovatelných automatů a většinou i způsob programování je stejný, konstrukčním pojetím a uživatelskou koncepcí jsou však obě kategorie výrazně odlišné.

- **Kompaktní programovatelné automaty (KPA)** měly původně pevně danou konfiguraci integrovaných modulů a byly uzavřeny v jednom pouzdře. Toto pouzdro se montuje přímo do výrobku nebo na rozváděcí DIN lišty do rozvaděče. V poslední době je i u KPA snaha o určitý stupeň modularity, takže je i u malých aplikací možnost přizpůsobit sestavu programovatelného automatu k potřebám konkrétní aplikace. Příkladem KPA je například Simatic S5-95U, Simatic S7 200, S7 300, Modicon Micro, Tecomat TC600 apod.
- **Modulární programovatelné automaty (MPA)** jsou svými funkčními schopnostmi a bohatým vybavením vhodné pro automatizační úlohy středního a velkého rozsahu. U MPA se na nosnou desku (lištu, rám) umísťují jednotlivé moduly – napájecí zdroj, procesorová jednotka, moduly vstupů a výstupů apod. Kromě běžných funkcí dostupných u kompaktních automatů, jako jsou binární i analogové vstupně výstupní jednotky zde bývá možnost volby z dalších jednotek pro rychlé čítání, pro polohování, pro nejrůznější typy komunikace, pro regulaci, i pro speciální funkce. Příkladem modulárního programovatelného automatu je Simatic S5 115U, Simatic S7 400, Tecomat TC700 apod.

Rozdělení typů programovatelných automatů podle velikosti

- **mikro PLC** - nejmenší a nejlevnější kompaktní PLC systémy (mikro PLC) nabízejí uživateli pevnou sestavu vstupů a výstupů, obvykle jen binárních, například 6 binárních vstupů / 6 binárních výstupů pro nejmenší systém, pro větší pak sestavy 8/6, 8/8, 12/12 atd. Svým kompaktním provedením, malými rozměry a nízkou cenou (v jednotkách tisíc Kč) se mikro PLC řadí do kategorie "spotřebního materiálu". Typickým použitím programovatelných automatů této kategorie (mikro PLC) je realizace logické výbavy jednoduchých strojů a mechanismů, která se tradičně řešila pevnou reléovou logikou. Vezmeme-li v úvahu ceny ovládacích prvků, relé, stykačů, časových relé a časových programátorů, pak je zjevné, že použití mikro PLC je účelné již u nejprostších aplikací, kde nahrazuje "hrst relé".
- **malé PLC** – jsou to programovatelné automaty, které mají možnost zpracovávat desítky vstupů a výstupů. Jsou vhodné pro úlohy malého rozsahu a poskytují většinou již kompletní

řadu funkcí pro logické i aritmetické operace. Konstrukčně se může jednat jak o kompaktní tak o modulární PLC.

- **střední PLC** – programovatelné automaty této kategorie jsou schopny zpracovávat stovky vstupně-výstupních signálů. Jsou vhodné pro řídicí úlohy středního rozsahu. Bývají již většinou modulární konstrukce.
- **velké PLC** – nejvyšší třída PLC, jsou schopny zpracovávat tisíce vstupů a výstupů. Používají se pro nejnáročnější aplikace. Bývají již prakticky výhradně modulární konstrukce.

Shrnutí pojmů

V dnešní automatizaci se dnes používá řada prostředků na bázi procesoru, které slouží jako základní komponenty řídicích systémů. Jedná se o **programovatelné automaty**, **mikrokontrolery**, **počítače typu PC**, **průmyslové počítače** apod. Každý z těchto systémů je vhodný pro určitou oblast aplikací. V oblasti řízení průmyslových procesů je dnes nejpoužívanějším systémem programovatelný automat. V dnešní automatizaci můžeme vidět rostoucí důraz na distribuovanost řídicích systémů a zároveň na integraci funkcí jednotlivých řídicích prvků.

Programovatelný automat je nejpoužívanější prostředek řídicí techniky v současné automatizaci. Programovatelný automat se používá na nejnižší úrovni počítačově řízené výroby – **na úrovni přímého řízení**. Programovatelné automaty lze podle velikosti rozdělit na **mikro**, **malé**, **střední** a **velké**, podle konstrukce na **kompaktní** a **modulární**. Použití programovatelných automatů má své výhody i nevýhody.

Kontrolní otázky

1. Jaké prostředky na bázi procesoru se používají v dnešní automatizaci?
2. Jaké trendy lze v oblasti automatizace pozorovat?
3. Jaké prostředky na bázi procesoru se v současné době v automatizaci používají? Charakterizujte jejich výhody a nevýhody.
4. Co je to programovatelný automat a ve které části řízení výroby se používá?
5. Nakreslete strukturu počítačově řízené výroby.
6. Co znamená zkratka PLC?
7. Nakreslete blokové schéma programovatelného automatu.
8. Jaké výhody a nevýhody programovatelné automaty mají?
9. Jak lze rozdělit PLC z konstrukčního hlediska?
10. Jak lze rozdělit PLC z hlediska velikosti?

Další zdroje

1-1. Martinásková M., Šmejkal L.: Řízení programovatelnými automaty. ČVUT Praha, 1998.

1-2. Martinásková M., Šmejkal L.: PLC a automatizace. Základní pojmy, úvod do programování. BEN, Praha 1999.

1-3. Stenerson J.: Fundamentals of Programmable Logic Controllers, Sensors, and Communications. Prentice Hall 1999. ISBN 0-13-746124-0.

1-4. Kováč F.: Distribuované riadiace systémy. Vydavatelstvo STU, Bratislava 1998.

Řešená úloha 1.1

Napište program, který provede logickou funkci zadanou následující pravdivostní tabulkou.

Pravdivostní tabulka:

č.	a2	a1	a0	y
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	1
8	1	1	1	1

a0 – a2: M0.0-M0.2

Y: M0.3

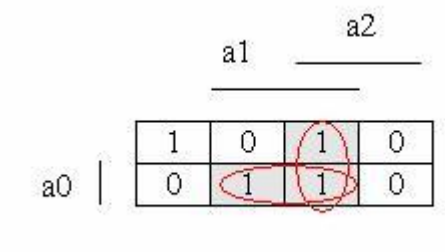
Z tabulky vypíšeme pro jaké kombinace vstupů je výstup roven log.1. Výsledná funkce je pak rovna logickému součtu jednotlivých součinů vstupních proměnných. K získání logické funkce z tabulky je možné použít i opačný postup (vyjádření kombinací pro log.0)

Výsledná funkce podle tabulky:

$Y = \overline{a0}\ \overline{a1}\ \overline{a2} + a0\ a1\ \overline{a2} + \overline{a0}\ a1\ a2 + a0\ a1\ a2$

Tuto funkci je možné dále minimalizovat (zjednodušit) čímž také zjednodušíme její realizaci (méně hradel). K tomu účelu lze požít Karnaughovy mapy. Princip spočívá v zakreslení logických jedniček do tabulky v niž pak hledáme sudé počty nazvaném sousedících jedniček, které se označí jako jeden celek. Takto vzniklé skupiny společně s ostatními jedničkami (které nesousedí s jinými) dávají dohromady zjednodušenou funkci.

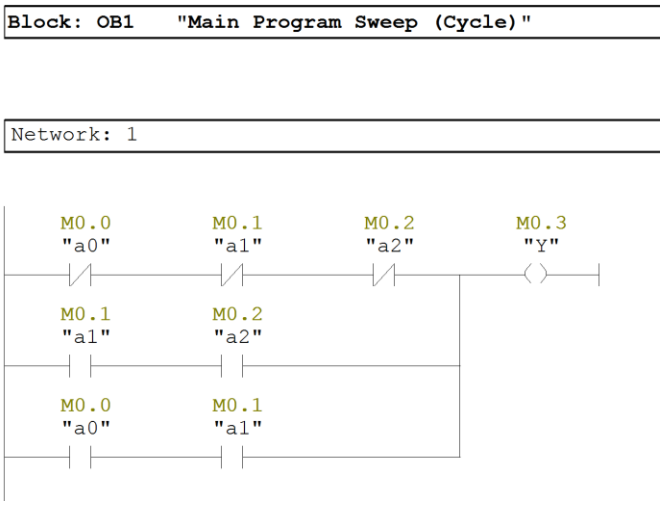
Karnaughova mapa:



Výsledná zjednodušená logická funkce má tvar:

Y = a0 a1 a2+ a1 a2 + a0 a1

Řešení programu v LAD



DVD-ROM

Řešený příklad naleznete na DVD:\cvičení\cvičení1\pr_1_1.zip

2. PROGRAMOVATELNÉ AUTOMATY SIMATIC S7, HARDWARE, ZÁKLADNÍ PRINCIPY

Čas ke studiu: 2 hodiny

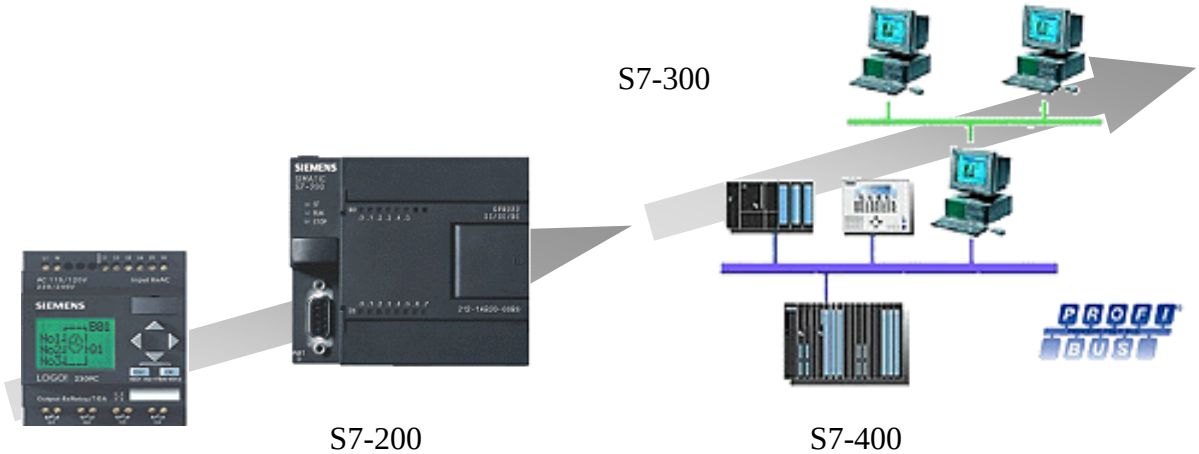
Cíl

V této kapitole se seznámíte s programovatelnými automaty **Siemens Simatic S7**. Jsou zde popsány základní produkty této řady programovatelných automatů. Největší pozornost je věnována automatu **Simatic S7 300**, protože je to velice často používaný programovatelný automat a veškeré řešené úlohy v tomto kurzu, které se týkají automatů Simatic S7, jsou řešeny pro tento automat.

Kapitola se zabývá vlastnostmi **procesorových jednotek** používaných u tohoto programovatelného automatu, strukturou paměti, způsoby **adresování** a použitelnými datovými typy.

Výklad

2.1 Přehled programovatelných automatů SIMATIC S7



Obr. 2.1: Výkonové spektrum nabídky Siemens.

2.1.1 LOGO!

LOGO! je univerzální řídicí a spínací modul určený pro nejjednodušší aplikace. LOGO! zahrnují řídicí člen, klávesnici, zobrazovací jednotku a zdroj. Modulární design znamená možnost nasazení v průmyslu i v automatizaci veřejných a soukromých budov.

Nabízí předprogramované základní funkce, které jsou při každodenní práci často používané, např. logické funkce, čítače, funkce zpožděného zapnutí a vypnutí, proudová pulzní relé nebo zobrazení zpráv na displeji.

LOGO! je možné rozšířit přesně podle požadavků aplikace. Základní modul nabízí 8 vstupů a 4 výstupy, maximální konfigurace je však až 24 vstupů, 16 výstupů, 8 analogových vstupů a 2 analogové výstupy.

Každé LOGO! nabízí 37 předprogramovaných funkcí pro vytváření uživatelského programu. [2-1, 2-13, 2-17, 2-18, 2-19]

Obr.2.2. LOGO!.

LOGO! nabízí:

- Přeprogramované základní funkce, PI regulátor, funkce ramp, analogový multiplexer.
- PI regulátor má nastavenou periodu vzorkování na 0,5ms (není možno měnit).
- Základní modul nabízí 8 vstupů a 4 výstupy z nichž vstupy I7 a I8 jsou analogové 0-10V.
- vysokorychlostní vstupy do 2kHz - I5 a I6 (platí pro LOGO! 12/24 RC/RCo a LOGO! 24/24o – generace OBA5).
- Celkem obsahuje 37 integrovaných funkcí a je možno vytvořit program až o 130 funkcích.
- Různé druhy rozšiřovacích analogových modulů pro PT100, analogové vstupy 4-20mA,0-10V.
- Komunikační moduly AS-interface Slave, instabus EiB.

2.1.2 S7-200

Kompaktní řada malých programovatelných automatů určených k řízení v jednodušších automatizačních aplikacích. Kompaktní design, flexibilní konfigurace a výkonný instrukční soubor jsou důvody, proč je řídicí systém S7-200 výborným řešení pro široké škály automatizačních aplikací.

Flexibilní design umožňuje rozšíření dle požadavků aplikace o digitální a analogové vstupy a výstupy nebo speciální technologické moduly. [2-1, 2-14, 2-17, 2-18, 2-19]

Vlastnosti PLC:

- Doba vykonání 1K bitových instrukcí, min. 0,22µs.
- Maximální rozšíření 128DI/120DO / 28AI/14AO.
- Vysokorychlostní čítače (typicky 30kHz).
- Dva digitální vstupy mohou být konfigurovány jako přerušovací.
- Může řídit až 8 smyček s PID regulátory.
- PID regulátor s možností samočinného nastavení.
- Mohou být připojeny rozšiřovací moduly (polohovací modul pro krokové motory, měření teploty (TC, RTD), vážící systém SIWAREX MS, modul AS – interface, modem modul , PROFIBUS-DP modul, Ethernet modul.

Obr.2.3.: Micro PLC - S7-200.

2.1.3 S7-300

Modulárně rozšiřitelný, volně programovatelný automat pro všechny aplikace v automatizační technice, určený zejména k řízení strojní výroby [2-1, 2-2, 2-15, 2-17, 2-18, 2-19]

Obsahuje:

- Technologické funkce (např. vysokorychlostní čítání, zpětnovazební řízení, motion control, apod.).
- Rychlejší zpracování instrukcí.
- MMC paměťová karta jednodušší údržba – program je možno zálohovat přímo v paměti.

Obr.2.4.: Modulární automat S7-300 (s CPU 319 – 3 PN/DP).

2.1.4 S7-400

Průmyslový řídicí systém SIMATIC S7-400 (obr.2.5.) je určen především pro náročnější automatizační úlohy velkého rozsahu. Jeho doménou jsou zejména velké výrobní celky navazující na celopodnikové řízení zdrojů a systémy pro sběr, archivaci a zpracování technologických dat, jež jsou typické např. pro energetiku, farmacii, chemii, potravinářský průmysl apod. Nižší řady PLC (S7-300; C7; S7-200) předčí svou modularitou a výkonností.

Multicomputing – tj. provoz více než jedné CPU v centralizované konfiguraci řídicího systému; výhod je několik, např. možnost rozdělit výkon podle technických funkcí, jako jsou řízení, početní operace nebo komunikace, které lze tímto způsobem navzájem oddělit a přiřadit je samostatným CPU. Každá z těchto CPU může mít dále připojeny vlastní lokální v/v a může zpracovávat vlastní program, nebo je možné jednu CPU vyčlenit na zpracování časově kritických úloh, jinou na běžné úlohy apod. V režimu multicomputing pracují všechny CPU jako jedna CPU, tzn. jestliže se jedna CPU zastaví, zastaví se současně i všechny ostatní. Synchronizační funkce umožňují koordinovat všechny činnosti „rozdělené“ CPU pro každou instrukci zvlášť.
Izochronní režim – tj. stručně řečeno časová synchronizace procesoru a vzdálených periferií po sběrnici PROFIBUS. Díky přesnému taktování lze spolehlivě obsluhovat i rychlé procesy. Pro úlohy z oblasti řízení pohybu, měření a regulace je k dispozici řada komponent, které izochronní režim podporují.

Možnost změny konfigurace za chodu (CiR) – tj. možnost výměny modulů či změna jejich parametrů za provozu systému. Díky této systémové funkci lze bez jakýchkoliv negativních dopadů na provozované technologické zařízení za plného provozu připojit nové senzory či akční členy. Není-li třeba provoz a výrobní zařízení zastavovat a poté znovu spouštět pokaždé, když se mění hardware, je možné velmi pružně reagovat na změny v procesu a velmi snadno jej různým způsobem optimalizovat. Značně se tak zkracuje doba potřebná k vykonání servisního zásahu a klesají náklady spojené s přerušením výroby. Distribuované sestavy v/v (tzv. slaves, řízené jednotky na sběrnici PROFIBUS-DP, popř. PROFIBUS-PA) i jednotlivé moduly v/v v systémech ET 200M lze přidávat a odebírat a přiřazovat jim nové parametry (např. vybrat jiné hranice přerušení apod.). 2-1, 2-16, 2-17, 2-18, 2-19]

Obr.2.5.: SIMATIC S7-400 – CPU.

Tab. 2.1.: Porovnání základních parametrů programovatelných automatů.

	CPU 224 XP	**CPU 314C-2DP**	**CPU 416-3 DP**
Pracovní paměť max.	12kB – program, (10kB data)	64kB 64kB až 8MB	2,8MB, (2,8MB program)
Doba vykonání 1K bin instrukcí, min.	0,22μs	0,1 μs	0,04 μs
Max velikost paměti.	256kB	8MB	64MB
Čítače	256	256	2048
Časovače	256	256	2048
Počet digitálních vstupů/výstupů,	Max. 168 I/O (14 DI /10 DO integrovaných v CPU)	Max. 1016 I/O (24 DI /16 DO integrovaných v CPU)	131072DI / 121072DO
Počet analog. vstupů/výstupů	30/15 (2AI / 2AO integrated v CPU)	253 (4AI / 2AO integrovaných v CPU)	8192AI / 8192AO
Síťové možnosti	PPI,MPI, Freeport, AS-Interface, PROFIBUS, Ind.Ethernet	PPI,MPI, AS-Interface, PROFIBUS/ PROFINET Ind. Ethernet,	PPI,MPI, PROFIBUS / PROFINET Ind.Ethernet
Real time clock	Integrován	Integrován	Integrován

2.2 PLC S7-300 a jeho základní charakteristiky

Průmyslový řídicí systém SIMATIC S7-300 je nejprodávanějším řídicím systémem z široké nabídky firmy Siemens AG. Je určen pro realizaci rozmanitých automatizačních úloh středního rozsahu. Poskytuje univerzální automatizační platformu pro systémová řešení s hlavním důrazem na výrobní technologii.

Jádrem řídicího systému řady S7-300 je jednotka CPU, která zpracovává uživatelský program.

Podle použití mohou být CPU vybrány podle integrovaných funkcí např. technologické funkce, komunikační rozhraní apod. [2-15, 2-18]

2.2.1 Standardní CPU:

V kategorii standardních CPU lze volit z několika typů. Všechny jednotky jsou standardně osazeny programovacím a komunikačními rozhraními MPI, v některých je zabudováno i rozhraní PROFIBUS (typy 315-2DP, 317-2DP). Novým trendem v současné automatizaci je orientace na standard Ethernet i ve výrobních provozech. Tomu plně vyhovují nové CPU s integrovaným ethernetovým rozhraním (315-2PN/DP, 317-2PN/DP, CPU 319 – 3 PN/DP). Právě díky nim je nyní připojení a obsluha distribuovaných jednotek přes Ethernet jednodušší a zmíněné jednotky lze přes toto rozhraní rovněž programovat. Parametry jednotlivých typů CPU jsou rovnoměrně odstupňovány tak, aby si uživatel mohl velmi snadno vybrat vhodnou jednotku pro danou automatizační úlohu.

CPU nabízí:

- Programování v SCL.
- Sekvenční programovaní S7-GRAPH.
- PLC nabízí jednoduché řízení s přídavnými moduly pro Motion Control a zpětnovazební řízení.
- Motion Control.
- Řešení se STEP 7 bloky a runtime softwarem Standard/Modular PID Control.

Rozšířené procesní diagnostiky se softwarem SIMATIC S7-PDIAG.

- CPU: CPU 312, CPU 314, CPU 315-2DP, CPU 315 – 2 PN/DP, CPU 317-2DP, CPU 317-2PN/DP, CPU 319 – 3 PN/DP.

Obr. 2.6. Příklad standardních CPU a CPU s rozhraním pro PROFINET.

Tab. 2.2.: Přehled standardních CPU.

CPU				
SIMATIC S7-300	**CPU 312**	**CPU 314**	**CPU 315-2DP**	**CPU 317-2DP**
Pracovní paměť	32kB	96kB	128kB	512kB
Doba vykonání				
Bitové operace	0,2 µs	0,1 µs	0,1 µs	0,05 µs
slovo	0,4 µs	0,2 µs	0,2 µs	0,2 µs
Pevná čárka	5 µs	2 µs	2 µs	0,2 µs
Pohyblivá čárka	6 µs	3 µs	3 µs	1 µs
S7 časovače/čítače	128/128	256/256	256/256	512/512
Rozsah adresace				
Digitální kanály	256	1024	1024	1024
Analogové kanály	64	256	256	256
Rozhraní				
MPI	■	■	■	■
PROFIBUS DP			■	■
PROFINET				
PtP - komunikace				

Tab. 2.3.: Přehled standardních CPU s rozhraním pro PROFINET.

CPU			
SIMATIC S7-300	CPU 315-2 PN/DP	CPU 317-2 PN/DP	CPU 319-3 PN/DP
Pracovní paměť	128kB 64kB až 8MB	512kB 64kB až 8MB	1,4MB 64kB až 8MB
Čas zpracování			
Bitové operace	0,1 µs	0,05 µs	0,01 µs
Slovo	0,2 µs	0,2 µs	0,02 µs
Pevná čárka	2 µs	0,2 µs	0,02 µs
Pohyblivá čárka	3 µs	1 µs	0,04 µs
S7 časovače/čítače	256/256	512/512	2048/2048
Digitální kanály	1024		
Analogové kanály	256		
Rozhraní			
MPI	■	■	■
PROFIBUS DP	■	■	■
PROFINET CBA, IO	■	■	■

2.2.2 Kompaktní automaty:

Jako kompaktní se označují CPU doplněné digitálními a analogovými v/v a nejčastěji vyžadovanými základními technologickými funkcemi jako rychlé čítání, měření frekvence, polohování a PID regulace.Všechny typy jsou standardně vybaveny komunikačním rozhraním MPI. Výkonnější procesorové jednotky jsou pak doplněny ještě o rozhraní PROFIBUS (313C-2DP, 314C-2DP) nebo RS422/RS485 (313C-2PtP, 314C-2PtP). Jsou cenově velmi výhodné pro úlohy, které vystačí s příslušným počtem vstupů a výstupů. Jinak je lze samozřejmě doplňovat o další moduly v/v ve stejném rozsahu jako standardní CPU.

- CPU: CPU 312C, CPU 313C, CPU 313 C – 2DP, CPU 313 C – 2PtP, CPU 314 C – 2DP, CPU 314 C – 2PtP.

Obr. 2.7. Příklad kompaktního automatu.

Tab. 2.4.: Přehled kompaktních CPU.

Kompaktní PLC				
SIMATIC S7-300	CPU 312C	CPU 313C	CPU 313C-2DP CPU 313C-2PtP	CPU 314C-2DP CPU 314C-2PtP
Pracovní paměť	32 kB	64 kB	96 kB	96 kB
Čas zpracování				
Bitové operace	0,2 µs	0,1 µs	0,1 µs	0,1 µs
Slovo	0,4 µs	0,2 µs	0,2 µs	0,2 µs
Pevná čárka	5 µs	2 µs	2 µs	2 µs
Pohyblivá čárka	6 µs	3 µs	3 µs	3 µs
S7 časovače/čítače	128/128	256/256	256/256	256/256
Rozsah adresace				
Digitální kanály	266	1016	1008	1016
Analogové kanály	64	253	248	253
Rozhraní MPI PROFIBUS DP PROFINET PtP - communication	 ■	 ■	 ■ ■	 ■ ■ ASCII, 3964 R pouze u PtP

Integrované I/O DI/DO AI/AO	10/6	24/16 4/2	16/16	24/16 4/2
Integrované funkce Čítač Pulzní výstup Polohování Regulace	 2(10 kHz) 2(10 kHz) --- PID Controller	 3(30 kHz) 3(2,5 kHz) --- PID Controller	 3(30 kHz) 3(2,5 kHz) --- PID Controller	 4(60 kHz) 4(2,5 kHz) ■ PID Controller

2.2.3 Bezpečnostní PLC

Bezpečnostní systémy se používají všude tam, kde je třeba zajistit co nejvyšší stupeň bezpečnosti obsluhy, výrobního zařízení či okolního prostředí – např. je-li potřeba předejít nehodám a poškození zdraví či životního prostředí v důsledku poruchy. Uživatel může vytvářet bezpečnostní řídicí systémy v centrálním i distribuovaném provedení. Hlavním znakem je spojení standardní provozní automatizace a bezpečnostní techniky do jediného systému. To znamená, že po síti PROFIBUS-DP mezi centrálním řídicím systémem a distribuovanými moduly v/v probíhá nejen „běžná“ komunikace, ale také bezpečnostně orientovaná komunikace (použití profilu Profisafe) a není nutná žádná samostatná bezpečnostní komunikační linka. Toto spojení standardní a bezpečnostně orientované automatizace značně snižuje výdaje na moderní zabezpečené provozy. Vše je v souladu s osvědčenými a platnými standardy dle světových a evropských norem.

Bezpečnostně orientovaný program se vytváří ve standardních programovacích jazycích reléových schémat (LD) a funkčních bloků (FBD) podle IEC 611131-3 při použití certifikovaných příkladů, které jsou k dispozici ve speciální tzv. F-knihovně. Pro decentralizované struktury jsou k dispozici bezpečnostní komponenty z řad ET 200S a ET 200M.

- CPU: CPU 315F-2DP, CPU 315F-2PN/DP, CPU 317F-2DP, CPU 317F-2PN/DP.

Obr 2.8. Příklad bezpečnostního PLC.

Tab. 2.5.: Přehled bezpečnostních CPU.

Bezpečnostní CPU				
SIMATIC S7-300	**CPU 315-F 2 DP**	**CPU 315-F 2 PN/DP**	**CPU 317-F 2 DP**	**CPU 317-F 2 PN/DP**
Pracovní paměť	192kB	256kB	1MB	
Čas zpracování				
Bitové operace	0,1 µs		0,05 µs	
slovo	0,2 µs		0,2 µs	
Pevná čárka	2 µs		0,2 µs	
Pohyblivá čárka	3 µs		1 µs	
S7 časovače/čítače	256/256		512/512	
Rozhraní				
MPI	■	■	■	■
PROFIBUS DP	■	■	■	■
PROFINET CBA,IO		■		■

2.2.4 Technologické CPU:

V technologickém CPU 317T-2 DP jsou přímo začleněny výkonné technologické funkce a funkce pro řízení polohy a pohybu. Je navrženo pro dynamické řízení pohybu v několika osách současně. Předprogramované funkce pro řízení pohybu podle standardu vydaného organizací PLCopen, integrované v/v, izochronní režim sběrnice PROFIBUS-DP – to vše přispívá k pohodlnému a flexibilnímu řízení pohybu současně v několika osách (např. nastavování polohy, synchronizace a spínání s použitím vaček). Osy lze snadno konfigurovat a parametrizovat ve vývojovém prostředí STEP 7.

- CPU: CPU 315T-2DP, CPU 317T-2DP.

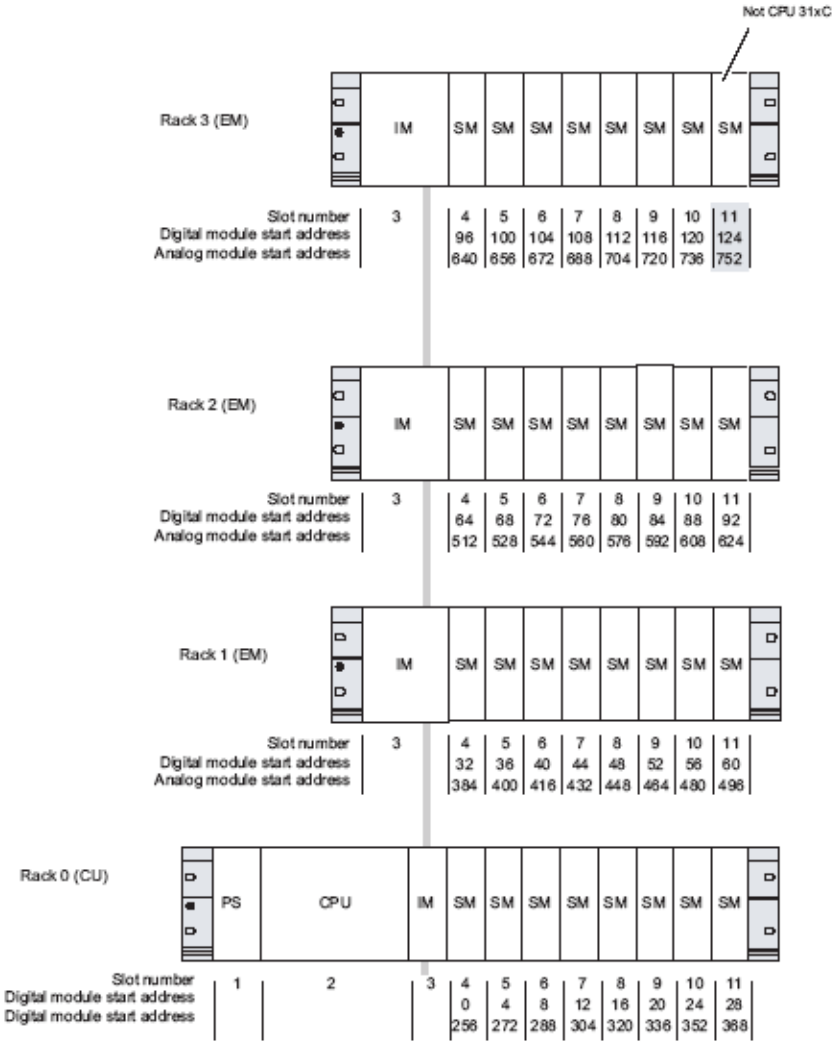
Obr. 2.9.. Příklad technologického PLC.

Tab. 2.6.: Přehled technologických CPU.

Technologické CPU		
SIMATIC S7-300	CPU 315T-2DP	CPU 317T-2DP
Pracovní paměť	128kB	512kB
Čas zpracování		
Bitové operace	0,1 µs	0,05 µs
slovo	0,2 µs	0,2 µs
Pevná čárka	2 µs	0,2 µs
Pohyblivá čárka	3 µs	1 µs
S7 časovače/čítače	256/256	512/512
Digitální kanály	1024	1024
Analogové kanály	256	256
Rozhraní		
MPI	■	■
PROFIBUS DP	■	■
PROFINET		
PtP - komunikace		

Adresování modulů

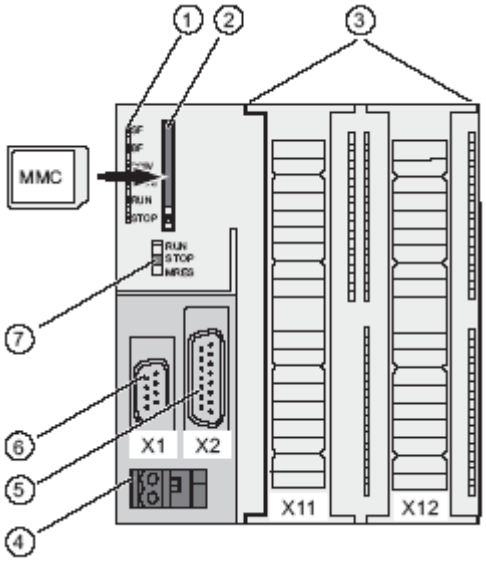
Simatic S7 300 má slotově orientované adresování (jako základní možnost) – každý slot má vyhrazenu počáteční adresu modulu. Následující obrázek a tabulka ukazuje adresaci jednotlivých slotů.



Obr 2.10: Konfigurace S7-300 + rozšíření.

2.3 Technické specifikace PLC S7-300

U PLC S7 31x-2 DP je podporováno rovněž uživatelsky orientované adresování. To znamená, že uživatel si může zvolit adresu u jednotlivých modulů. Výhodou je optimalizace využití adresového prostoru a také to, že programové adresy jsou nezávislé na hardwarové konfiguraci PLC. [2-4, 2-5, 2-10, 2-11, 2-12]



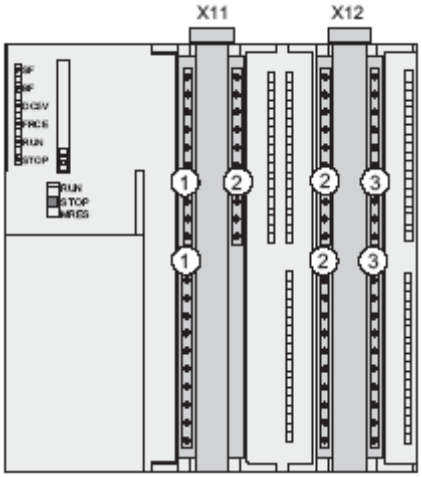
Obr 2.11: Kompaktní PLC S7-300 (CPU xxx C-2PtP(DP)).

Tab. 2.7: Popis vysvětlivek.

Obrázek ukazuje na pozici	**Popis**
(1)	Status a chybový displej
(2)	Slot pro Micro Memory Card (MMC)
(3)	Připojení integrovaných I/O
(4)	Konektor pro připojení napájení
(5)	2. připojení X2(PtP nebo DP)
(6)	1.připojení X1(MPI)
(7)	Přepínač režimů

Tab. 2.8: Popis stavů LED, který signalizuje chybový display.

LED	**Barva**	**Popis**
SF	Červená	Hardwarová nebo softwarová chyba
BF (u CPU s DP)	Červená	Chyba sběrnice
DC5V	zelená	5V napájení pro CPU a signalizace S7-300 sběrnice je OK
FRCE	Žlutá	Force
RUN	Zelená	CPU v režimu RUN
STOP	Žlutá	CPU je v režimu STOP nebo startup



Obr 2.12: Rozmístění I/O na kompaktním PLC.

Tab. 2.9: Popis vysvětlivek.

Obrázek ukazuje na pozici	**Integrované I/O**
(1)	Analogové vstupy/výstupy
(2)	Každý s 8 digitálními vstupy
(3)	Každý s 8 digitálními výstupy

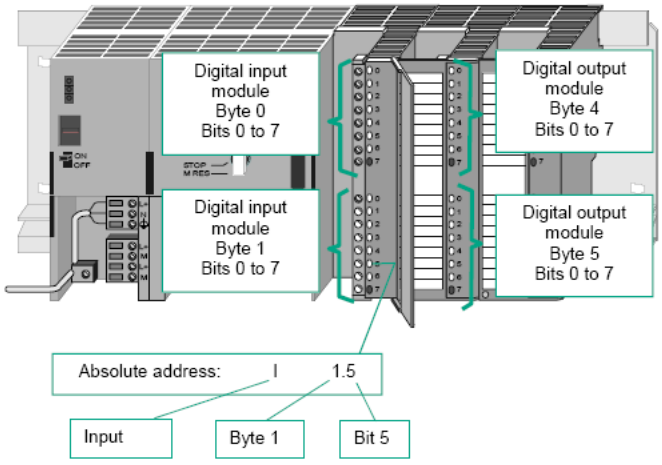
Adresování digitálních I/O modulů

Adresa vstupu nebo výstupu se skládá z bytové a bitové části.

Např. I 124.2 – vstup (I – input), byte 124, bit 2

Adresování analogových I/O modulů

Adresa pro vstupní a výstupní analogový kanál je vždy slovo, tedy 16b. Adresy se tedy zvyšují po dvou (bytech).

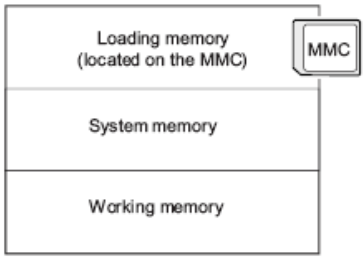


Obr. 2.13: Příklad adresování na PLC.

2.4 Paměť PLC

Paměť PLC je rozdělena na 3 paměťové oblasti.

- Load memory je umístěna v SIMATIC Micro Memory Card (MMC). Velikost load memory je shodná s velikostí SIMATIC Micro Memory Card (MMC). Paměť se používá pro uložení bloků, datových bloků a systémových dat (konfigurace, spojení, parametry modulu, apod.). [2-3]
- System memory: RAM system memory je integrovaná do CPU a nemůže být rozšířena. Obsahuje:
  - Rozsah adresace memory bitů, časovačů a čítačů.
  - I/O process image.
  - Lokální data.
- RAM: RAM je integrována do CPU a nemůže být rozšířena. Běží zde uživatelský program. Program běží pouze v RAM a system memory.



Obr. 2.14: Rozdělení paměti.

2.4.1 Remanence load memory, system memory a RAM

CPU obsahuje zálohovanou paměť tzn. že data a paměť není vymazána, když vypnete napájení nebo restartujete PLC (warm start)

- Zálohovaná paměť v load memory : Program je vždy zálohován, je uložen v SIMATIC Micro Memory Card, je uložen i v případě výpadku napájení nebo restartu PLC.
- Zálohovaná data v system memory: Diagnostic buffer, MPI adresa (a přenosová rychlost) a počítadlo provozních hodin jsou zapsány do retentive memory oblasti CPU. Záloha MPI adresy a přenosové rychlosti zajistí, že vaše CPU může pokračovat v komunikaci dokonce po výpadku napětí, memory resetu nebo ztrátě nastavení komunikace
- Retentive data v RAM: Obsah zálohovaných DB jsou vždy zálohovány při restartu, zapnutí a vypnutí napájení.

2.4.2 SIMATIC Micro Memory Card (MMC)

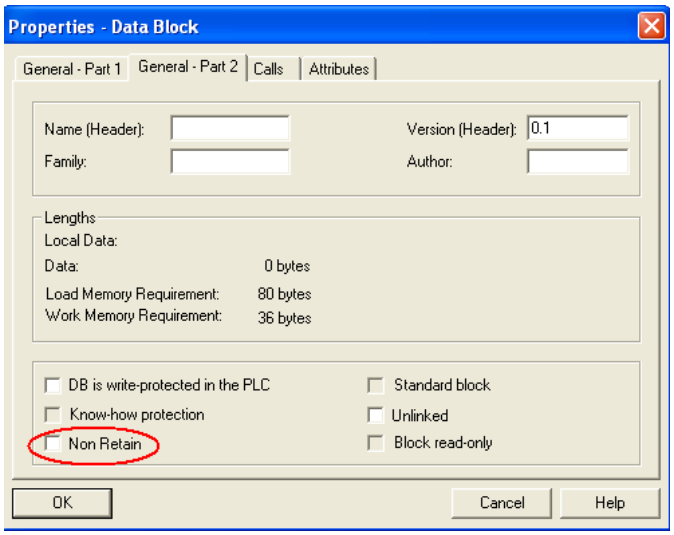
MMC karty se začaly používat v programovatelných automatech od poloviny roku 2001. MMC karty se začaly používat pro load memory. Výhody MMC karet jsou značné. Umožňuje např. zálohu dat do MMC karty, možnost uložení celého programu včetně symboliky a komentářů. Ztráta dat nenastane ani když je vypnuto napájení. Tím je také zajištěna jednoduchá údržba, že nepotřebujete baterie v PLC. Nyní MMC karty mohou mít velikost až 8MB. [2-3]

Když se vytváří program pro PLC mohou se aktuální hodnoty ukládat anebo také nemusí. Mohou být ukládány merkery, časovače, čítače a hodnoty v datových blocích. To jestli se data ukládají, závisí jednak na typu procesoru, na hardwarové konfiguraci a také na nastavení vlastností jednotlivých datových bloků.

Zálohování jednotlivých dat závisí na použitém CPU a firmwaru. Detailnější popis lze získat v dokumentaci [2-3].

Zálohování datových bloků

Stavy proměnných v datových blocích mohou být zálohovány. Datové bloky, které jsou nahrány do load memory, jsou vždy zálohovány. S S7-300 procesory a C7 mohou mít defaultně nastavenou vlastnost zálohy jednotlivých bloků anebo S7-300 CPU nemusí mít tuto vlastnost nastavenu defaultně tzn. lze tuto vlastnost nastavit pro každý datový blok zvlášť, jak ukazuje obr. 2.15.



Obr. 2.15: Nastavení remanence datového bloku.

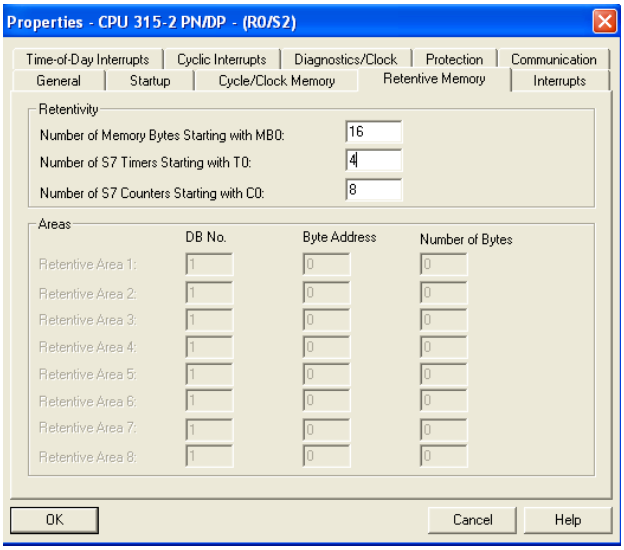
Přečtení sériového čísla MMC

Sériové číslo přečtete pomocí systémové funkce SFC51 „RDSSYST“.

Na vstupní parametr SSL_ID je nutné nastavit číslo W#16#011C a na vstup INDEX nastavit W#16#0008.

Zálohování merkerů, časovačů, čítačů ve STEP7

Zálohování časovačů, čítačů, merkerů se provádí v hardwarové konfiguraci, kliknutím na CPU pravým tlačítkem myši a výběrem *Object Properties*…. Automaticky se zálohuje diagnostický buffer, čas, program v MMC a počítadlo provozních hodin.



Obr. 2.16: Nastavení remanence datového bloku.

Formátování a reset MMC

Pokud bliká led STOP s frekvencí 0,5Hz může MMC obsahovat následující chyby:

- MMC karta není naformátovaná.
- MMC karta obsahuje konfigurace různých CPU typů.
- Aktuální verze CPU nepodporuje velikost paměti.
- MMC obsahuje operační systém, který není kompatibilní s CPU.

Uživatel nemůže formátovat MMC kartu. Všechno co se může udělat je pouze ji resetovat.

Pokud na CPU bliká žlutá led tj. STOP bliká pomalu 0,5Hz, přepínač stiskněte do polohy MRES a držte přepínač zde asi 9s dokud nebude LED STOP svítit. Během příštích 3 sekund musíte přepínač znovu nastavit do polohy MRES. LED STOP bude nyní blikat během procesu mazání.

Pokud ani toto nevyřeší problém s tím, že bude LED pomalu blikat je nutné kartu vyměnit.

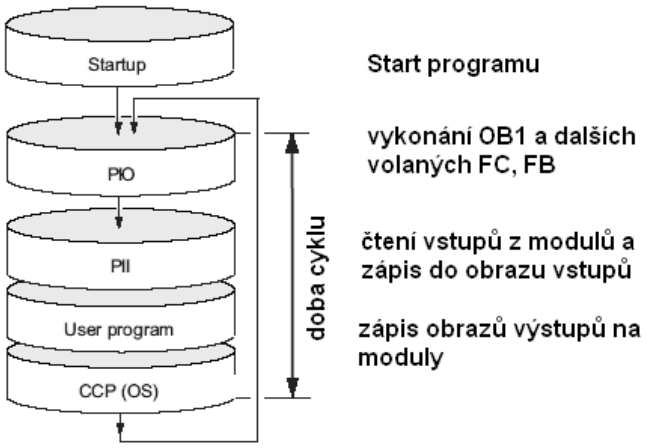
Na kartu můžete zapisovat nebo z ní číst pouze, pomocí následujícího zařízení:

- Field PG.
- Power PG.
- MMC programming adapter (MLFB 6ES7798-0BA00-0XA0) for use with PG 720 or PG 740.
- USB prommer (MLFB 6ES7792-0AA00-0XA0).

!!!!!! Pokud MMC kartu vymažete v nějakém zařízení, zrušíte vnitřní strukturu a nemůžete již tuto MMC kartu použít pro aplikaci v PLC.!!!!!

2.5 Process Images

Process image je speciální část paměti, ve které jsou uloženy stavy vstupů nebo výstupů. Operační system provádí update process image periodicky.



Obr.2.17: Cyklus programu.

2.6 Logické rozdělení paměti –adresování

System memory CPU je rozdělena do paměťových oblastí uvedených v tabulce níže. Pomocí těchto instrukcí můžete vytvářet instrukce ve vašem programu.

Tab. 2.10. Oblasti paměti.

Paměťové oblasti	**Přístup do paměti**	**S7 označení (IEC)**	**Popis**
Obraz vstupů	Input (bit)	**I**	Na začátku každého cyklu, CPU čte výstupy ze vstupních modulů a zapisuje jednotlivé stavy vstupů do této paměti např. I124.0
	Input byte	**IB**	Např. IB 124
	Input word	**IW**	Např. IW 126
	Input double word	**ID**	Např. ID 128
Obraz výstupů	Output (bit)	**Q**	Na konci cyklu se z obrazu výstupů z této paměťové oblasti zapíší stavy výstupů na výstupní moduly např. Q124.0
	Output byte	**QB**	Např. QB 0
	Output word	**QB**	Např. QW 2
	Output double word	**QB**	Např. QD 4
Bit memory	Memory (bit)	**M**	Tato paměťová oblast se využívá např. pro výpočty
	Memory byte	**MB**	Např.MB 20
	Memory word	**MW**	Např. MW 200
	Memory double word	**MD**	Např. MD 220

Paměťové oblasti	**Přístup do paměti**	**S7 označení (IEC)**	**Popis**
Časovače	Timer (T)	**T**	Paměťová oblast vyhrazená pro časovače např. T32
Čítače	Counter (C)	**C**	Paměťová oblast vyhrazená pro čítače
Datové bloky	Data block, opened with „OPN DB“:	**DB**	Datové bloky obsahují informace pro program z datových bloků mohou vyčítat informace různé FB, FC. Instanční datové bloky jsou přiřazeny vždy nějakému FB nebo SFB
	Data bit	**DBX**	Např. DBX 0.0
	Data byte	**DBB**	Např. DBB 2
	Data word	**DBW**	Např. DBW 4
	Data double word	**DBD**	Např. DBD 6
	Data block, opened with „OPN DI“	**DI**	
	Data bit	**DIX**	
	Data byte	**DIB**	
	Data word	**DIW**	
	Data double word	**DID**	
Lokální data	Local data bit	**L**	Tato paměťová oblast obsahuje dočasná data, když je datový blok vykonáván
	Local data byte	**LB**	
	Local data word	**LW**	
	Local data double word	**LD**	
Peripheral (I/O) area: **inputs**	Peripheral input byte	**PIB**	Peripheral input and output dovolují pomocí přímého čtení přímo přistupovat na vstupně výstupní moduly
	Peripheral input word	**PIW**	Např. PIW272
	Peripheral input double word	**PID**	Např. PID274
Peripheral (I/O) area: **Outputs**	Peripheral output byte	**PQB**	
	Peripheral output word	**PQW**	Např. PQW272
	Peripheral output double word	**PQD**	Např. PQD274

2.7 Přímé, nepřímé, absolutní, symbolické adresování

Přímé a nepřímé adresování

První část operandu je pro přímé i nepřímé adresování stejná. Druhá část operandu (parametr je rozdílný pro každou z adresovacích metod.

Přímé adresování: Druhá část operandu ukazuje přímo na pozici v paměti, touto částí je adresa. Adresa ukazuje přímo na pozici slova.

Např.

```
A   M0.0
R   M10.0
=   Q124.0
```

Nepřímé adresování: Druhá část adresy udává pozici slova nebo číslo nepřímo ukazatelem. Ukazatel je:

- Word, který obsahuje slovo nebo číslo čítače, datového bloku , funkce nebo funkčního bloku.
- Double word, který obsahuje přesnou pozici slova v paměti.

Příklad nepřímého adresování

```
A   I[MW2]      //provede logickou operaci AND na vstupu, jehož adresa je uložena v MW2
L   IB[DBD4]    // nahraje vstupní bit, jehož adresa je v double wordu 4 do akumulátoru
```

Absolutní adresování obsahuje identifikátor a typ dat (např. MW 100, I 0.5). Nepotřebujeme symboliku, ale program je bez ní hůře čitelný a podává méně informací o tom, co se v něm vykonává.

Symbolické adresování: adresa obdrží jméno, pod kterým je v programu používána. Symboly pro vstupy, výstupy, časovače, čítače, merkery a bloky jsou uloženy v symbolické tabulce. Symbolické adresování se týká i lokálních proměnných, parametrů bloků a návěští. Symbolická jména mohou mít také vysvětlující komentáře. Symbolické jméno může mít délku max. 24 znaků, komentář 80 znaků.

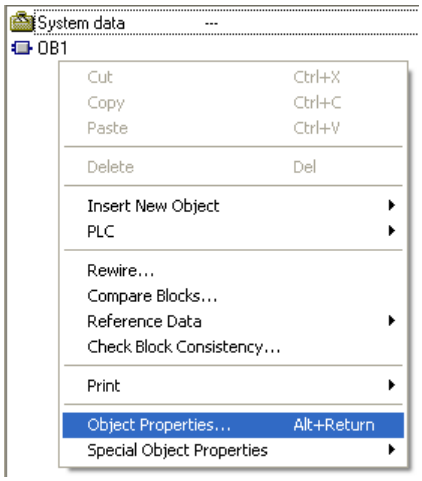
Programátor může přiřadit jednotlivým adresám symboly, aby byl program čitelnější – symbolické adresování.

Nastavení priority pro symbolické adresování

Když zakládáte nový projekt ve STEP7, měli by jste si také nastavit symbolické adresování, protože často je potřeba, aby jste např. v projektu dodržovali symboliku bez ohledu na změnu adresy např. v datovém bloku. Toto vám zajistí, že budete moci např. datový blok rozšiřovat a stále budete mít správnou adresu v datovém bloku.

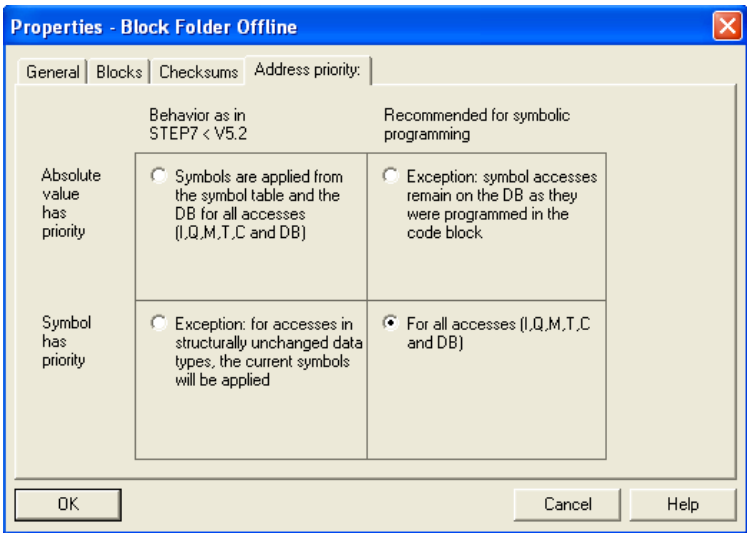
Vytvoření nastavení pro sledování symboliky:

Klikněte pravým tlačítkem myši někde do prostoru, kde se zakládají organizační bloky, FC, FB, DB. Zvolte z nabídky *Object Properties* (obr.2.18).



Obr. 2.18: Vyvolání vlastností projektu.

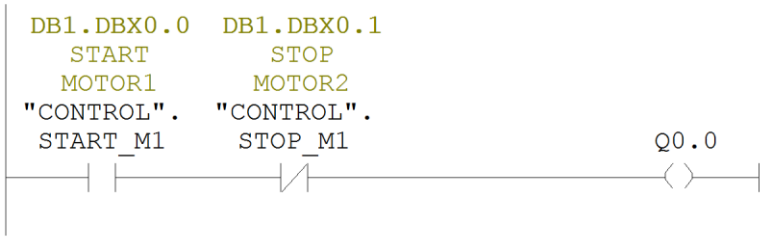
Z následujícího dialogového okna klikněte na záložku *Address priority* a vyberte *For all accesses (I, Q, M, T, C and DB)* obr. 2.19.



Obr. 2.19: Nastavení priority pro symbolické adresování.

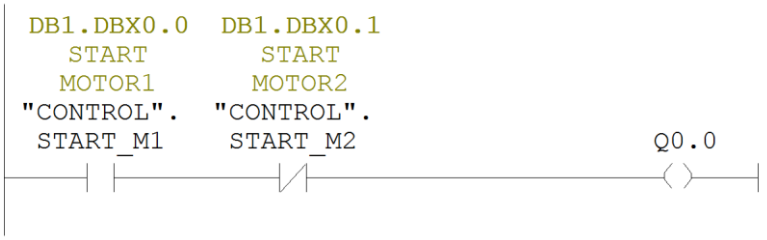
Nyní již je nastavena priorita pro symbolické adresování. Co nabízí tato možnost vysvětluje následující příklad.

Např. vytvořím si sdílený datový blok a v něm nadefinuji dvě proměnné typu boolean, které budu používat pro spouštění a zastavování motoru. Řešení ukazuje obr. 2.20.

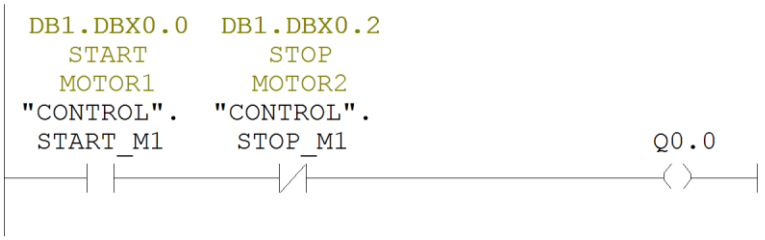


Obr. 2.20: Původní program.

Nyní ale budeme chtít provést změnu v datovém bloku a přidat na adresu 0.1 další proměnnou, která bude spouštět motor č. 2. tzn. že se mi proměnná STOP_M1 musí přesunout na adresu 0.2. Kdyby nebyla nastavená priorita sledování symbolů, vypadalo by řešení podle obr. 2.21. Je vidět, že řešení není vhodné, protože se přepsaly adresy, tzn. bylo by třeba opravit adresy, abych bylo dosaženo řešení, jaké ukazuje obrázek 2.20. Pokud byla nastavena prioritu pro symbolické adresování, bude řešení správné, i když byly změněny v datovém bloku (obr.2.22).



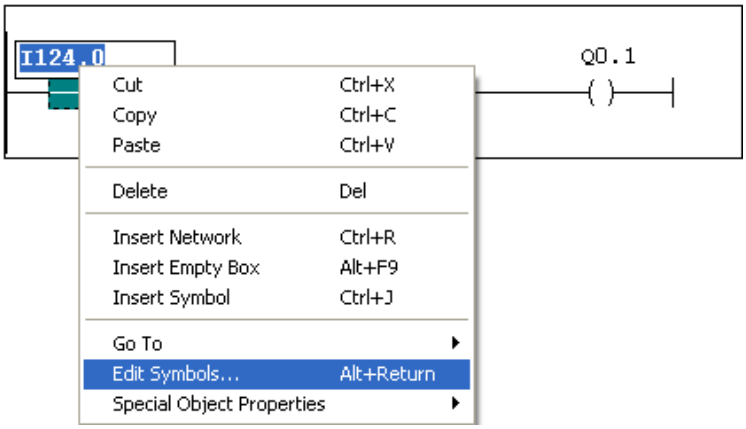
Obr. 2.21: Změna adres.



Obr. 2.22: Změna adres s nastavenou prioritou.

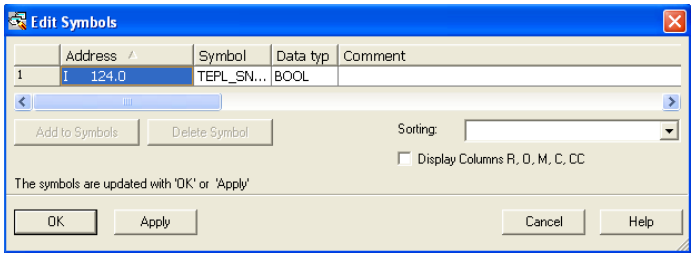
Pojmenování adres symbolickými výrazy

Označením např. vstupu a kliknutím pravým tlačítkem myši na označený vstup lze z nabídky vybrat *Edit Symbols* a pojmenovat daný vstup obr. 2.23



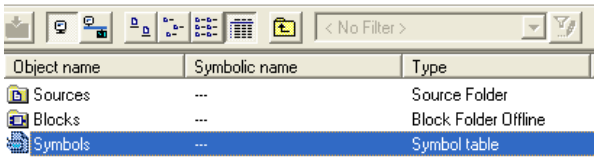
Obr.2.23: Vyvolání editoru symbolů.

V dialogovém okně nyní stačí pouze napsat symbolické pojmenování (obr.2.24) a po stisku tl. OK již bude vstup I124.0 pojmenován TEPL_SNIMAC_1.



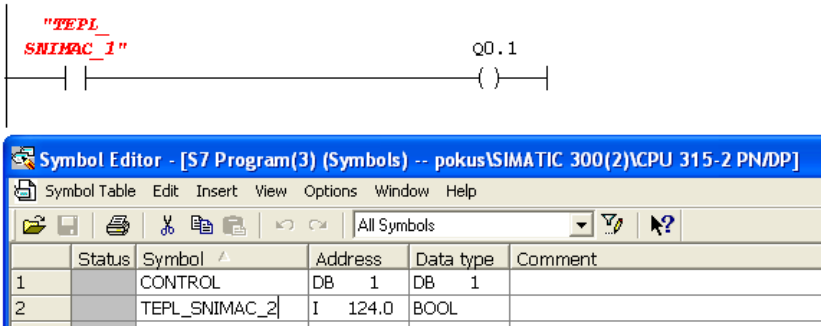
Obr. 2.24: Editování symbolů.

Více symbolů najednou lze vkládat přímo do tabulky symbolů. Dvojklikem levým tlačítkem myši na *Symbols* obr. 2.25. můžete vkládat v symbolické tabulce do jednotlivých sloupců symboly s aktuálními adresami.

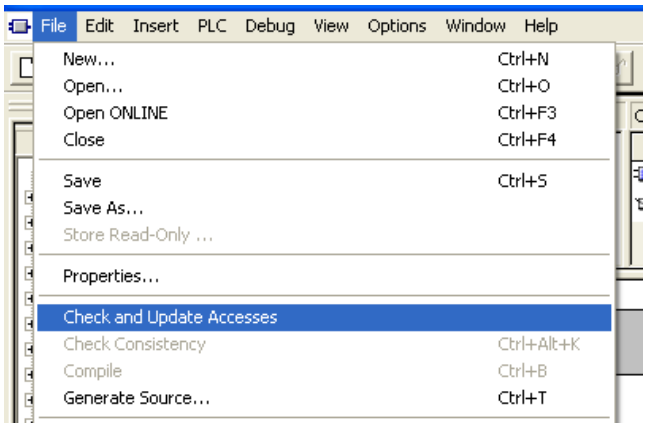


Obr.2.25: Vyvolání symbolické tabulky.

Pokud se stane, že např. po přepsání symbolického pojmenování např. v našem případě TEPL_SNIMAC_1 na TEPL_SNIMAC_2, se symbolické vyjádření nad kontaktem v LAD např. v OB1 zčervená, musíte zvolit v menu *File* → *Check and Update Accesses* (obr. 2.26). STEP7 provede update symbolů a symbol se přepíše na TEPL_SNIMAC_2.



Obr. 2.26: Změna symbolu u vstupu I124.0.

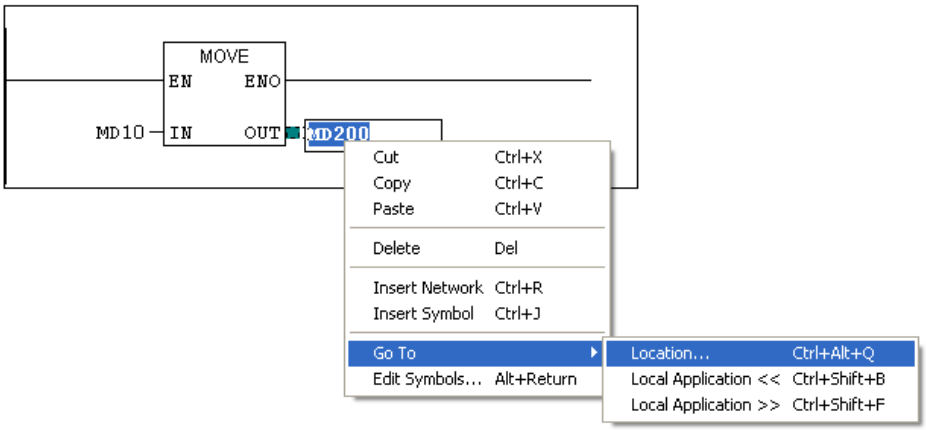


Obr. 2.27: Update symbolů.

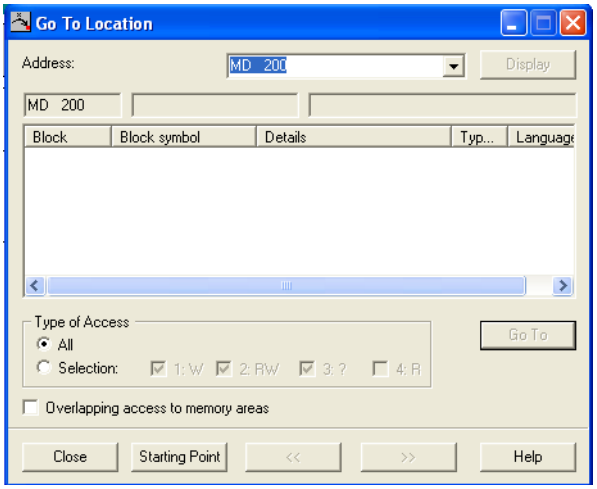
Vkládání nových proměnných

Při vytváření proměnné se může stát, že si jednotlivé proměnné můžete přepisovat např. používáte-li již proměnnou MD 220 a znovu si ji vytvoříte. Pokud si jednotlivé proměnné nevypisujte do symbolické tabulky, tak může být někdy i obtížné najít tuto chybu zvláště u rozsáhlých projektů. Proto je vhodné si proměnnou při jejím vytvoření zkontrolovat, zda již není použita v programu. Pokud jste si již vytvořili odkazy na data (Reference data), můžete postupovat podle níže uvedeného popisu.

Označte proměnnou, klikněte na tuto proměnnou pravým tlačítkem myši a z menu vyberte *Go To* → *Location* (obr.2.28). STEP7 sám prohledá program, jestli ji již nepoužíváte. Pokud proměnnou nenajde v okně *Go To Location,* nevypíše cestu k proměnné.



Obr. 2.28: Vyhledání proměnné.



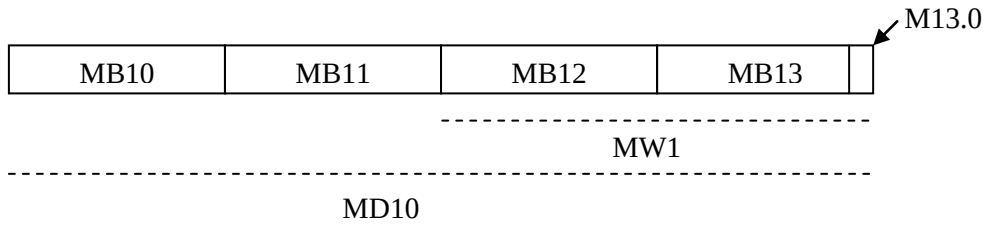
Obr. 2.29: Okno se zobrazením hledané proměnné.

2.8 Datové typy ve Step7

Analogové hodnoty se v programovatelném automatu zpracovávají při práci s analogovými vstupy a výstupy, při řešení různých aritmetických operací, u čítačů a časovačů apod. Analogové proměnné musí být určitého datového typu. Ve STEP7 existují následující datové typy:

Elementární datové typy

- BOOL, BYTE, WORD, DWORD – základní typy o velikosti 1, 8, 16 a 32 bitů. Používají se tehdy, pokud chceme načíst nebo zapsat hodnotu určité paměťové buňky v binárním tvaru.
- BCD čísla – jedná se o hodnoty zapsané v BCD kódu, tedy v podstatě hexadecimálně, přičemž jsou využity pouze číslice 0-9. To se využívá hlavně u čítačů a časovačů.
- CHAR – představuje jednotlivý znak v ASCII formátu, je to 8 bitová hodnota.
- INT – proměnná Integer – 16-bitové celé číslo. Bit 15 je znaménkový, je-li v něm „1“, jedná se o číslo záporné.
- DINT – obdoba INT, ale jedná se o 32-bitové číslo.
- REAL – tento typ představuje desetinné číslo, je uložen ve 32 bitech. Je složeno ze dvou částí – exponent (bity 23-30) a mantisa (bity 0-22).
- S5TIME – datový typ, který se používá pro předvolbu časových funkcí. Má velikost 16 bitů. Hodnota je v BCD kódu v rozsahu 000-999. Nejvyšší 4 bity slova jsou vyhrazeny pro časovou základnu.
- DATE – datový typ uložený ve slově, udává počet dnů od 1.1.1990.
- TIME – datový typ zabírá dvě slova, udává čas – den, hodina, minuta, sekunda, milisekunda.
- TIME_OF_DAY – zabírá dvě slova a udává počet ms od počátku dne.



Obr. 2.30: Umístění hodnot v paměti.

Z obrázku 2.30 je vidět, že například slovo MW 12 se skládá z bytů MB12 a MB13, přičemž MB 13 je nižší! Je také jasné že slova MW 11 a MW 12 se částečně překrývají.

Složené datové typy

- DATE_AND_TIME – datum a čas, typ zabírá 8 bytů, hodnoty jsou zde uloženy v BCD formátu.
- STRING – řetězec o délce 254 znaků. Maximální délka stringu je uložena v prvním bytu. Aktuální délka je uložena ve druhém bytu, pak následuje vlastní text.
- ARRAY – představuje pole sestavené z pevného počtu komponent (max. 65536). Jeho prvky je možné v programu používat jako běžné proměnné. Index pole nelze v programu dynamicky měnit.
- STRUCT – datová struktura složená z pevného počtu komponent, které mohou být různého datového typu. Je možné vytvářet i vnořené struktury, vnoření lze provést až do šesté úrovně.

Tab. 2.11: Přehled elementárních datových typů.

Typ popis	**Velikost v bitech**	**Datový typ**	**Rozsah**	**Příklad**
BOOL(Bit)	1	Boolean text	TRUE/FALSE	TRUE
BYTE (Byte)	8	Hexadecimal number	B#16#0 to B#16#FF	L B#16#10 L byte#16#10
WORD (Word)	16	Binary number	2.0 to 2#1111_1111_1111_1111	L 2#0001_0000_0000_0000
		Hexadecimal number	W#16#0 to W#16#FFFF	L W#16#1000
		BCD Decimal	C#0 to C#999	L C#998
		Decimal number unsigned	B#(0.0) to B#(255.255)	L B#(10,20)
DWORD (Double word)	32	Binary number	2#0 to 2#1111_1111_1111_1111 1111_1111_1111_1111	2#1000_0001_0001_1000_ 1011_1011_0111_1111
		Hexadecimal number	DW#16#0000_0000 to DW#16#FFFF_FFFF	L DW#16#00A2_1234 L dword#16#00A2_1234
		Decimal number unsigned	B#(0,0,0,0) to B#(255,255,255,255)	L B#(1, 14, 100, 120) L byte#(1,14,100,120)
INT (Integer)	16	Decimal number signed	-32768 to 32767	L 1
DINT (Integer, 32 bits)	32	Decimal number signed	L#-2147483648 to L#2147483647	L L#1
REAL (Floating-point number)	32	IEEE Floating-point number	Upper limit: 3.402823e+38 Lower limit: 1.175 495e-38	L 1.234567e+13
S5TIME (SIMATIC time)	16	S7 time in steps of 10 ms (default)	S5T#0H_0M_0S_10MS to S5T#2H_46M_30S_0MS and S5T#0H_0M_0S_0MS	L S5T#0H_1M_0S_0MS L S5TIME#0H_1H_1M_0S_0MS
TIME (IEC time)	32	IEC time in steps of 1 ms, integer signed	-T#24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS L TIME#0D_1H_1M_0S_0MS
DATE (IEC date)	16	IEC date in steps of 1 day	D#1990-1-1 to D#2168-12-31	L D#1996-3-15 L DATE#1996-3-15
TIME OF	32	Time in	TOD#0:0:0.0 to	L TOD#1:10:3.3

Typ popis	Velikost v bitech	Datový typ	Rozsah	Příklad
DAY (Time)		steps of 1 ms	TOD#23:59:59.999	L TIME_OF_DAY#1:10:3.3
CHAR (Character)	8	ASCII characters	'A','B' etc.	L 'E'

Uživatelské datové typy

Uživatelský datový typ je složen z komponent libovolného datového typu. Na jednotlivé komponenty UDT se dotazuje jako na komponenty struktury.

Shrnutí pojmů

Řada programovatelných automatů Siemens Simatic S7 obsahuje následující produkty - **LOGO!**, **Simatic S7 200**, **Simatic S7 300**, **Simatic S7 400**. Programovatelný automat se používá pro malé až střední aplikace, dokáže zpravovat stovky vstupně/výstupních signálů, může mít buď kompaktní nebo modulární konstrukci. Disponuje celou řadu procesorových jednotek (CPU), včetně tzv. technologických CPU.

Paměť PLC Simatic S7 300 je rozdělena na 3 paměťové oblasti - Load memory, System memory RAM a uživatelská RAM. Paměť obsahuje celou řadu **datových oblastí** – logické členění paměti.

Přístup k datům nebo periferiím lze realizovat pomocí **přímého** (absolutního nebo symbolického) nebo **nepřímého adresování.**

Programovací jazyk Step7 pro automaty Simatic S7 300/400 umožňuje používat **elementární**, **složené** a **uživatelské** datové typy.

Kontrolní otázky

1. Jaké modely programovatelných automatů jsou v řadě Siemens Simatic S7?
2. Popište základní vlastnosti PLC Simatic LOGO!.
3. Popište základní vlastnosti PLC Simatic S7 200.
4. Popište vlastnosti PLC Simatic S7 300.
5. Z jakých součástí je sestaven modulární programovatelný automat Simatic S7 300?
6. Jak je členěna paměť u PLC Simatic S7 300? Fyzické i logické členění.
7. Jaký způsob adresování má PLC Simatic S7 300?
8. Popište přímé adresování u PLC Simatic S7 300.
9. Popište nepřímé adresování u PLC Simatic S7 300.
10. Jaké datové typy se používají ve Step7.

Další zdroje

2-1. Siemens: SIMATIC Products for Totally Integrated Automation and Micro Automation, Katalog ST 70, 2005.

2-2. Siemens: SIMATIC Automation System S7-300 Getting Started Collection, 01/2006, A5E00123662-05.

2-3. Siemens: Memory Concepts for SIMATIC S7-300 CPUs and for C7 Devices, ID:7302326.

2-4. Siemens: SIMATIC S7-300 and M7-300 Programmable Controllers Module Specifications, Edition 2, EWA 4NEB 710 6067-02 01.

2-5. Siemens: SIMATIC S7-300 Automation System CPU 31xC Technological Functions, Edition 05/2003, A5E00105484-03.

2-6. Siemens: SIMATIC Programming with STEP7 V5.3, Edition 01/2004, A5E00261405-01.

2-7. Siemens: SIMATIC Working with STEP7 V5.3 Getting Started, 01/2004, A5E00261403-01.

2-8. Siemens: SIMATIC Configuring Hardware and Communication Connections STEP7 V5.3, 01/2004, A5E00261404-01.

2-9. Software STEP7 V5.3 + SP3, Help.

2-10. Siemens: SIMATIC S7-300, CPU 31Xc and CPU 31x: Installation Operating Instructions, Release 01/2006 A5E00105492-06.

2-11. Siemens: SIMATIC CPU 31xC, CPU 31x, IM151-7 CPU, BM 147-1CPU, BM 147-2 CPU, Edition 01/2006, A5E00105517-07.

2-12. Siemens: SIMATIC S7-300 CPU 31xC and CPU 31x, Technical Data, Manual, 01/2006 Edition A5E00105475-06.

2-13. www.siemens.com/logo

2-14. www.siemens.com/S7-200

2-15. www.siemens.com/S7-300

2-16. www.siemens.com/S7-400

2-17. www.siemens.cz/tia-nadosah

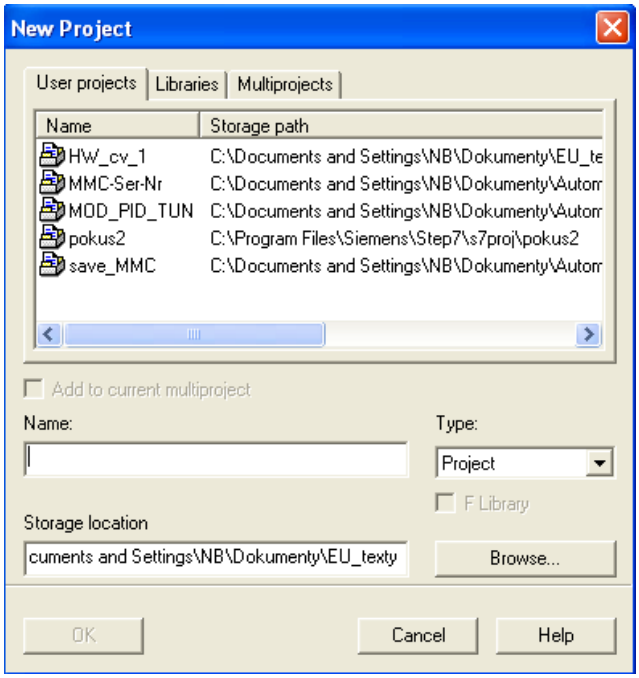
2-18. http://www1.siemens.cz/ad/current/prezentace/as/index.php?mi=1

2-19. http://support.automation.siemens.com

Řešená úloha 2.1

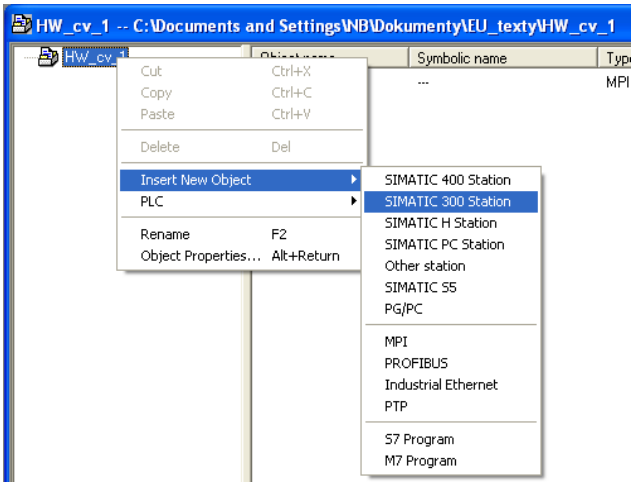
Zadání: Vytvořte hardwarovou konfiguraci pro PLC S7-300.

Kliknutím na ikonu  se spustí SIMATIC Manager. SIMATIC Manager je rozhraní z kterého můžeme editovat celou aplikaci. Nejjednodušší způsob jak vytvořit novou aplikaci je zvolit z nabídky **File** položku **'New Project' Wizard...** . Tento způsob vytvoření hardwarové konfigurace, ale není ve většině případů vhodný, protože průvodce neobsahuje všechny druhy PLC. Proto zvolíme v menu z nabídky File položku New. Následuje okno pro pojmenování projektu.



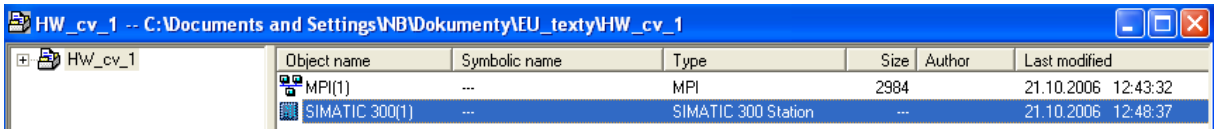
Obr. 2.31: Založení nového projektu.

Kliknutím na tlačítko OK se založí nový projekt. Kliknutím pravým tlačítkem myši na ikonu Vámi pojmenovaného projektu, se zobrazí dialogové menu, kde výběrem položky Insert New Object → SIMATIC 300 Station se specifikuje, vytvoření hardwarové konfigurace pro PLC S7-300 (obr. 2.32).



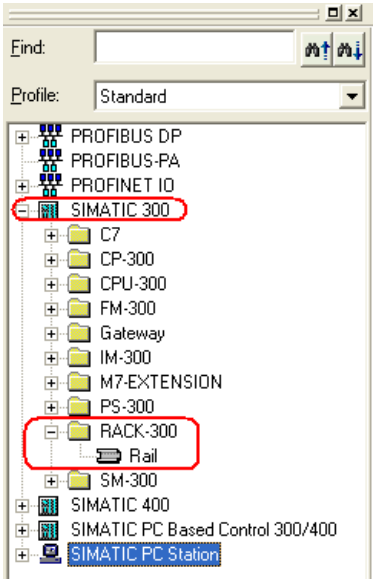
Obr. 2.32: Vložení stanice pro PLC S7-300.

Dvojklikem levým tlačítkem myši na SIMATIC 300 (obr. 2.33), v následujícím okně kliknutím na hardware, se otevře hardwarová konfigurace.



Obr.2.33: Přechod do hardwarové konfigurace.

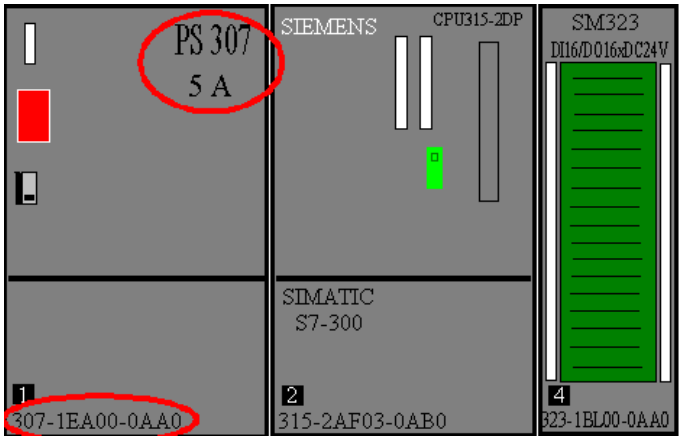
V pravé části okna by se měl zobrazit katalog. Pokud se tak nestane, musí se v menu vybrat položka View a v pak položku *Catalog* nebo použít klávesovou zkratku Ctrl +K. Základní jednotka CPU se spolu s rozšiřujícími kartami, funkčními moduly a komunikačními jednotkami umístí na lištu Rail. Proto se nová konfigurace nejprve zahájí výběrem z katalogu položky *RACK 300* → *Rail* podle (obr. 2.34).



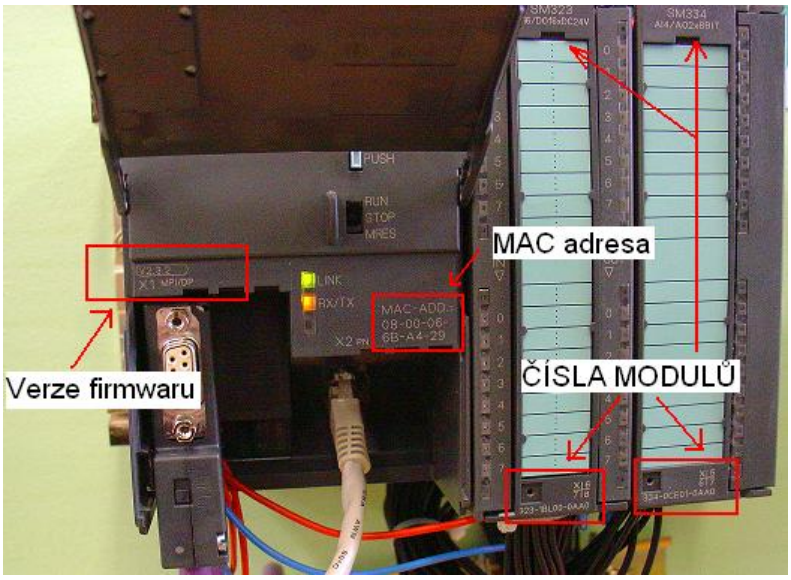
Obr. 2.34: Katalog.

Na první pozici RACKu se vkládá napájecí zdroj. Napájecí zdroje jsou umístěny ve složce PS-300. Označením pozice 1 v RACKu - kliknutím na místo levým tlačítkem myši, lze tento zdroj vložit dvojklikem na vybraný zdroj např. potřebujeme napájecí zdroj 5A s označením 6ES7 307-1EA00-0AA0.

Na pozici číslo dva se vkládá centrální procesorová jednotka. Tato volba je asi nejdůležitější na celé konfiguraci. Každá jednotka se liší svým označením např. CPU 314 nebo CPU 315PN/DP. Jak bylo na přednášce číslo 2 uvedeno, SIEMENS vyrábí několik řad těchto PLC. Každé PLC nese na svém vnějším krytu číslo např. PLC s CPU 315-2DP má dole specifikováno přesné označení tj. 315-2AF03-0AB0 (obr.2.35).

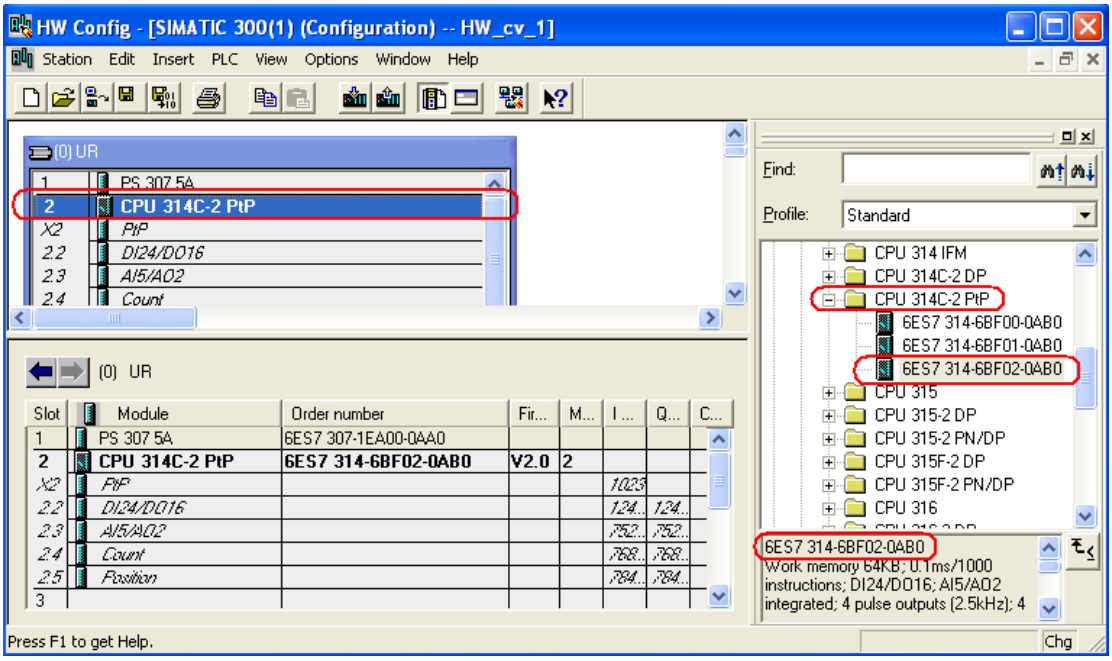


Obr. 2.35: Označení jednotlivých modulů.



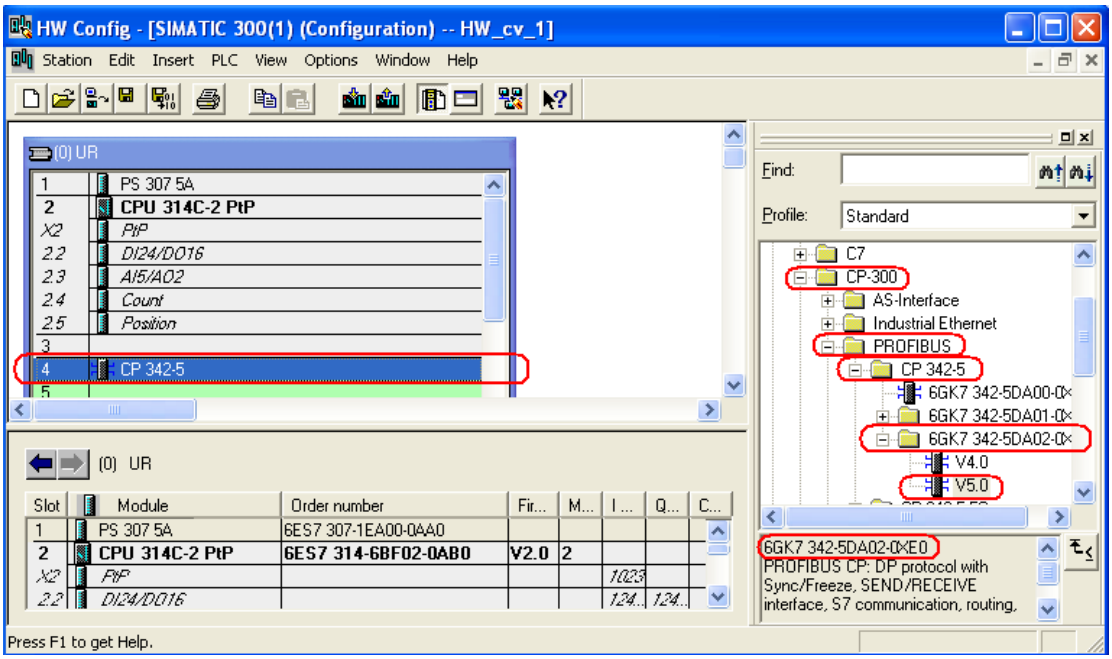
Obr. 2.36: PLC s rozhraním pro PROFINET.

Odklopením spodní části PLC (kde je umístěn konektor pro připojení kabelu s rozhraním RS 232 se nachází další označení CPU tj. verze firmwaru. Většina PLC disponuje firmwarem verze 2.x, ale některé mají také firmware verze 1.x. Do druhého slotu budeme vkládat programovatelný automat s označením ***CPU 314C-2PtP*** s označením ***6ES7 314-6BF02-0AB0***. Kliknutím na složku ***CPU-300*** se otevřou jednotlivé podsložky. Otevřením složky s označením ***CPU 314C-2PtP*** a kliknutím na označení PLC ***6ES7 314-6BF02-0AB0*** se přetažením levým tlačítkem myši na slot 2 umístí centrální jednotka obr. 2.37.



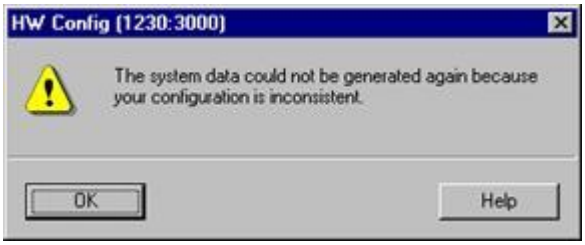
Obr.2.37: Vložení CPU 314C-2PtP.

Za jednotkou CPU je umístěn ve slotu 4 také komunikační procesor s označením ***CP 342-5*** s ***firmwarem 5.4*** a označením ***6GK7 342-5DA02-0XE0***. Tyto komunikační procesory jsou umístěny ve složce ***CP-300 → PROFIBUS→CP 342-5 → V5.0***. Kliknutím na tento modul a přetažením do slotu 4 je hardwarová konfigurace ukončena obr. 2.38.



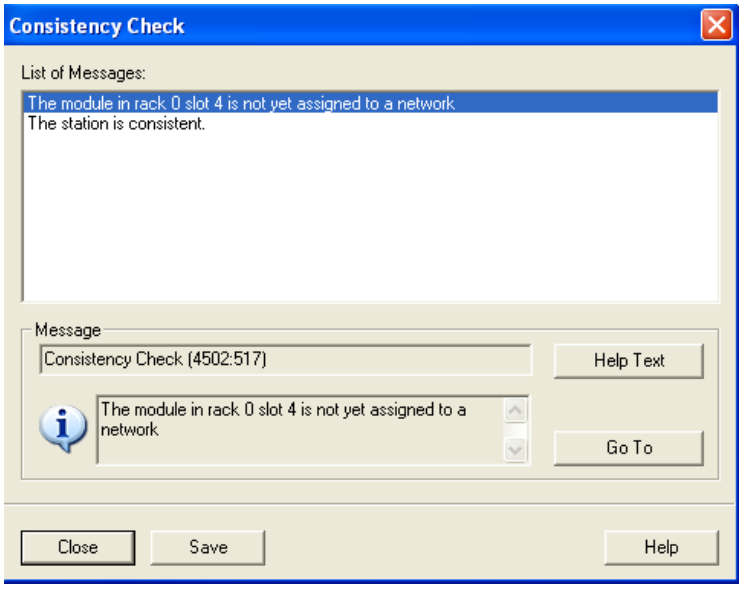
Obr. 2.38: Vložení komunikačního procesoru CP 342-5.

Po ukončení vkládání modulů je nutné vytvořenou konfiguraci uložit a zkompilovat. Volbou v menu ***Station → Save and Compile*** nebo klávesovou zkratkou ***Ctrl+S*** se provede uložení a kompilace hardwarové konfigurace. Pokud je HW konfigurace neúplná zobrazí se hlášení obr. 2.39.



Obr. 2.39: Hlášení o neúplnosti HW konfigurace.

Problém u jednoduchých projektů se odhalí poměrně snadno. Ale u rozsáhlejších projektů to může být problém. Volbou v menu ***Station → Consistency Check*** se vyvolá okno s přesným popisem úplnosti nebo neúplnosti HW konfigurace.

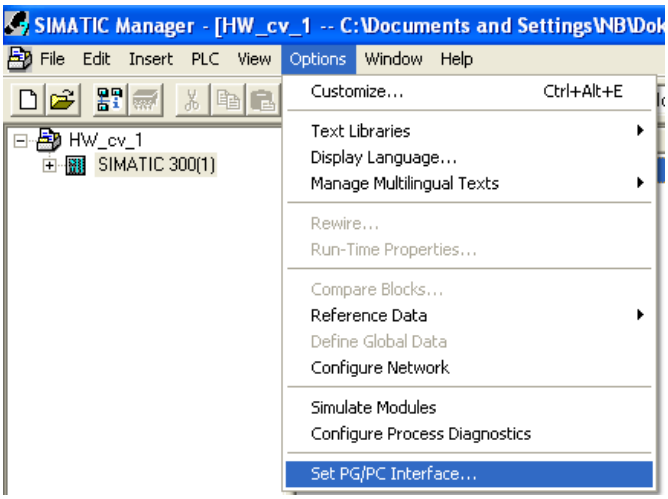


Obr. 2.40: Výpis HW konfigurace.

Kliknutím na jednu z hlášek a poté kliknutím na tlačítko ***Go To*** se přejde na rack a slot, ve kterém může být chyba. Tento výpis o úplnosti HW konfigurace může být také uložen kliknutím na tlačítko ***Save***.

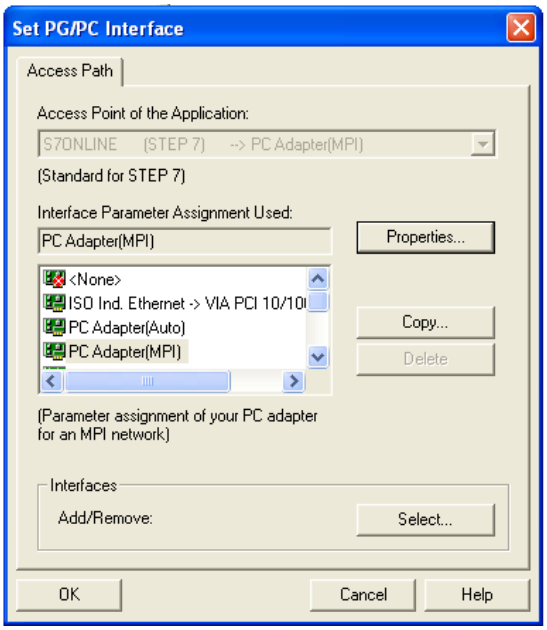
Download do PLC

Download do PLC je možné provést buď přes MPI kabel nebo přes Ethernet. Nastavení komunikace přes Ethernet je popsáno ve cvičení 11. MPI kabel může být připojitelný k PC přes rozhraní RS 232 nebo přes USB. Nastavení daného rozhraní se provádí v základním okně volbou v menu Options → Set PG/PC Interface obr.2.41.

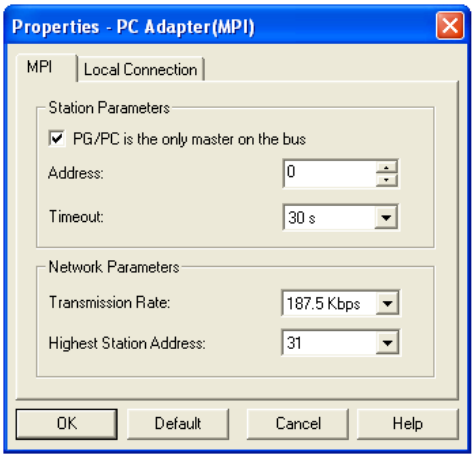


Obr. 2.41: Nastavení rozhraní pro MPI kabel.

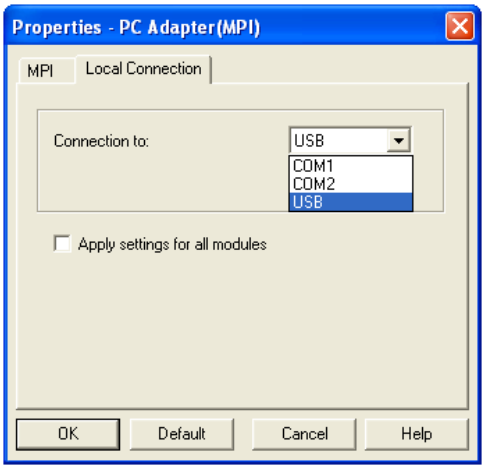
PLC má přepínač pro volbu režimu. Pouze v režimu RUN nebo STOP (u PLC s firmwarem 2.x a vyšším) lze provést download hardwarové konfigurace do PLC. U starších PLC s firmwarem 1.x měl přepínač čtyři polohy. Pokud byl přepínač v poloze RUN-P nebo STOP bylo možné provést download programu a hardwarové konfigurace.



Obr. 2.42: Volba komunikace např. přes MPI.



Obr. 2.43: Parametry komunikace přes MPI.



Obr. 2.44: Nastavení připojení adaptéru k PC.

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 2\pr_2_1.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 1\ anim1.avi.

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 1\ anim2.avi.

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 1\ anim3.avi.

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 2\ anim1.avi.

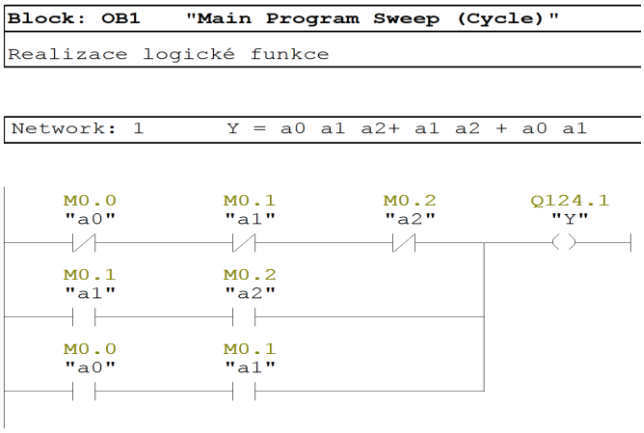
Řešená úloha 2.2

Zadání:

Vytvořte program, který bude vykonávat funkci Y = a0 a1 a2+ a1 a2 + a0 a1. Místo použitých vstupů použijte M bity (M0.0 – M0.2, M1.0 – M1.2).

Výsledná funkce:Y = a0 a1 a2+ a1 a2 + a0 a1

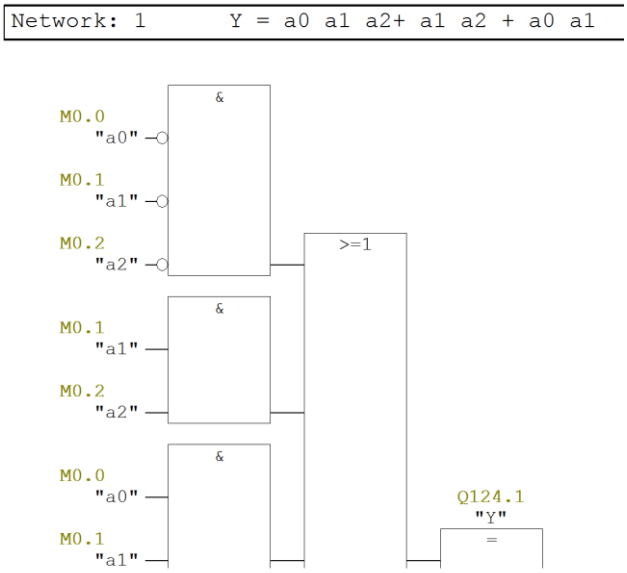
Řešení programu v LAD



Řešení programu v STL

```
Network: 1      Y = a0 a1 a2+ a1 a2 + a0 a1

      AN    "a0"  M0.0
      AN    "a1"  M0.1
      AN    "a2"  M0.2
      O
      A     "a1"  M0.1
      A     "a2"  M0.2
      O
      A     "a0"  M0.0
      A     "a1"  M0.1
      =     "Y"   Q124.1
```

Řešení programu v FBD



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 2\pr_2_2.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 7\ anim1.avi.

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 7\ anim2.avi.

Řešená úloha 2.3

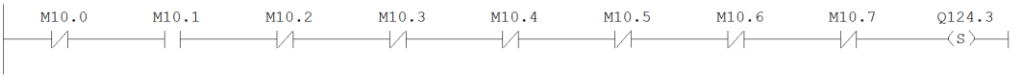
Zadání:

Nastavte Q124.3 v případě že, MB10 = 2 a resetujte Q124.3 v případě, že MB10=7.

Řešení programu v LAD

```
Block: OB1    "Main Program Sweep (Cycle)"
```

```
Network: 1      Logická funkce
nastaveni Q124.3 v případě, e MB10 = 2
```




```
Network: 2      Logická funkce
reset Q 124.3 v případě, že MB10 = 7
```

Řešení programu v STL

```
Block: OB1    "Main Program Sweep (Cycle)"
```

```
Network: 1      Logická funkce
nastaveni Q124.3 v případě, e MB10 = 2
```

```
AN    M     10.0
A     M     10.1
AN    M     10.2
AN    M     10.3
AN    M     10.4
AN    M     10.5
AN    M     10.6
AN    M     10.7
S     Q    124.3
```

```
Network: 2      Logická funkce
reset Q 124.3 v případě, že MB10 = 7
```

```
A     M     10.0
A     M     10.1
A     M     10.2
AN    M     10.3
AN    M     10.4
AN    M     10.5
AN    M     10.6
AN    M     10.7
R     Q    124.3
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 2\pr_2_3.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 3\ anim1.avi.

3. PROGRAMOVATELNÉ AUTOMATY SIMATIC S7 300, ZÁKLADY PROGRAMOVÁNÍ V JAZYCE STEP7, LOGICKÉ FUNKCE

Čas ke studiu: 2 hodiny

Cíl

Kapitola popisuje **strukturu programu** u PLC S7 300/400. Definuje pojem **cyklus programu** a komentuje vlivy jednotlivých činností na délku tohoto cyklu. Dále jsou zde popsány jednotlivé typy **programových a datových bloků**, které je možné u uvedených programovatelných automatů použít.

Kapitola popisuje typy **vstupně/výstupních signálů**, se kterýma je možné u programovatelných automatů setkat a popisuje základní instrukce pro programování logických funkcí. Jsou zde uvedeny příklady zápisů logických funkcí v jazycích LAD, STL a FBD. Kapitola má zásadní význam pro pochopení využití logických instrukcí při pokročilejším programování.

Kapitola seznamuje s **časovači a čítači** u programovatelných automatů Simatic S7300/400. Nabízí výčet všech typů časovačů a čítačů a demonstruje jejich použití v programu. Po prostudování kapitoly bude student schopen zvolit správný časovač nebo čítač pro příslušnou aplikaci.

Výklad

3.1 Program v S7 300

Má dvě hlavní části – operační systém a uživatelský program [3-4].

- Operační systém – řídí činnost CPU – např. řídí chování při restartu, aktualizuje PII, PIQ, volá uživatelský program, detekuje chyby programu, spravuje paměť, komunikuje s programovacím prostředím.
- Uživatelský program – vytváří si jej uživatel. Obsahuje všechny funkce potřebné pro řízení příslušného úkolu. Jazyk Step7 umožňuje strukturovat tento uživatelský program, tedy rozložit jej na individuální, samostatné sekce – program je pak přehlednější, jeho organizace, modifikace a ladění je jednodušší, uvádění systému do provozu je snazší.

Uživatelský program se může skládat z následujících elementů:

Organizační bloky – OB – určují strukturu programu. Jsou rozhraním mezi operačním systémem a uživatelským programem. Řídí start systémy, cyklické, časové provádění, obsluhují přerušení atd. Jsou volány OS.

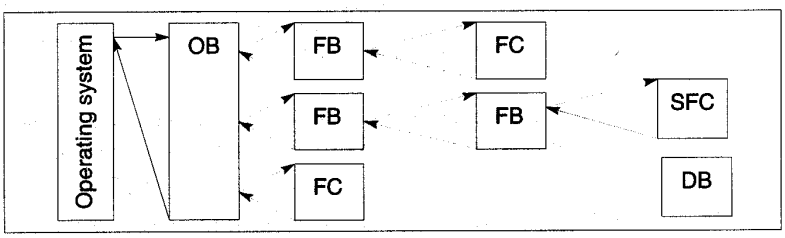
Funkční bloky – FB, funkce – FC – logické bloky, které programuje uživatel. Funkční bloky jsou bloky, které mají přiřazeny paměťovou oblast, která slouží pro ukládání parametrů. Funkce jsou bloky, které tuto paměťovou oblast nemají.

Datové bloky – DB - jsou to datové oblasti obsahující uživatelská data. Mohou to být buď instanční datové bloky pro FB nebo sdílené DB, který může používat jakýkoliv logický blok.

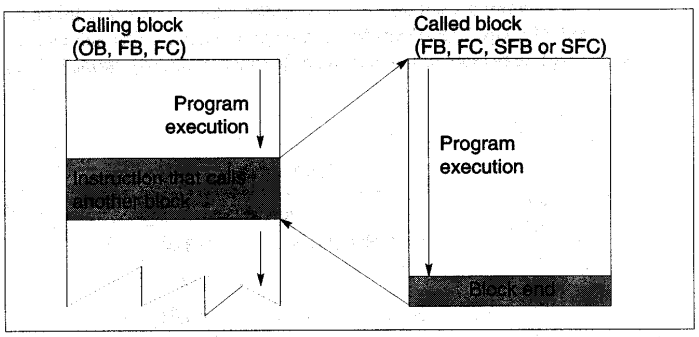
Systémové funkce – SFC, systémové funkční bloky SFB – standardní, předprogramované funkční bloky a funkce. Uživatel je může používat, nemusí je programovat. Tyto bloky jsou součástí operačního systému, nemusí být přenášeny s programem.

Hierarchie volání bloků

Každý blok musí být volán, pokud má být proveden. Počet bloků a hloubka vnoření závisí na procesoru.



Obr. 3.1: Struktura programu.



Obr. 3.2: Volání bloku.

Proměnné bloku

Při psaní bloku můžeme definovat proměnné bloku. Mohou to být:

- parametry, které se mezi bloky přenášejí
- statické proměnné, které jsou uloženy v instančních datových blocích
- dočasné proměnné, které jsou dostupné pouze, když je blok prováděn. Potom jsou přepsány.

Parametry programových bloků mohou být následující:

- Formální parametry – jsou specifikovány při deklaraci proměnných.
- Aktuální parametry – nahrazují formální parametry při volání bloku.

U parametrů je třeba definovat jeho použití – vstupní (IN – FB/FC), výstupní (OUT– FB/FC) a vstupně/výstupní (IN_OUT– FB/FC), statické (STAT- uloženy v instančním DB – FB) a dočasné (TEMP– FB/FC, OB). Dále je třeba specifikovat jeho datový typ a počáteční hodnotu.

3.1.1 Organizační bloky

Organizační bloky určují pořadí, ve kterém jsou jednotlivé programové sekce prováděny. Provádění OB voláním jiného OB. Který OB je oprávněn přerušit jiný OB závisí na prioritě. Bloky s vyšší prioritou mohou přerušit bloky s nižší prioritou. Nejnižší priorita je 1. Prioritu OB u S7 300 nelze měnit. Organizační bloky na pozadí mají nejnižší prioritu, konkrétně 0.29.

Organizační bloky pro start programu – jsou dva rozdílné typy restartu – kompletní restart a restart. S7 300 mají pouze kompletní restart. Během startu systému volá operační systém příslušné OB – OB100 (kompletní restart) a OB 101 (restart). Tyto bloky jsou vyvolány následujícími událostmi – zapnutí napájení, přepnutí STOP-RUN, pokud je restart iniciován programovacím zařízením. Napsáním programu do těchto OB lze specifikovat chování PLC při startu. Není zde omezení délky ani prováděcí doby programu v těchto OB. Přerušení těchto OB není možné. Všechny DO jsou během provádění těchto OB rovny 0.

Organizační blok pro cyklické provádění – cyklické provádění programu je základní způsob provádění programu u PLC. Operační systém volá OB1 cyklicky. Jelikož CPU má obrazy I/O, neadresuje vstupy a výstupy přímo na I/O moduly, ale přistupuje k interní paměťové oblasti, které tyto obrazy obsahují.

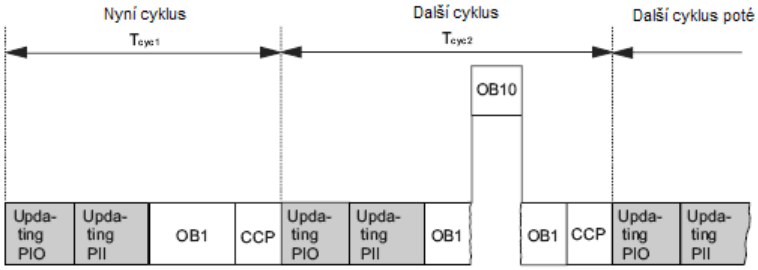
Cyklický program začíná ihned po ukončení startovacího programu. Může být přerušen přerušením, příkazem STOP, výpadkem napájení nebo chybou. Doba cyklu – viz. následující část.

Tab. 3.1: Seznam organizačních bloků.

Typ přerušení	Organizační blok	Priorita	Popis
Hlavní program	OB1	1	Cyklicky vykonávaný organizační blok
Denní časové přerušení (time-of-day interrupts)	OB10-OB17	2	Organizační bloky pro denní časové přerušení (OB10-OB17)
Zpožděné přerušení (time-delay interrupts)	OB20 OB21 OB22 OB23	3 4 5 6	Organizační bloky pro zpožděné přerušení (OB20-OB23)
Cyklické přešení (cyclic interrupts)	OB30 OB31 OB32 OB33 OB34 OB35 OB36 OB37 OB38	7 8 9 10 11 12 13 14 15	Organizační bloky pro cyklické přerušení

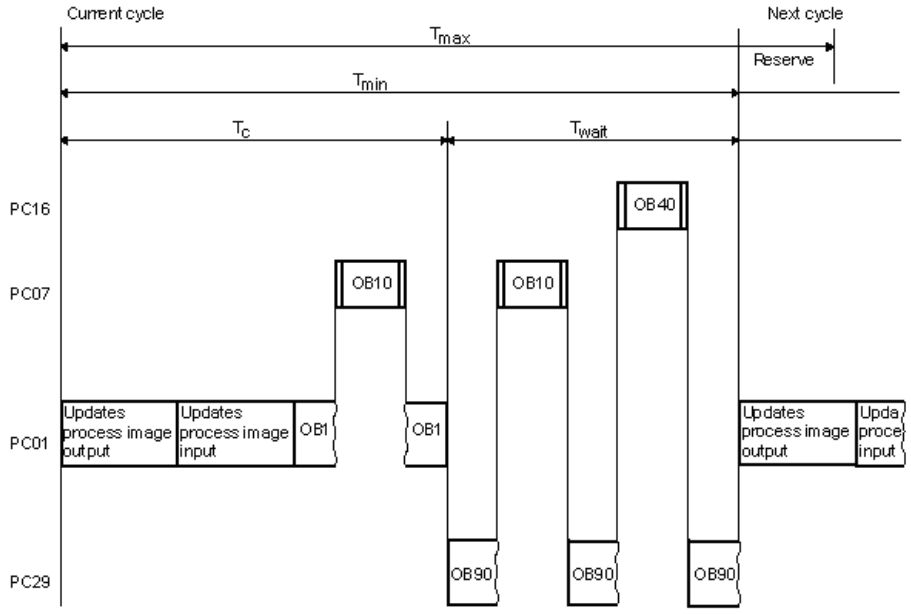
Typ přerušení	Organizační blok	Priorita	Popis
Hardwarové přerušení (hardware interrupts)	OB40 OB41 OB42 OB43 OB44 OB45 OB46 OB47	16 17 18 19 20 21 22 23	Organizační bloky pro hardwarové přerušení
DPV1 přerušení (DPV1 interrupts	OB55 OB56 OB57	2 2 2	Organizační bloky pro zařízení podporující DPV1
Multicomputing interrupt	OB60	25	Organizační blok pro synchronizaci operací na několika CPU
Synchronní cyklické přerušení (Synchronous cycle interrrupt)	OB61 OB62 OB63 OB64	25	Organizační bloky využívající reakční doby na PROFIBUS-DP
Redundantní chyby (redundancy errors)	OB70 I/O redundancy Error (pouze v H systémech) OB72 CPU Redundancy Error (pouze v H systémech)	25 28	Organizační bloky pro detekci chyb (OB70-OB87 /OB121-OB122)

Typ přerušení	Organizační blok	Priorita	Popis
Asynchronní chyby (Asynchronnous errors)	OB80 Časová chyba (Time Error) OB81 Chyba napájení (Power Supply Error) OB82 Diagnostické přerušení (Diagnostic Interrupt) OB83 Vložení/vyjmutí modulu (Insert/Remove Module Interrupt) OB84 Hardwarová chyba CPU (CPU Hardware Fault) OB85 Zacyklení programu (Program Cycle Error) OB86 Rack Failure OB87 Chyba komunikace (Communication Error)	25	Organizační bloky pro detekci chyb (OB70-OB87 /OB121-OB122)
Cyklus který běží na pozadí (Background Cycle)	OB90	29	Organizační blok OB90
Spuštění (Startup)	OB100 Restart OB101 teplý restart (Hot Restart) OB102 studený restart (Cold Restart)	27 27 27	Organizační bloky pro detekci chyb (OB70-OB87 /OB121-OB122)
Synchronní chyby	OB121 Programová chyba (Programming Error) OB122 Chyba přístupu (Access Error)		Organizační bloky pro detekci chyb (OB70-OB87 /OB121-OB122)



Obr. 3.3: Organizační blok OB1.

Organizační bloku pro vykonávání programu na pozadí – pokud specifikujeme minimální dobu cyklu ve STEP7 a ta je delší než aktuální doba cyklu, má CPU na konci cyklického programu volný čas. Tento čas se používá pro provádění organizačního bloku na pozadí – OB90. Pokud OB90 neexistuje, CPU během zbylého času pouze čeká. OB90 má úroveň priority 29, což odpovídá 0.29. Je to tedy nejnižší priorita. OB90 může být jakkoliv dlouhý, jeho prováděcí čas není CPU kontrolován.



Tmax = maximální doba cyklu, která může být nastavena
Tmin = minimální doba cyklu, která může být nastavena
Tc = aktuální doba cyklu
Twait = rozdíl mezi Tmin a aktuální dobou cyklu. V tomto čase může nastat také přerušení a OB 90 může probíhat dále
PC = priorita (Priority class)

Obr. 3.4: Organizační blok OB 90.

Organizační bloky pro přerušením spouštěné programy – STEP 7 nabízí několik různých organizačních bloků, které mohou přerušit OB1 v určitém intervalu nebo na základě nějaké podmínky. Tyto bloky lze konfigurovat buď pomocí STEP7 nebo prostřednictvím systémových funkcí (SFC). STEP7 nabízí následující typy vykonávání programu:

- Časově spouštěné vykonávání programu.
- Spouštění programu hardwarovým přerušením.
- Spouštění programu diagnostickým přerušením.
- Sspouštění programu pomocí přerušení multicomputingu.
- Obsluha chyb.

Tab. 3.2: Časově volané organizační bloky.

Typ přerušení	**Organizační bloky přerušení**	**Příklad**
Denní časové přerušení (Time-of-day interrupt)	OB10 to OB17	Výpočet průtoku na konci míchání surovin a jejich přesun
Zpožděné přerušení (Time-delay interrupt)	OB20 to OB23	Řízení otáček ventilátoru na motoru, které musí běžet ještě 20s po vypnutí motoru
Cyklické přerušení (Cyclic interrupt)	OB30 to OB38	Pravidelné vzorkování vstupů pro regulaci
Hardwarové přerušení (Hardware interrupt)	OB40 to OB47	Signalizace dosažení minimální hladiny v nádrži

Pomocí SFC funkcí lze maskovat, odložit nebo zakázat startovací událost pro některý OB.

Provádění programu po přerušení – když operační systém detekuje startovací událost organizačního bloku s vyšší prioritou, provádění programu je přerušeno po právě aktivní instrukci. Operační systém uloží data přerušeného bloku, která budou nutná při návratu k provádění daného bloku.

Aby operační systém prováděl přerušovací OB, musí být naprogramován.

Organizační bloky obsluhu chyb – lze je rozdělit na dvě skupiny:

- Synchronní chyby - tj. chyby v určité specifické části uživatelského programu. Chyba nastane při vykonávání určité části programu. Pokud není naprogramován příslušný OB, program se zastaví.
- Asynchronní chyby – tyto chyby nejsou přímo svázány s uživatelským programem. Jsou to chyby priorit nebo PLC (chybný modul). Pokud není naprogramován příslušný OB, program se zastaví (kromě OB81).

3.1.2 Funkční bloky- FB

Jsou to bloky „s pamětí“. Uživatel si je programuje sám. Mají přiřazen tzv. instanční datový blok. Parametry, které jsou přenášeny do FB a statické proměnné jsou v tomto bloku ukládány. Dočasné proměnné jsou ukládány v datovém zásobníku. Data v instančním DB se neztratí, jakmile je provádění FB ukončeno.

FB obsahuje program, který se provede vždy, když je FB zavolán z jiného bloku. Funkční bloky velice usnadňují programování opakujících se a komplexních algoritmů.

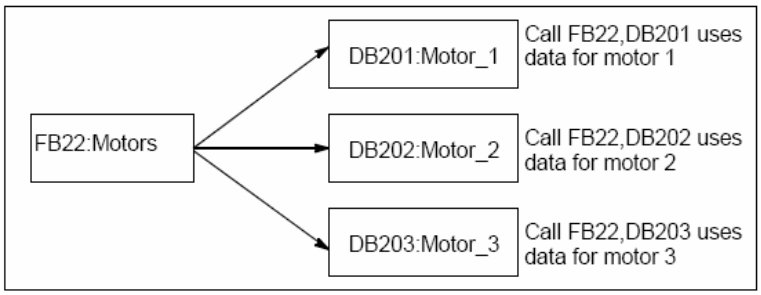
Instanční DB je přiřazen FB při každém volání, při kterém jsou přenášeny parametry. Voláním více než jedné instance funkčního bloku můžeme řídit více zařízení pomocí jednoho bloku. Např. funkční blok pro motor, může řídit několik motorů při použití rozdílného nastavení parametrů pro každý motor.

Pokud z FB voláme další FB, pak lze uvést tento volaný FB v deklaraci proměnných jako statickou proměnnou typu FB. To umožní vnořit proměnné a koncentrovat je v jednom instančním DB.

Formálním parametrům lze při deklaraci FB přiřadit počáteční hodnoty. Tyto hodnoty jsou zapsány do instančního DB, který je FB přiřazen.

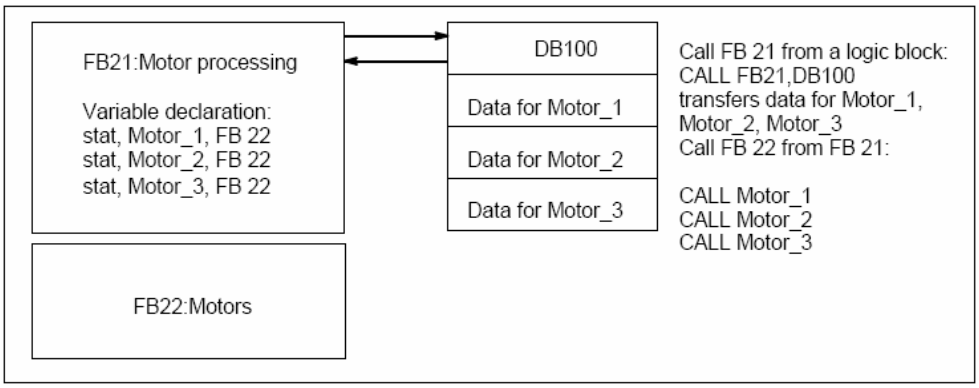
3.1.3 Datové bloky

Instanční DB



Obr. 3.5: Využití instančních DB.

Instanční DB je přiřazen každému volání FB, které přenáší parametry. Jsou v něm uloženy aktuální parametry a statická data. Parametry deklarované v FB určují strukturu DB. Instance FB znamená volání funkčního bloku. Pokud např. FB voláme 5x, má pět instancí. Pokud přiřadíme několik instančních DB funkčnímu bloku, který řídí motor, můžeme řídit několik motorů.

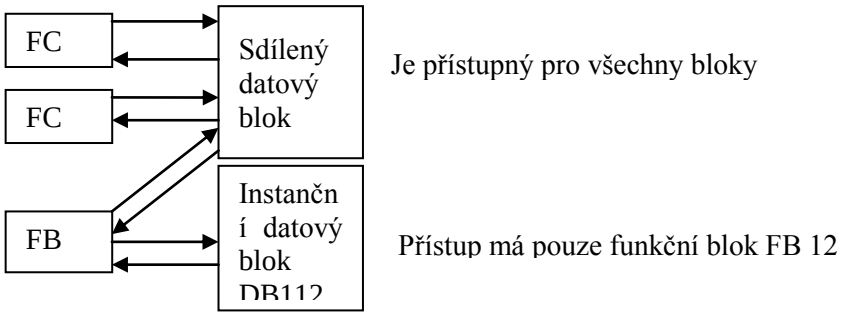


Obr. 3.6: Využití jednoho instančního DB pro více FB.

Je rovněž možné přenést instanční data pro několik motorů do jednoho instančního DB. V tomto případě je však nutné daný FB volat z jiného funkčního bloku a deklarovat FB jako formální parametr.

Sdílené DB

Datové bloky se používají k ukládání dat. Sdílené DB obsahují data, které jsou přístupné ze všech programových bloků. Sdílené DB lze strukturovat podle vlastních požadavků.



Obr. 3.7: Příklad použití sdílených DB.

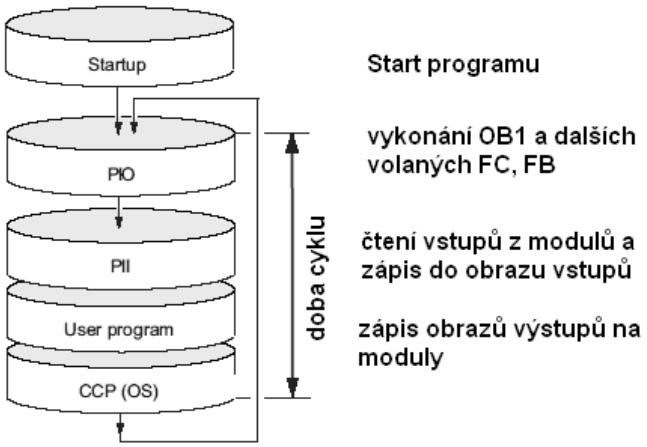
Pokud je volán logický blok (FB, FC, OB), může dočasně využívat místo v lokální datové oblasti (L stack). Kromě toho může využívat datový prostor ve sdíleném DB. V DB, narozdíl od L, nejsou data po provedení programu mazána. Sdílený a instanční DB může být otevřen současně.

Uložení dat při přerušení bloku

Pokud je provádění bloku přerušeno (voláním jiného bloku nebo blokem s vyšší prioritou), uloží se informace přerušeného bloku do zásobníku „block stack“. Ukládá se tam číslo bloku, typ, adresa návratu a čísla datových bloků, které byly v době přerušení otevřeny.

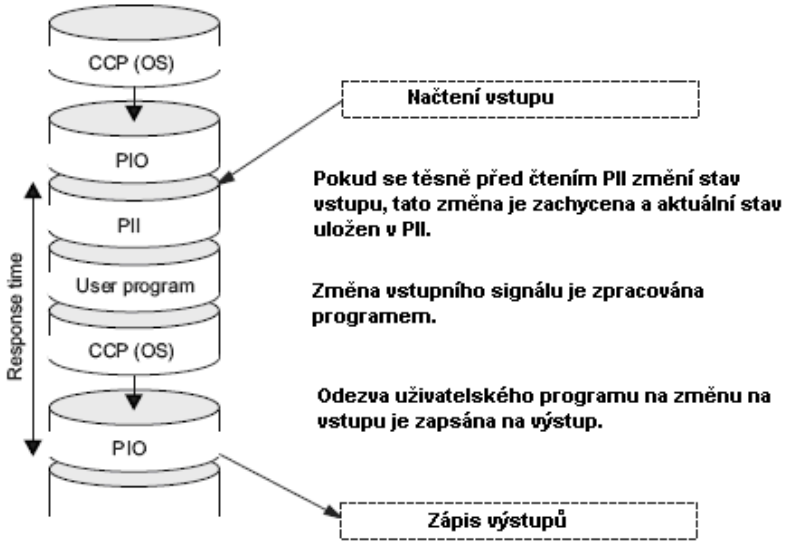
3.2 Cyklus provádění programu

Doba cyklu (cycle time) – čas jednoho programového cyklu + všechny programové sekce, které cyklický program přeruší. Tato doba závisí na – rychlosti CPU, době přenosu PII/PIQ, délce programu, na S7 časovačích, komunikacích přes MPI a PROFIBUS, na obsluze přerušení.

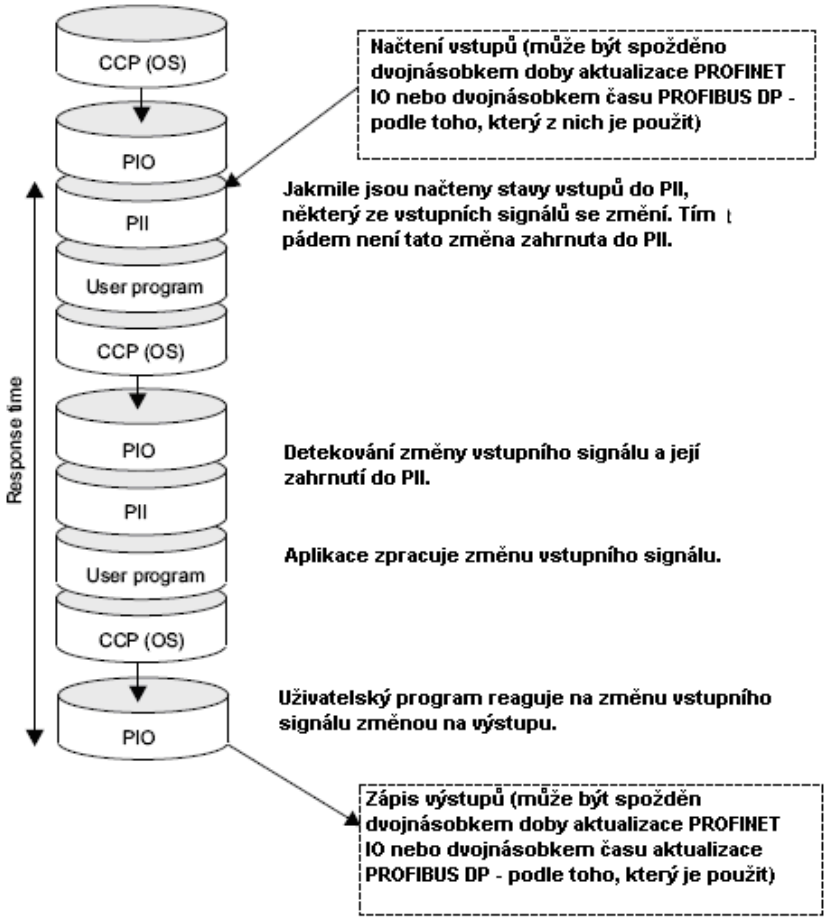


Obr. 3.8: Operační cyklus.

Doba odezvy (response time) – čas mezi detekcí vstupního signálu a modifikací příslušného výstupu. Závisí na zpoždění I/O, zpoždění zavedené PROFIBUSEM.



Obr. 3.9: Minimální doba odezvy.



Obr. 3.10: Maximální doba odezvy.

Doba činnosti operačního systému PLC – je to doba nutná pro zpracování systémových činností.

Tab. 3.3: Doby činnosti operačního systému.

CPU	Doba cyklu do vykonání kontroly cyklus (cycle check point (CCP))
312C	500 µs
314C-2	500 µs
315	500 µs
319	77 µs

Provedení uživatelského programu – je dán součtem prováděcích dob instrukcí vynásobený určitým faktorem:

Tab. 3.4: Násobící faktor.

CPU	**CPU 312C**	**CPU 314C-2DP**	**CPU 315**	**CPU 319**
Factor	1,06	1,1	1,1	1,05

Aktualizace časovačů – časovače se aktualizují každých 10ms. Každá aktualizace zabere asi 10µs.

Profibus rozhraní zvýší dobu cyklu typicky o 5%.

Integrované funkce zvyšují dobu cyklu o max. 10%.

Zvýšení doby cyklu přerušovacími rutinami – odskok do přerušovací rutiny prodlouží dobu cyklu o určitou dobu (300-800µs) plus vlastní prováděcí čas instrukcí v přerušovací rutině.

Doba odezvy na přerušovací signál – doba od prvního detekování přerušovacího signálu do provedení první instrukce přerušovací rutiny. Skládá se z času odezvy CPU, času odezvy signálového modulu a runtime Profibus sběrnice. Pohybuje se od 0,3µs do 1,5µs.

3.3 Technologické signály

Program běžící v programovatelném automatu je připojen k řízenému systému pomocí technologických signálů. Pomocí těchto signálů je automatu schopen ovlivňovat činnost tohoto systému. Technologické signály mají vzhledem k automatu buď vstupní nebo výstupní charakter. Vstupní signály automatu jsou takové, které přicházejí zvenčí a poskytují automatu určitou informaci o stavu okolního procesu. Mají svůj zdroj ve snímačích a čidlech, tlačítcích, klávesnicích apod. Může se jednat například o údaj o teplotě, o poloze nějakého zařízení, o stisku tlačítka. Výstupní signály automatu vycházejí z automatu a dávají informaci okolnímu procesu. Slouží pro ovládání jednotlivých akčních členů, pohonů, pro rozsvěcování indikačních světel apod.

Vstupní a výstupní signály jsou přivedeny k vstupním a výstupním modulům programovatelného automatu. Jejich hodnoty jsou zpracovávány uživatelským programem.

Digitální vstupy

Digitální (binární) vstupy jsou určeny pro připojení dvouhodnotových veličin k programovatelnému automatu. Jejich zdrojem jsou snímače, čidla, tlačítka apod., které dávají informaci „ano/ne“, „pravda/nepravda“ o sledované části procesu.

Nejčastěji používané fyzikální rozsahy digitálních vstupů jsou:

- 0/24V DC – snad nejpoužívanější rozsah digitálních vstupů.
- 0/5V DC – rozsah odpovídající TTL logice, v průmyslu se používá zřídka.
- 230V AC – rozsah odpovídající běžné střídavé silové síti, při použití vstupních modulů pro tento rozsah je výhodné to, že signály není nutné převádět např. na 0/24V DC.

Digitální výstupy

Digitální (binární) výstupy slouží pro připojení akčních členů s dvouhodnotovým signálem k programovatelnému automatu. Jedná se například o různé indikátory, jednoduché pohony, stykače, relé apod.

Nejčastěji používané fyzikální rozsahy digitálních výstupů jsou:

- 0/24V DC – snad nejpoužívanější rozsah digitálních vstupů.
- 0/5V DC – rozsah odpovídající TTL logice, v průmyslu se používá zřídka.
- 230V AC – rozsah odpovídající běžné střídavé silové síti, takže je možné napájet odběrově nenáročná zařízení přímo z digitálního výstupu (např. cívku stykače).

Analogové vstupy

Analogové vstupy slouží pro připojení analogových (spojitých) veličin k programovatelnému automatu. Jedná se hlavně o signály ze snímačů a čidel, které poskytují spojitý signál. Bývají to hlavně snímače fyzikálních veličin jako jsou teploměry, tlakoměry apod. Analogové vstupní signály mají většinou charakter napěťový nebo proudový, ale jsou i rozsahy, kdy se měří odpor, nebo „přímo“ teplota. Obecně platí, že proudové signály jsou odolnější proti rušení, jejich použití je výhodné zvláště u průmyslových aplikací a tam, kde je vyšší možnost případného rušení.

Nejčastěji používané fyzikální rozsahy analogových vstupů jsou:

- 0/10V – běžně používaný napěťový rozsah.
- +10/-10V – obdoba předchozího, ale spodní mez je posunuta do záporné polarity.
- 1/5V.
- 0/20mA – běžně používaný proudový rozsah.
- +20/-20mA – obdoba předchozího, ale spodní mez je posunuta do záporné polarity.
- 4/20mA – proudový rozsah umožňující detekovat přerušení signálového vodiče.
- Pt100 – rozsah pro přímé připojení odporového teploměru Pt100.

Analogové výstupy

Analogové výstupy slouží pro připojení akčních členů s analogovými signály k programovatelnému automatu. Jedná se například o servopohony, frekvenční měniče, ventily apod. Analogové výstupní signály mají charakter napěťový nebo proudový.

- 0/10V – běžně používaný napěťový rozsah.
- +10/-10V – obdoba předchozího, ale spodní mez je posunuta do záporné polarity.
- 0/20mA – běžně používaný proudový rozsah.
- +20/-20mA – obdoba předchozího, ale spodní mez je posunuta do záporné polarity.
- 4/20mA – proudový rozsah umožňující detekovat přerušení signálového vodiče.

Přerušovací vstupy

Přerušovací vstupy jsou digitální vstupy, které umožňují přerušení probíhajícího programu po příchodu náběžní případně sestupné hrany. Program je přerušen a je provedena obslužná rutina, která umožní okamžitou reakci na vzniklou událost. Přerušovací vstupy se používají tehdy, pokud chceme snímat digitální signál, který se může měnit rychleji něž je doba cyklu provádění programu. Připojení takového signálu jsme schopni zajistit jeho kvalitní snímání bez ztráty informací. Na přerušovací vstupy se připojují obvykle signály z inkrementálních čidel, světelných závor apod.

Fyzikální rozsahy přerušovacích vstupů jsou podobné jsou u digitálních vstupů (230V AC se zde nepoužívá).

Čítačové vstupy

Čítačové vstupy slouží pro připojení impulsních digitálních signálů k programovatelnému automatu. Jedná se o signály, jejichž frekvence (perioda) je větší, než kterou by bylo možno snímat běžným digitálním nebo přerušovacím vstupem (obvykle 20, 40, 60kHz). Čítačové vstupy jsou většinou podporovány speciální instrukcí nebo funkcí. Používají se hlavně pro připojení signálu od inkrementálních snímačů.

Speciální digitální výstupy

U některých programovatelných automatů nebo rozšiřujících modulů existují speciální digitální výstupy, které umožňují realizaci například PWM modulace (pulzně šířkové modulace) apod. V podstatě se jedná o možnost přesného řízení změny úrovně na digitálním výstupu. Tyto výstupy se využívají například u řízení pohonů, u smyčkového řízení apod.

3.4 Logické instrukce

Logické instrukce se používají pro práci s binárními veličinami. Jedná se o zpracování digitálních vstupů a výstupů a o práci s paměťovými binárními hodnotami. Pomocí logických instrukcí lze realizovat kombinační i sekvenční logické funkce [3-7].

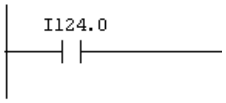
Mezi základní logické instrukce patří:

- Kontakty.
- Výstup.
- Set.
- Reset.
- Testy hran.

Kontakty

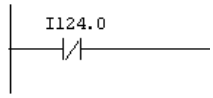
Kontakty slouží pro načtení hodnoty binární proměnné (vstupu nebo paměťové buňky). S jejich pomocí lze vytvořit v podstatě libovolnou logickou funkci. Existují dva základní typy kontaktů:

- Spínací kontakt (normally open contact) – je-li na testované buňce logická jednička, kontakt je sepnut.



Obr. 3.11: Spínací (NO) kontakt.

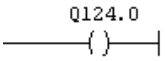
- Rozpínací kontakt (normally closed contact) – je-li na testované buňce logická nula, kontakt je sepnut.



Obr. 3.12: Rozpínací (NC) kontakt.

Výstup

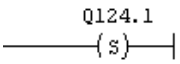
Instrukce přirazení na výstup provede zapsání výsledku logické operace na uvedenou výstupní buňku (digitální výstup nebo paměťovou buňku).



Obr. 3.13: Přiřazení na výstup.

Instrukce SET

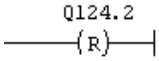
Instrukce SET nastaví příslušnou paměťovou buňku nebo výstup na logickou jedničku, pokud byl výsledek logické operace roven logické jedničce. Při výsledku rovném logické nule instrukce neprovede nic.



Obr. 3.14: Instrukce SET.

Instrukce RESET

Instrukce RESET nastaví příslušnou paměťovou buňku nebo výstup na logickou nulu, pokud byl výsledek logické operace roven logické jedničce. Při výsledku rovném logické nule instrukce neprovede nic.

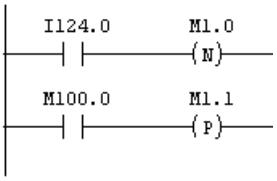


Obr. 3.15: Instrukce RESET.

Detekce náběžné (sestupné) hrany

Instrukce pro detekci hran se používají tehdy, pokud chceme detekovat změnu stavu proměnné z logické nuly na logickou jedničku nebo naopak. Instrukce nastaví výsledek logické operace na jedničku po dobu jednoho programového cyklu po vzniku příslušné události.

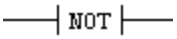
Instrukce vyžaduje zadání pomocné paměťové buňky (na obrázku M1.0 nebo M1.1), do které si uloží mezivýsledek nutný pro vyhodnocení změny stavu. Uvedené buňky by pak již neměly být v programu dále použity, aby nebylo ovlivněno provádění uvedené instrukce.



Obr. 3.16: Detekce náběžné a sestupné hrany.

Instrukce NOT

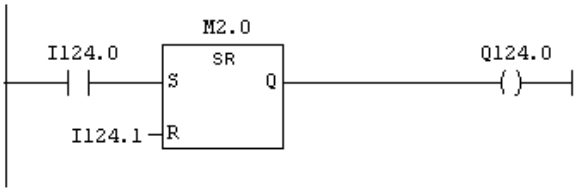
Instrukce NOT neguje výsledek logické operace.



Obr. 3.17: Instrukce negace.

Klopné obvody

Paměťové buňky a výstupu programovatelného automatu mají charakter RS klopného obvodu. Je možné je nastavovat a resetovat pomocí instrukcí SET a RESET. Využití klopného obvodu je vidět na následujících obrázcích:



Obr. 3.18: Použití úplné instrukce RS klopného obvodu.

3.5 Logické funkce

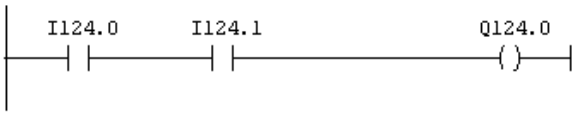
Logický součet (OR)

Logický součet je jednou ze základních logických funkcí. Popisuje situaci buď a nebo, tzn. výsledkem součtu je logická jednička, pokud alespoň jedna ze sčítaných veličin má hodnotu jedna. Je reprezentován paralelním spojením kontaktů.

Obr. 3.19: Logický součet.

Logický součin (AND)

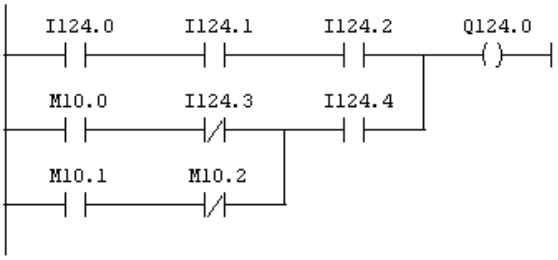
Logický součin je rovněž jednou ze základních logických funkcí. Popisuje situaci A a zároveň B, tzn. výsledkem součinu je logická jednička, pokud všechny veličiny součinu mají hodnotu jedna. Logický součin je reprezentován sériovým spojením kontaktů.



Obr. 3.20: Logický součin

Kombinovaná funkce

Na následujících obrázcích je příklad kombinované funkce, která zahrnuje jak logický součet, tak součin.



Obr. 3.21: Kombinovaná funkce.

3.6 Časovače

Časovače jsou instrukce, které slouží k práci s časovými intervaly. S jejích pomocí lze generovat časový interval, který je možné potom využít v dalším programu.

Každý časovač má v paměti PLC vyhrazenu 16-ti bitovou oblast – slovo časovače, která obsahuje při běhu časovače aktuální načítaný čas. Hodnota v tomto slově je za běhu časovače dekrementována z hodnoty dané časovou konstantou při startu časovače k nule. Rychlost dekrementace je dána časovou základnou. V programu je možné použít 256 časovačů.

Slovo časovače obsahuje hodnotu času v binárním tvaru uloženou na bitech 0-9. Časová hodnota udává počet jednotek (počet časových intervalů daných časovou základnou). Hodnotu časovače je možné přednastavit dvěma způsoby:

- Pomocí zápisu slova W#16#wxyz (kde w je časová základna a xyz je hodnota času)
- Pomocí zápisu časové konstanty ve tvaru S5T#aH_bM_cS_dMS (kde a,b,c,d jsou hodnoty definované uživatelem, H označuje hodiny, M minuty, S sekundy, MS milisekundy). V tomto případě se časová základna zvolí automaticky.

Maximální nastavitelný čas je 9,990 sekund nebo 2H_46M_30S.

Časová základna – bity 12 a 13 časovačového slova jsou vyhrazeny pro časovou základnu.

Příklady časových konstant: S5TIME#4S = 4 seconds, s5t#2h_15m = 2 hours and 15 minutes, S5T#1H_12M_18S = 1 hour, 12 minutes, and 18 seconds.

Tab. 3.5: Časové základny časovačů.

Časová základna	Binární vyjádření pro časovou základnu
10ms	00
100ms	01
1s	10
10s	11

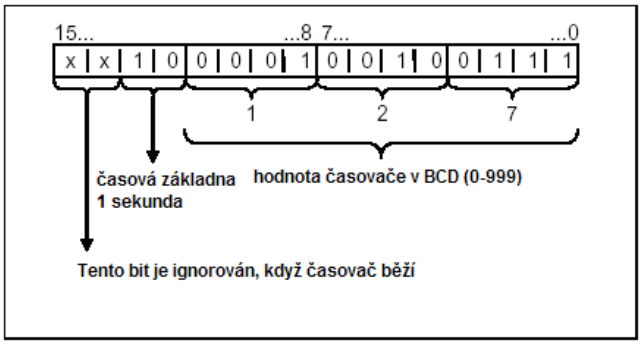
Hodnota času větší než 2H_46M_30S není časovačem akceptována. Hodnota, která překračuje možnosti přesnosti (jako např. 2h10ms) je zaokrouhlena.

Tab. 3.6 : Hranice nastavitelných hodnot pro jednotlivé časové základny.

Rozlišení	Rozsah
0,01 s	10ms-9s_990ms
0,1s	100ms-1M_39s_900ms
1s	1s-16m_39s
10s	10s-2h_46m_30s

Formát konstanty časovače

Konstanta časovače obsahuje jednak hodnotu, která specifikuje počet časovaných jednotek a také hodnotu časové základny (viz. následující obrázek).



Obr. 3.22: Formát konstanty časovače.

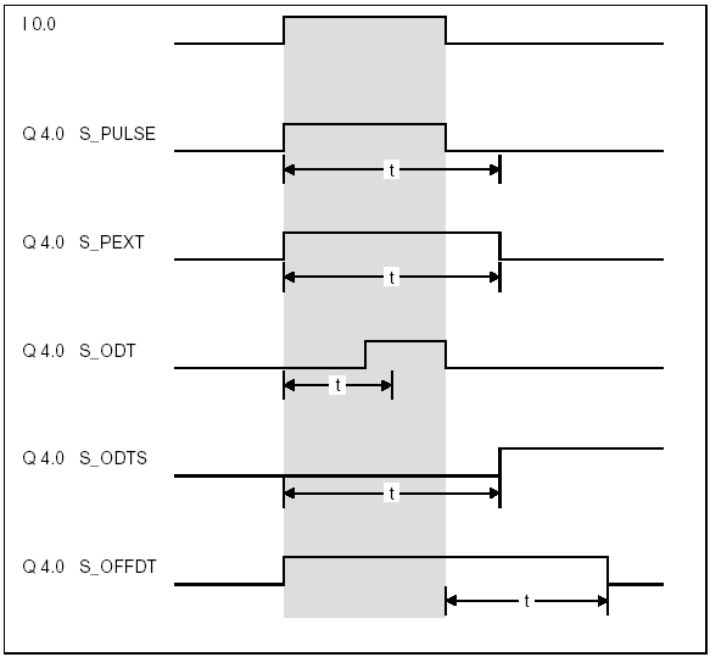
Každá instrukce časovače poskytuje dva výstupy - binární a BCD.

Typy časovačů

- S_PULSE – impulzní časovač (Pulse Timer) – maximální délka výstupního impulsu je stejná jako nastavená časová konstanta. Pokud během nastavené doby spadne vstupní signál na nulu, je nulován i výstup.
- S_PEXT – prodloužený impuls (Extended Pulse Timer) – výstupní signál má délku danou časovou konstantou bez ohledu na délku vstupního impulsu.

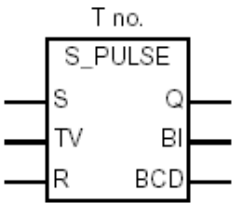
- S_ODT – zpožděné zapnutí (On-Delay Timer) – výstup se nastaví na log. 1 až po době dané časovou konstantou a zůstane nastaven tak dlouho, dokud je vstupní signál v log. 1.
- S_ODTS – zpožděné zapnutí s pamětí (Retentive On-Delay Timer) – výstupní signál se nastaví na log. 1 až po době dané časovou konstantou a zůstane nastaven bez ohledu na stav vstupu.
- S_OFFDT – zpožděné vypnutí (Off-Delay Timer) – výstupní signál se nastaví na log. 1 když je vstupní signál 1 nebo když časovač běží. Čas je odstartován, když se vstupní signál změní z 1 do 0.

Výběr vhodného časovače



Obr. 3.23: Časové průběhy jednotlivých časovačů.

Časovač SP – impulz



Obr. 3.24: Instrukce časovače SP.

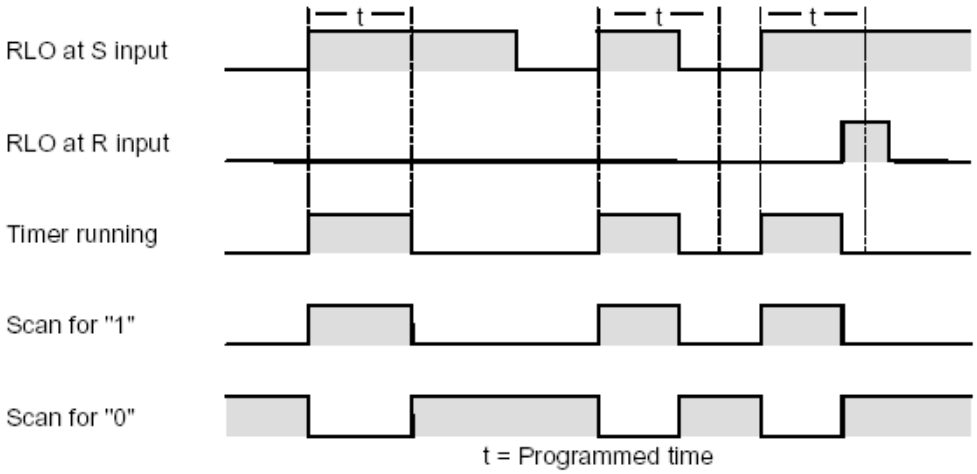
Tab. 3.7: Popis jednotlivých parametrů časovače.

Parametr	**Datový typ**	**Oblast paměti**	**Popis**
T no.	TIMER	T	Identifikační číslo časovače (rozsah hodnot závisí na typu CPU)
S	BOOL	I,Q,M,L,D	Startovací vstup
TV	S5TIME	I,Q,M,L,D	Přednastavený čas

Parametr	Datový typ	Oblast paměti	Popis
R	BOOL	I,Q,M,L,D	Resetovaní vstup
BI	WORD	I,Q,M,L,D	Hodnota časovače v binární formátu
BCD	WORD	I,Q,M,L,D	Hodnota časovače v BCD formátu
Q	BOOL	I,Q,M,L,D	Stav časovače

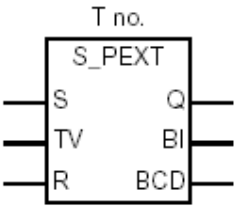
Na výstupu tohoto časovače je generován impuls, který má délku danou zadanou časovou konstantou. Pokud je vstupní signál kratší než zadaný čas, je výstup rovněž nulován.

Vlastnosti časovače:



Obr. 3.25: Časový průběh signálu časovače SP.

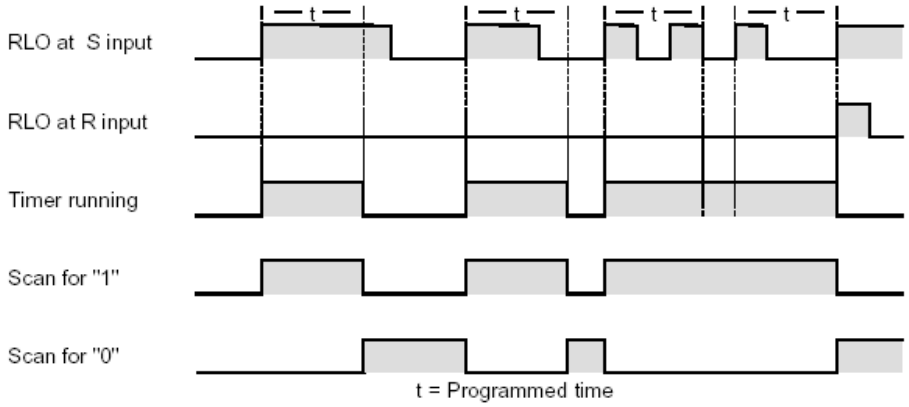
Časovač SE – prodloužený impuls



Obr. 3.26: Instrukce časovače SE .

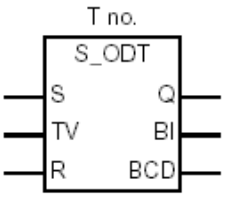
Popis parametrů je stejný jako u časovače SP.

Po přivedení vstupního signálu generuje časovač na výstupu impulz o délce zadané časové konstanty, bez ohledu na délce vstupního signálu.



Obr. 3.27: Časový průběh signálu časovače SE.

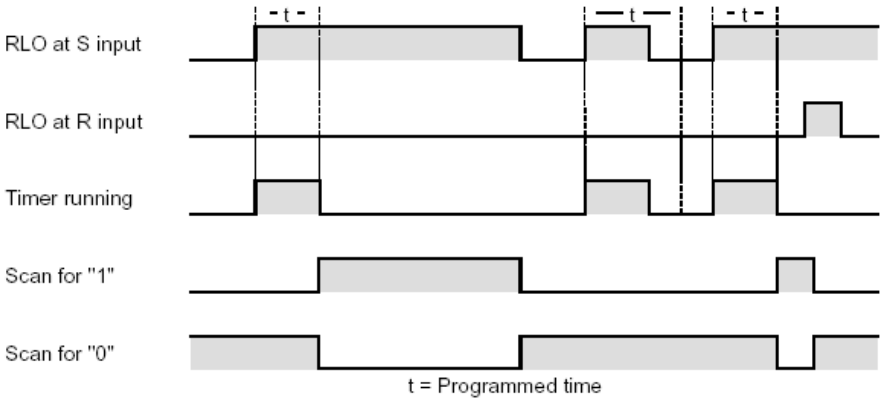
Časovač SD - zpožděné zapnutí



Obr. 3.28: Instrukce časovače SD.

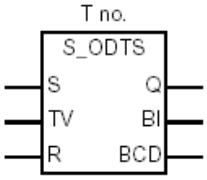
Popis parametrů je stejný jako u časovače SP.

Je-li na vstup časovače přivedena log. 1, výstup je po zadaném čase nastaven na log.1. Pokud se výstup změní zpět na log. 0, je nulován i výstup.



Obr. 3.29: Časový průběh signálu časovače SD.

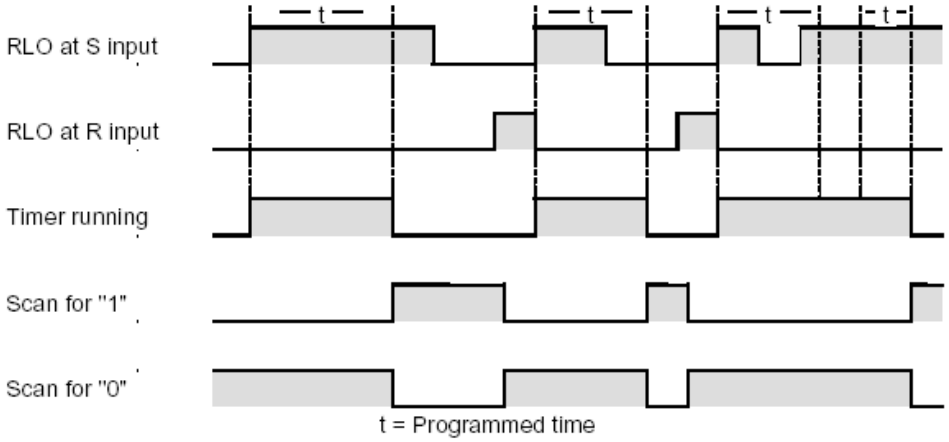
Časovač SS - zpožděné zapnutí s pamětí



Obr. 3.30: Instrukce časovače SS.

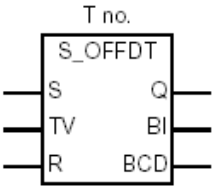
Popis parametrů je stejný jako u časovače SP.

Časovač nastartuje časování, pokud se na jeho vstupu objeví náběžná hrana. Po uplynutí zadaného času se výstup časovače nastaví log. 1. Resetován může být pouze signálem RESET. Časovač může být restartován náběžnou hranou na vstupu, zatímco časovač běží.



Obr. 3.31: Časový průběh signálu časovače SS.

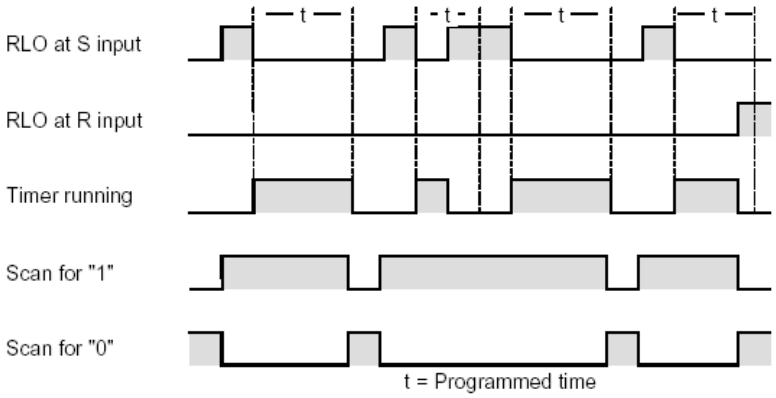
Časovač SF - zpožděné vypnutí



Obr. 3.32: Instrukce časovače SF.

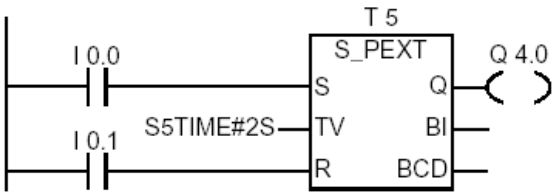
Popis parametrů je stejný jako u časovače SP.

Výstupní signál se nastaví na log. 1 když je vstupní signál 1 nebo když časovač běží. Časovač je startován negativní hranou na vstupu. Po uplynutí zadané doby od negativní hrany se výstup vynuluje.



Obr. 3.33: Časový průběh signálu časovače SF.

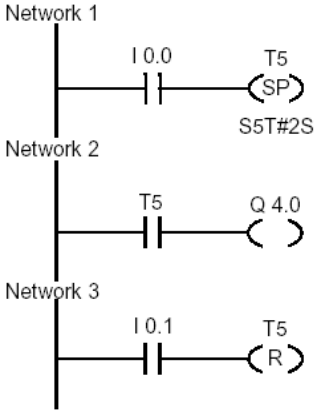
Příklad použití časovače v LAD:



Obr. 3.34: Příklad použití časovače v LAD.

Instrukce časovače

V jazyku liniových schémat STEP 7 LAD je možné využívat úplnou nebo zkrácenou verzi instrukce časovače. Úplné verze jsou ty, které byly uvedeny na předchozích obrázcích. Zkrácená verze časovače je podobná instrukci přiřazení na logický výstup (output coil) a umožňuje zadat pouze číslo časovače a nastavit čas. Použijeme ji tehdy, pokud chceme v daném okamžiku jen odstartovat časovač a nechceme využívat další parametru, které nám nabízí úplná instrukce (reset, BI, BCD, Q). Tyto parametry je pak možné využít v dalších networcích bloku.



Obr. 3.35: Příklad použití zkrácené instrukce časovače.

Časovače podle IEC 61131-3

Všechny výše uvedené časovače se u PLC Siemens Simatic7 běžně používají. Jejich použití je zavedené už od řady S5. Ve standardizační normě IEC 61131-3 však tyto typy uvedeny nejsou a norma zavádí jako standardní tři časovače – TP, TON a TOF, pomocí kterých lze realizovat jakýkoliv typ časování. U PLC Simatic S7 300/400 tyto tři časovače neexistují jako instrukce, lze je však využít ve formě standardních funkčních bloků (TP – SFB3, TON – SFB4 a TOF –SFB5):

- TP – generování pulzu – generuje pulz zadané délky. Je to obdoba časovače SE.
- TON – Timer On Delay – zpožděné zapnutí – nastaví výstup po uplynutí zadaného času. Obdoba časovače SD.
- TOF – Timer Off Delay – zpožděné vypnutí – resetuje výstup po uplynutí zadaného času. Obdoba časovače SF.

3.7 Čítače

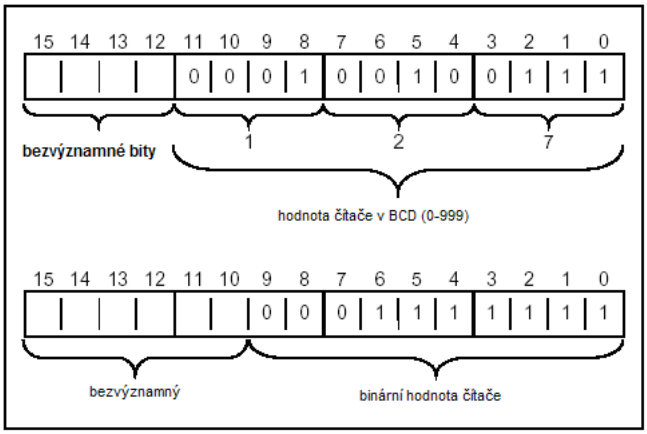
Čítače jsou instrukce, které slouží k čítání impulsů na příslušném vstupu. Každý čítač má v paměti PLC vyhrazenu 16-ti bitovou oblast – slovo čítače. V programu je možné použít 256 čítačů.

Načítaná hodnota je uložena v bitech 0-9 čítačového slova. Rozsah čítače je 0-999.

Typy čítačů

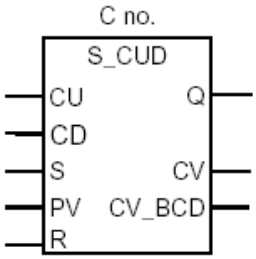
- S_CUD – čítač dolu/nahoru - Up-Down Counter.
- S_CD - čítač dolu - Down Counter.
- S_CU čítač nahoru - Up Counter.

Čítač je možné přednastavit vložením hodnoty ve tvaru C#127, která má rozsah 0-999. C# určuje, že hodnota je v BCD formátu. Hodnota čítače má následující tvar:



Obr. 3.40: Hodnota čítače.

S_CUD – čítač dolu/nahoru - Up-Down Counter



Obr. 3.41: Instrukce čítače S_CUD.

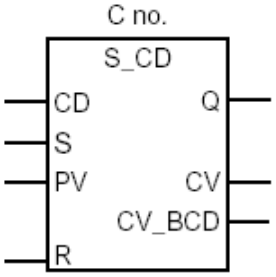
Tab. 3.8: Popis jednotlivých parametrů čítače.

Parametr	Datový typ	Oblast paměti	Popis
C no.	COUNTER	C	Identifikační číslo čítače (rozsah hodnot závisí na typu CPU)
CU	BOOL	I,Q,M,L,D	Čítací vstup - nahoru
CD	BOOL	I,Q,M,L,D	Čítací vstup – dolu
S	BOOL	I,Q,M,L,D	Vstup pro nastavení čítače

PV	S5TIME	I,Q,M,L,D	Přednastavená hodnota čítače
R	BOOL	I,Q,M,L,D	Resetovaní vstup
CV	WORD	I,Q,M,L,D	Hodnota čítače v binární formátu
CV_BCD	WORD	I,Q,M,L,D	Hodnota čítače v BCD formátu
Q	BOOL	I,Q,M,L,D	Stav čítače

Čítač je přednastaven hodnotou PV v okamžiku náběžné hrany na vstupu S. Po přivedení log. 1 na signál R, je čítač nastaven na 0. Čítač zvyšuje svoji hodnotu o 1 vždy při náběžné hraně na CU a snižuje svoji hodnotu o 1 vždy při náběžné hraně na CD.

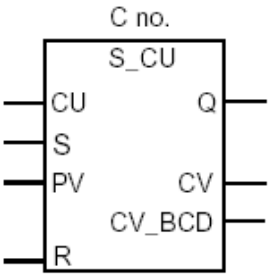
S_CD - čítač dolu - Down Counter



Obr. 3.42: Instrukce čítače S_CD.

Funkce i parametry čítače S_CD je obdobná jako u S_CUD, čítá však pouze dolů.

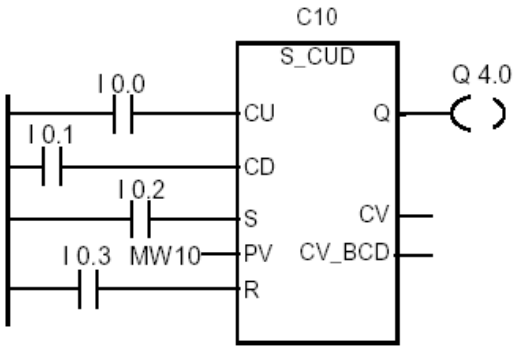
S_CU čítač nahoru - Up Counter



Obr. 3.43: Instrukce čítače S_CU.

Funkce i parametry čítače S_CU je obdobná jako u S_CUD, čítá však pouze nahoru.

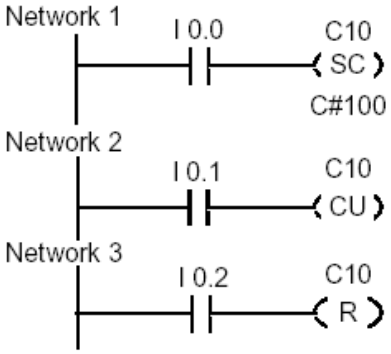
Příklad použití čítače v LAD:



Obr. 3.44: Příklad použití čítače v LAD.

Instrukce čítače

V jazyku liniových schémat STEP 7 LAD je možné využívat úplnou nebo zkrácenou verzi instrukce čítače. Úplné verze jsou ty, které byly uvedeny na předchozích obrázcích. Zkrácená verze čítače je podobná instrukci přiřazení na logický výstup (output coil) a umožňuje zadat pouze číslo čítače. Použijeme ji tehdy, pokud chceme v daném okamžiku jen inkrementovat, dekrementovat nebo nastavit čítač a nechceme využívat dalších parametrů, které nám nabízí úplná instrukce (reset, BI, BCD, Q). Tyto parametry je pak možné využít v dalších networcích bloku.



Obr. 3.24: Příklad použití zkrácené instrukce čítače v LAD.

Časovače podle IEC 61131-3

Norma IEC 61131-3 definuje své časovače – CTD – čítač dolů, CTU – čítač nahoru a CTUD – čítač nahoru/dolu. Tyto časovače nejsou zahrnuty v instrukčním souboru STEP 7, ale je možné je využít ve formě standardních funkčních bloků (podobně jako u časovačů). Čítač CTU je v SFB 0 a je velice podobný čítači S_CU ze STEP 7. Čítač CTD je v SFB 1 a je velice podobný čítači S_CU ze STEP 7. Čítač CTUD je v SFB 2 a je velice podobný čítači S_CUD ze STEP 7.

Shrnutí pojmů

Program je u programovatelného automatu Simatic S7 300 prováděn cyklicky. **Operační cyklus** obsahuje blok čtení vstupů, zpracování uživatelského programu, blok zápisu výstupů a systémový blok. V textu jsou popsány další pojmy týkající se operačního cyklu, zejména doba odezvy.

U programovatelných automatů se lze setkat s **digitálními vstupy a výstupy, analogovými vstupy a výstupy, přerušovacími a čítačovými vstupy a se speciálními digitálními výstupy**. Nejběžnějším rozsahem u digitálních signálů je 0/24V DC a u analogových 0/10V nebo 0/20mA.

Logické instrukce slouží pro realizaci logických funkcí. Existuje několik základních logických instrukcí, pomocí kterých lze realizovat většinu kombinačních i sekvenčních logických funkcí.

Časovače jsou instrukce, které se používají ke generování určitých časových intervalů v programu. Ve STEP 7 existuje **pět typů časovačů** – SP, SE, SD, SS a SF. Každý z těchto časovačů má svojí konkrétní funkci. Dále je možné v programu využívat standardní časovače, které definuje norma IEC 61131-3. Tyto časovače jsou k dispozici jako standardní funkční bloky.

Čítače slouží k čítání počtu impulsů. Ve STEP 7 existují tři typy čítačů – S_CU, S_CD a S_CUD. Zajišťují čítání nahoru, dolu a oběma směry. Podobně jako u časovačů, i zde jsou k dispozici standardní funkční bloky s čítači definovanými v normě IEC 61131-3.

Kontrolní otázky

1. Popište operační cyklus u PLC Simatic S7 300.
2. Co je to doba odezvy?
3. Jaké druhy technologických signálů se používají u programovatelných automatů?
4. Jaké jsou základní fyzikální rozsahy pro digitální vstupy?
5. Jaké jsou základní fyzikální rozsahy pro analogové vstupy?
6. Co je to Normally Open Contact?
7. Pomocí kterých instrukcí lze detekovat náběžnou a sestupnou hranu?
8. Jak se v LAD zakresluje logický součet a součin? Co jsou to časovače a k čemu se používají?
9. Jak vypadá konstanta časovače u PLC Siemens Simatic S7 300/400 ?
10. Co je to časová základna?
11. K čemu slouží časovač typu SE?
12. K čemu slouží časovač typu SD?
13. Co jsou to čítače a k čemu se používají?
14. Popište vlastnosti čítače S_CUD.

Další zdroje

3-1. Siemens: SIMATIC Automation System S7-300 Getting Started Collection, 01/2006, A5E00123662-05.

3-2. Siemens: Memory Concepts for SIMATIC S7-300 CPUs and for C7 Devices, ID:7302326.

3-3. Siemens: SIMATIC S7-300 and M7-300 Programmable Controllers Module Specifications, Edition 2, EWA 4NEB 710 6067-02 01.

3-4. Siemens: SIMATIC Programming with STEP7 V5.3, Edition 01/2004, A5E00261405-01.

3-5. Siemens: SIMATIC Working with STEP7 V5.3 Getting Started, 01/2004, A5E00261403-01.

3-6. Siemens: SIMATIC Configuring Hardware and Communication Connections STEP7 V5.3, 01/2004, A5E00261404-01.

3-7. Siemens: SIMATIC S7 Ladder Logic (LAD) for S7-300 and S7-400 Programming, 6ES7810-4CA07-8BW1.

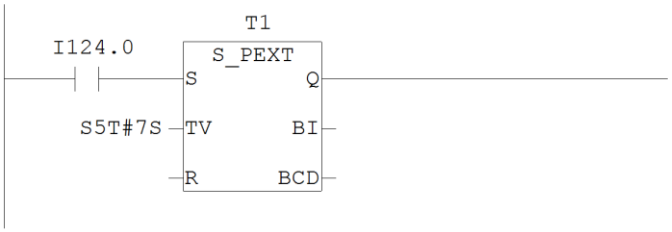
Řešená úloha 3.1

Napište program, který po přivedení impulsu na I 124.0 vytvoří na výstupu Q 124.0 impuls o délce 7s. (SE)

Řešení programu v LAD

```
Block: OB1    "Main Program Sweep (Cycle)"

Napište program, který po přivedení impulsu na I 124.0 vytvoří na výstupu Q
124.0  impuls o délce 7s. ( SE )
```

```
Network: 1

Aktivace časovače
```




```
Network: 2

impulz na výstupu Q124.0
```

Řešení programu v STL

```
Network: 1

Aktivace časovače

      A     I    124.0
      L     S5T#7S
      SE    T      1
      NOP   0
      NOP   0
      NOP   0
      NOP   0
```

```
Network: 2

impulz na výstupu Q124.0

      A     T      1
      =     Q    124.0
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 3\pr_3_1.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 3\ anim2.avi.

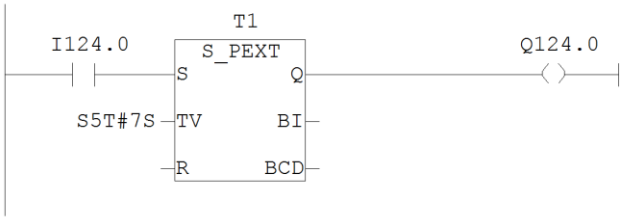
Řešená úloha 3.2.

Doplňte program tak, aby na výstupu Q 124.1 byl impuls delší o 10s než impuls na Q 124.0. (SF)

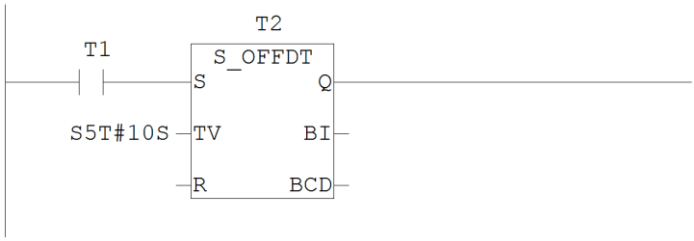
Řešení programu v LAD

```
Block: OB1    "Main Program Sweep (Cycle)"

Doplňte program tak, aby na výstupu Q 124.1 byl impuls delší o 10s než impuls
na
Q 124.0.   ( SF )
```

```
Network: 1

7s impulz na výstupu Q124.0
```




```
Network: 2

Impulz na Q124.1, ktery je o 10s delsi nez impulz na Q124.0
```




```
Network: 3
```

Řešení programu v STL

```
Network: 1

7s impulz na výstupu Q124.0

      A     I    124.0
      L     S5T#7S
      SE    T      1
      NOP   0
      NOP   0
      NOP   0
      A     T      1
      =     Q    124.0
```

```
Network: 2

Impulz na Q124.1, ktery je o 10s delsi nez impulz na Q124.0

      A     T      1
      L     S5T#10S
      SF    T      2
      NOP   0
      NOP   0
      NOP   0
      NOP   0
```

```
Network: 3

      A     T      2
      =     Q    124.1
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 3\pr_3_2.zip

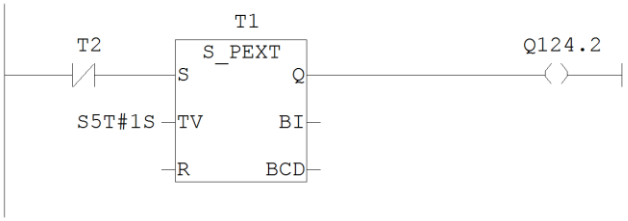
Řešená úloha 3.3.

Vytvořte program tak, aby na výstupu Q 124.2 byl impulsní signál s délkou periody 2s a střídou 1/1.

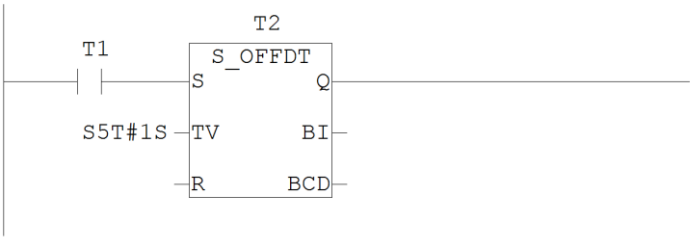
Řešení programu v LAD

```
Block: OB1    "Main Program Sweep (Cycle)"
Vytvořte program tak, aby na výstupu Q 124.2 byl impulsní signál s délkou
periody 2s a střídou 1/1
```

```
Network: 1
```




```
Network: 2
```



Řešení programu v STL

```
Block: OB1    "Main Program Sweep (Cycle)"
Vytvořte program tak, aby na výstupu Q 124.2 byl impulsní signál s délkou
periody 2s a střídou 1/1
```

```
Network: 1
```

```
AN    T      2
L     S5T#1S
SE    T      1
NOP   0
NOP   0
NOP   0
A     T      1
=     Q    124.2
```

```
Network: 2
```

```
A     T      1
L     S5T#1S
SF    T      2
NOP   0
NOP   0
NOP   0
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 3\pr_3_3.zip

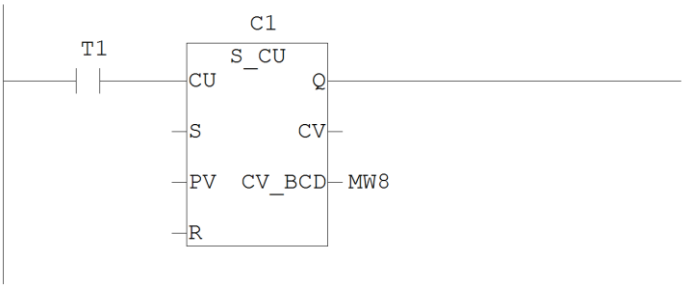
Řešená úloha 3.4.

Doplňte program 3.3 čítačem tak, aby čítač čítal impulsy z výstupního bitu Q 124.2 a jejich počet ukládal v binárním tvaru memory wordu MW8.

Řešení programu v LAD

```
Block: OB1    "Main Program Sweep (Cycle)"

Doplňte program 3.3 čítačem tak, aby čítač čítal impulsy z výstupního bitu Q
124.2 a jejich počet ukládal v binárním tvaru memory wordu MW8.
```

```
Network: 3

Ukládání počtu impulzů na výstupu Q 124.2 do MW8
```



Řešení programu v STL

```
Network: 3

Ukládání počtu impulzů na výstupu Q 124.2 do MW8
```

```
      A     T     1
      CU    C     1
      BLD   101
      NOP   0
      NOP   0
      NOP   0
      NOP   0
      LC    C     1
      T     MW    8
      NOP   0
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 3\pr_3_4.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 3\ anim3.avi.

4. PROGRAMOVATELNÉ AUTOMATY SIMATIC S7 300, ROZŠÍŘENÝ INSTRUKČNÍ SOUBOR

Čas ke studiu: 2 hodiny

Cíl

Kapitola popisuje základní instrukce jazyka STEP7, které slouží hlavně pro práci s analogovými veličinami. Jedná se o:

- Instrukce LOAD a TRANSFER, MOVE.
- Aritmetické instrukce.
- Instrukce posuvů a rotací.
- Instrukce porovnávání.
- Instrukce skoků.
- Převodní instrukce.

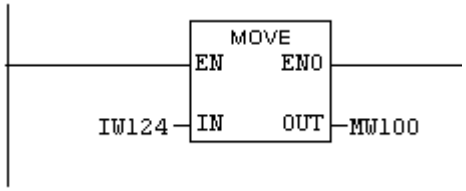
Výklad

4.1 Instrukce MOVE (resp. LOAD a TRANSFER) – instrukce přenosu

Instrukce MOVE (resp. LOAD a TRANSFER) slouží pro kopírování hodnot z jedné paměťové buňky do druhé. Během přenosu se využívá Akumulátor 1. Instrukce LOAD načte hodnotu z příslušné paměťové buňky do Akumulátoru 1. Původní obsah Akumulátoru 1 se přesune do Akumulátoru 2. Instrukce TRANSFER načte hodnotu z Akumulátoru 1 a uloží ji do příslušné paměťové buňky [4-3].

Pokud je program zapsán v jazyku LAD, je dvojice instrukcí LOAD a TRANFER nahrazena instrukcí MOVE, která však plní stejnou funkci [4-4].

LAD



STL

```
L     IW   124
T     MW   100
NOP   0
```

Obr. 4.2: Instrukce LOAD/TRANSFER resp. MOVE v programu.

Pomocí instrukcí LOAD a TRANSFER nebo MOVE lze pracovat jen s číselnými hodnotami (nikoliv binárními!). Pro práci s binárními hodnotami (bity) slouží logické instrukce.

4.2 Aritmetické instrukce

Aritmetické instrukce se používají pro zpracování číselných hodnot. Může se jednat například o úpravu hodnoty načteného analogového vstupu apod. Ve Step 7 se rozlišují instrukce pro práci v pevné a pohyblivé desetinné čárce. Mezi aritmetické instrukce patří:

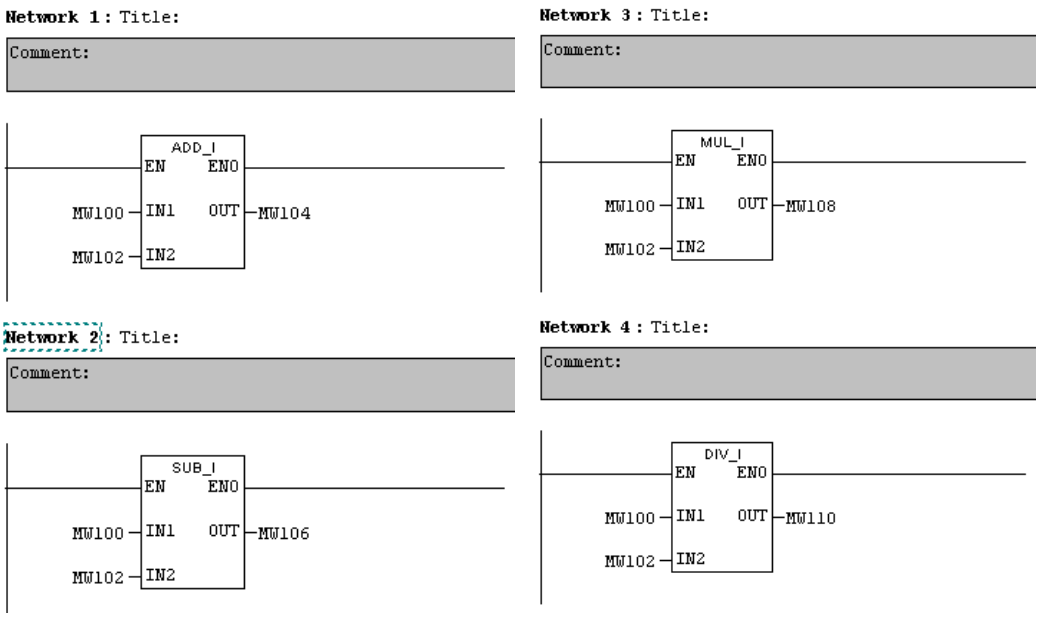
Tab. 4.1: Aritmetické instrukce.

Operace	**Pevná řádová čárka**	**Pohyblivá řádová čárka**
Sčítání	ADD_I, ADD_DI	ADD_R
Odčítání	SUB_I, SUB_DI	SUB_R
Násobení	MUL_I, MUL_DI	MUL_R
Dělení	DIV_I, DIV_DI	DIV_R
Goniometrické funkce		SIN, COS, TAN, ASIN, ACOS, ATAN
Další funkce		LN, EXP, SQR, ABS…

U programovatelných automatu obecně platí, že jejich procesory bývají optimalizovány pro logické instrukce. Proto logické operace a operace v pevné řádové čárce trvají podstatně kratší dobu, než operace v pohyblivé čárce. Výjimku tvoří hlavně automaty, které používají procesor Intel jako běžné počítače typu PC.

Příklady použití aritmetických instrukcí v programu:

LAD



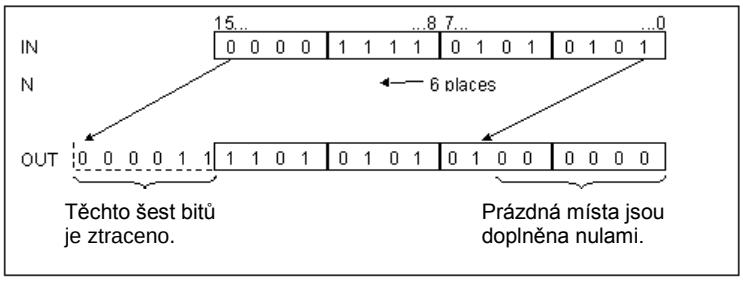
Obr. 4.3: Použití aritmetických instrukcí.

4.3 Instrukce posuvů a rotací

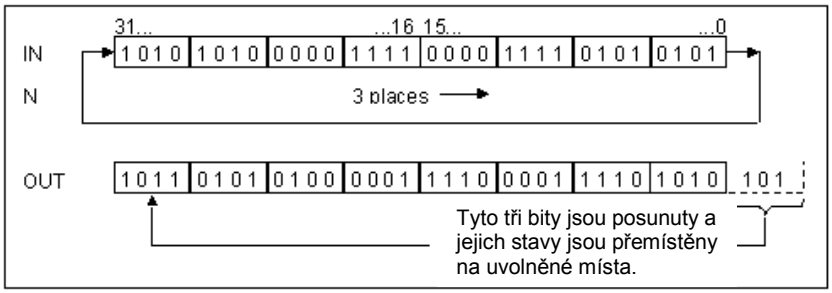
Instrukce posuvů a rotací slouží k bitovému posouvání (rotaci) obsahu paměťové buňky. Posuv je taková operace, kdy se obsah buňky posouvá na příslušnou stranu a krajní bit mizí mimo buňku. Na uvolněný bit se zapíše logická nula. Naopak rotace je taková operace, kdy krajní bit nemizí, ale přesouvá se na uvolněné místo, takže obsah buňky neustále rotuje dokola.

Instrukcí rotací a posuvů je ve STEP7 několik, ale základní jsou tyto:

- SHR_W – posun slova doprava.
- SHL_W- posun slova doleva.
- ROR_DW – rotace dvojitého slova doprava.
- ROL_DW – rotace dvojitého slova doleva.



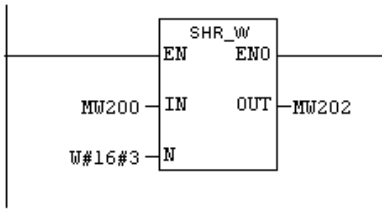
Obr. 4.4: Princip provádění instrukce SHL_W.



Obr. 4.5: Princip provádění instrukce ROR_DW.

Příklad použití instrukce posuvu SHR_W:

LAD

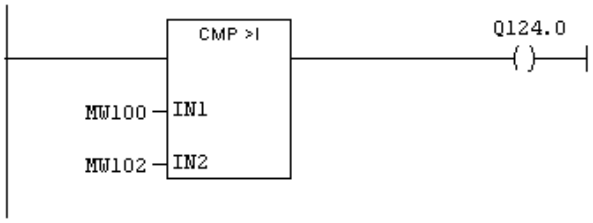


Obr. 4.6: Příklad použití instrukce SHR_W.

4.4 Instrukce porovnávání

Porovnávací instrukce slouží k porovnávání číselných hodnot. Pomocí porovnávacích funkcí lze realizovat běžné operace porovnávání – je rovno, je vetší, je menší, apod. Všechny porovnávací instrukce mají podobný tvar, liší se pouze konkrétní operací a použitým datovým typem. Je třeba si uvědomit, že výstupem všech porovnávacích instrukcí je binární hodnota (ano/ne).

LAD



Obr. 4.6: Příklad porovnávání dvou hodnot typu Integer instrukcí „je větší“.

4.5 Instrukce skoků

Instrukce skoků jsou užitečné instrukce, které umožňují určitým způsobem větvit program. Jedná se hlavně o skoky na návěští, které mohou být podmíněny různými událostmi. Mezi instrukce podobné skokům patří volání funkce nebo funkčního bloku, podmíněné ukončení bloku apod.

Skoky na návěští

- JU – nepodmíněný skok na návěští.
- JL – skok na návěští, který umožňuje naprogramovat několikanásobný skok (několik návěští). Konkrétní číslo návěští, na které instrukce skočí se bere z ACCU 1.
- JC, JNC – podmíněný skok na návěští. Skočí, když RLO = 1 (resp. RLO = 0).
- JBI, JNBI – podmíněný skok na návěští. Skočí, pokud BR = 1 (resp. BR = 0). (BR je Binary Result – stavový bit ve stavovém registru).
- JO - podmíněný skok na návěští. Skočí, pokud OV = 1 (OV je Overflow – přetečení – stavový bit ve stavovém registru).
- JOS - podmíněný skok na návěští. Skočí, pokud OS = 1 (OS je uložené přetečení – stavový bit ve stavovém registru).
- JZ, JNZ - podmíněný skok na návěští. Skočí, pokud výsledek aritmetické operace je 0 (resp. není 0).
- LOOP – realizace cyklu LOOP. Dekrementuje obsah Akumulátoru 1 a skáče na návěští, pokud je nenulový.
- Apod.

Ukončení bloku

- BEU – nepodmíněné ukončení bloku, touto instrukcí se ukončí provádění aktuálního bloku a program se vrací o úroveň výše.

- BEC - podmíněné ukončení bloku, touto instrukcí se ukončí provádění aktuálního bloku na základě výsledky logické operace a program se vrací o úroveň výše.
- CALL – slouží pro volání funkce nebo funkčního bloku.

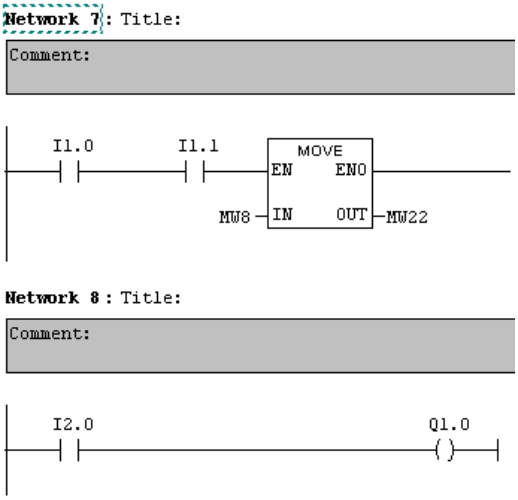
```
Network 7: Title:
Comment:

      A     I     1.0
      A     I     1.1
      JNB   _001
      L     MW     8
      T     MW    22
_001: NOP   0

Network 8 : Title:
Comment:

      A     I     2.0
      =     Q     1.0
```

Obr..4.7: Příklad použití instrukce skoku JC.

Na tomto příkladu je vidět, že pokud chceme provést podmíněný zápis do paměťové buňky pomocí instrukcí LOAD a TRANSFER, musíme použít instrukci skoku. Instrukce LOAD a TRANSFER nejsou podmíněné a proto nezávisí na výsledku předchozí logické operace!

LAD

Uvedený příklad se v LAD realizuje poněkud jinak, než by se dalo předpokládat. Řada instrukcí skoků v LAD chybí, protože se dají realizovat jednoduše pomocí liniového schématu. Instrukce LOAD a TRANSFER jsou zde nahrazeny pomocí instrukce MOVE, které se dá volat podmíněně v rámci určitého liniového schématu.



Obr. 4.8: Podmíněný zápis pomocí instrukce MOVE.

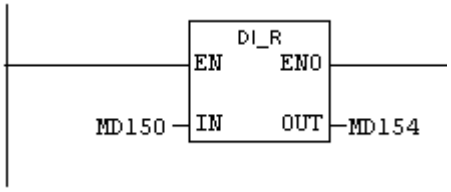
4.6 Převodní instrukce

Převodní instrukce převádí hodnoty z jednoho datového typu do druhého nebo převádějí hodnoty uvedené v běžných datových typech do nějakého speciálního kódu. Všechny převodní instrukce mají podobný tvar. Ve Step7 existují následující základní převodní instrukce:

- I_DINT - Integer to Double Integer.
- DI_REAL - Double Integer to Floating-Point.
- I_BCD - Integer to BCD.
- ROUND - Round to Double Integer.
- Apod.

Příklad použití převodní instrukce DI_REAL:

LAD



Obr. 4.9: Příklad použití převodní instrukce DI_REAL.

Shrnutí pojmů

Ve Step 7 existují **elementární**, **složené** a **uživatelské** datové typy. Jednotlivé proměnné v programu pak mají přiřazen jeden z konkrétních datových typů.

Přenosové instrukce LOAD a TRANSFER jsou základní instrukce pro práci s číselnými hodnotami. Slouží pro přesun hodnot z jedné paměťové buňky do druhé.

Aritmetické instrukce slouží pro zpracování číselných hodnot. Ve Step 7 existuje široká nabídka aritmetických instrukcí pro práci jak v pevné tak pohyblivé řádové čárce.

Instrukce posuvů a rotací slouží pro realizaci posunu či rotace bitů v rámci nějaké paměťové buňky.

Instrukce skoků slouží pro určité větvení programu a skoky mohou být podmíněný různými podmínkami.

Porovnávací instrukce slouží k porovnávání dvou číselných hodnot.

Převodní instrukce slouží pro převod proměnných na jiný datový typ nebo pro převod do nějakého speciálního kódu.

Kontrolní otázky

1. Jaké jsou elementární datové typy ve Step7?
2. K čemu slouží instrukce LOAD a TRANSFER?
3. Jaký je rozdíl mezi posuvem a rotací?

4. K čemu se používají instrukce skoků JC a JNC?
5. K čemu slouží převodní instrukce a jaké znáte?

Další zdroje

4-1. Siemens: SIMATIC Programming with STEP7 V5.3, Edition 01/2004, A5E00261405-01.

4-2. Siemens: SIMATIC Working with STEP7 V5.3 Getting Started, 01/2004, A5E00261403-01.

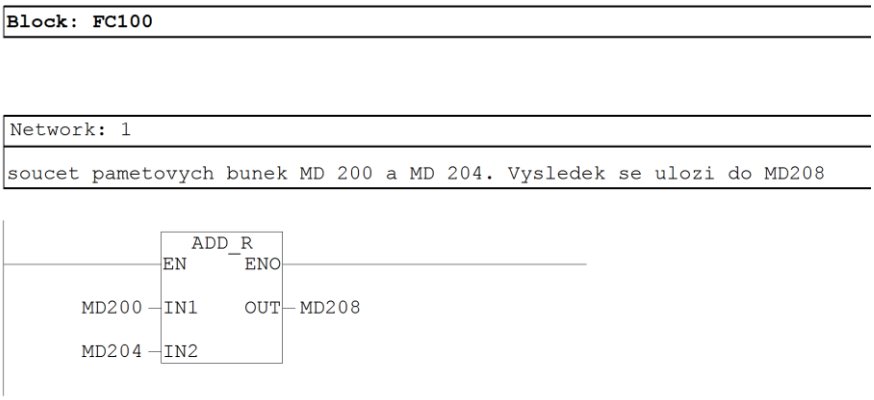
4-3. Siemens: SIMATIC S7 Statement List (STL) for S7-300 and S7-400 Programming, 6ES7810-4CA04-8BR0.

4-4. Siemens: SIMATIC S7 Ladder Logic (LAD) for S7-300 and S7-400 Programming. 6ES7810-4CA07-8BW1.

Řešená úloha 4.1.

Vytvořte funkci FC100, která sečte obsah paměťových buněk MD200 a MD204 a výsledek uloží do MD 208. Paměťové buňky obsahují reálná čísla

Řešení programu v LAD



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 4\pr_4_1.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 2\ anim2.avi.

Řešená úloha 4.2.

Vytvořte „univerzální“ funkci, která opět sečte dvě čísla a zapíše výsledek. Hodnoty na vstupu funkce bude možné měnit. Předpokládejte vícenásobné použití funkce.

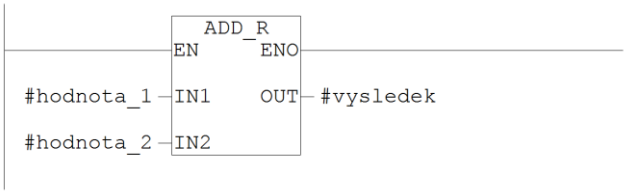
Řešení programu v LAD

FC 1

Name	Data Type	Address	Comment
IN		0.0	
hodnota_1	Real	0.0	
hodnota_2	Real	4.0	
OUT		0.0	
vysledek	Real	8.0	
IN_OUT		0.0	
TEMP		0.0	
RETURN		0.0	
RET_VAL		0.0	

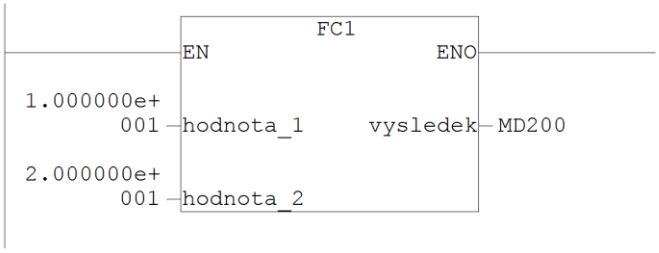
```
Block: FC1
```

```
Network: 1

secte dve promenne, ktere jsou na vstupu funkce a zapise na vystup funkce
```




```
Block: OB1   "Main Program Sweep (Cycle)"
```

```
Network: 1

priklad pouziti funkce FC1 secte hodnotu 10 a 20
vysledek zapise do promenne MD200
```



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 4\pr_4_2.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 5\ anim2.avi.

Řešená úloha 4.3.

Vytvořte sdílený datový blok DB100, v tomto DB deklarujte 4 proměnné, z nichž 3 budou typu real. Pojmenujte proměnné hodnota1, hodnota2 a výsledek. Datový blok pojmenujte symbolicky jako DATA. Proměnnou hodnota1 nastavte v datovém bloku na hodnotu 100 a proměnnou hodnota2 na hodnotu 200.

Řešení programu v LAD

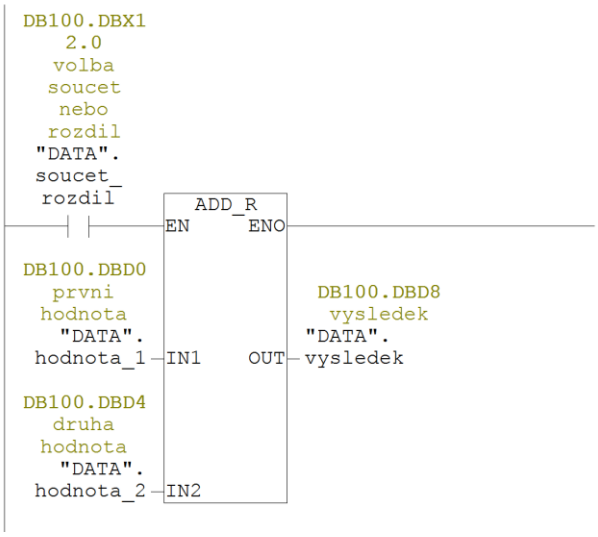
Block: DB100

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	hodnota_1	REAL	1.000000e+002	prvni hodnota
+4.0	hodnota_2	REAL	2.000000e+002	druha hodnota
+8.0	vysledek	REAL	0.000000e+000	vysledek
+12.0	soucet_rozdil	BOOL	FALSE	volba soucet nebo rozdil
=14.0		END_STRUCT		

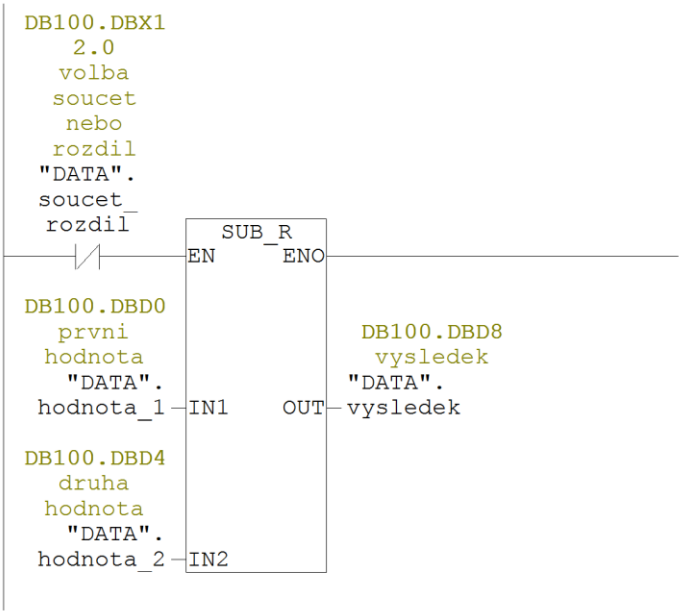
Block: FC1

Network: 1

vypočítá součet hodnot, jestlize soucet_rozdil = true

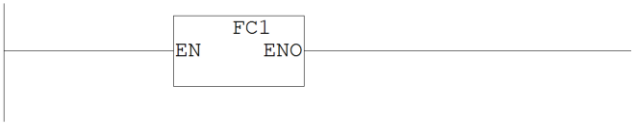


```
Network: 2

vypocita rozdil, jestlize soucet_rozdil = false
```




```
Block: OB1    "Main Program Sweep (Cycle)"
```

```
Network: 1
```



Řešení programu v STL

```
Block: FC1
```

```
Network: 1

vypočítá součet hodnot, jestlize soucet_rozdil = true
```

```
      A     "DATA".soucet_rozdil  DB100.DBX12.0      -- volba soucet nebo rozdil
      JNB   _001
      L     "DATA".hodnota_1      DB100.DBD0         -- prvni hodnota
      L     "DATA".hodnota_2      DB100.DBD4         -- druha hodnota
      +R
      T     "DATA".vysledek       DB100.DBD8         -- vysledek
_001: NOP   0
```

```
Network: 2

vypocita rozdil, jestlize soucet_rozdil = false
```

```
      AN    "DATA".soucet_rozdil  DB100.DBX12.0      -- volba soucet nebo rozdil
      JNB   _002
      L     "DATA".hodnota_1      DB100.DBD0         -- prvni hodnota
      L     "DATA".hodnota_2      DB100.DBD4         -- druha hodnota
      -R
      T     "DATA".vysledek       DB100.DBD8         -- vysledek
_002: NOP   0
```

```
Block: OB1    "Main Program Sweep (Cycle)"
```

```
Network: 1
```

```
      CALL   FC      1
      NOP    0
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 4\pr_4_3.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 6\ anim1.avi.

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 6\ anim2.avi.

Řešená úloha 4.4.

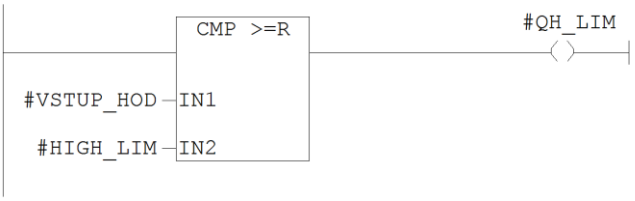
Vytvořte funkční blok, který bude porovnávat zda vstupní hodnota leží v definovaných mezích. Meze bude možné zadávat na vstup funkčního bloku, překročení nebo podkročení limitu signalizujte na výstupu funkčního bloku dvěmi bity. Vstupní proměnné funkčního bloku pojmenujte VSTUP_HOD, HIGH_LIM, LOW_LIM, výstupní bity pojmenujte QH_LIM, QL_LIM. K funkčnímu bloku vytvořte instanční datový blok, ve kterém budou uloženy limitní hodnoty. Stavy jednotlivých hodnot sledujte ve VAT tabulce.

Řešení programu v LAD

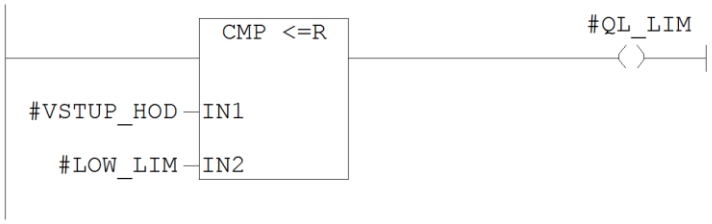
FB 100

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
VSTUP_HOD	Real	0.0	0.000000e+000	
HIGH_LIM	Real	4.0	0.000000e+000	
LOW_LIM	Real	8.0	0.000000e+000	
OUT		0.0		
QH_LIM	Bool	12.0	FALSE	
QL_LIM	Bool	12.1	FALSE	
IN_OUT		0.0		
STAT		0.0		
TEMP		0.0		

```
Network: 1

porovnává, jestli vstupní hodnota je vetsi
nebo rovna hornimu limitu
```




```
Network: 2

porovnává, jestli vstupní hodnota je vetsi
nebo rovna dolnimu limitu
```



Řešení programu v STL

```
Block: FB100
```

```
Network: 1

porovnává, jestli vstupní hodnota je vetsi
nebo rovna hornimu limitu
```

```
      L     #VSTUP_HOD
      L     #HIGH_LIM
      >=R
      =     #QH_LIM
```

```
Network: 2

porovnává, jestli vstupní hodnota je vetsi
nebo rovna dolnimu limitu
```

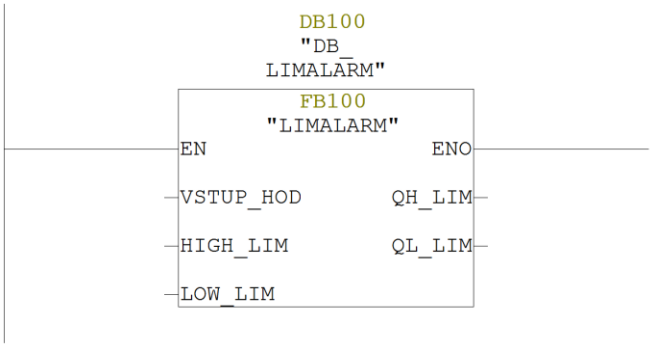
```
      L     #VSTUP_HOD
      L     #LOW_LIM
      <=R
      =     #QL_LIM
```

Řešení programu v LAD

OB1

```
Block: OB1   "Main Program Sweep (Cycle)"
```

```
Network: 1
volani funkciho bloku FB100
```




```
Network: 1
volani funkciho bloku FB100

      CALL  "LIMALARM" , "DB_LIMALARM"  FB100 / DB100
       VSTUP_HOD:=
       HIGH_LIM :=
       LOW_LIM  :=
       QH_LIM   :=
       QL_LIM   :=
      NOP   0
```

VAT tabulka

	Address	Symbol	Display format	Status value	Modify value
1	//vstupní hodnota				
2	DB100.DBD 0	"DB LIMALARM".VSTUP HOD	FLOATING POINT		
3	//horni limit				
4	DB100.DBD 4	"DB LIMALARM".HIGH LIM	FLOATING POINT		
5	//dolni limit				
6	DB100.DBD 8	"DB LIMALARM".LOW LIM	FLOATING POINT		
7					
8	//dosazen horni limit				
9	DB100.DBX 12.0	"DB LIMALARM".QH LIM	BOOL		
10	//dosazen dolni limit				
11	DB100.DBX 12.1	"DB LIMALARM".QL LIM	BOOL		

DB100

Address	Declaration	Name	Type	Initial value	Actual value
0.0	in	VSTUP_HOD	REAL	0.000000e+000	0.000000e+000
4.0	in	HIGH_LIM	REAL	0.000000e+000	0.000000e+000
8.0	in	LOW_LIM	REAL	0.000000e+000	0.000000e+000
12.0	out	QH_LIM	BOOL	FALSE	FALSE
12.1	out	QL_LIM	BOOL	FALSE	FALSE

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 4\pr_4_4.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 5\ anim1.avi.

Řešená úloha 4.5.

Vytvořte jednoduchou recepturu pro výrobu chleba. Předpokládejte, že se bude vyrábět 5 druhů chleba. Všechny receptury se budou ukládat do datového bloku. Receptura pro výrobu chleba bude obsahovat pouze pro jednoduchost tři položky a to mouka, sul, voda.

Příklad budu řešit tím způsobem, že vytvořím jednu recepturu, která bude obsahovat položky mouka, sul, voda. Tyto položky zapíši do UDT a symbolicky pojmenuji recept_chleba. Následně tuto UDT vložím do datového bloku DB100, který symbolicky pojmenuji „receptury_chleba“.

Řešení programu v LAD

UDT1 - <offline>

"recept_chleba"

Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 1.6.2007 19:19:57odp.
Interface: 1.6.2007 19:19:57odp.
Lengths (block/logic/data): 00000 00000 00000

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	mouka	REAL	0.000000e+000	
+4.0	sul	REAL	0.000000e+000	
+8.0	voda	REAL	0.000000e+000	
=12.0		END_STRUCT		

DB100 - <offline> - Declaration view

"receptury_chleba"
Global data block DB 100
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 1.6.2007 19:25:57odp.
Interface: 1.6.2007 19:25:57odp.
Lengths (block/logic/data): 00194 00060 00000

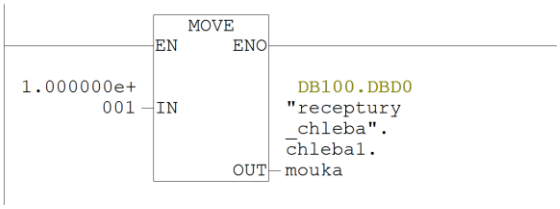
Block: DB100

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	chleba1	"recept_chleba"		
+12.0	chleba2	"recept_chleba"		
+24.0	chleba3	"recept_chleba"		
+36.0	chleba4	"recept_chleba"		
+48.0	chleba5	"recept_chleba"		
=60.0		END_STRUCT		

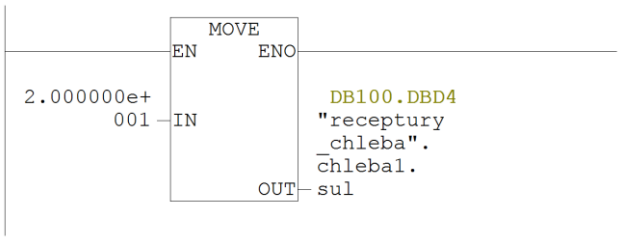
V OB1 zapíši do první receptury pro chleba 1 data.

Block: OB1 "Main Program Sweep (Cycle)"

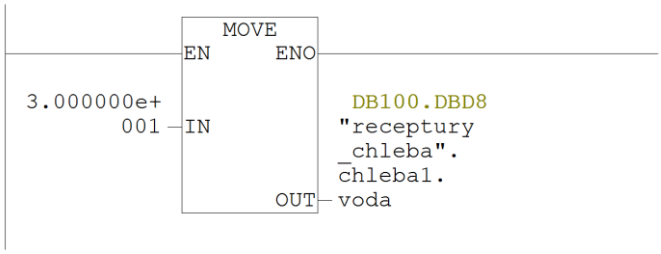
Network: 1



Network: 2



Network: 3



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 4\pr_4_5.zip

5. PROGRAMOVATELNÉ AUTOMATY SIMATIC S7 300, ZPRACOVÁNÍ ANALOGOVÝCH VELIČIN

Čas ke studiu: 2 hodiny

Cíl

Po prostudování této kapitole budete umět použít **analogové vstupy a výstupy** u programovatelného automatu Simatic S7 300. Jsou zde popsány možnosti čtení a zápisu analogových vstupů/výstupů a formát jejich hodnot. Jednotlivé způsoby jsou zde porovnávány a jsou popsány jejich výhody a nevýhody. Další část kapitoly se zabývá využitím mechanizmu **přerušení**. Jsou zde popsány typy přerušení, které jsou u programovatelného automatu Simatic S7 300 k dispozici a jsou demonstrovány možnosti jejich použití.

Výklad

5.1 Analogové vstupy a výstupy

Analogové vstupy a výstupy slouží pro připojení analogových (spojitých) technologických signálů. Analogové technologické signály jsou takové, jejichž hodnota se může měnit spojitě v určitém rozsahu. Používají se pro měření teplot, tlaků, poloh, napětí apod., případně jako výstupní signály pro ovládaní různých akčních členů, apod. Nejčastěji používané rozsahy analogových technologických signálů byly popsány v kapitole 3. Zbývá tedy ještě objasnit, jakým způsobem jsou analogové signály měřeny ze vstupů a zapisovány na výstupy.

Je zřejmé, že pokud má být analogový signál v programovatelném automatu zpracováván, musí být nějakým způsobem změřen. K tomu slouží tzv. analogové vstupy. Analogové vstupy jsou vývody programovatelného automatu, které jsou připojeny na A/D převodník a jsou tedy schopny převést přivedenou analogovou hodnotu na číslo. Toto číslo pak představuje velikost změřeného signálu a může být dále zpracováváno. Po zpracování je často potřebné výslednou hodnotu opět převést na analogový signál a vyvést ji z programovatelného automatu k ovládanému zařízení. K tomu slouží analogové výstupy. To jsou vývody, které vycházejí z D/A převodníku. D/A převodník umožňuje převést číslo na analogový signál.

V praxi je velice důležitá přesnost převodu obou typů převodníků. Základní parametr, který ovlivňuje přesnost převodu, je bitová šířka převáděné hodnoty (to znamená, kolika bity je reprezentováno převáděné číslo) - rozlišení. V praxi jsou běžně používané osmi, dvanácti, šestnácti bitové i jiné převodníky.

Příklad:

Změříme-li analogovou hodnotu osmi-bitovým převodníkem, rozdělíme měřicí rozsah na 256 kroků. V rozsahu 0-10 V je tedy jeden krok 0,039 V. Tím je dána i možná nepřesnost měření.

Pokud použijeme šestnácti-bitový převodník, rozdělíme měřicí rozsah na 65536 kroků. V rozsahu 0-10 V je jeden krok 0,000153 V. Měření je v tomto případě podstatně přesnější.

Je jasné, že přesnost měření ovlivňují i další vlivy, nejenom bitová šířka převodníku. Jejich popis však není náplní této přednášky a bližší informace je možné najít v celé řadě odborných publikací.

Analogové vstupy a výstupy bývají umístěny na přídavných modulech analogových vstupů, resp. výstupů. Tyto moduly se připojí k programovatelnému automatu a tím získáme možnost zpracovávat analogové signály. Výběrem vhodného přídavného modulu můžeme ovlivnit počet a parametry vstupů a výstupů. Způsob připojení a adresování analogových modulů byl popsán v kapitole 3. U některých kompaktních programovatelných automatů jsou analogové vstupy a výstupy umístěny i na základním kompaktu (označují se jako vestavěné vstupy/výstupy).

5.2 Čtení analogových vstupů

U programovatelného automatu Simatic S7 300/400 existují dva základní způsoby čtení analogových vstupů [4-3, 5-4, 5-5]:

- Přímé čtení – čtení peripherial wordu pomocí instrukce LOAD v STL nebo v LAD pomocí instrukce MOVE.
- Čtení pomocí funkce SCALE – FC 105.

Přímé čtení

Při přímém čtení analogového vstupu využíváme instrukce LOAD, přičemž čteme přímo z periferie – jako operand je peripherial word (PIW). To, že instrukce čte hodnotu vstupu přímo z periferie znamená, že se při této operaci nepoužívá PII (Process Input Image), ale program přistupuje ke vstupu přímo v okamžiku, kdy je instrukce prováděna.

Základní nevýhodou tohoto způsobu je to, že změřená hodnota je v rozsahu převodníku a je většinou nutné ji pro další zpracování určitým způsobem upravit. Tato výhoda se projeví hlavně tehdy, kdy je nutné v programu pracovat s daným analogovým signálem ve fyzikálních jednotkách. Pak bychom museli provést převod z hodnoty v rozsahu převodníku do fyzikálních jednotek sami v programu, což by mohlo být zbytečně náročné. Pokud však není nutné pracovat v programovatelném automatu s fyzikálním rozsahem (což je poměrně často), pak hlavní nevýhoda přímého čtení odpadá.

Naopak hlavní výhoda tohoto způsobu čtení je rychlost. Je to nejrychlejší způsob čtení analogového vstupu. Je třeba o něm uvažovat hlavně v případě, kdy je nutné v jednom okamžiku změřit více analogových vstupů. Při použití funkce SCALE by mohlo v takovém případě dojít k nežádoucímu prodloužení programového cyklu.

Měřená analogová hodnota je po převodu reprezentována 16-ti bitovým binárním číslem v pevné desetinné čárce.

Tab. 5.1: Reprezentace analogové hodnoty.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hodnota bitu	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Bit číslo 15 je vždy vyhrazen pro znaménko, přičemž 0 znamená kladné číslo a 1 záporné.

Pokud má A/D převodník menší rozlišení než 16 bitů, je změřená hodnota zarovnána v rámci 16-ti bitového slova vlevo a volné bity jsou doplněny nulami. Například:

Tab. 5.2: Úprava hodnoty při použití převodníku s nižším rozlišením.

Rozlišení	**Analogová hodnota**															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16ti-bitové rozlišení	0	1	0	0	0	1	1	0	0	1	1	1	0	0	1	1
13ti-bitové rozlišení	0	1	0	0	0	1	1	0	0	1	1	1	0	**0**	**0**	**0**

Tab. 5.3: Binární reprezentace bipolárního měřeného rozsahu.

Jednotky	**Měrená hodnota v %**	**Datové slovo**																**Rozsah**
		2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
32767	>118.515	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Přetečení
32511	117.589	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	Přes rozsah
27649	>100.004	0	1	1	0	1	1	0	0	0	0	0	0	0	0	1	1	
27648	100.000	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	Platný rozsah
1	0.003617	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-1	-0.003617	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
-27648	-100.000	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	
-27649	≤ -100.004	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	Pod rozsah
-32511	-117.589	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
-32767	≤ -118.515	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Podtečení

Tab. 5.4: Binární reprezentace unipolárního měřeného rozsahu.

Jednotky	**Měrená hodnota v %**	**Datové slovo**																**Rozsah**
		2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
32767	>118.515	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Přetečení
32511	117.589	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	Přes rozsah
27649	>100.004	0	1	1	0	1	1	0	0	0	0	0	0	0	0	1	1	
27648	100.000	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	Platný rozsah
1	0.003617	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
-1	-0.003617	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Pod rozsah
-4868	-17.589	1	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	
-32767	≤ -17.589	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Podtečení

Čtení pomocí funkce SCALE – FC 105

Funkce SCALE je standardní funkce, která je dodávána v knihovně funkcí ve STEP 7. Slouží pro konverzi integer hodnoty na reálné číslo (na hodnotu veličiny v inženýrských jednotkách). Velice

často se tato funkce používá pro čtení analogových vstupů. Hlavní výhoda tohoto způsobu je zřejmá – změřenou hodnotu dostaneme jako reálné číslo v požadovaném rozsahu, například tedy ve fyzikálním rozsahu veličiny.

Naopak hlavní nevýhoda je to, že provedení funkce SCALE trvá mnohem déle, než přímé čtení (závisí to na použitém procesoru).

Převodní vztah je následující:

OUT = [((FLOAT (IN) –K1)/(K2–K1)) * (HI_LIM – LO_LIM)] + LO_LIM

Konstanty K1 a K2 jsou automaticky voleny podle toho, jedná-li se o bipolární nebo unipolární rozsah:

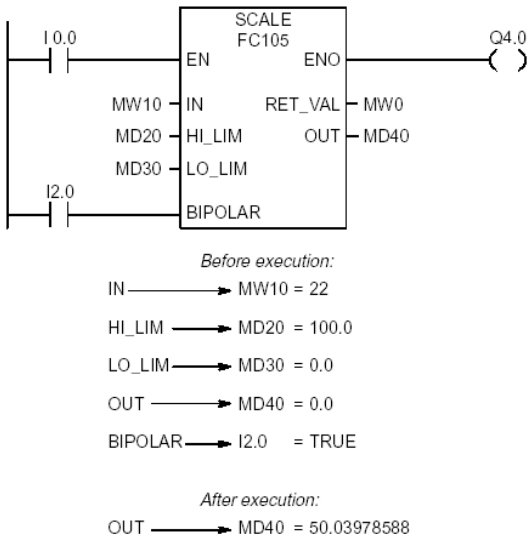
- bipolární – K1=-27648, K2=27648
- unipolární - K1=0, K2=27648

Pozn.: Hodnota 27648 je maximální velikost hodnoty rozsahu převodníku.

Tab. 5.5: Parametry funkce SCALE.

Parametr	**Deklarace**	**Typ**	**Paměťová oblast**	**Popis**
EN	Input	BOOL	I,Q,M,D,L	Aktivuje funkci
ENO	Output	BOOL	I,Q,M,D,L	Funkce byla provedena bez chyby
IN	Input	INT	I,Q,M,D,L,P, Const	Vstup, který má být konvertován
HI_LIM	Input	REAL	I,Q,M,D,L,P, Const	Horní limit
LO_LIM	Input	REAL	I,Q,M,D,L,P, Const	Dolní limit
BIPOLAR	Input	BOOL	I,Q,M,D,L	Rozlišení, zda se jedná o bipolární nebo unipolární rozsah
OUT	Output	REAL	I,Q,M,D,L,P	Výsledek převodu
RET_VAL	Output	WORD	I,Q,M,D,L,P	Vrácená hodnota – obsahuje chybovou informaci o převodu.

Příklad použití funkce SCALE:



Obr. 5.1: Příklad použití funkce SCALE.

5.3 Zápis analogových výstupů

U programovatelného automatu Simatic S7 300/400 existují dva základní způsoby zápisu analogových výstupů:

- Přímý zápis – zápis peripherial wordu pomocí instrukce TRANSFER v STL nebo instrukce MOVE v LAD.
- Zápis pomocí funkce UNSCALE – FC 106.

Přímý zápis

Zápis na analogový výstup se provede pomocí instrukce TRANSFER na peripherial word (PQW). Platí zde stejné výhody a nevýhody jako u čtení. Před provedením vlastního zápisu je nutné hodnotu převést na rozsah D/A převodníku, což většinou není problém, pokud nepracujeme s fyzikálními jednotkami. Pokud ano, je lepší zvolit pro zápis na výstup funkci UNSCALE.

Bitová reprezentace hodnoty pro zápis na analogový výstup vypadá stejně jako u analogového vstupu.

Zápis pomocí funkce UNSCALE – FC 106

Funkce UNSCALE je standardní funkce, která je dodávána v knihovně funkcí ve STEP 7. Je opakem funkce SCALE. Slouží pro konverzi reálného čísla (hodnoty veličiny v inženýrských jednotkách) na integer hodnotu. Velice často se tato funkce používá pro zápis analogových výstupů.

Hlavní výhoda tohoto způsobu je zřejmá – zapisovanou hodnotu máme jako reálné číslo a její převod na rozsah převodníku zajistí uvedená funkce.

Naopak hlavní nevýhoda je to, že provedení funkce UNSCALE trvá mnohem déle, než přímý zápis (závisí to na použitém procesoru).

Převodní vztah je následující:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

Konstanty K1 a K2 jsou automaticky voleny podle toho, jedná-li se o bipolární nebo unipolární rozsah:

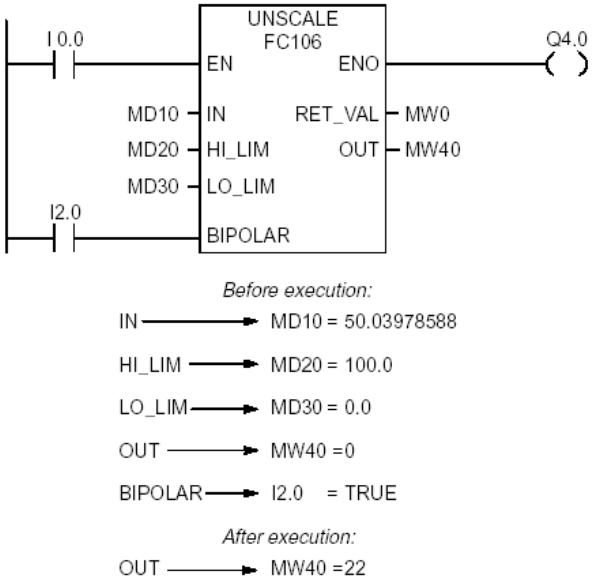
- bipolární – K1=-27648, K2=27648
- unipolární - K1=0, K2=27648

Pozn.: Hodnota 27648 je maximální velikost hodnoty rozsahu převodníku.

Tab. 5.6: Parametry funkce UNSCALE.

Parametr	**Deklarace**	**Typ**	**Paměťová oblast**	**Popis**
EN	Input	BOOL	I,Q,M,D,L	Aktivuje funkci
ENO	Output	BOOL	I,Q,M,D,L	Funkce byla provedena bez chyby
IN	Input	REAL	I,Q,M,D,L,P, Const	Vstup, který má být konvertován
HI_LIM	Input	REAL	I,Q,M,D,L,P, Const	Horní limit
LO_LIM	Input	REAL	I,Q,M,D,L,P, Const	Dolní limit
BIPOLAR	Input	BOOL	I,Q,M,D,L	Rozlišení, zda se jedná o bipolární nebo unipolární rozsah
OUT	Output	INT	I,Q,M,D,L,P	Výsledek převodu
RET_VAL	Output	WORD	I,Q,M,D,L,P	Vrácená hodnota – obsahuje chybovou informaci o převodu

Příklad použití funkce UNSCALE:



Obr. 5.2: Příklad použití funkce UNSCALE.

5.4 Přerušení

Jako „přerušení“ se označuje mechanizmus, který zajistí přerušení provádění rozpracovaného programu, vykonání určité obslužné rutiny, návrat zpět do programu na místo, kde byl přerušen a pokračování v jeho provádění. Během této operace je zajištěno, že nejsou ztracena rozpracovaná data. K přerušení programu dochází na základě nějaké vnější asynchronní události, kterou je nutné ihned zpracovat. Dokončí se tedy rozpracovaná instrukce uživatelského programu a následuje provedení obslužné rutiny, která je reakcí na vzniklou událost.

U programovatelných automatů se můžeme setkat s následujícími typy přerušovacích událostí:

Hardwarové přerušení – reaguje na náběžnou nebo sestupnou hranu na tzv. přerušovacím vstupu.

- Časové přerušení – reaguje na uplynutí určitého časového intervalu.
- Diagnostická a chybová přerušení.
- Komunikační přerušení – reaguje na příchod znaku na komunikační rozhraní.

Obecně platí, že obslužné rutiny by měly být co nejkratší, aby časově nezatěžovaly hlavní program.

V hlavním programu je možno povolit a zakázat obsluhu přerušení. Tím je možné zajistit, že určitá část programu nebude přerušena.

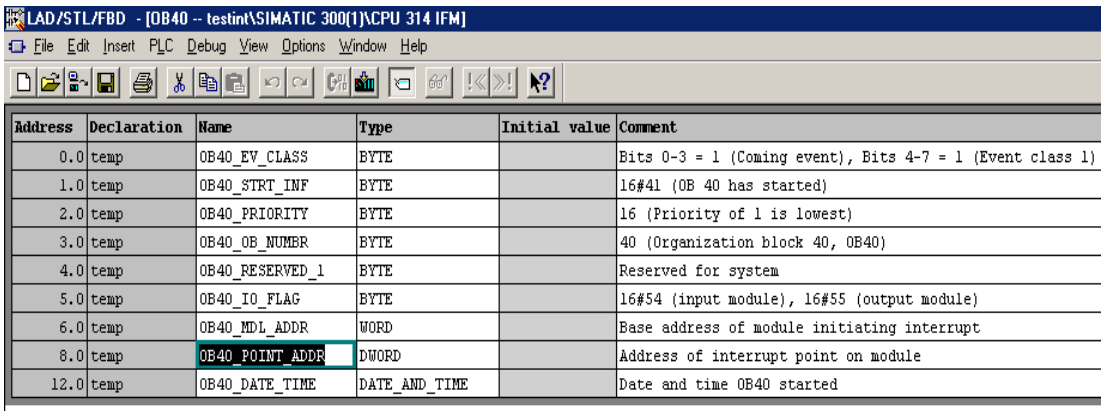
Hardwarové přerušení

Zpracování hardwarového přerušení je velice důležitá vlastnost programovatelných automatů řady Simatic S7 300/400. Umožňuje detekovat i ty digitální signály, u kterých se předpokládá rychlost změn kratší (nebo srovnatelná), než je doba cyklu programu. Bez hardwarového přerušení by nebylo možné zajistit, aby byly zachyceny všechny změny takového signálu (viz. Process Input Image a cyklus programu).

Pro zapsání obslužné rutiny slouží u Simaticu organizační blok OB40-47. V hardwarové konfiguraci je nutné specifikovat, na kterou hranu signálu bude příslušný blok reagovat. Automaty S7 300 mají možnost využívat jen OB40, S7 400 pak všechny uvedené bloky.

Pokud je přerušovací rutina vyvolávána několika událostmi (několika přerušovacími vstupy, případně náběžnou i sestupnou hranou), je nutné rozlišit, která událost rutinu vyvolala, aby bylo možné správně reagovat. K tomuto rozlišení slouží temporary proměnná OB40_POINT_ADDR, která je jedním z lokálních parametrů bloku OB40.

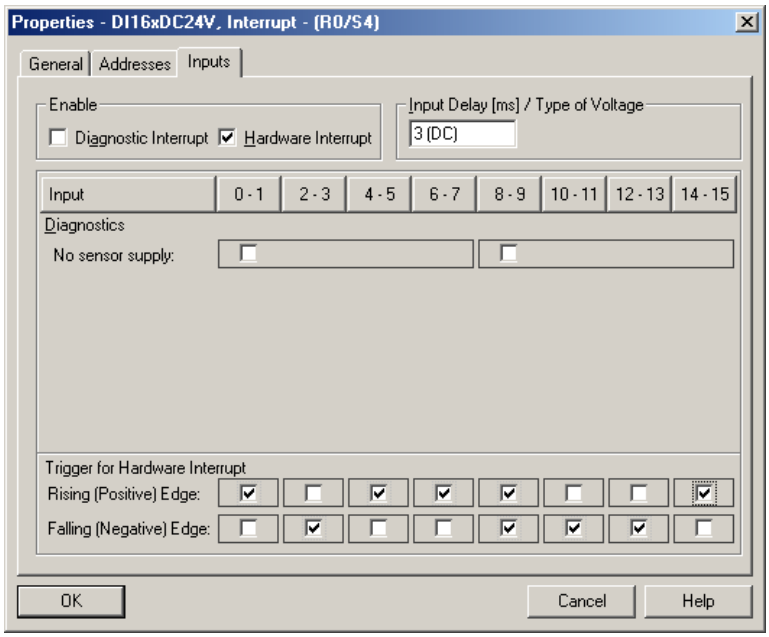
Lokální parametry bloku OB 40:



Address	Declaration	Name	Type	Initial value	Comment
0.0	temp	OB40_EV_CLASS	BYTE		Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
1.0	temp	OB40_STRT_INF	BYTE		16#41 (OB 40 has started)
2.0	temp	OB40_PRIORITY	BYTE		16 (Priority of 1 is lowest)
3.0	temp	OB40_OB_NUMBR	BYTE		40 (Organization block 40, OB40)
4.0	temp	OB40_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB40_IO_FLAG	BYTE		16#54 (input module), 16#55 (output module)
6.0	temp	OB40_MDL_ADDR	WORD		Base address of module initiating interrupt
8.0	temp	OB40_POINT_ADDR	DWORD		Address of interrupt point on module
12.0	temp	OB40_DATE_TIME	DATE_AND_TIME		Date and time OB40 started

Obr. 5.3: Lokální parametry bloku OB 40.

Nastavování hran signálu, které budou vyvolávat přerušení:



Obr. 5.4: Nastavování spouštěcích hran signálu.

Pokud dojde během zpracovávání přerušovací rutiny k další přerušovací události, nedojde k přerušení rutiny, ale požadavek je zařazen do fronty a k jeho obsloužení dojde až po ukončení provádění rozpracované rutiny.

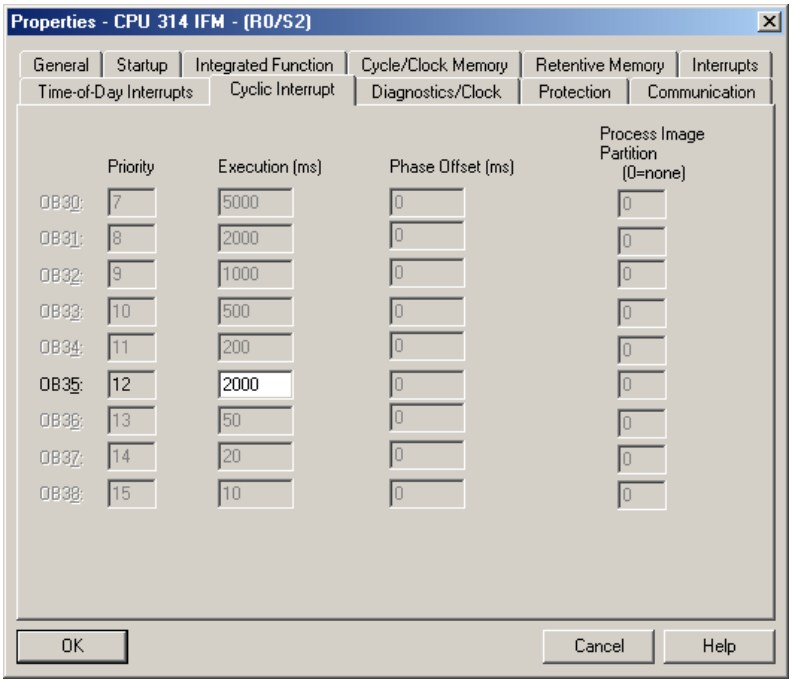
Časová přerušení

Časová přerušení přerušují program v daných časových intervalech. Tyto intervaly mohou (ale nemusí) být vztaženy k reálnému času. U Simaticu S7 300/400 je celá řada časových přerušení. Jejich obslužné rutiny jsou umístěny v příslušných organizačních blocích. Tyto bloky již byly popsány v kapitole 6.

Proto je zde již pouze uveden příklad nastavení parametrů bloků cyklického časového přerušení z hardwarové konfigurace:

Tab. 5.7: Organizační bloky časového přerušení.

OB cyklického přerušení (Cyclic Interrupt OB)	**Interval v ms**	**Priorita (Priority Class)**
OB30 (only S7-400)	5000	7
OB31 (only S7-400)	2000	8
OB32 (only S7-400)	1000	9
OB33 (only S7-400)	500	10
OB34 (only S7-400)	200	11
OB35	100	12
OB36 (only S7-400)	50	13
OB37 (only S7-400)	20	14
OB38 (only S7-400)	10	15



Obr. 5.5: Nastavování časového intervalu u cyklického přerušení.

Veškeré parametry časových přerušovacích bloků se nastavují v hardwarové konfiguraci.

Diagnostická a chybová přerušení

Umožňují reagovat na chyby hardwaru a chyby v provádění uživatelského programu. Pro zápis jejich obslužných rutin slouží příslušné organizační bloky. Jejich využití v uživatelském programu je velice výhodné. Umožní zabránit přepnutí programovatelného automatu do STOP modu, pokud dojde k určité chybě. Je-li tato chyba ošetřena v příslušné rutině k zastavení automatu nedojde.

Tab. 5.7: Organizační bloky diagnostických a chybových přerušení.

Asynchronní chyby/Chyby redundance	**Synchronní chyby**
OB70 I/O Redundancy Error (pouze H CPU)	OB121 Programming Error (například DB není zaveden do CPU)
OB72 CPU Redundancy Error (pouze H CPU, například, chyba CPU)	OB122 I/O Access Error (například přístup k signálovému modulu neexistuje)
OB80 Time Error (například překročení doby cyklu)	
OB81 Power Supply Error (například chyba baterie)	
OB82 Diagnostic Interrupt (například zkrat na vstupním modulu)	
OB83 Remove/Insert Interrupt (například vyjmutí vstupního modulu)	
OB84 CPU Hardware Fault (chyba MPI rozhraní)	

Asynchronní chyby/Chyby redundance	Synchronní chyby
OB85 Priority Class Error (například OB není zaveden do CPU)	
OB86 Rack Failure	
OB87 Communication Error (například nekorektní identifikátor u globální datové komunikace)	

Komunikační přerušení

Komunikační přerušení se zpravidla používá tehdy, chceme-li ošetřit příchod znaku na komunikační rozhraní. Využívá se to hlavně tehdy, je-li umožněno naprogramovat volnou komunikaci v rámci uživatelského programu (například u Simaticu S7 200). Komunikační přerušení v tomto smysly není u automatů S7 300 a 400 použito. Komunikace je zde řešena jiným způsobem.

Shrnutí pojmů

Analogové vstupy a výstupy slouží k měření a zápisu spojitých signálů programovatelným automatem. Pro převod těchto signálů se používají moduly s A/D a D/A převodníky. U programovatelných automatů S7 300/400 se používají dva způsoby čtení (zápisu) analogových vstupů (výstupů) – přímé čtení (zápis) a čtení pomocí funkce SCALE (UNSCALE). Každý ze způsobů má svoje výhody a nevýhody.

Přerušení je důležitý mechanizmus, který se používá pro okamžitou obsluhu asynchronních událostí. Existují následující základní typy přerušení:

Hardwarové přerušení.

- Časové přerušení.
- Diagnostická a chybová přerušení.
- Komunikační přerušení.

U PLC Simatic S7 300/400 jsou přerušovací rutiny zapisovány do příslušných organizačních bloků.

Kontrolní otázky

1. Jak závisí přesnost měření analogového vstupu na bitové šířce použitého převodníku?
2. Jaké jsou možnosti čtení analogových vstupů u PLC Simatic S7 300/400?
3. Popište výhody použití přímého čtení analogových vstupů oproti využití funkce Scale.
4. Popište výhody použití funkce Scale oproti využití přímého čtení analogových vstupů.
5. Popište formát slova získaného přímým čtením analogového vstupu.
6. Jak se liší měřená hodnota analogového vstupu při použití 8-mi a 12-ti bitového převodníku?
7. Co je to mechanizmus přerušení?
8. Jaké jsou typy přerušení?
9. K čemu se používá hardwarové přerušení?
10. Jaké typy časových přerušení máme k dispozici u PLC Simatic S7 300/400?

Další zdroje

5-1. Siemens: SIMATIC Programming with STEP7 V5.3, Edition 01/2004, A5E00261405-01.

5-2. Siemens: SIMATIC Working with STEP7 V5.3 Getting Started, 01/2004, A5E00261403-01.

5-3. Siemens: SIMATIC S7 Statement List (STL) for S7-300 and S7-400 Programming, 6ES7810-4CA04-8BR0.

5-4. Siemens: SIMATIC S7 Ladder Logic (LAD) for S7-300 and S7-400 Programming. 6ES7810-4CA07-8BW1.

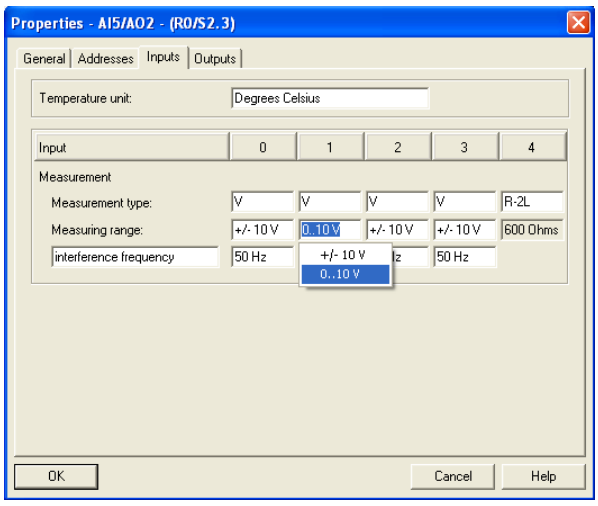
5-5. Siemens: SIMATIC S7-300 and M7-300 Programmable Controllers Module Specifications, Edition 2, EWA 4NEB 710 6067-02 01.

Řešená úloha 5.1.

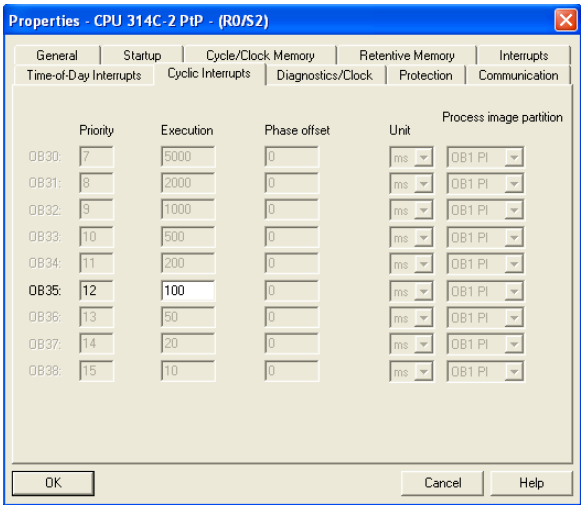
Vytvořte program, který bude v pravidelných intervalech (každých 100ms) vzorkovat analogový vstup na adrese 754. Analogový vstup upravte na rozsah 0-100 pomocí funkce SCALE FC105 a zapište do paměťové oblasti MD220. Vstupní signál je napěťový o úrovni 0-10V. Jako PLC použijte programovatelný automat S7-300 s CPU 314C-2PtP.

Slot	Module	Order number	Firmware	MPI address	I address	Q address	Comment
1							
2	CPU 314C-2 PtP	6ES7 314-6BF02-0AB0	V2.0	2			
X2	PtP				1023		
2.2	DI24/DO16				124..126	124..125	
2.3	AI5/AO2				752..761	752..755	
2.4	Count				768..783	768..783	
2.5	Position				784..799	784..799	
3							

Hardwarová konfigurace



Nastavení analogového vstupu na napěťový



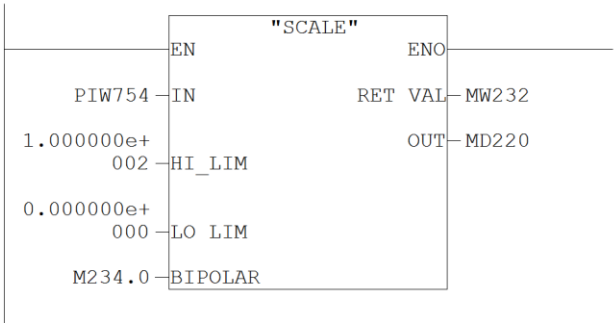
Nastavení vzorkovací periody signál o úrovni 0-10V

Vložení funkce SCALE FC105 do organizačního bloku OB35 (funkci najdeme v : **Libraries →Standard Library → TI-S7 Converting Blocks).**

Řešení programu v LAD

```
Block: OB35   "Cyclic Interrupt"

Vytvořte program, který bude v pravidelných intervalech (každých 100ms)
vzorkovat analogový vstup na adrese 754. Analogový vstup upravte na rozsah
0-100
pomocí funkce SCALE FC105 a zapište do paměťové oblasti MD220. Vstupní signál
je
napěťový o úrovni 0-10V. Jako PLC použijte programovatelný automat S7-300 s CPU
314C-2PtP.
```

```
Network: 1
```



Řešení programu v STL

```
Block: OB35   "Cyclic Interrupt"

Vytvořte program, který bude v pravidelných intervalech (každých 100ms)
vzorkovat analogový vstup na adrese 754. Analogový vstup upravte na rozsah
0-100
pomocí funkce SCALE FC105 a zapište do paměťové oblasti MD220. Vstupní signál
je
napěťový o úrovni 0-10V. Jako PLC použijte programovatelný automat S7-300 s CPU
314C-2PtP.
```

```
Network: 1
```

```
A     M    234.0
=     L     20.0
BLD   103
CALL  "SCALE"
 IN     :=PIW754
 HI_LIM :=1.000000e+002
 LO_LIM :=0.000000e+000
 BIPOLAR:=L20.0
 RET_VAL:=MW232
 OUT    :=MD220
NOP   0
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 5\pr_5_1.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 4\ anim2.avi.

Řešená úloha 5.2.

Vytvořte program, který bude v pravidelných intervalech (každých 200ms) vzorkovat druhý analogový vstup na kartě analogových vstupů. Analogový vstup upravte na rozsah 0-100 a zapište do paměťové oblasti MD222. Výstupní hodnota bude zobrazena s platností na dvě desetinná místa. Vstupní signál je napěťový o úrovni 0-10V. Jako PLC použijte programovatelný automat S7-300 s CPU 315 PN-DP a kartu analogových vstupů SM 334 (334-0CE01-0AA0). Pro uložení mezivýsledků převodu použijte lokální proměnnou organizačního bloku OB35 (TEMP).

Slot	Module	Order number	Firmware	MPI address	I address	Q address	Comment
1							
2	CPU 315-2 PN/DP	6ES7 315-2EG10-0AB0	V2.3	2			
X1	MPI/DP			2	2047*		
X2	PN-IO				2046*		
3							
4	AI4/AO2x8/8Bit	6ES7 334-0CE01-0AA0			256...263	256...259	
5							

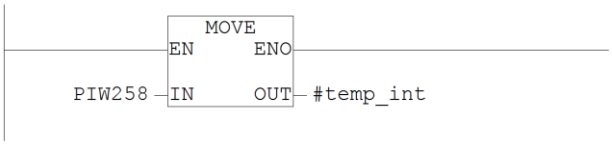
Řešení programu v LAD

Name	Data Type	Address	Comment
TEMP		0.0	
OB35_EV_CLASS	Byte	0.0	Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB35_STRT_INF	Byte	1.0	16#36 (OB 35 has started)
OB35_PRIORITY	Byte	2.0	Priority of OB Execution
OB35_OB_NUMBR	Byte	3.0	35 (Organization block 35, OB35)
OB35_RESERVED_1	Byte	4.0	Reserved for system
OB35_RESERVED_2	Byte	5.0	Reserved for system
OB35_PHASE_OFFSET	Word	6.0	Phase offset (msec)
OB35_RESERVED_3	Int	8.0	Reserved for system
OB35_EXC_FREQ	Int	10.0	Frequency of execution (msec)
OB35_DATE_TIME	Date_And_Time	12.0	Date and time OB35 started
temp_int	Int	20.0	pomocna promenna integer
temp_dint	DInt	22.0	pomocna promenna double integer
temp_real	Real	26.0	pomocna promenna real

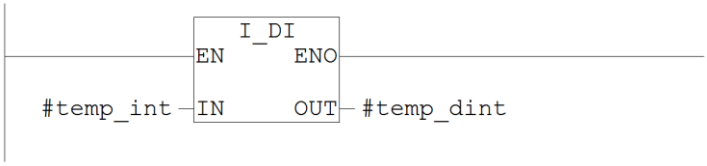
Block: OB35 "Cyclic Interrupt"

Vytvořte program, který bude v pravidelných intervalech (každých 200ms) vzorkovat druhý analogový vstup na kartě analogových vstupů. Analogový vstup upravte na rozsah 0-100 a zapište do paměťové oblasti MD220. Výstupní hodnota bude zobrazena s platností na dvě desetinná místa. Vstupní signál je napěťový o úrovni 0-10V. Jako PLC použijte programovatelný automat S7-300 s CPU 315 PN-DP a
kartu analogových vstupů SM 334 (334-0CE01-0AA0).

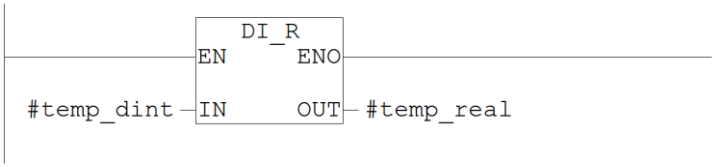
Network: 1 Načtení analogového vstupu



Network: 2 Převod z integer na double integer

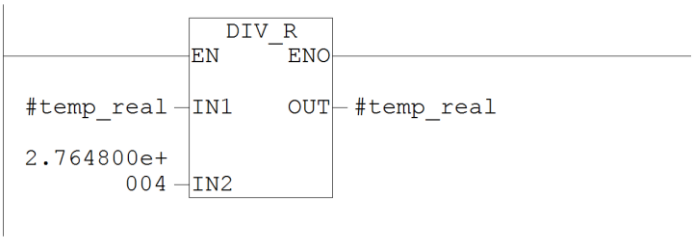


Network: 3 Převod z double integer na real



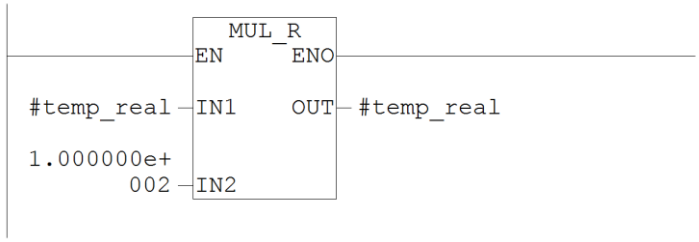
Network: 4 Změna rozsahu

Vydělení rozsahem převodníku tj. 27648



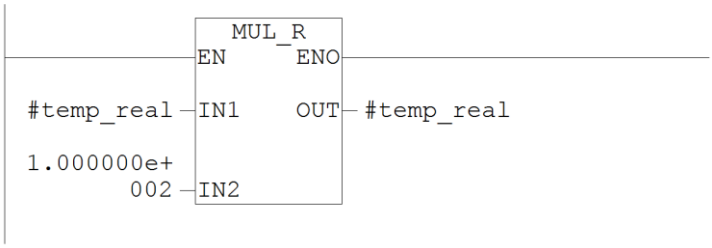
Network: 5 Změna rozsahu

Vynásobení 100 tj. úprava na rozsah 0-100



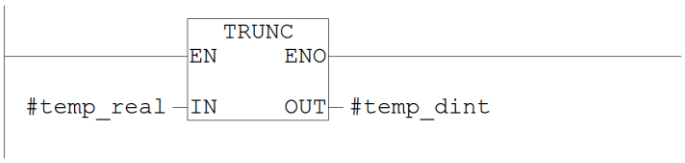
Network: 6 Úprava na dvě desetinná místa

Nejprve je nutno nejdříve hodnotu vynásobit hodnotou 100



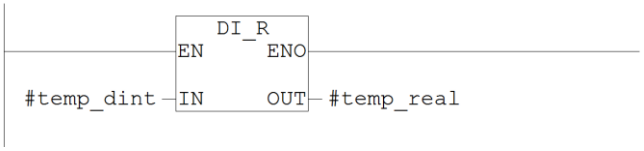
Network: 7 Úprava na dvě desetinná místa

Převod na double integer (zruším si tím desetinná místa)



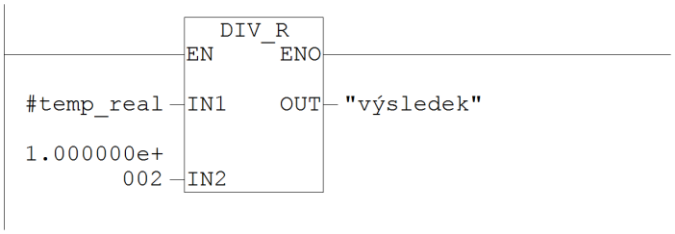
Network: 8 Úprava na dvě desetinná místa

Převod z double integer na real



Network: 9 Úprava na dvě desetinná místa

Obnovení počtu desetinných míst tj. mám dvě desetinná místa



Řešení programu v STL

```
Block: OB35   "Cyclic Interrupt"

Vytvořte program, který bude v pravidelných intervalech (každých 200ms)
vzorkovat druhý analogový vstup na kartě analogových vstupů. Analogový vstup
upravte na rozsah 0-100 a zapište do paměťové oblasti MD220. Výstupní hodnota
bude zobrazena s platností na dvě desetinná místa. Vstupní signál je napěťový o
úrovni 0-10V. Jako PLC použijte programovatelný automat S7-300 s CPU 315 PN-DP a
kartu analogových vstupů SM 334 (334-0CE01-0AA0).
```

```
Network: 1

      L     PIW  258        //hodnota je zapsána do ACCU1
      ITD                   //převod integer na double integer
      DTR                   //převod DI na reálné číslo

//úprava rozsahu z 0-27468 na 0-100
      L     2.764800e+004
      /R                    //vydělení rozsahem převodníku
      L     1.000000e+002
      *R
//úprava na dvě desetinná místa
      L     1.000000e+002
      *R
      TRUNC
      DTR
      L     1.000000e+002
      /R

       T     "vysledek"                                    MD222
```

DVD-ROM

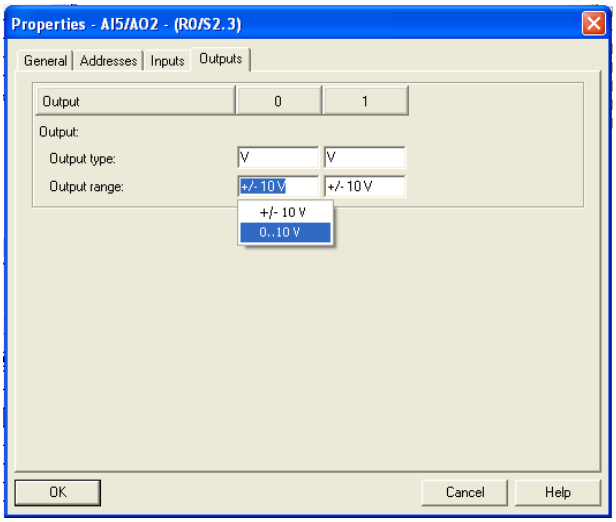
Řešený příklad naleznete na DVD: cvičení\cvičení 5\pr_5_2.zip

DVD-ROM

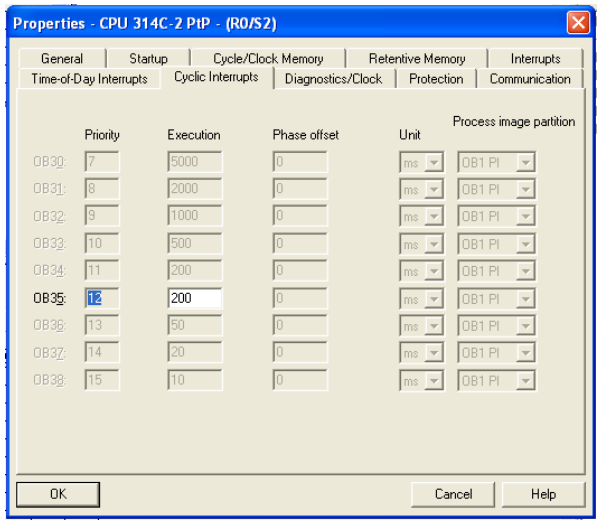
K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 4\ anim1.avi.

Řešená úloha 5.3.

Vytvořte organizační blok OB35, který bude v pravidelný časových intervalech 200ms zapisovat na analogový výstup obsah paměťové buňky MD100, kde vstupní hodnota bude v rozsahu 0-100. Rozsah zvolte unipolární. Pro zápis na analogový výstup využijte funkci UNSCALE



Nastavení analogového rozsahu na 0-10V

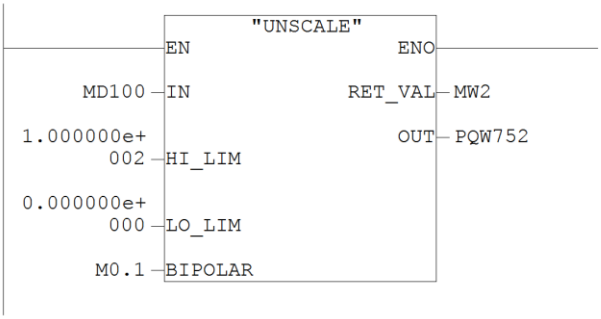


Nastavení vzorkování

Řešení programu v LAD

```
Block: OB35   "Cyclic Interrupt"
```

```
Network: 1

Tento network zapisuje obsah pametove bunky MD100 na analogovy vystup PQW 752.
Rozsah je zvolen jako unipolarni.
```



Řešení programu v STL

```
Block: OB35   "Cyclic Interrupt"
```

```
Network: 1

Tento network zapisuje obsah pametove bunky MD100 na analogovy vystup PQW 752.
Rozsah je zvolen jako unipolarni.
```

```
A     M      0.1
=     L     20.0
BLD   103
CALL  "UNSCALE"
 IN     :=MD100
 HI_LIM :=1.000000e+002
 LO_LIM :=0.000000e+000
 BIPOLAR:=L20.0
 RET_VAL:=MW2
 OUT    :=PQW752
NOP   0
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 5\pr_5_3.zip

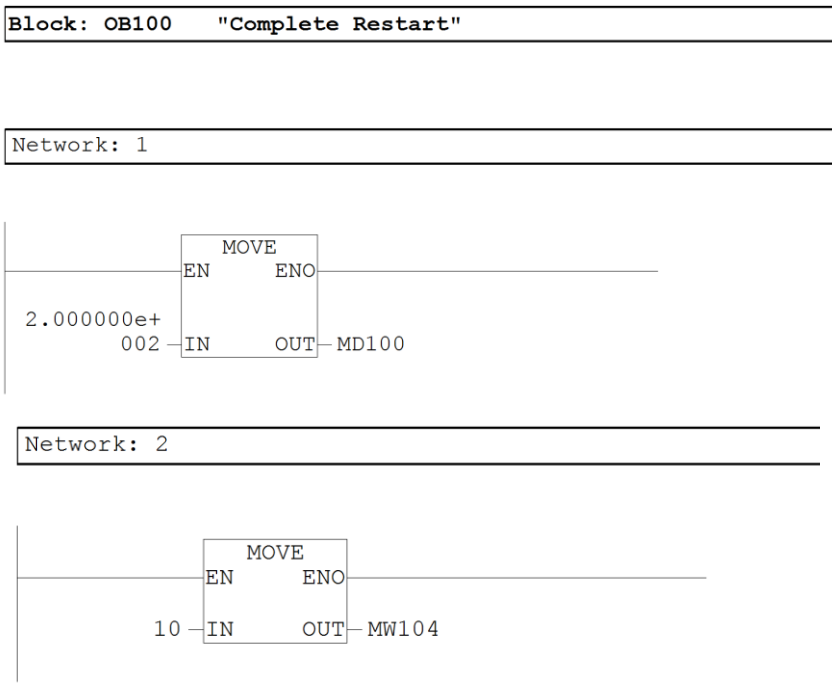
DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 4\ anim3.avi.

Řešená úloha 5.4.

Vytvořte organizační blok OB100, který při prvním cyklu PLC nastaví do paměťových buněk MD100 a MW104 hodnoty 200.0 a 10.

Řešení programu v LAD



Řešení programu v STL

```
Block: OB100    "Complete Restart"
```

```
Network: 1
```

```
      L     2.000000e+002
      T     MD   100
      NOP   0
```

```
Network: 2
```

```
      L     10
      T     MW   104
      NOP   0
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 5\pr_5_4.zip

6. PROGRAMOVATELNÉ AUTOMATY BERNECKER-RAINER

Čas ke studiu: 2 hodiny

Cíl

Kapitola popisuje **programovatelné automaty firmy Bernecker&Rainer**. Jsou zde prezentovány základní produkty výrobní řady programovatelných automatů této firmy. Část textu je věnována rovněž typům použitých **procesorových jednotek** a jejich vývojovým generacím.

V další části kapitoly je popsán softwarový nástroj **Automation Studio** pro tvorbu řídicích aplikací pro uvedené automaty.

Student se rovněž seznámí s principy zpracování programu těchto automatů, zejména je zde popsán tzv. **deterministický multitasking**. Jsou zde rovněž základní informace o **programovacích jazycích**, **datových typech** a využití **simulátoru** programovatelného automatu v programovacím prostředí.

Výklad

6.1 Programovatelné automaty Bernecker Rainer

Programovatelné automaty, power panely, frekvenční měniče od firmy Bernecker Rainer B&R splňují všechny požadavky pro komplexní řízení v automatizovaném systému řízení výkon, funkčnost a provozní spolehlivost. Systémy B&R 2003, B&R 2005, X20, X67 a další zahrnují celou řadu aplikací od nejjednodušších logický procesorů po komplexní decentralizované systémy automatizovaného řízení. Každý systém má rozdílnou strukturu, montáž, modularitu a výkon CPU. Tímto svým širokým spektrem výkonu mohou být použity v centralizovaném i decentralizovaném automatickém řízení. PCC – Programmable Computer Controller. Díky přechodu využívaných procesorů z řady procesorů Motorola na procesory Intel poskytují také tyto programovatelné automaty a Power panely různé přídavné možnosti jako WebServer, VNC Server, USB rozhraní, paměť Compact Flash apod. Ve většině procesorů a Power panelů také existuje spolu se základním rozhraním RS232 a USB také možnost připojení do sítě Ethernet a tím využívat např. i FTP komunikaci.

Obr. 6.1: Přehled produktů Bernecker Rainer.

6.1.1 B&R System 2003

B&R System 2003 může být použit pro kompletní řízení právě tak jako pro distribuovaný systém. Může být rozšířen o jednotlivé moduly digitálních a analogových vstupů a výstupů. Může komunikovat např. s průmyslovými počítači nebo s automaty z celé řady B&R. Různé druhy komunikačních modulů pro fieldbus sítě nebo sítě zajišťující bezporuchovou komunikaci.

Centrální procesorová jednotka zahrnuje široké spektrum použití . Optimální cena k výkonu umožňuje, že se tyto systémové jednotky stále používají. Jednotky CPU jsou výkonnostně odstupňované s pamětí, integrovaným komunikačním interfacem a u některých jednotek nabízejí možnost připojení screw-in modulů. K naprogramování těchto CPU slouží B&R Automation Studio. [6-15, 6-17]

Obr. 6.2: Příklad procesorové jednotky.

Pokud je systém B&R 2003 použit také ve spojení se vzdálenými vstupy a výstupy, mohou být tyto řídicí jednotky spojeny pomocí sběrnice CAN, ETHERNET Powerlink a X2X spojení.

B&R 2003 nabízí velké množství vstupně výstupních modulů v různém provedení. Pomocí analogových, digitálních signálů spolu s časovači a čítači, PID regulátory můžete vytvářet různé programové struktury.

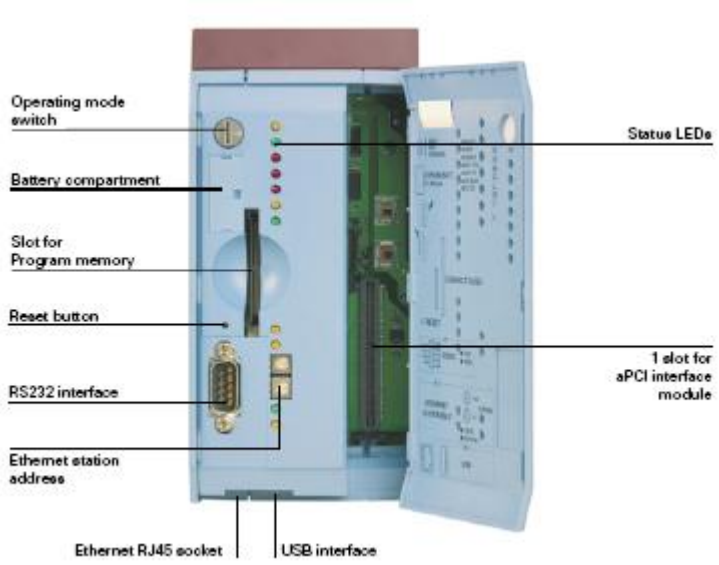
Tab. 6.1: Přehled procesorových jednotek.

Modul	Popis
CP 430	2003 CPU, 100KB SRAM, 256 KB Flash PROM, 24VDC, 7W supply, 1 RS232 interface, 1 CAN interface, CAN max. 64 digital/32 analog I/O points
CP 470	2003 CPU, 100KB SRAM, 256 KB Flash PROM, 24VDC, 14W supply, 1 RS232 interface, 1 CAN interface, CAN max. 128 digital/64 analog I/O points
CP 474	2003 CPU, 100KB SRAM, 512 KB Flash PROM, 24VDC, 12,6W supply, 1 RS232 interface, 1 CAN interface, CAN, 4 sloty pro screw in moduly, max. 208 digital/80 analog I/O points
CP 476	2003 CPU, 750KB SRAM, 512 KB Flash PROM, 24VDC, 12,5W supply, 1 RS232 interface, 1 CAN interface, CAN, 4 sloty pro screw in moduly, max. 272 digital/80 analog I/O points
CP 770	2003 CPU, 100KB SRAM, 512 KB Flash PROM, 100-240VAC, 14W supply, 1 RS232 interface, 1 CAN interface, CAN, 4 sloty pro screw in moduly, max. 128 digital/64 analog I/O points
CP 774	2003 CPU, 100KB SRAM, 512 KB Flash PROM, 100-240VDC, 12,6W supply, 1 RS232 interface, 1 CAN interface, CAN, 4 sloty pro screw in moduly, max. 208 digital/80 analog I/O points

6.1.2 B&R System 2005

B&R System 2005 je výkonný programovatelný automat z řady B&R. Zahrnuje široké spektrum uplatnění od velkého počtu digitálních a analogových vstupů a výstupů, použití při řízení různých strojů, apod. Může být zapojen do různých sítí spolu s průmyslovými počítači a automaty z nabídky B&R.

Novější B&R System 2005 CPU jsou vybaveny kompatibilními procesory Intel s nejmodernější architekturou. Spolu s možností využívat paměti Compact Flash, komunikací přes Ethernet a PCI rozšiřujícími sloty dovolují začlenit tyto systémy do světa komunikačních technologií. Mohou být také začleněny do distribuovaného řízení prostřednictvím jednotek vstupů a výstupů. Jednotlivé moduly mohou být např. Spojeny prostřednictvím sítě ETHERNET Powerlink. Do základních jednotek jsou také zahrnuty různé komunikační rozhraní. Protože celá řada těchto procesorů je postavena na procesorech Intel, poskytují také tyto procesory další služby jako je Web Server, možnost připojit se k nim přes FTP a také VNC Server. Nevýhoda Systemu 2005 je ta (představme si sestavu CPU a za ním karty DI, DO, AI, AO), že jednotlivé rozšiřovací moduly DI, DO apod. musí být řazeny na backplane bezprostředně za sebou. Nelze mít neobsazen jeden slot mezi dvěma moduly I/O, protože moduly umístěné za touto mezerou ve slotu již system nerozpozná!!!!! [6-16, 6-17]



Obr. 6.3: Základní jednotka CP 360.

- Vysoko výkonnostní systémy díky paralelnímu procesorovému zpracování.
- Velmi rychlé I/O v jednotkách μs.
- Základní deska může mít maximálně 15 slotů.
- Komunikace ETHERNET, PROFIBUS, B&R NET2000, CAN, možnost připojení k systémům od jiných výrobců.
- Rozšiřitelnost – max. 52 modulů a 32 vzdálených stanic.

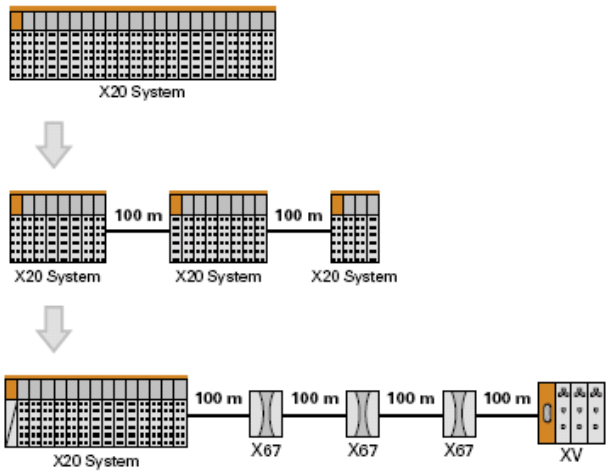
Tab. 6.2: Přehled procesorových jednotek.

Modul	**Popis**
CP 260	2005 CPU, 4MB DRAM, 850 kB SRAM, 512 KB Flash PROM, 1 PCMCIA slot, 1 RS232
CP 340	2005 CPU, x86 233 Intel compatible, 16MB DRAM, 512KB SRAM, vyměnitelná paměť Compact Flash, 1 slot pro aPCI modul, 1 USB interface, 1 RS232, 1 Ethernet 100Base
CP 360	2005 CPU, Pentium 266, 32MB DRAM, 512 SRAM, výměnná paměť Compact Flash, 1 slot pro aPCI, 1 USB interface, 1 RS232, 1 Ethernet 100 Base T
CP 380	2005 CPU, Pentium III 500, 64MB DRAM, 512 SRAM, výměnná paměť Compact Flash, 1 slot pro aPCI, 1 USB interface, 1 RS232, 1 Ethernet 100 Base T
CP 382	2005 CPU, Pentium III 500, 64MB DRAM, 512 SRAM, výměnná paměť Compact Flash, 3 slot pro aPCI, 1 USB interface, 1 RS232, 1 Ethernet 100 Base T

6.1.3 Systémy X20, X67

Systém X20 jako standardní I/O systém rozšiřuje možnosti řídicích systémů. Princip je stejně tak jednoduchý jako hospodárný: Sestavíte si systém podle svých požadavků. Nehraje roli, zda jedno-,

dvou- nebo třívodičové zapojení nebo zda jsou signály analogové. Všechny způsoby připojení a typy vstupů a výstupů jsou k dispozici. Systém X20 je inteligentní řídicí a I/O systém, centralizovaný nebo decentralizovaný. Architekturu a topologii určují uživatelé. Skok ve flexibilitě přináší decentralizovaná základnová deska (backplane). Kdekoli a kolikrát je potřeba můžete překonat libovolnou vzdálenost mezi jednotlivými modulu: mohou být umístěny vedle sebe na liště, v jednom rozvaděči, až 100m od sebe nebo v kombinaci s ostatními díly B&R. [6-12, 6-13, 6-14, 6-17]



Obr.6.4: Distribuované aplikace s X20.

Systém I/O pro těžké provozní podmínky se jmenuje X67. Je logickým pokračováním systému X20, chráněným proti vlivům prostředí a určeným pro distribuovanou montáž přímo na stroje a zařízení. Připojení ventilových pneumatických ostrůvků nezávislých na sběrnici s označením XV uzavírá požadavky na decentralizované I/O systémy.

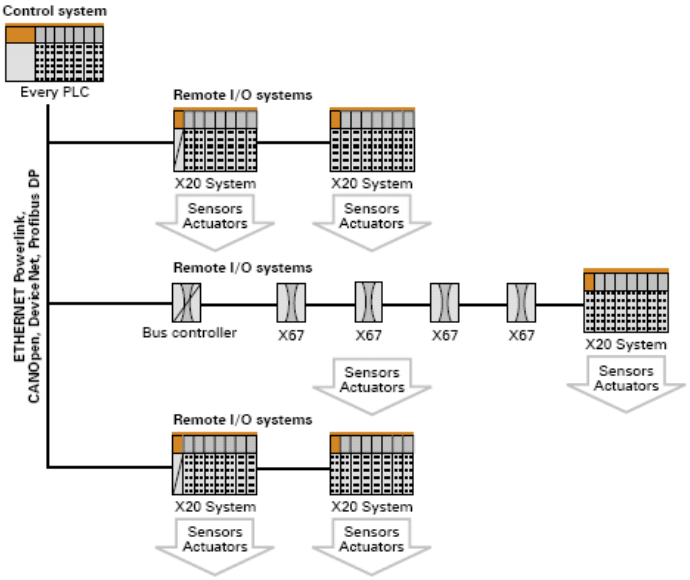
Obr.6.5: Systém X67.

Základní myšlenka spočívá v decentralizované struktuře. Všechny moduly spojuje X2X Link – jednotná „základnová deska“. Mezi dvěma sousedícími moduly X20, X67 nebo XV může být vzdálenost až 100m, rozměry rozvaděče přestávají být limitujícím faktorem. Přitom X2X Link na bázi krouceného měděného kabelu zaručuje nejvyšší odolnost proti rušení. Poprvé tak vzniká průchozí decentralizovaná základnová deska, která zajišťuje komunikaci jak mezi moduly bezprostředně vedle sebe, tak vzdáleně přes kabel X2X Link bez převodů a ztrát výkonů. Jedinečnou výhodou systému X20 je možnost nechat neobsazené základnové moduly pro rozšíření nebo varianty stroje nebo zařízení bez nutnosti změny adresování.

Celý systém je otevřený a tím přináší možnosti zahrnout tyto I/O systémy do sběrnic jako jsou PROFIBUS-DP, CAN a odvozené protokoly CANopen a DeviceNet jsou klasické sběrnice, které se na trhu prosadily. Na trh automatizace také stále silněji proniká i protokol ETHERNET Powerlink.

Pomocí řadičů sběrnic (bus controller) lze systém X20, X67 a XV připojit jako výkonné standardní systémy I/O ke všem uvedeným sběrnicím a používat je v běžný systémech PLC. Připravují se i další řadiče sběrnic vycházející ze sběrnice Ethernet – Ethernet /IP, Profinet a Modbus/IP.

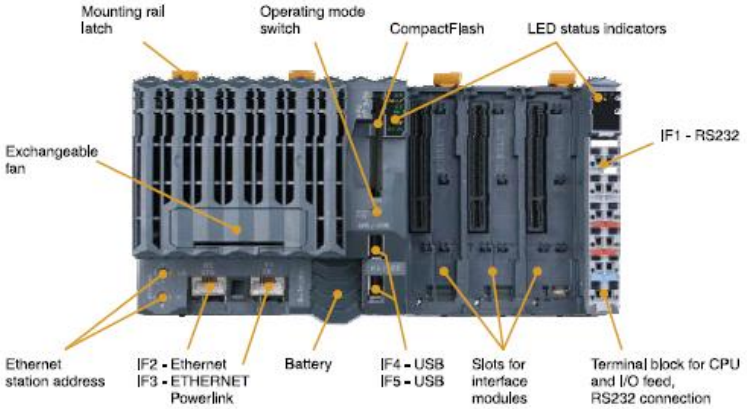
Příklad komunikace s B&R moduly: Chcete-li rychlost systému B&R využít i pro sběrnici PROFIBUS-DP, je možné nezávisle na cyklu sběrnice PROFIBUS snížit cyklus decentralizované základnové desky (X2X Linku) na 200µs. Velmi rychlé vstupy jsou zachyceny a zapamatovány a PROFIBUS Master je může vyčítat pomaleji.



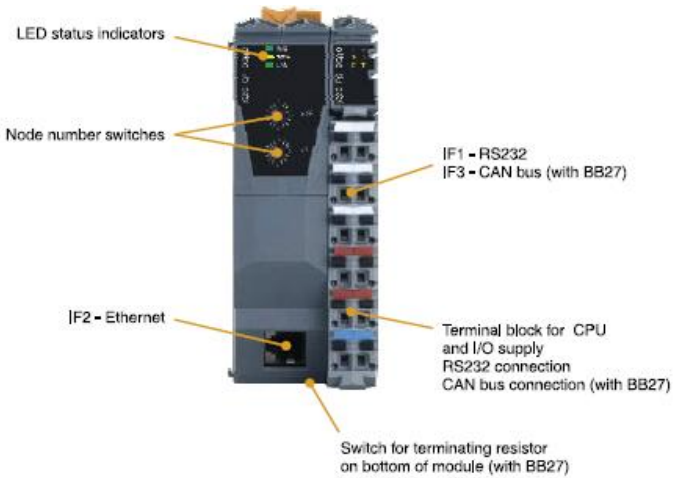
Obr. 6.6: Příklad konfigurace s moduly X67.

6.1.4 Systém X20 – CPU

CPU s v systému X20 lze najít v podobě standardních CPU nebo jako kompaktní CPU. Níže v tabulce je uveden přehled jednotlivých CPU.



Obr.6.7: Standardní CPU .



Obr.6.8: Kompaktní CPU.

Tab. 6.3: Přehled standardních CPU v systému X20.

X20CP3486	X20 CPU, Celeron 650, 64 MB DRAM, 1MB SRAM, vyjímatelná paměť Compact Flash, CPU obsahuje 3 sloty pro X20 IF moduly (komunikační moduly), 2 USB, 1 RS232, 1 Ethernet interface 10/100 Base-T, 1 ETHERNET Powerlink interface
X20CP1485	X20 CPU, Celeron 650, 64 MB DRAM, 1MB SRAM, vyjímatelná paměť Compact Flash, CPU obsahuje 1 slot pro X20 IF modul (komunikační modul), 2 USB, 1 RS232, 1 Ethernet interface 10/100 Base-T, 1 ETHERNET Powerlink interface
X20CP3485	X20 CPU, Celeron 400, 32 MB DRAM, 1MB SRAM, vyjímatelná paměť Compact Flash, CPU obsahuje 3 sloty pro X20 IF moduly (komunikační moduly), 2 USB, 1 RS232, 1 Ethernet interface 10/100 Base-T, 1 ETHERNET Powerlink interface
X20CP1485	X20 CPU, Celeron 400, 32 MB DRAM, 1MB SRAM, vyjímatelná paměť Compact Flash, CPU obsahuje 1 slot pro X20 IF modul (komunikační modul), 2 USB, 1 RS232, 1 Ethernet interface 10/100 Base-T, 1 ETHERNET Powerlink interface
X20CP3484	X20 CPU, Celeron 256, 32 MB DRAM, 1MB SRAM, vyjímatelná paměť Compact Flash, CPU obsahuje 3 sloty pro X20 IF moduly (komunikační moduly), 2 USB, 1 RS232, 1 Ethernet interface 10/100 Base-T, 1 ETHERNET Powerlink interface
X20CP1484	X20 CPU, Celeron 256, 32 MB DRAM, 1MB SRAM, vyjímatelná paměť Compact Flash, CPU obsahuje 1 slot pro X20 IF modul (komunikační modul), 2 USB, 1 RS232, 1 Ethernet interface 10/100 Base-T, 1 ETHERNET Powerlink interface

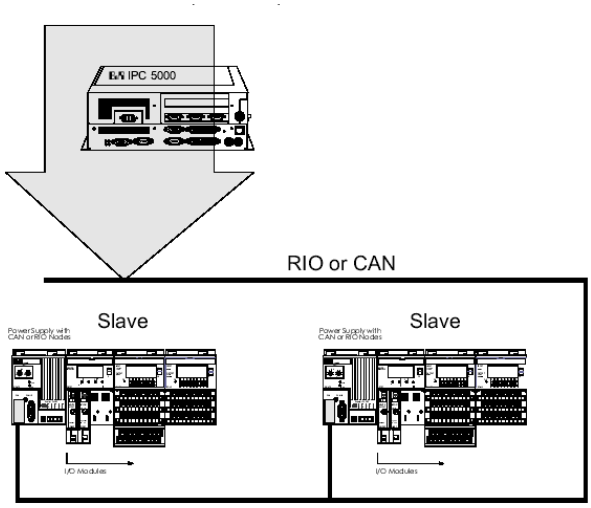
Tab.6.4: Přehled kompaktních CPU.

X20CP0292	X20 CPU, kompaktní CPU µP25, 750KByte SRAM, 3,4MB FPROM, RS232 a podpora CAN (rozhraní přímo na CPU), 1 Ethernet interface 100 Base - T
X20CP0291	X20 CPU, kompaktní CPU µP16, 100KByte SRAM, 512KB FPROM, RS232 a podpora CAN (rozhraní přímo na CPU), 1 Ethernet interface 100 Base - T
X20CP0201	X20 CPU, kompaktní CPU µP16, 100KByte SRAM, 512KB FPROM, RS232 a podpora CAN (rozhraní přímo na CPU)

Obr. 6.9: Příklad sestavy standardního CPU s I/O jednotkami.

6.1.5 Slot PLC

- Jedná se o PCI kartu, která se vkládá do průmyslového PC.
- Kombinuje vizualizační a ovládací úlohy v jediném zařízení.
- IPC5xxx ovládání vizualizačních úloh.
- LS251 ovládání ovládacích úloh, které je zcela nezávislé na IPC5xxx operačním systému.
- Vysokorychlostní výměna dat mezi LS251 a IPC5xxx prostřednictvím PCI sběrnice.
- Možnost rozšíření připojení k síti prostřednictvím CAN sběrnice nebo RIO.



Obr 6.10: Slot PLC.

6.2 Maximální rozšíření B&R System 2003, B&R System 2005, Systém X20

Každý programovatelný automat má omezené možnosti rozšíření a není možné rozšiřovat množství analogových/digitálních vstupů a výstupů do nekonečna. V systému B&R je každému modulu přidělena fyzická adresa a podle této adresy je možné daný CPU dále rozšiřovat. Maximální rozšíření pro B&R System 2003 ukazuje následující tabulka 6.5.

Každá rozšiřující jednotka např. digitálních vstupů zaujímá nějaké číslo pro rozšíření. Např. modulu DI 439 jsou přiřazeny dvě adresy tab. 6.6.

Když provedete upload hardwaru do B&R Automation Studia, vidíte v levé části obrazovky aktuální pozice modulů obr. 6.11.

Tab. 6.5: Maximální rozšíření vstupů/výstupů.

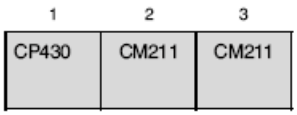
CPU	Maximální číslo slotu	Max. analogových modulů	Možné adresy pro analogové vstupy
CP 430	4	2	1-4
CP 470/CP770	6	4	1-8
CP474/CP774	12	4	1-8
CP 476	16	4	1-8
EX 270	4	2	1-2
EX 470/EX770	8	4	1-4
EX 477/EX777	6	4	1-6

Tab. 6.6: Rozšiřovací moduly.

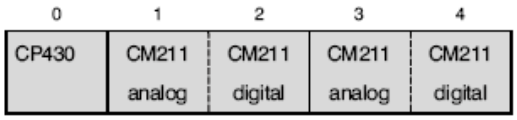
Rozšiřovací moduly	Maximální číslo slotu	Max. analogových modulů	Možné adresy pro analogové vstupy
AF 101	1	1	1
DI 439	2	-	1
DM 465	2	-	1
CM 211	2	1	1
CM 411	2	2	1

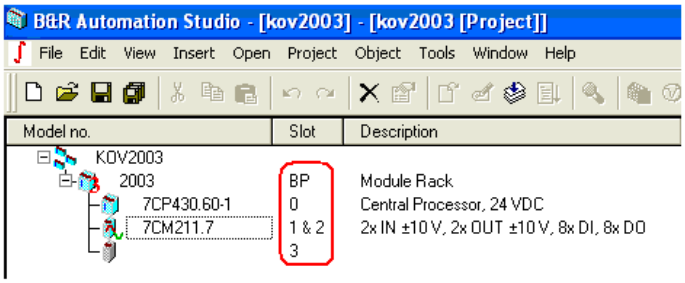
Příklad hardwarové konfigurace

Fyzická adresa



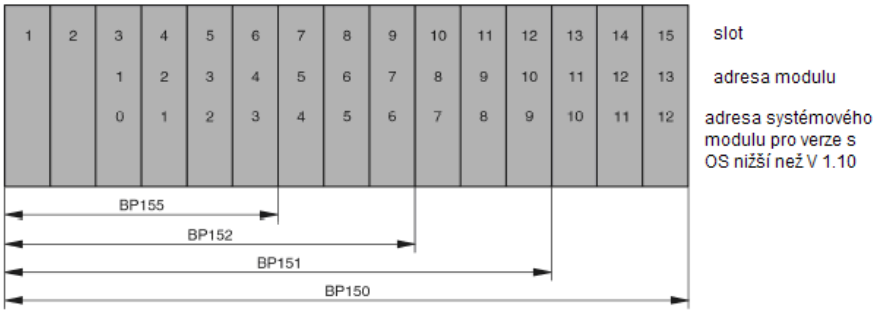
Logická adresa



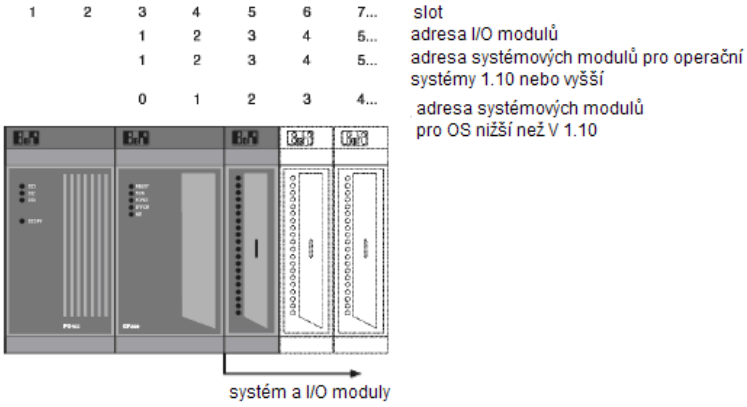


Obr. 6.11: Hardwarová konfigurace v B&R Automation Studiu B&R System 2003.

Maximální rozšíření pro B&R System 2005 ukazují následující obrázky obr. 6.12 a 6.13.

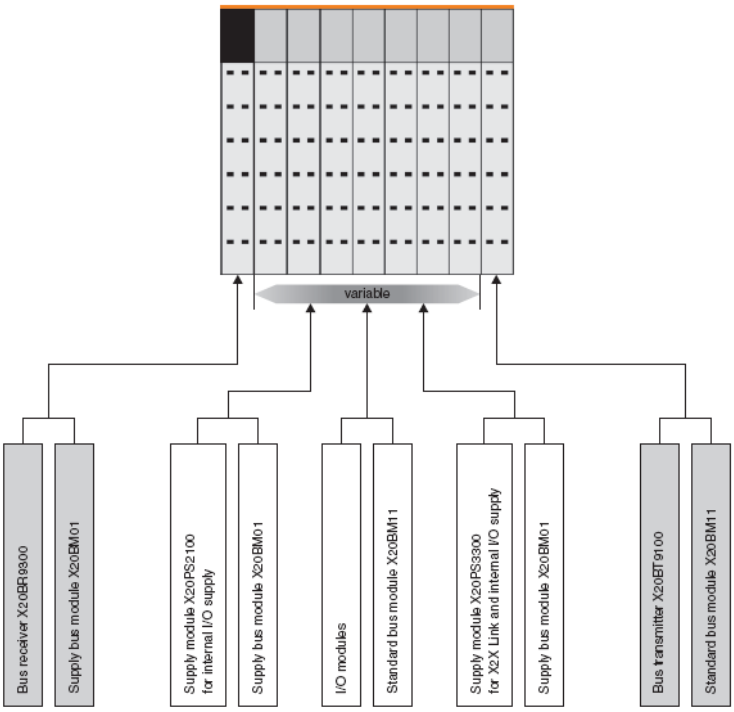


Obr.6.12: Maximální rozšíření pro B&R System 2005.



Obr.6.13: B&R System 2005.

Systémy X20 jsou především určeny do distribuovaných systémů s připojením na různé sítě, zejména pro komunikaci po X2X Linku. Aby mohlo CPU komunikovat po X2X linku je nutné CPU vybavit tímto rozhraním. Vždy také musí platit na decentrálních periferiích, kde kabel X2X linku musí vstupovat vždy do modulu receiver a pokud bude dál sběrnice rozšířena i za tuto decentrální periferii je nutné, aby na konci této periferie, kde bude kabel „vycházet“ je nutné, aby byl umístěn modul transmitter, který zaručí napěťovou stabilitu sítě. Uvádí se, že tímto způsobem mohu rozšiřovat sběrnici teoreticky bez limitu



Obr. 6.14: Řazení X20 modulů na periferii.

6.3 Generace programovatelných automatů Bernecker Rainer

Programovatelné automaty Bernecker Rainer jsou vyráběny ve dvou základních generacích SG3 (System Generation 3) a SG4. Základní rozdíl je v použitém procesoru u těchto automatů. Programovatelné automaty generace SG3 využívají procesor Motorola a automaty generace SG4 využívají procesor Intel.

Přehled základních typů procesorů:

Procesory patřící do generace SG3: CP260, IF260, IP161, XP152, IF100, IF101, CP100, CP104, CP152, CP153, CP200, CP210, **CP430**, CP470, CP474, CP476, CP770, CP774, **PP21, PP41**, PP15, LS251.

Procesory patřící do generace SG4: PP1x0, PP2x0, PP3x0, CP380, CP382, **CP360**, AR102, AR105, AR010, AR000, X20.

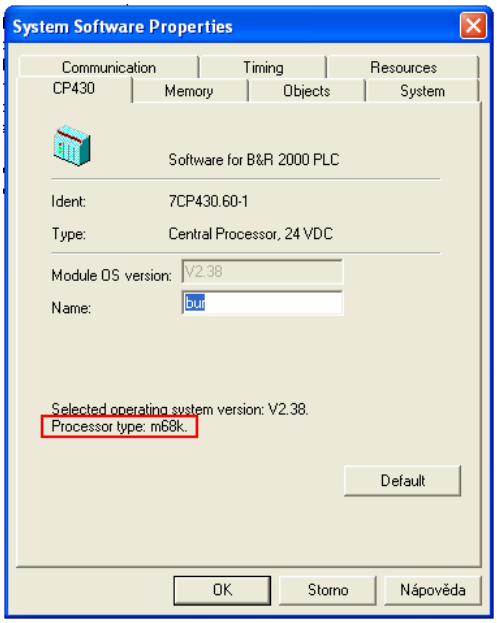
Programovatelné automaty s procesory Motorola SG3

Program je uložen v paměti flash a také se z této paměti bootuje při startu. Dále s tímto procesorem nemůžete komunikovat po ethernetu (neobsahuje CPU na sobě toto rozhraní), proto je většinou na CPU jenom rozhraní pro RS232 a CAN. Také je zde rozdíl v použití knihoven, protože tyto automaty nemají cooprocesor nemohou počítat v plovoucí řádové čárce např. u PID regulátorů.

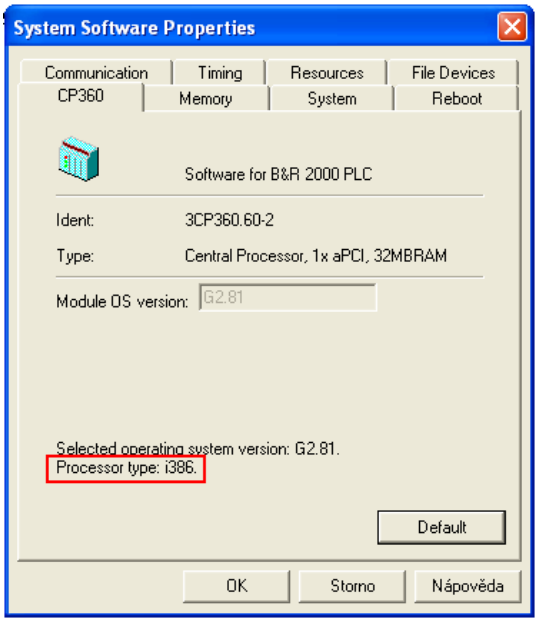
Programovatelné automaty s procesory Intel SG4

Program se zde ukládá nejprve na HDD nebo Compact Flash. Při bootování se program z paměti Compact Flash kopíruje do paměti DRAM, protože je tato paměť rychlejší a také se remanentní data

kopírují do DRAM. Další rozdíl je v tom, že systémy SG4 jako základní rozhraní využívají komunikaci přes Ethernet. CPU moduly jsou vybaveny přímo tímto rozhraním. Často také obsahují na CPU možnost připojení USB zařízení od klávesnice do USB disk. Dále také obsahují integrované funkce pro vzdálenou správu jako WebServer, VNC Server možnost připojit se k CPU přes FTP. I na programovatelném automatu může běžet vizualizace a pomocí VNC Klienta mohu tuto vizualizaci zobrazit na PC.



Obr. 6.15: Procesor řady SG3.



Obr. 6.16.: Procesor řady SG4.

6.4 Módy programovatelných automatů B&R

Každý programovatelný automat může být přepnut do následujících 4 režimů.

- RUN v tomto módu se vykonává uživatelský program, uživatelský program je poprvé nakopírován do paměti DRAM.
- SERVICE v tomto módu neběží program, běží pouze operační systém a automat očekává zásah od programátora.
- DIAGNOSTIC tento stav dovoluje vymazat jednotlivé stavy proměnných z PLC a může být také načten logbook, paměť může být vymazána, může být zvolen warm/cold restart. Pouze v tomto režimu může být paměť vymazána.
- BOOT v tomto režimu může být zapsán do Compact Flash nový operační systém.

Start PLC vždy probíhá od režimu BOOT. Pokud je tento režim a všechny požadavky na bezproblémový start splněny, přechází do režimu DIAGNOSTIC, pokud vše je OK přechází se do režimu SERVICE. Pokud je vše v pořádku přechází se do režimu RUN.

6.5 Paměť PLC

Každý automat může obsahovat různé typy paměti. Paměť můžeme rozdělit na RAM a ROM. Do těchto dvou částí pamětí může uživatel zasahovat.

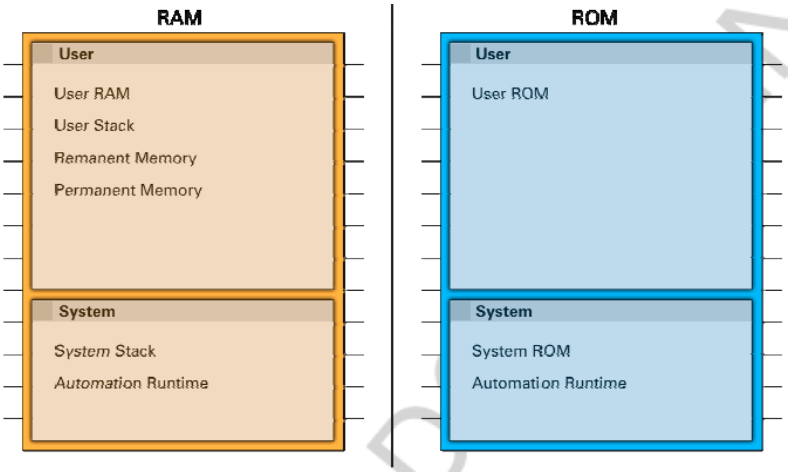
RAM

Paměť RAM je přizpůsobena požadavkům na rychlé čtení a zápis, obsahuje potřebná data pro běh programu. Data jsou po výpadku napětí z paměti vymazána (DRAM). Paměť RAM lze dále rozdělit do dvou skupin

- DRAM paměť je paměť, která se používá pro ukládání softwarových objektů stejně tak jako hodnot proměnných, tabulek, apod. Vyznačuje se rychlým přístupem. Tato část paměti je po výpadku napětí vymazána, protože není zálohovaná baterií.
- SRAM je statickou pamětí RAM. Tato paměť je zálohovaná baterií a při studeném restartu je obsah paměti vymazán. Do této části paměti se také ukládají permanentní a remanentní data. Rozdíl mezi remanentní a permanentní datovou oblastí v paměti je ten, že při studeném restartu je obsah remanentní paměti vymazán. Obsah permanentní paměti zůstává nezměněn.

ROM

Paměť ROM je vyvinuta pro dlouhodobé uložení dat. Data nejsou ztracena i po výpadku napájení. Pro paměť ROM se používá Compact Flash. Zde je uložen operační systém automatu a také zde mohou být uložena uživatelská data.



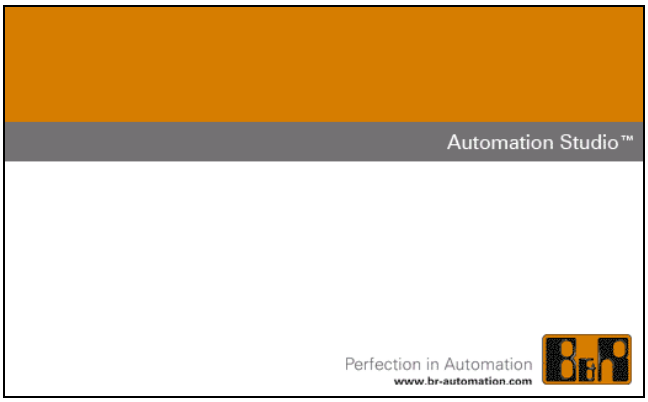
Obr.6.17: Rozdělení paměti.

Všechny data jsou ukládána za běhu programu do paměti DRAM, protože přístup do této části paměti je rychlejší. V případě SRAM slouží jako „sklad“ pro remanentní data a datové objekty. Remanentní data jsou data, která musí být uložena, když dojde k výpadku napětí nebo když systém je restartován (warm restart). SYSROM a USRROM je umístěna v paměti Compact Flash. Paměťová oblast REMMEM a USRRAM jsou umístěna v SRAM.

Během výpadku napětí jsou remanentní data z DRAM umístěna do paměti SRAM s určitým omezením. Není dovoleno vytvářet pole větší struktury než 4095 prvků. Maximální velikost paměti proměnných pro každý task nebo globální paměti proměnných je v projektu omezena na 32kB pro systém SG3 a 64kB pro systémy SG4.

6.6 B&R Automation Studio

B&R Automation Software zahrnuje všechny softwarové balíky nutné pro konfiguraci, programování, diagnostiku a všechny operace v B&R Control Systems, B&R Motion Systems a B&R Panel Systems.



Obr. 6.18: B&R Automation Studio .

B&R Automation Studio

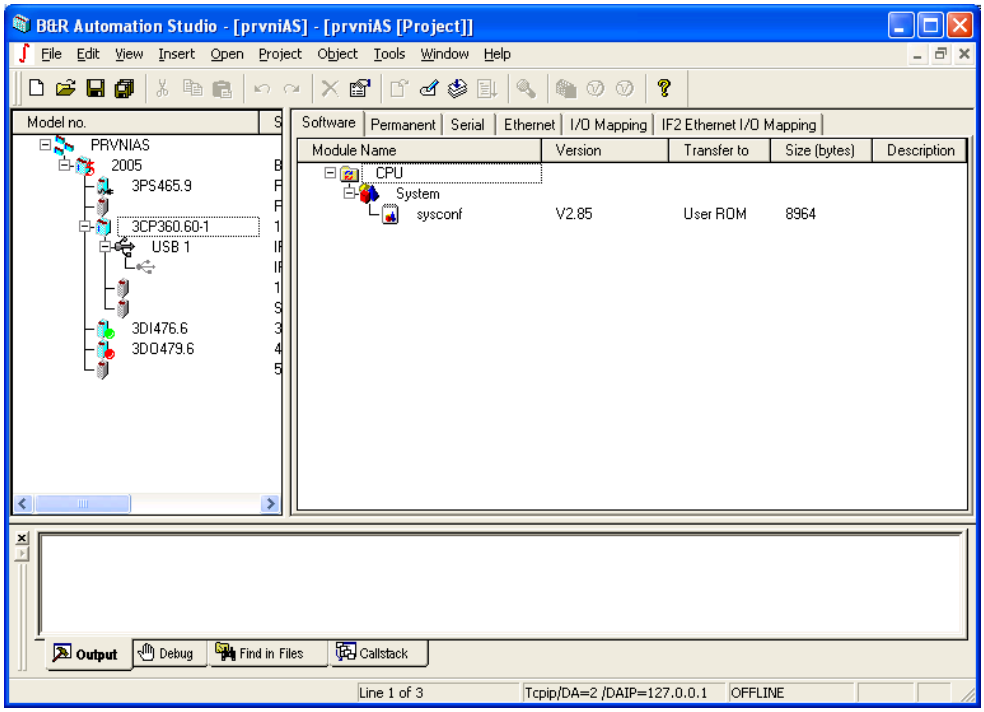
Automation Studio™ - základní sada - slouží k jednoduché konfiguraci a programování automatizačních úloh. Grafická konfigurace hardwaru a softwaru nabízí jasný přehled o projektu. Proměnné a procesní data jsou adresovány symbolicky. Cílový hardware je systémem automaticky rozpoznán a podporován.

V programu Automation Studio™ je integrováno široké spektrum standardních funkčních bloků - sahají od jednoduchých logických a matematických operací až ke komunikačním protokolům a složitým řídicím algoritmům. Pomocí správce knihoven lze vytvářet a spravovat vlastní funkční bloky.

Síť Automation Net™ umožňuje snadné uvádění aplikací do provozu. K připojení slouží RS232, CAN, TCP/IP nebo modem. Výhody přinášejí i monitor procesních proměnných s funkcí "force" a nástroj pro sledování a vyhodnocování proměnných v čase (tracer). K dalším výhodám patří i ladicí program pro zdrojovou úroveň s body přerušení (debugger), možnostmi vstupování do procedur i jejich provádění v jednom kroku, provádění jednoho cyklu, sledování linky (online test programových součástí - line coverage) a chybový žurnál pomáhající při hledání chyb.

Výhody B&R Automation Studia:

- Prostředí zahrnuje možnost použití pro všechny druhy aplikací.
- Jednoduchá integrace a komunikace mezi nástroji.
- Jednoduchá správa prostředí.
- Jeden dodavatel hardwaru a softwaru.
- Je zaručena kompatibilita při použití jednotlivých nástrojů.



Obr. 6.19: B&R Automation Studio.

Operační systém **B&R Automation Runtime™** poskytuje programům vytvořeným v Automation Studiu™ na všech cílových systémech jednotné provozní prostředí. To umožňuje jednotné programování a provoz na libovolném zařízení.

Systém Automation Runtime™ zajistí, aby byly všechny úlohy zpracovány v definovaném časovém rastru, a pracuje tak jako nastavitelný, deterministický, víceúlohový systém reálného času.

Velké projekty je možné rozdělit do malých podprojektů. Tento postup zvyšuje modularitu a usnadňuje údržbu projektu.

Nabízí tyto výhody:

- Zajišťuje nejvyšší možný výkon hardwaru.
- Běží na všech B&R zařízeních.
- Zajišťuje cyklický běh programu.
- Podporuje všechny programovací jazyky IEC 61131-3 a C.
- Rozsáhlá knihovna funkcí podle IEC 61131-3 stejně tak jako rozšířená B&R Automation library.
- Integrací prostředí B&R Automation Runtime do prostředí Automation NET se získá přístup do všech sítí prostřednictvím funkcí konfigurovaných v B&R Automation Studio.

Komunikace s Automation NET

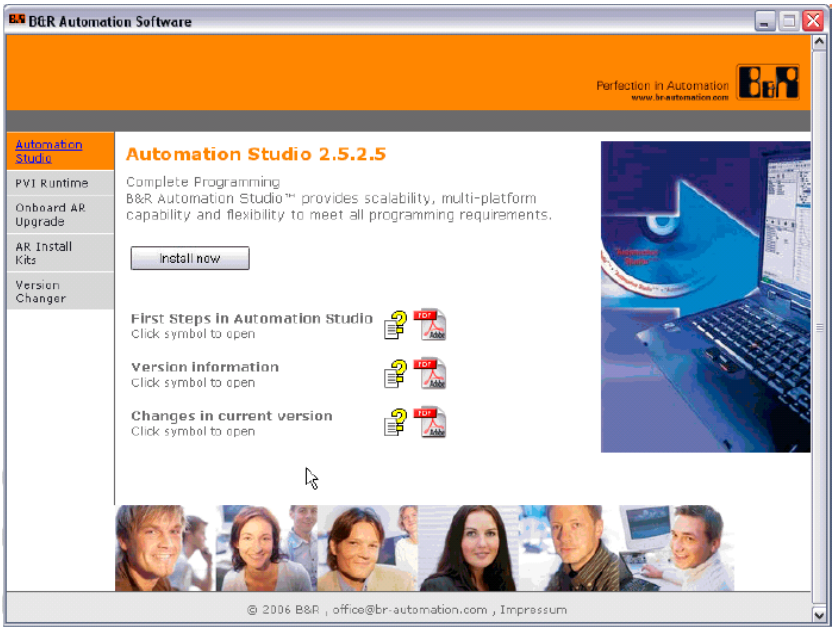
Komunikační platformu Automation Net™ používají všechny součásti systému B&R AutomationSoftware™. Všechny stanice komunikující v síti si mohou vyměňovat a zpracovávat programové objekty a procesní data nezávisle na cílovém systému, používaném přenosovém médiu a protokolu.

Platforma Automation Net™ využívá standardní rozhraní Windows a umožňuje systémům B&R propojení s kancelářskými nástroji.

Instalace B&R Automation Software

Po vložení CD do mechaniky Vašeho PC se spustí průvodce instalací. Ale hned v první nabídce se nabízí několik možností - software pro vývoj vašeho programu:

- Automation Studio: Tento software je nezbytný pro vývoj aplikací pro systémy B&R.
- PVI Runtime: software, který umožňuje komunikovat mezi PC a PLC a naopak.
- Onboard AR upgrade - software, který umožňuje upgrade operačních systémů.
- AR Install Kits – pomocí tohoto software můžete aktualizovat operační systém v průmyslových PC.
- Version changer - tento software je užitečný pokud máte na jednom PC nainstalováno více verzí Automation Studia. Software se Vás při spuštění zeptá, ve které verzi AS chcete Automation Studio spustit např. ve verzi 2.5.2.21 apod.



Obr. 6.20: Instalace Automation Studia.

6.7 Operační systém – deterministický multitasking a jeho popis, taskové třídy, tasky, cyklus programu

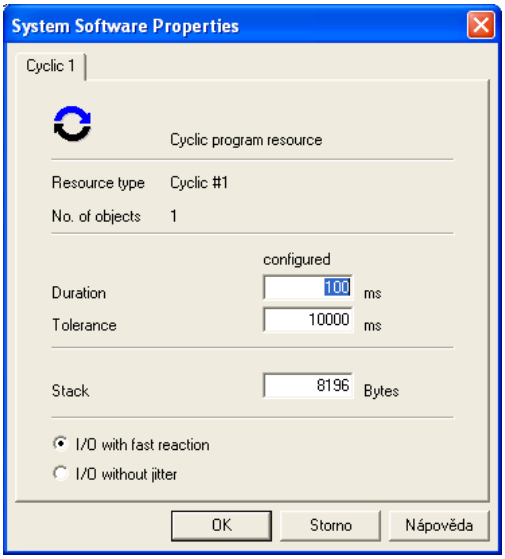
Operační systém u systému Bernecker Rainer je charakteristický tím, že využívá deterministický multitasking. Operační systém je napsán ve VxWorks (pro Intel).

V programu se rozeznávají tasky a taskové třídy. Task je nezávislá část programu, která vykonává nějakou úlohu v řídicím systému – programu. Dělením programu do jednotlivých tříd umožňuje programátorovi napsat si každou úlohu – tasku v jiném jazyku a také zvyšují přehlednost programu, protože každý task může vykonávat určitou část řídicí aplikace. Každý task má svoji inicializační a cyklickou část.

Tyto tasky se vkládávají do taskových tříd, které se spouští v definovaných časových okamžicích, kde spouštění těchto taskových tříd a v nich vnořených tasků řídí operační systém - Scheduler.

Cyklické taskové třídy

Tyto taskové třídy jsou vykonávány cyklicky. Doba je definována uživatelem a je monitorována systém managerem. U systémů SG3 a SG4 je různý počet těchto taskových tříd. Každá tasková třída má také svoji prioritu, kde nejvyšší prioritu má tasková třída Cyclic #1. Taskové třídy mají defaultně přidělen interval spouštění - to znamená, že když založíte cyklickou třídu Cyclic #1, která je defaultně nastavena na volání každých 10ms, že tento čas již nelze měnit. Tento čas je jako parametr této taskové třídy a lze změnit. **Tato tasková třída Cyclic #1 má nejvyšší prioritu, a proto kdyby jste potřebovali např. volat tuto taskovou třídu každých 100ms místo 10ms je možné změnit tento čas!!!!**



Obr. 6.21: Nastavení časového intervalu pro task.

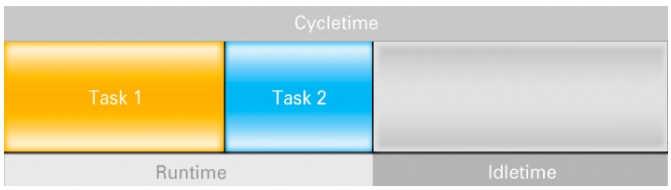
Všechny taskové třídy jsou vykonávány tak rychle, jak je to možné. Tabulka 6.7. ukazuje přehled interval cyklů, podle kterých je možné volit jednotlivé taskové úlohy.

Tab. 6.7 : Jednotlivé tasky a doby pro CP 360.

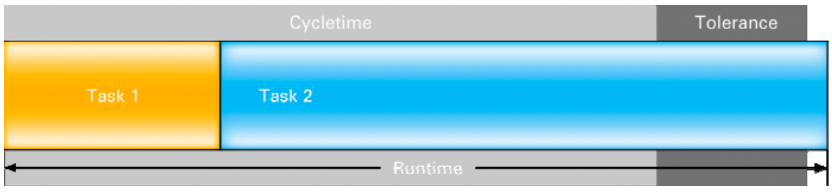
Task class	Doba cyklu (Cycle Time) [ms]	Tolerance [ms]
Cyclic #1	10	10
Cyclic #2	20	20
Cyclic #3	50	50
Cyclic #4	100	100
Cyclic #5	200	100
Cyclic #6	500	100
Cyclic #7	1000	100
Cyclic #8	10	3000

Jak bylo výše uvedeno, každá tasková třída může obsahovat několik tasků např. Task1 a Task2. Každá tasková třída má nějaký cyklus volání – cycle time. Podmínkou je to, aby se tasky Task1 a Task2 vykonaly v době označovanou jako cycle time. Z obrázku je vidět, že tasky by neměly vyplňovat veškerý čas - cycle time a měl by zde zbýt ideálně i nějaký zbytkový čas označovaný jako Idletime.

V tomto čase probíhá např. komunikace s PC apod. Tento čas cycle time by neměl být těmito tasky Task1 a Task2 překročen. Proto ještě existuje za Cycle time jistá tolerance, avšak jestli tasky překročí i tuto toleranci, automat přejde do stopu a bude v režimu DIAGNOSTIC.



Obr. 6.22: Vykonávání tasků.

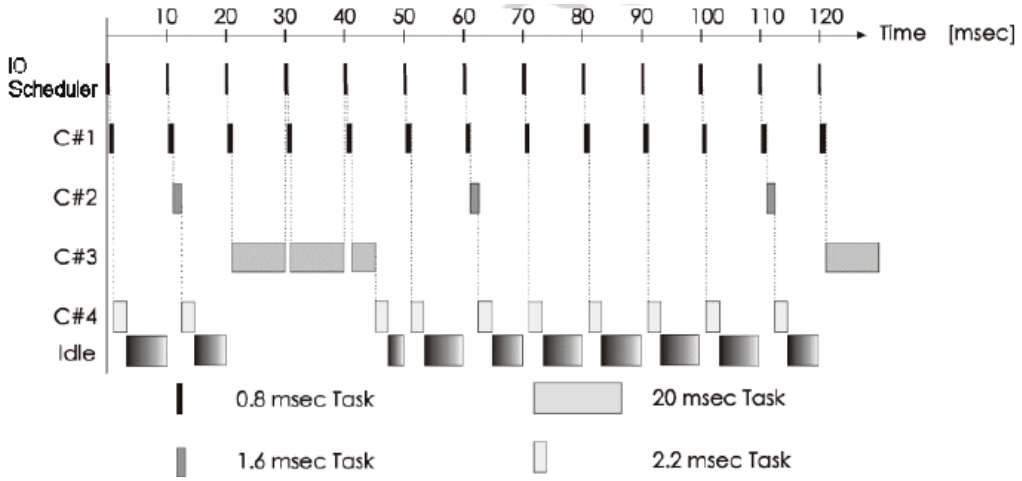


Obr. 6.23: Překročení tolerance.

Deterministický multitasking

B&R systém využívá deterministický multitasking tzn. že každá tasková třída se provádí v pravidelných časových intervalech.

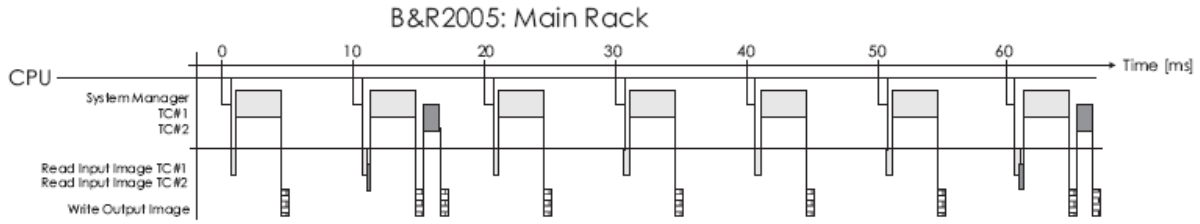
Deterministický multitasking si vysvětlíme na dalším obrázku. Při startu PLC tj. při přechodu do režimu RUN se nejprve vykonávají inicializační části tasků. Tento čas není monitorován Schedulerem tedy dá se říct, že může probíhat „libovolně dlouho“. Po provedení inicializace tasků, si operační systém rozdělí volání těchto taskových tříd tak, aby se procesor v jedné chvíli nevyužíval na 100% a ve zbývajícím čase se CPU využíval z 2%. Nejprve se spustí tasková třída Cyclic #1 (jak je vidět na následujícím obrázku). Tato tasková třída se vykoná a v dalším časovém intervalu se spustí opět tasková třída C#1 a po vykonání také C#2. V dalším kroku se vykoná opět C#1 a spustí se po vykonání tasková třída C#3., která se ale nevykoná v jednom časovém intervalu a proto operační systém rozdělí toto vykonávání do několika časových okamžiků. Tím, že se také tyto taskové třídy uváděly v činnost postupně, se také dosáhne optimálního využití CPU. Z obrázku 6.24 je vidět, že C#3 je vykonávána přerušovaně, protože ji přerušuje tasková třída C#1, která má vyšší prioritu. Z obrázku 6.24 je také vidět, že např. tasková třída C#2 je volána každých 50ms a po vykonání ještě také zbývá dostatečné množství času IdleTime, který slouží např. pro komunikaci s PC. Tasková třída C#4 má nejnižší prioritu a z obrázku je vidět, že se nemusí v některých časových okamžicích provést.



Obr. 6.24: Časový průběh vykonávání tasků.

I/O Image

Nechť jsou definovány dvě taskové třídy TC#1 a TC#2. Image vstupů jsou čteny odděleně pro každou taskovou třídu TC#1 a TC#2 zvlášť. Z obrázku 6.25 je vidět, že se nejprve provede načtení vstupů pro taskovou třídu TC#1, vykoná se tasková třída TC#1 a na konci tasku 1 se zapíší výstupy. V dalším 10ms časovém intervalu se opět nejprve vykoná tasková třída TC#1, vykoná se načtení vstupů, vykoná se tasková třída TC#1 a zapíší se výstupy a protože ještě zbývá čas do uplynutí časového 10ms intervalu, vykoná se také tasková třída TC#2. Opět se nejprve vytvoří image vstupů vykoná se tasková třída TC#2 a zapíše se image výstupů. Z obrázku je vidět, že se tasková třída TC#2 nevykoná vždy, protože je tato tasková třída volaná v 50ms intervalech. Viz tabulka 6.7.

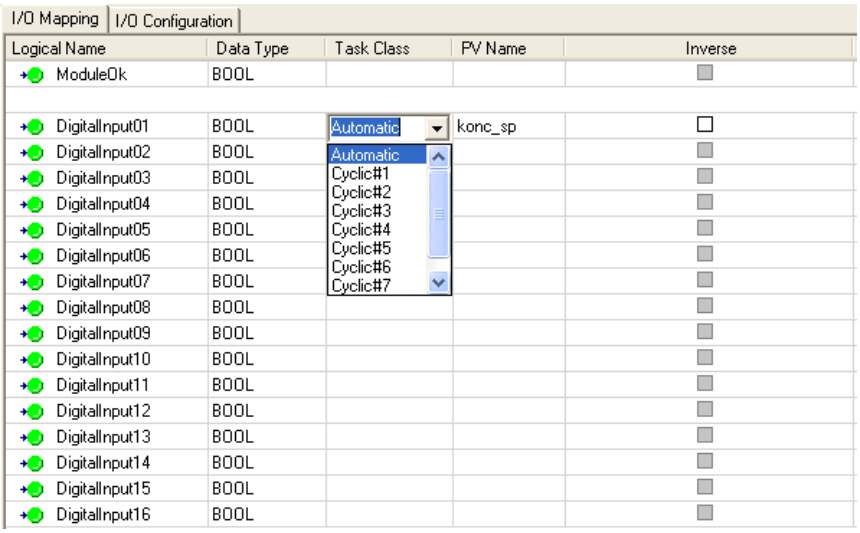


Obr. 6.25: Vykonání I/O Image.

Výše uvedené tvrzení o vykonání image I/O je nutné ještě trochu upřesnit, protože záleží jakou platformu využíváte, jestli CPU je Motorola nebo Intel.

Čtení vstupů

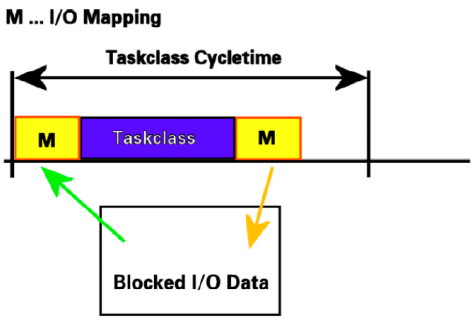
Pokud je CPU jako Motorola tak tento CPU resp. jeho OS čte vstupy všechny najednou. U procesorů řady Intel je možné si nastavit, kdy chci abych si přečetl tyto vstupy, jak ukazuje následující obrázek 6.26. Když nastavíte u digitálního vstupu, že se tento vstup má aktualizovat automaticky → Automatic, tak Automation Studio si samo určí, kdy tento vstup bude načítat tj. provádět jeho image. Když ale např. tomuto vstupu nadefinujete, že má tento vstup vyčítat např.jenom při vykonávání taskové třídy C#1, musíte nastavit Cyclic#1. Doporučuje se ale, aby se zanechávalo nastavení Automatic, avšak pro časové kritické aplikace je vhodné nastavit, kdy se má načíst image vstupu.



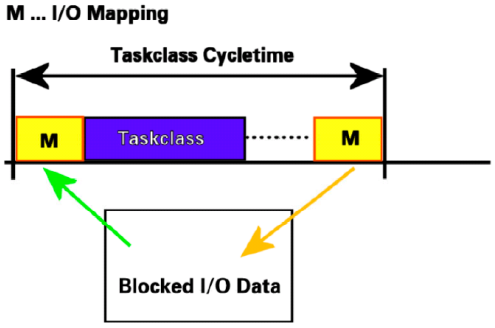
Obr. 6.26: Nastavení čtení vstupu.

Zápis výstupů

Výstupy se mohou opět zapisovat dvěma způsoby. Když využíváte platformu Motorola a Intel, tak se výstupy zapisují ihned po vykonání dané taskové třídy jak ukazuje obr. 6.27. Pokud využíváte platformu Intel spolu s pohony (frekvenční měniče) tak zde je důležité, aby se aktualizace výstupů prováděly vždy na konci cyklu - cycletime. Tento způsob zápisu až na konci cycle time je možný pouze u procesorů řady Intel obr. 6.28!!!



Obr. 6.27: Zápis výstupů.



Obr. 6.28: Zápis výstupů na konci cyklu.

Ostatní taskové třídy

EXC- Exception task classes – EXC task je task, pomocí kterého se ošetřují vzniklé chyby. Exception task může být vytvořen pro každou chybu, kterou podporuje expansion handler. Expansion handler je rozšířený operační systém uložený v systém ROM. Exception task má nejvyšší prioritu v systému. Rozeznávají se následující chyby uvedené v tabulce (tab.6.8). Pokud nastane chyba uvedená v tabulce 6.8 , přejde PLC do režimu Service. Exception task může být používán pouze, jestli je exception task class konfigurována. Pouze jeden exception task může být konfigurován pro jednu výjimku uvedenou v tabulce 6.8 .

Např. nastane-li chyba v taskové třídě Cyclic #1 je task 1 přerušen, protože EXC má vyšší prioritu. Vykoná se tento EXC task a pokud je chyba ošetřena provede se návrat na adresu, kde nastalo přerušení. EXC task sdílí globální datovou oblast s taskem Cyclic #1. To je výhodné, protože zkracuje přístupovou dobu, aby si mohl vyměnit data s Cyclic #1.

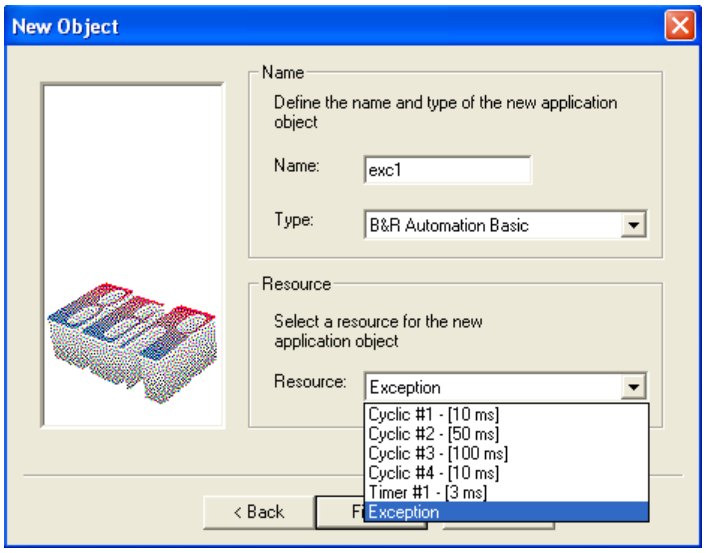
Nepoužívejte tento exception task pro běžné chyby, které Vám mohou nastat v programu. Tímto stylem rozhodně není napsán správně program, protože program musí řešit i nějaké chyby, které se mohou vyskytnout neočekávaně a řešením je využití Exception tasků.

Tab 6.8.: Vyjímky, které mohou nastat za běhu programu na PCC.

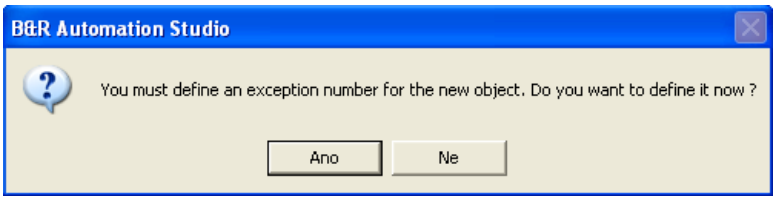
Výjimka číslo	**Popis**	**B&R System**
2	Bus error	Procesor
3	Address error	
4	Neplatná instrukce	
5	Dělení nulou	
6	Přetečení rozsahu	
7	Null pointer	
8	Porušení práv	
10	Neimplementovaný příkaz	

24	Nežádoucí přerušení	
128	I/O Exception	I/O sběrnice
144	Nedodržení doby cyklu	Doba cykly (Cycle Time)
145	Nedodržení maximální doby cyklu	
146	TC vstup nedodržení odezvy na vstupu (TC - task class)	
147	TC nedodržení odezvy na výstupu	
160	HSTC nedodržení maximálního času pro task	
161	HSTC I/O nedodržení času pro cyklus	
162	Systém cycle time violation	
170	CAN I/O exception	Remote I/O
176	RIO HSTC – cycle time violation	
177	RIO exception when ready / error	

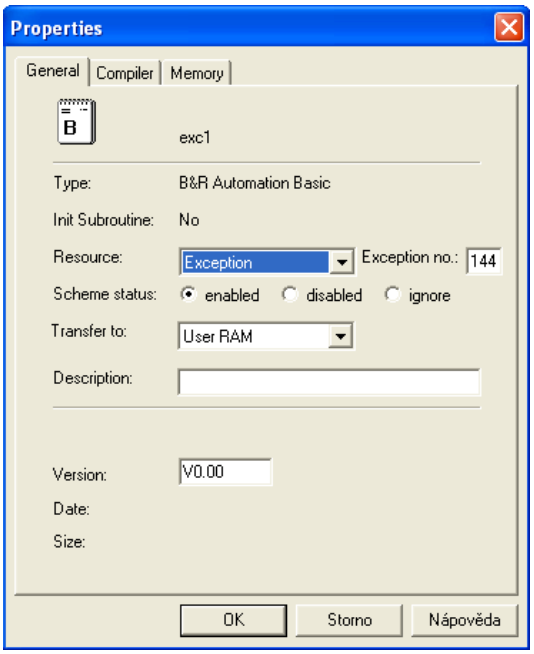
Příklad vytvoření task v taskové třídě Exception:



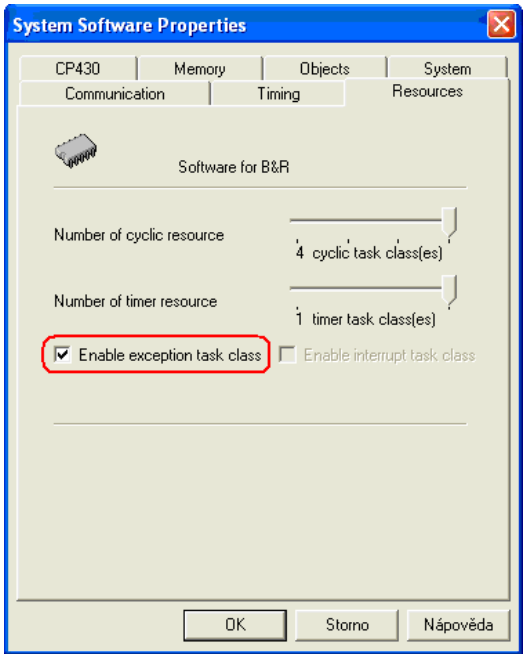
Obr. 6.29: Vytvoření Exception tasku.



Obr.6.30 : Upozornění na specifikování druhu chyby.



Obr.6.31: Nastavení čísla výjimky.



Obr.6.32: Povolení vykonání exception třídy.

6.8 Programovací jazyky, datové typy

Programovat je možné ve všech standardních jazycích: Automation Basic, ANSI-C, jazyky podle IEC 61131-3 - kontaktní schémata (LAD), jazyk mnemokódů (IL), strukturovaný text (ST), sekvenční vývojové diagramy (SFC). Dále je k dispozici editor datových modulů a editor datových typů.

- B&R Automation Basic
- ANSI C
- IEC61131 Ladder Diagram (LD)
- IEC61131 Instruction List (IL)
- IEC61131 Structured Text (ST)
- IEC61131 Sequential Function Chart (SFC)

B&R Automation Basic - tento programovací jazyk vyvinula firma Bernecker Rainer pro své automaty. Jak již sám název napovídá, program se vytváří v Basicu.

ANSI C - využitím jazyka C můžete také vytvářet jednotlivé úlohy pro PLC.

Dále se může také program vytvářet v jazycích, které definuje norma IEC. Tyto jazyky jsou Ladder Diagram (LD), Instruction List (IL), Structured Text (ST), Sequential Function Chart (SFC).

Tab. 6.9: Programovací jazyky.

Operace	**Programovací jazyky**					
	LAD	**SFC**	**IL**	**ST**	**AB**	**C**
Logické operace	ANO	ANO	ANO	ANO	ANO	ANO
Aritmetické operace			ANO	ANO	ANO	ANO

Přechody (decisions)	ANO		ANO	ANO	ANO	ANO
Smyčky				ANO	ANO	ANO
Step sequencers				ANO	ANO	ANO
Dynamické proměnné				(ANO)	ANO	ANO
Funkční bloky	ANO	ANO	ANO	ANO	ANO	ANO

Datové typy

Data, která mají být použita v programu, musí být přiřazena určitému datovému typu. Přiřazením typů dat se předepíše jejich velikost a struktura jejich bitů. Rozdílné typy dat musí být známé, neboť některé instrukce vyžadují určité typy dat.

Tab. 6.10a: Datové typy.

Typ	Velikost	Rozsah	Použití
BOOL	1	0..1	Např. Digitální I/O
DINT	32	-2 147 483 648.. 2 147 483 647	
INT	16	-32 768 .. 32767	Např. Analogové I/O
SINT	8	-128..127	
UDINT	32	0.. 4 294 967 295	
UINT	16	0..65 535	
USINT	8	0..255	
REAL	32	-3,4e38.. 3,4e38	

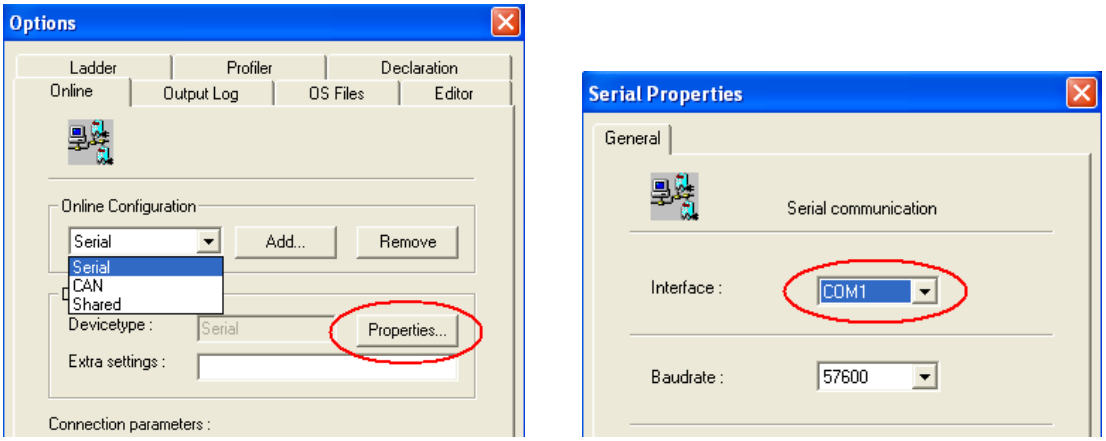
Tab. 6.10b: Datové typy.

Typ	Velikost	Rozsah	Použití
STRING	1-xx byte	2 znaků – 32767 znaků	"Text"
TIME	4 bytes	0..4 294 967 295 ms	Time differences
DATE_AND_TIME	4 bytes	Second since 1970	Date calculations

Příklad vytvoření nového projektu, upload hardware

Protože B&R systém může být tvořen různými moduly, je nejvýhodnější si tuto hardwarovou konfiguraci načíst do prostředí B&R Automation Studia.

Před zahájením uploadu hardwaru je také vhodné zkontrolovat, na který port PC je připojen kabel od PLC. Volbou v menu *Tools* → *Options* se provádí nastavení portu pro komunikaci a v okně *Options* v záložce *Online* se volí typ spojení např. Serial, kde kliknutím na tlačítko *Properties* se volí příslušný port PC, jak ukazuje obr.6.33.



Obr. 6.33: Nastavení komunikace mezi PLC a PC.

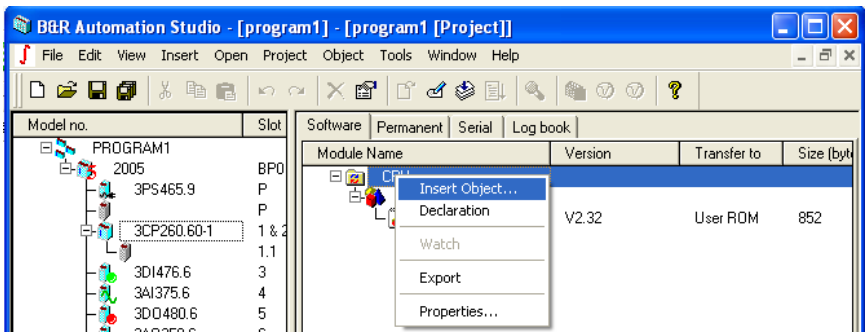
Vytvoření nového projektu je jednoduché *File* → *New project*. V okně *New Project Wizard* je nezbytné zatrhnout *Upload hardware from target*, jak ukazuje následující obrázek 6.34.

Obr.6.34: Upload hardware.

Vkládání tasků – inicializace, cyklická část

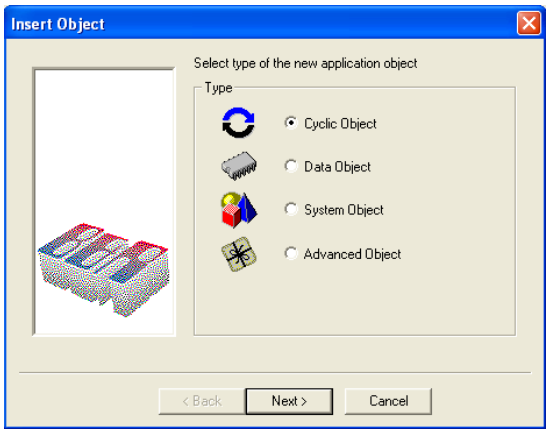
Jak již bylo výše uvedeno, program je členěn do jednotlivých tasků, kde každý task může mít odlišnou dobu vykonávání.

Vložení nového tasku se provede kliknutím pravým tlačítkem myši v pravém okně na *CPU* a v dialogovém okně se vybere *Insert Object*, čímž se spustí průvodce vytvořením tasku.

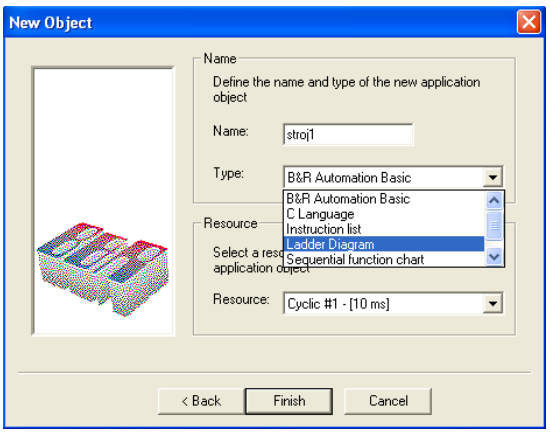


Obr.6.35: Vytvoření nového tasku.

Např. při vytváření cyklického tasku, volíme *Cyclic Object*, jak ukazuje obr.6.36 V následujícím okně *New Object* se vybere jazyk, ve kterém se bude psát program např. *Ladder Diagram*, pojmenuje se tento task např. stroj1 přiřadí se mu odpovídající doba volání např. v 10ms intervalech (obr.6.37.). Kliknutím na tlačítko *Finish* průvodce vytvoří tento task. Tím jste vložili task stroj1 do taskové třídy Cyclic#1, která je volána v pravidelných 10ms intervalech.

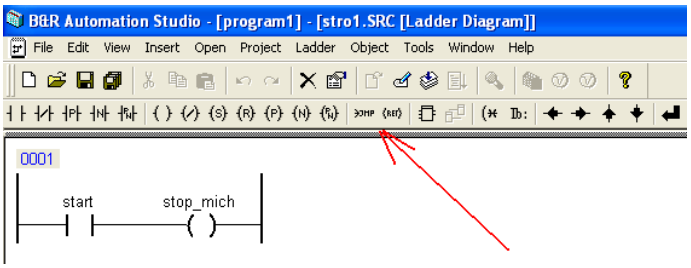


Obr.6.36:Vytvoření cyklického tasku.

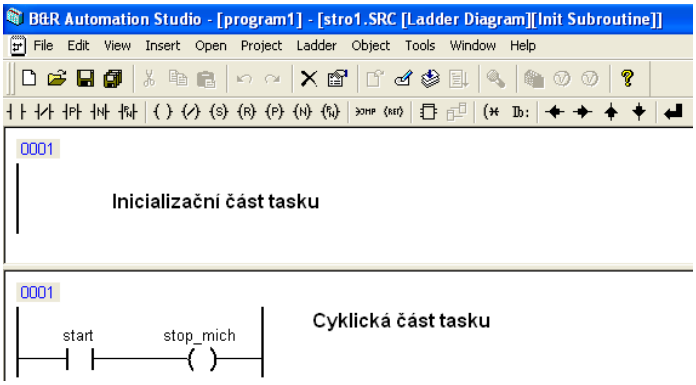


Obr.6.37:Volba jazyku, druhu tasku a pojmenování Tasku.

Každý task se skládá ze dvou částí, která se vykoná jenom jednou tj. inicializace, a části která se provádí cyklicky. Pokud je nutno provést něco jenom jednou tj. při inicializaci, otevřením příslušného tasku a kliknutím na lištu tam, kde ukazuje šipka na obrázku 6.38. a tahem dolů se objeví další okno, jak ukazuje následující obrázek 6.39. tzv. inicializační část tasku. Nebo výběrem v menu *View* → *Init Subroutine* se vyvolá inicializační část tasku.



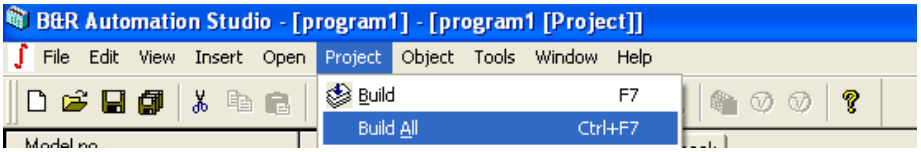
Obr.6.38: Otevření inicializace tasku.



Obr.6.39: Části tasku.

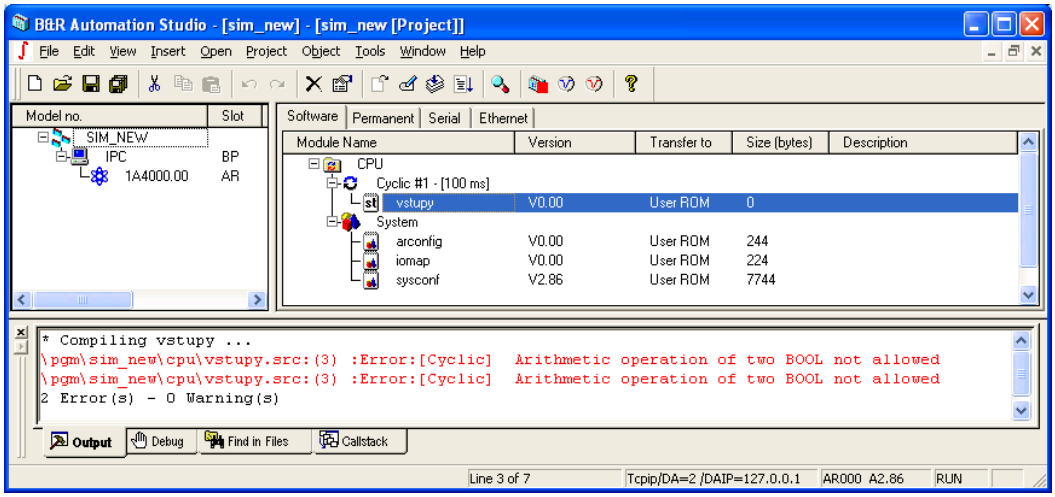
Download do PLC

Před zápisem programu do PLC je nutné ještě provést kompilaci programu. Kompilace se provede volbou příkazu *Project → Build All* nebo jenom pro překlad stačí příkaz Build v menu B&R Automation Studia.



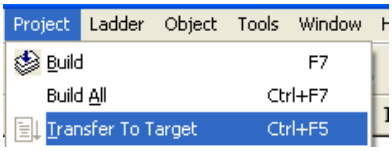
Obr.6.40: Spuštění kompilace programu.

Pokud se během překladu programu vyskytnou chyby, budou vypsány v dolní části obrazovky obr. 6.41. Zde vidíme celý soupis chyb a také je zde napsáno, ve kterém tasku k chybě došlo. Abych nemusel otevírat příslušný task, tak mi stačí stisknout na klávesnici klávesu F4, tím se dostanu do tasku, kde vznikla chyba. Tímto způsobem mohu procházet jednu chybu (varování) za druhou.



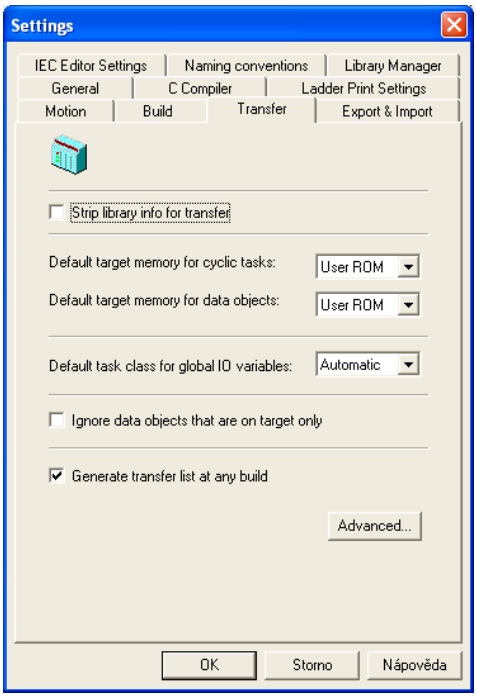
Obr. 6.41: Výpis chyb.

Pokud je vše správně, může se přistoupit k zápisu programu do PLC v menu *Project → Transfer To Target*, jak ukazuje následující obrázek a zahájí se zápis do PLC.

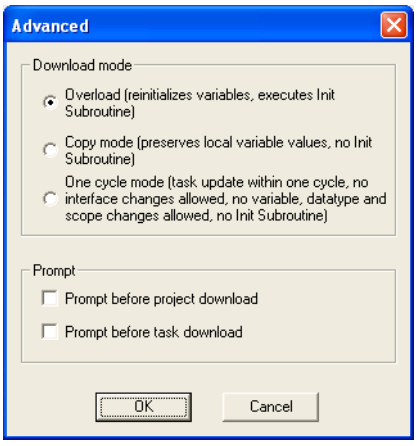


Obr.6.42 : Zápis programu do PLC.

Existují celkem tři druhy zápisu programu do PLC. Nastavení downloadu se provede výběrem v menu *Project → Settings…*, kde v okně Settings v záložce *Transfer* se zobrazí aktuální nastavení. Kliknutím na tlačítko *Advanced* je možné upřesnit typ zápisu do PLC.



Obr. 6.43: Nastavení kompilace programu.



Obr. 6.44: Nastavení zápisu do PLC.

- Overload - tento režim je nejčastější a používá se u procesorů Motorola a Intel. V tomto režimu je po downloadu do PLC vykonána také inicializace, dojde také k přepsání proměnných, protože kompilátor je při překladu může umístit na jinou adresu.
- Copy mode - když v tomto režimu se zapisuje program do PLC, tak se nevykonává inicializace. V prvním kroku se do PLC nahrává změněný task a až je kompletní, PLC zruší vykonávání „starého tasku“ a přejde se k vykonávání nového tasku s tím, že hodnoty proměnných, které již za běhu programu mají své hodnoty, nebudou znovu inicializovány.
- One Cycle Mode - tento režim je možný pouze, když se proměnné nezmění např. nevznikne nová proměnná, nová struktura, nezmění se datový typ proměnné apod. Opět i při tomto režimu zápisu se převezmou aktuální stavy proměnných.

6.8.1 Správný zápis programu do PLC

Zápis programu přes RS232, Ethernet, Routing

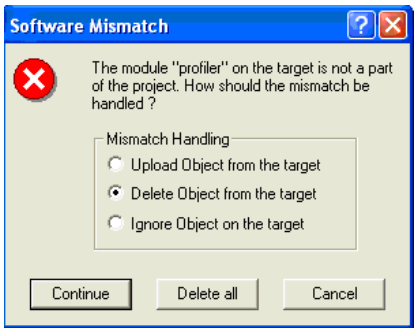
Před zahájením zápisu do PLC je nejvhodnější zastavit běh PLC kliknutím na ikonu  . Tím je běh programu v PLC zastaven. Pokud by jste vyžadovali opravdu bezproblémový start, je vhodné po zastavení běhu PLC také smazat paměť. *Project → Services →Clear Memory.*

Po zastavení běhu PLC, by se měl provést zápis do PLC s tím, že v průběhu zápisu programu do PLC můžete obdržet hlášení, že již nějaké objekty existují v paměti. Když PLC je v režimu, že nevykonává program je vhodné (pokud jste dříve již nesmazali paměť) vymazat stávající program. **Kdyby jste běh programu nezastavili tím, že PLC bude v režimu SERVICE, mohli by jste při této volbě, jak ukazuje následující obrázek, způsobit během downloadu do PLC chybu v paměti stávajícího programu, která by byla způsobena tím, že jste se potkali s místem, které se právě vykonávalo. V tomto případě by byla vhodnější volba Ignorovat stávající program.**

Program jste tímto úspěšně do PLC zapsali. Nyní je doporučené provést studený restart kliknutím na ikonu  . Tím způsobíte to, že se smaže statická RAM paměť. Nemusíte mít obavu, že smažete operační systém PLC, protože tento operační systém PLC je umístěn v jiné části paměti. Po znovu

obnovení činnosti PLC tj. přechod do režimu RUN, je v tuto chvíli program správně zapsán a také vykonáván.

Toto je jeden ze způsobů zápisu programu do PLC. Další možností jak zapsat program do PLC, je využít paměť Compact Flash anebo zápis programu pomocí routingu.

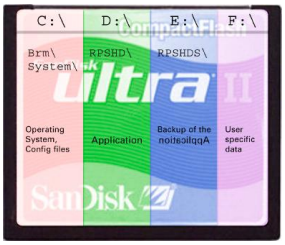


Obr. 6.45: Upozornění na různé verze programu v PLC.

Zápis programu na Compact Flash

Zapsat program na Compact Flash je možné pouze u systémů, které jsou vybaveny procesory Intel tj. SG4. Compact Flash je možné vygenerovat ve dvou variantách a to jako One partition systém nebo jako Three partition systém.

- One partition systém flash je vhodné vygenerovat, když pouze testujete program a víte, že program ještě nebude uveden do ostrého provozu. Systém vygeneruje flash, kterou rozdělí na dva oddíly, kde první oddíl C: bude obsahovat řídicí program a druhá část flash oddíl D: bude obsahovat uživatelská data např. soubory apod.
- Three partition systém v tomto případě je Compact Flash rozdělena do 3 a více částí první oddíl C: obsahuje operační systém a spouštěcí soubory, oddíl D: obsahuje aplikaci tj. řídicí program, oddíl E: obsahuje zálohovaný řídicí program, oddíl F: je vyhrazen uživateli pro ukládání souborů, pro přístup z FTP Serveru apod. Tento druh Compact Flash je vhodný zejména do ostrého provozu, protože napsaný program pro PLC je uložen v oddíle D:, kde z tohoto oddílu je také tento program překopírován do DRAM. Vždy je na začátku programu kontrolován tzv. Checksum, který kontroluje konzistenci dat. Kdyby nastala chyba v programu nebo spíš program by se při překopírování nějak poškodil, operační systém si začne kopírovat tento program z oddílu E:, protože zde je nakopírován při generování Compact Flash také. Tímto je zabezpečeno, že program je spuštěn vždy.

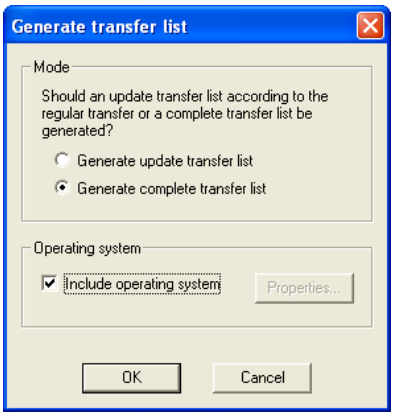


Obr. 6.46: Image Compact flash.

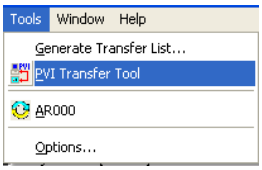
6.8.2 PVI Transfer Tool

PVI Transfer Tool může být buď jako součást Automation Studia anebo je k dostání ve verzi free. Program poskytuje veškeré potřebné servisní zásahy do PLC, jako je např. nastavení data a času, vygenerování Compact Flash, zjištění informací o PLC apod. Z Automation Studia se tento *PVI Transfer Tool* spouští příkazem *Tools → PVI Transfer Tool.* Před spuštěním PVI Transfer Toolu je

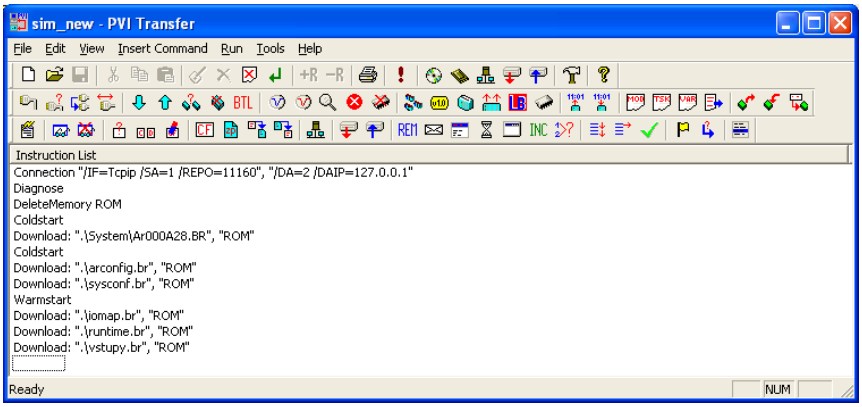
ještě vhodné nastavit tento software v menu příkazem *Tools* →*Generate Transfer List* obr. 6.48, protože je vhodné, aby se také na Compact Flash při generování této paměti zapsal i operační systém.



Obr. 6.47: Nastavení PVI Transfer Toolu pro vygenerování Transfer List s operačním systémem.



Obr. 6.48: Spuštění PVI Transfer Tool.



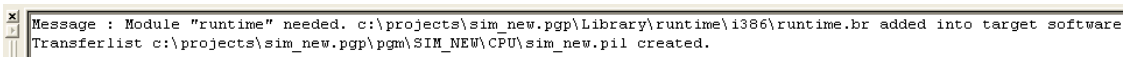
Obr. 6.49: Prostředí PVI Transfer.

Vygenerování programu pro Compact Flash se provádí pomocí příkazu Tools →Generate Compact Flash.

Záloha projektů.

Každý projekt je vhodné si archivovat, aby v případě, že se na PLC vyskytnou potíže, bylo možné nejjednodušší cestou obnovit zpět jeho činnost. Dále je také mít na paměti, že změnit program v PLC po např. 3 letech nemusí být tak jednoduché, protože se může změnit jak firmware pro tyto PLC tak samotné programovací prostředí. Automation Studio při svém překladu vygeneruje tzv. *.pil soubor, který je výsledek zkompilovaného projektu. Tento soubor je vhodné si uschovat, protože kdyby se na PLC vyskytla nějaká chyba, tak s použitím tohoto souboru by bylo možné obnovit opětovně jeho činnost. Další možností, jak si uchovat svůj projekt, je vytvořit si CD v prostředí PVI Transfer Tool.

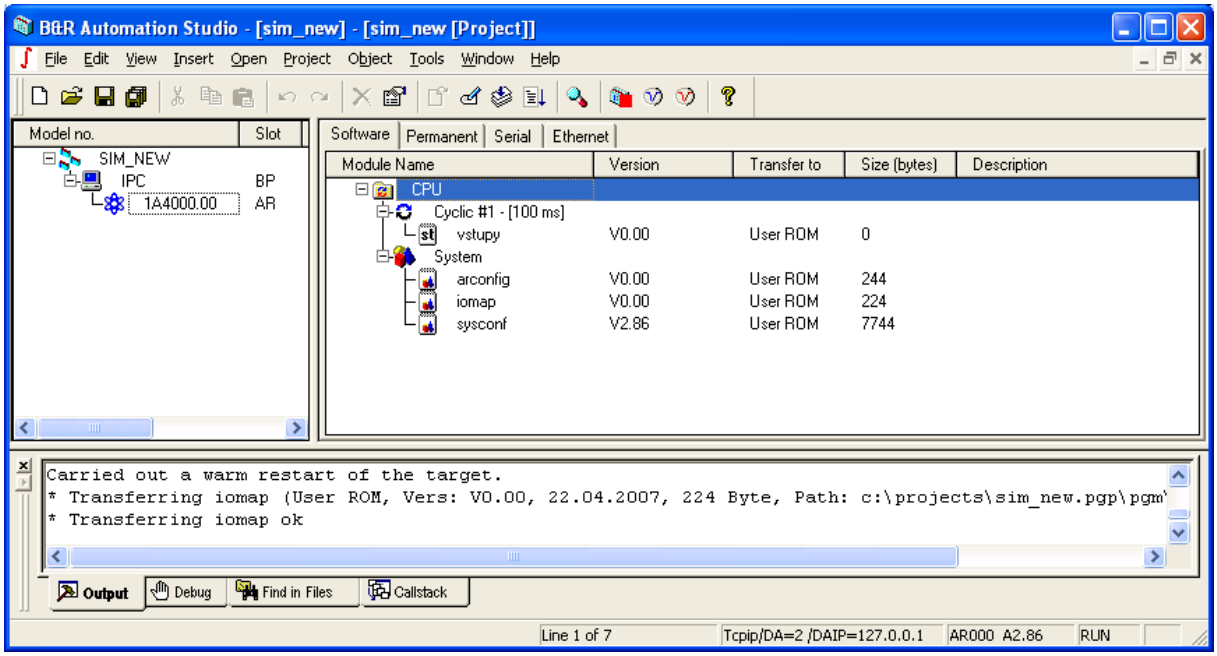
Mohli by jste si říct, proč využívat tento nástroj pro vypalování CD, když mohu využít běžné software pro vypalování CD. Tento software při vygenerování CD, vytvoří z CD plnohodnotnou zálohu, protože na jednom CD obdržíte jak zkompilovaný program, tak se zde také vypálí kompletní software PVI Transfer Tool. Takže pokud by jste potřebovali nutně SW, stačí nainstalovat si tento PVI Transfer Tool. Pak budete mít v rukou nástroj ve verzi SW, ve které jste program vytvořili, což může být v mnoha případech výhodou, protože software ve kterém jste napsali program se mohl již několikrát za tuto dobu změnit.



Obr.6.50: Informace o vygenerovaném *.pil souboru.

6.9 Využití simulátoru pro ladění programu bez nutnosti připojení k reálnému PLC

B&R Automation Studio nabízí také možnost využití emulátoru bez nutnosti připojení k reálnému programovatelnému automatu B&R. Neumožňuje testovat real time aplikace. Nevýhodou je také to, že neumožňuje testovat program pro konkrétní typ CPU, se kterým pracujete. Je možné z Vašeho projektu exportovat tasky, taskové třídy a poté je zase importovat do tohoto simulátoru a naopak. Tím získáte možnost otestovat si program.



Obr.6.51: PC Based Simulátor.

Shrnutí pojmů

V textu této kapitoly je popsaná technika programovatelných automatů Bernecker&Rainer, zejména produkty System 2003, System 2005, X20, X67 a SlotPLC. Tyto systémy lze rozšiřovat pomocí široké řady přídavných modulů. Základem procesorových jednotek jsou dva typy mikroprocesorů (Motorola a Intel) a podle nich lze rozdělit příslušné programovatelné automaty na dvě skupiny – dvě generace.

Programovatelné automaty Bernecker&Rainer mohou pracovat ve čtyřech provozních režimech – RUN, SERVICE, DIAGNOSTIC a BOOT.

Paměť programovatelných automatů Bernecker&Rainer má několik oblastí, ale lze ji rozdělit na dvě základní skupiny – RAM (přizpůsobena požadavkům na rychlé čtení a zápis, obsahuje potřebná data pro běh programu) a ROM (dlouhodobé uložení dat, data nejsou ztracena ani po výpadku napájení)

K programování se používá softwarový nástroj Automation Studio, které umožňuje tvorbu programů v několika jazycích - B&R Automation Basic, ANSI C, IEC61131 Ladder Diagram (LD), IEC61131 Instruction List (IL), IEC61131 Structured Text (ST) a IEC61131 Sequential Function Chart (SFC).

Operační systém uvedených programovatelných automatů se označuje jako deterministický multitasking, kde se program člení do tasků, odkud jsou volány funkce a funkční bloky. Jednotlivé tasky přísluší taskovým třídám, které zajišťují jejich spouštění.

B&R Automation Studio nabízí také možnost využití emulátoru bez nutnosti připojení k reálnému programovatelnému automatu B&R.

Kontrolní otázky

1. Jaké programovatelné automaty jsou v základní řadě firmy B&R?
2. Popište základní vlastnosti automaty B&R System 2003.
3. Popište základní vlastnosti automaty B&R X20.
4. Jaké jsou možnosti rozšíření programovatelných automatů B&R System 2003.
5. Jaké jsou možnosti rozšíření programovatelných automatů B&R X20.
6. Jaké existují generace procesorových jednotek programovatelných automatů B&R a čím se liší?
7. Jaké programovací jazyky lze využít v software Automation Studio pro psaní řídicích aplikací?
8. Popište deterministický multitasking.
9. Co je to tasková třída a jaké jsou jejich typy?
10. Co je to task?
11. Jak funguje čtení vstupů a zápis výstupů u programovatelných automatů B&R.

Další zdroje

6-1. B&R training document: The Basics of Automation Studio TM210.

6-2. B&R training document: Automation Studio Online Communication TM211.

6-3. B&R training document: Automation Runtime TM213.

6-4. B&R training document: The Service Technician on the Job TM220.

6-5. B&R training document: Automation Studio Diagnostics TM223.

6-6. B&R training document: Ladder diagram (LD) TM240.

6-7. B&R training document: Automation Basic (AB) TM247.

6-8. B&R training document: Memory Management and Data Storage TM250.

6-9. B&R training document: Automation Studio Libraries I TM260.

6-10. B&R training document: Closed Loop Control with LOOPCONR TM261.

6-11. B&R Automation Studio 2.5.2.21, Help.

6-12. B&R katalog X20 System.

6-13. B&R X20 System, User's Manual, version 1.20 (June 2006) Model No.: MAX20-ENG.

6-14. B&R Product Catalog 2007.

6-15. B&R Product Catalog System 2003.

6-16. B&R Product Catalog System 2005.

6-17. www.br-automation.com

Řešená úloha 6.1.

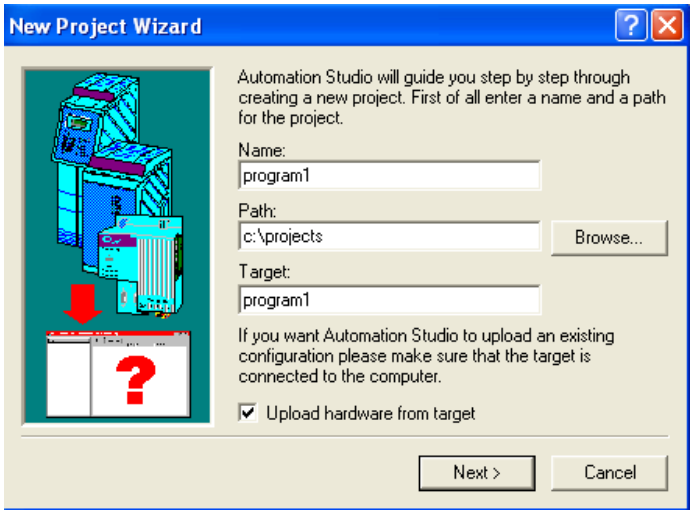
Seznámení se s vývojovým prostředím B&R Automation Studiem

Programovatelné automaty Bernecker Rainer poskytují možnost programátorovi stáhnout si hardwarovou konfiguraci do prostředí B&R Automation Studia. Toto cvičení obsahuje základní kroky v prostředí B&R Automation Studia. Kliknutím na ikonu  spustíte prostředí B&R Automation Studia

Příklad vytvoření nového projektu, upload hardware

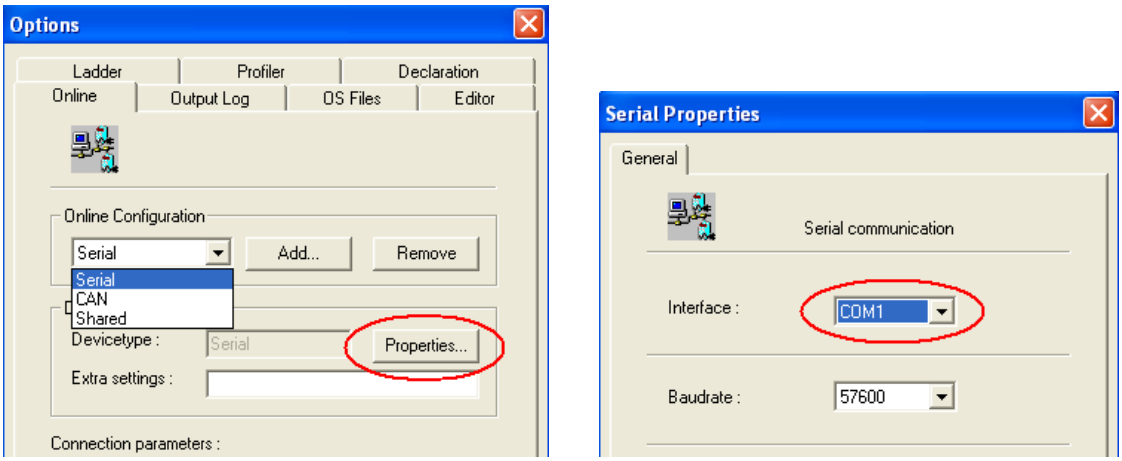
Protože B&R systém může být tvořen různými moduly, je nejvýhodnější si tuto hardwarovou konfiguraci načíst do prostředí B&R Automation Studia.

Vytvoření nového projektu je jednoduché *File* → *New project*. V okně *New Project Wizard* je nezbytné zatrhnout *Upload hardware from target*, jak ukazuje následující obrázek 6.52.



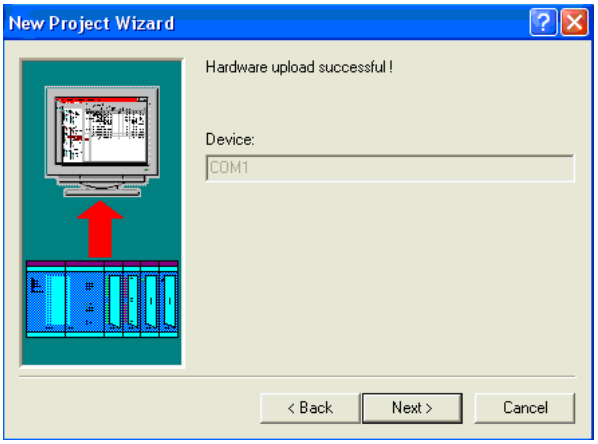
Obr.6.52 : Upload hardware.

Před zahájením uploadu hardwaru je také vhodné zkontrolovat, na který port PC je připojen kabel od PLC. Volbou v menu *Tools* → *Options* se provádí nastavení portu pro komunikaci a v okně *Options* v záložce *Online* se volí typ spojení např. Serial, kde kliknutím na tlačítko *Properties* se volí příslušný port PC, jak ukazuje obr.6.53.

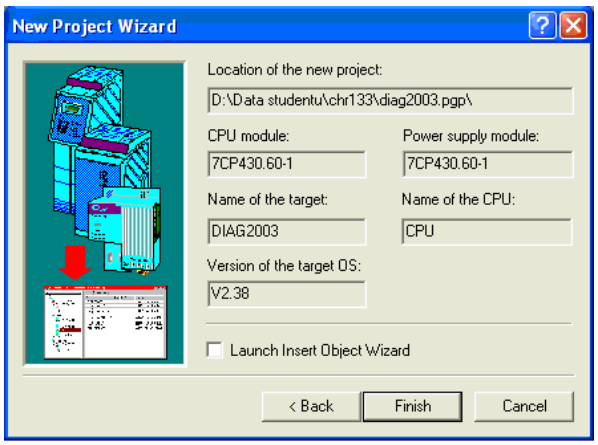


Obr.6.53: Nastavení komunikace mezi PLC a PC.

Pokud se upload hardwaru vykoná správně zobrazí se následující okno, které ukazuje obr. 6.54. Kliknutím na tlačítko Next přejdete do dalšího okna, kde získáte výpis aktuálního druhu CPU a verzi operačního systému.

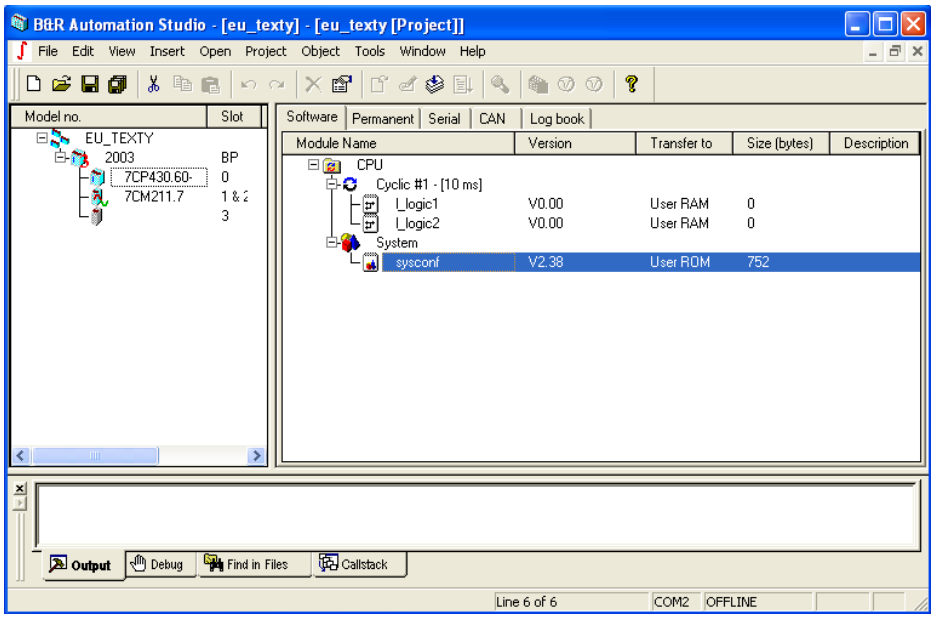


Obr. 6.54: Informace o dokončení uploadu hardwaru.



Obr. 6.55: Výpis druhu procesoru včetně typu operačního systému.

Po ukončení průvodce se zobrazí v levé části prostředí B&R Automation Studia hardwarová konfigurace.

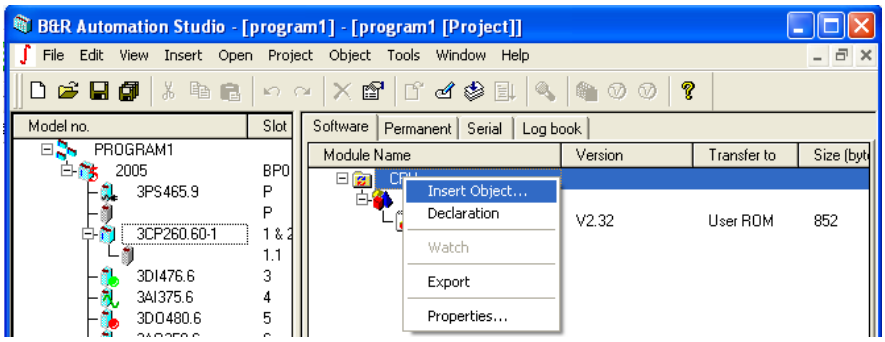


Obr. 6.56: B&R Automation Studio.

Vkládání tasků – inicializace, cyklická část

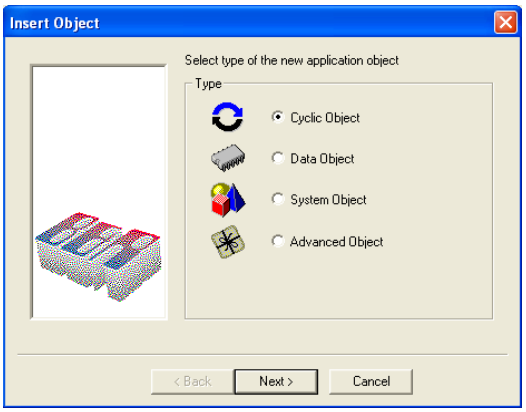
Jak bylo v přednášce uvedeno, program je členěn do jednotlivých tasků, kde každý task může mít odlišnou dobu vykonávání.

Vložení nového tasku se provede kliknutím pravým tlačítkem myši v pravém okně na *CPU* a v dialogovém okně se vybere *Insert Object*, čímž se spustí průvodce vytvořením tasku.

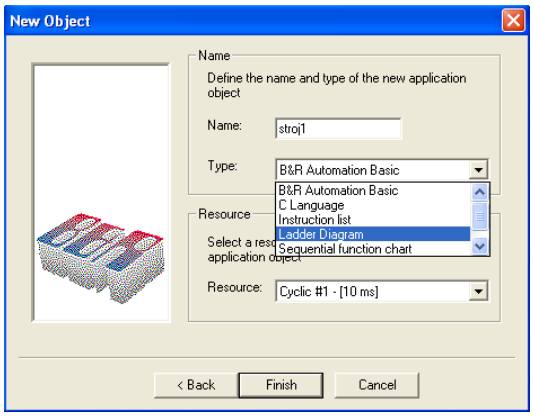


Obr.6.67 : Vytvoření nového tasku.

Např. při vytváření cyklického tasku, volíme *Cyclic Object*, jak ukazuje obr.6.6 V následujícím okně *New Object* se vybere jazyk, ve kterém se bude psát program např. *Ladder Diagram* a pojmenuje se tento task např. stroj1, přiřadí se mu odpovídající doba volání např. v 10ms intervalech (obr.6.68, 6.69). Kliknutím na tlačítko *Finish* průvodce vytvoří tento task.

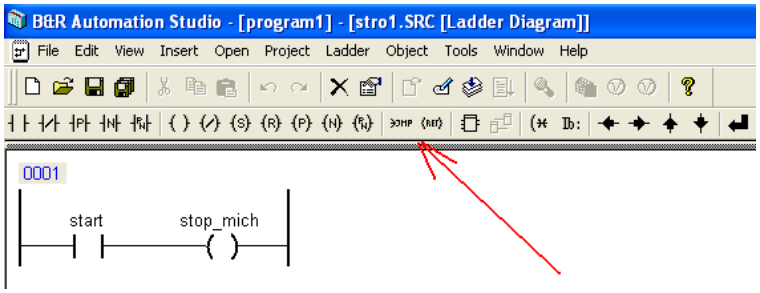


Obr.6.68 :Vytvoření cyklického tasku.

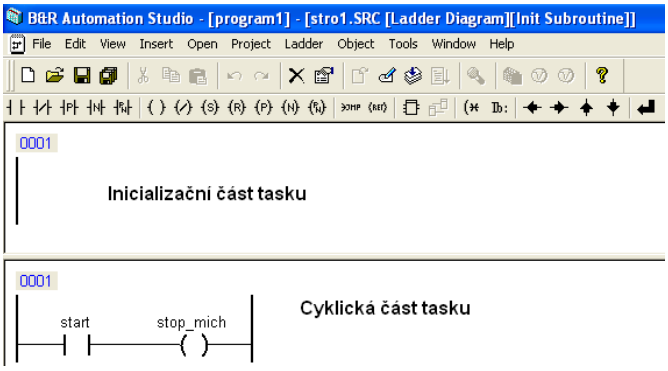


Obr.6.69 :Volba jazyku, druhu tasku a pojmenování Tasku.

Každý task se skládá ze dvou částí, která se vykoná jenom jednou tj. inicializace, a části která se provádí cyklicky. Pokud je nutno provést něco jenom jednou tj. při inicializaci, otevřením příslušného tasku a kliknutím na lištu tam, kde ukazuje šipka na obrázku 6.70 a tahem dolů se objeví další okno, jak ukazuje následující obrázek 6.70 tzv. inicializační část tasku. Nebo výběrem v menu *View → Init Subroutine* se vyvolá inicializační část tasku.



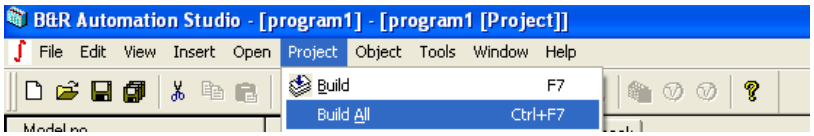
Obr. 6.70: Otevření inicializace tasku.



Obr. 6.71: Části tasku.

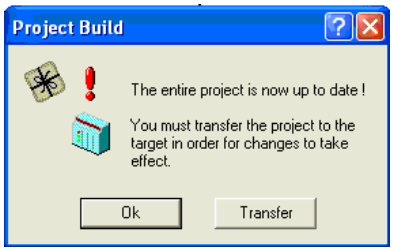
Download do PLC

Před zápisem programu do PLC je nutné ještě provést kompilaci programu. Kompilace se provede volbou příkazu *Project → Build All* v menu B&R Automation Studia (obr. 6.72).

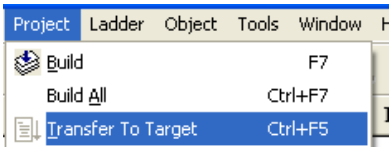


Obr.6.72 : Spuštění kompilace programu.

Pokud se kompilace programu vykoná správně a nejsou nalezeny žádné chyby v programu, budete vyzvání jestli chcete tento nový program také zapsat do automatu (obr.6.73). Další možností jak zapsat program do PLC je vybrat příkaz v menu Project → Transfer To Target, jak ukazuje následující obrázek a zahájí se zápis do PLC (obr. 6.74).



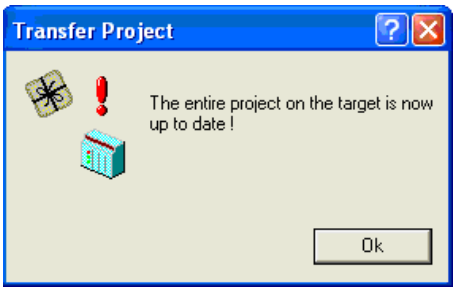
Obr. 6.73: Info o ukončení kompilace programu.



Obr.6.74: Zápis programu do PLC.

PLC přepněte do režimu STOP. Když se zahájí zápis do programu, software také porovnává obsah paměti, jestli nahrazujete program novým softwarem. Pokud se software změnil, objeví se hlášení podle obr. 6.75. Ponechte stávající nastavení na *Delete Object from the target* a klikněte na *Continue*. **Viz přednáška** Po dokončení zápisu budete informování o úspěšném zápisu a restartu PLC.

Obr. 6.75: Výmaz programu z PLC.



Obr. 6.76: Info o úspěšném zápisu.

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 6\pr_6_1.pgd.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 6\ anim1.avi.

DVD-ROM

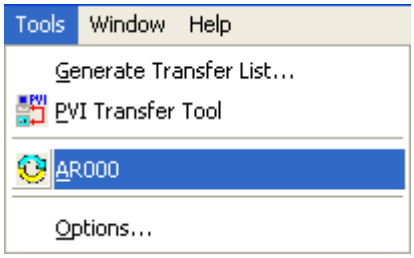
K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 6\ anim2.avi.

Řešená úloha 6.2.

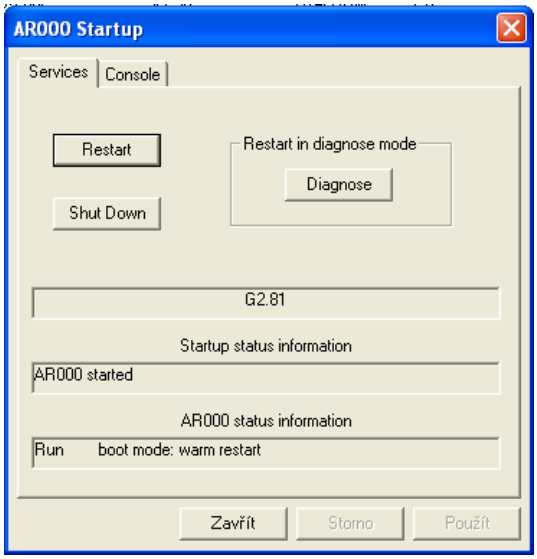
Vytvořte program, který bude vykonávat následující funkci y=a+b, kde a,b zvolte libovolné proměnné typu Boolean. Pro vyzkoušení programu využijte emulator v prostředí Automation Studia.

B&R Automation Studio nabízí také možnost využití emulátoru bez nutnosti připojení k reálnému programovatelnému automatu B&R. Neumožňuje testovat real time aplikace. Nevýhodou je také to, že neumožňuje testovat program na CPU připojeném k PC. Je možné z Vašeho projektu exportovat tasky, taskové třídy a poté je zase importovat do tohoto simulátoru a naopak. Tím získáte možnost otestovat si program.

Výběrem položky Tools v menu a výběrem AR2000 spustíte konfiguraci emulátoru, jak ukazují následující dva obrázky 6.77, 6.78.

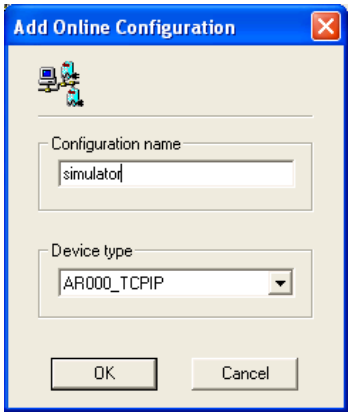


Obr.6.77: Spuštění konfigurace emulátoru.

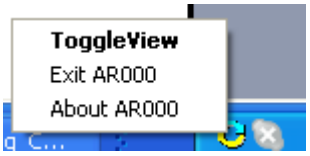


Obr.6.78: Spuštění emulátoru.

Poté je potřeba nastavit nové on-line spojení s automatem volbou v menu *Tools* → *Options*, kde kliknutím na tlačítko *Add*, pojmenováním nového spojení a volbou *Device type* jako AR000_TCPIP, je připraveno on-line spojení se simulátorem.

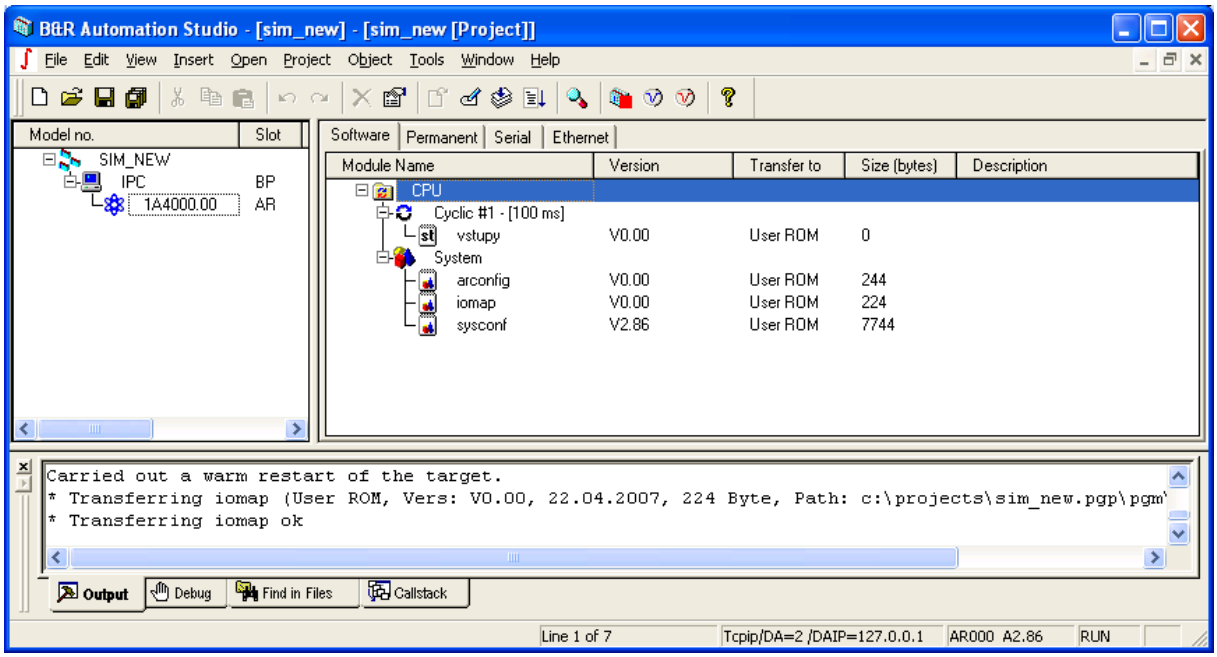


Obr.6.79: Nastavení spojení se simulátorem.



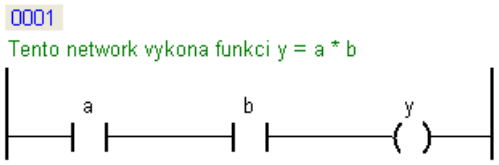
Obr.6.80: Zastavení běhu simulátoru.

Poté se musí vytvořit nový projekt, kde se musí zvolit možnost uploadovat hardware ze zařízení. Po vytvoření hardwarové konfigurace je možné již tento PCBasedSimulátor používat jako plnohodnotný automat. Můžete vytvářet cyklické objekty, sledovat chyby v Logbooku a využívat ladicí nástroje jako jsou PV Monitor, Watch a Trace, které jsou popsány v kapitole. Překlad programu a download do simulátoru se provádí stejně jako s připojeným reálným automatem. Zastavení běhu simulátoru se provede podle obrázku 6.80.



Obr.6.81: PC Based Simulátor.

Řešení v LAD



Řešení v Automation basic

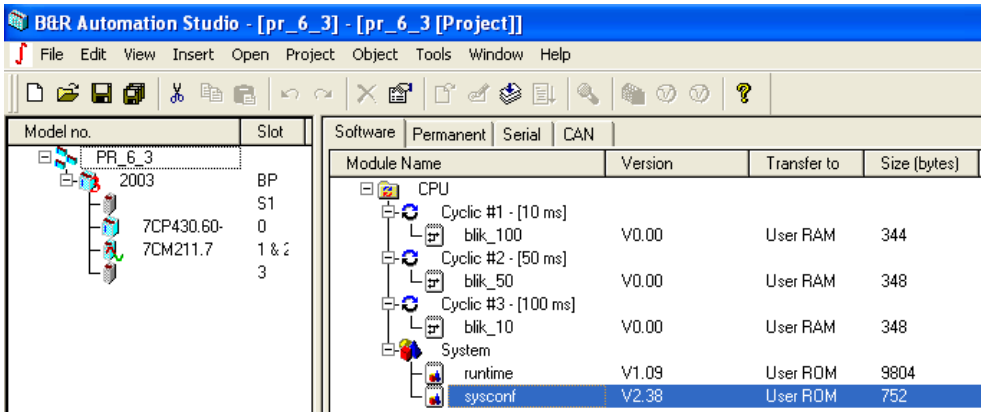
```
B&R Automation Basic : cycl_ab
0001 (* cyclic program *)
0002
0003 if a = 1 and b =1 then
0004 y = 1
0005
0006 else
0007 y = 0
0008
0009 endif
```

DVD-ROM

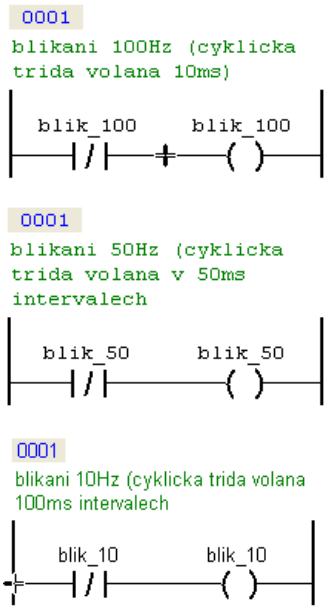
Řešený příklad naleznete na DVD: cvičení\cvičení 6\pr_6_2.pgd.zip

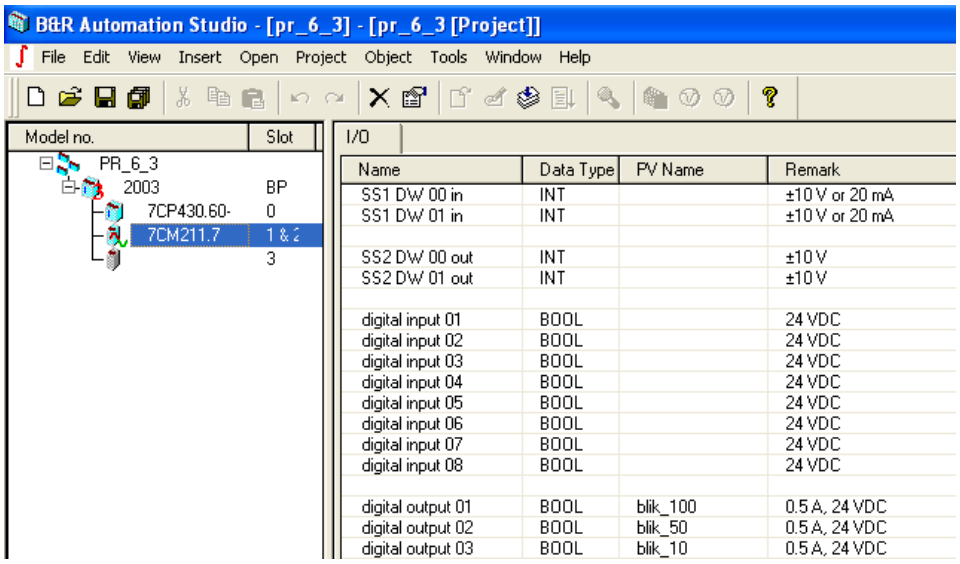
Řešená úloha 6.3.

Vytvořte program, který bude generovat signály o frekvenci 10Hz, 50Hz, 100Hz. Výstup DQ1 bude blikat s frekvencí 100Hz, výstup DQ2 bude blikat s frekvencí 50Hz, výstup DQ3 bude blikat s frekvencí 10Hz. Pro generování signálů využijte vlastnosti deterministického multitaskingu.



Řešení programu v LAD





DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 6\pr_6_3.pgd.zip

7. PROGRAMOVATELNÉ AUTOMATY BERNECKER-RAINER. PROGRAMOVÁNÍ, KOMUNIKACE

Čas ke studiu: 2 hodiny

Cíl

Kapitola popisuje základní možnosti programování automatů Bernecker&Rainer. Zabývá se zejména **knihovnami funkcí, binárními instrukcemi**, **čítači** a **časovači**.

Je zde rovněž popsaná tvorba a vlastnosti **funkcí** a **funkčních bloků**. Rovněž se zabývá **datovými objekty**, způsobem jejich tvorby a využití. Po prostudování těchto částí kapitoly, které se zabývají programováním, bude student schopen napsat jednoduchý program využívající logické funkce i zpracování analogových hodnot.

V závěrečných pasážích kapitoly se čtenář seznámí s modulem CM211 pro připojení vstupně/výstupních signálů, který bude využíván v praktických cvičeních v laboratoři.

Rovněž jsou zde uvedeny základní **komunikační možnosti** programovatelných automatů Bernecker&Rainer.

Výklad

7.1 Knihovny, Library Manager

Knihovny obsahují již předpřipravené funkční bloky, které již realizují základní druhy řízení např. komunikace, regulace apod. Nejvýznamnější je knihovna Standard, která obsahuje základní druhy časovačů a čítačů. Základní přehled dostupných knihoven v B&R Automation Studiu ukazuje tab. 7.1. Více o knihovnách se dozvíte v helpu Automation Studia.

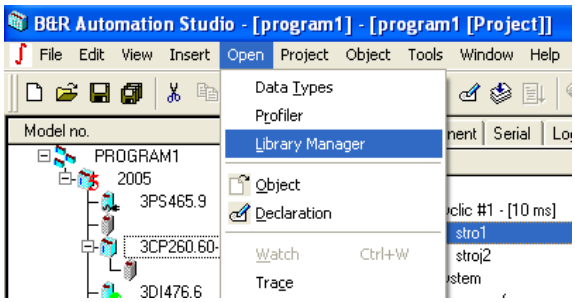
Tab. 7.1: Přehled knihoven.

Library	**Popis**
Acp10_mc	Knihovna pro práci s měniči ACOPOS
AsHW	Čtení informací ze zařízení
AsIMA	Knihovna pro komunikace s INA 2000
AsMath	Matematické funkce které nejsou v OPERATOR library
AsMem	Správa paměti
AsString	Práce s řetězci
AsTime	Podpora date and time funkcí
CAN_Lib	Pro práci se sběrnicí CAN
CANIO	Knihovna pro práci se sběrnicí CAN v systémech B&R2003
Commserv	Systémové rozšíření INAclnt library

Library	Popis
CONVERT	Konverzní funkce podle IEC61131-3
DataObj	Práce s datovými objekty
FileIO	Podpora práce se soubory
INAclnt	INA2000 client komunikace
IOConfig	Konfigurace HW na systémech 2003 – zapisuje i na subsloty
IOCtrl	Konfigurace HW
LoopConR	Zpětnovazební řízení počítá v REAL
LoopCont	Zpětnovazební řízení počítá v INTEGER
NET2000	NET2000 protocol
OPERATOR	Funkce podle IEC61131-3
SYS_lib	Systémové funkce
VNCServ	Vizualizace které běží na zařízeních SG4 a podporující VGA

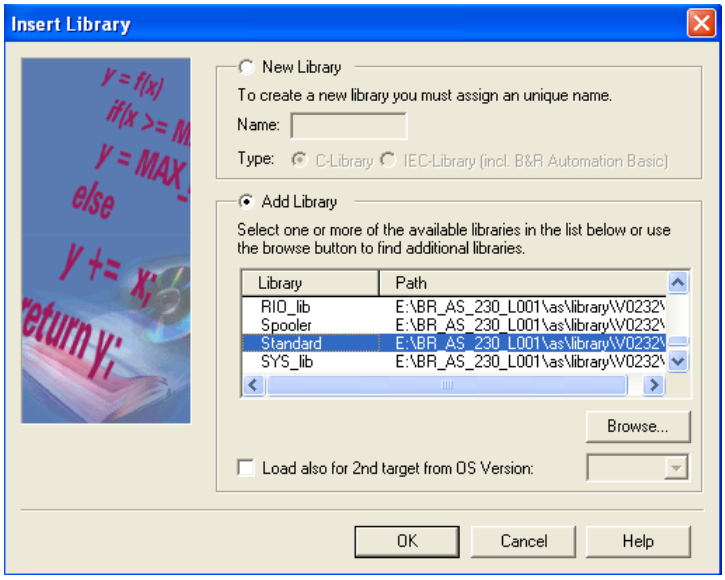
Library Manager obsahuje různé druhy knihoven, které obsahují již předpřipravené funkční bloky, pomocí nichž lze vytvářet různé druhy úloh od jednoduchých logických a matematických operací až po komunikace a komplexní řídicí algoritmy. Jednotlivé funkční bloky je možné vkládat do jakéhokoliv jazyka, který je podporován B&R Automation Studiem. Programátor si také může vytvářet a vkládat své funkční bloky do Library Manageru. Může si také vytvořit vlastní nápovědu k danému funkčnímu bloku.

Protože se často v různých programech vyskytují např. časovače, čítače a PID regulátory je dalším krokem při vytváření projektu si tyto FB - knihovny přidat do **Library Manageru**, který obsahuje aktuálně importované knihovny. *Library Manager* se otevře volbou příkazu v menu *Open → Library Manager* (obr. 7.1)



Obr.7.1 : Otevření Library Manageru.

Volbou *Insert → Library* se spustí průvodce přidáním nové knihovny. V okně *Insert Library* klikněme na *Add Library* a můžeme např. vybrat knihovna Standard (obr.7.2), která obsahuje časovače, čítače, apod. Stejným způsobem se může také přidat např. knihovna pro práci s regulátory - LoopCount.



Obr.7.2 : Přidání knihovny do Library Manageru.

7.2 Vytváření proměnných, binární instrukce

Aby si mohly jednotlivé události vyměňovat informace, je nutné v programu využívat proměnné. Proměnné lze rozdělit do základních dvou skupin tj. globální a lokální:

- Globální proměnné – stavy globálních je možné vyčítat ze všech částí tasků jednotlivých tříd.
- Lokální proměnné – stavy lokálních proměnných je možné vyčíst pouze z tasku, ve kterém jsou definovány .

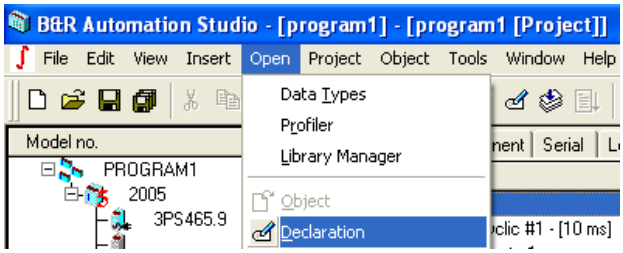
Dále se proměnné dělí na zálohované proměnné (remanent) a „nepřetržitě“ (permanent) zálohované datové paměti:

- Zálohované proměnné (Remanent variables) - pokud jsou proměnné při vytváření zvoleny jako zálohované remanent, budou stavy proměnných také známy i po warm restartu. Jestliže nastane výpadek napětí jsou zálohované paměti kopírovány do paměti SRAM. Po restartu PLC jsou opět obnoveny.
- Permanent proměnné (Permanent variables) jsou spolu se zálohovanými proměnnými kopírovány při restartu systému. Ačkoliv proměnné definované jako permanent jsou také uloženy navzdory cold restartu.

Zálohované a permanent variable by měly být jenom používány pokud je to nezbytně nutné.

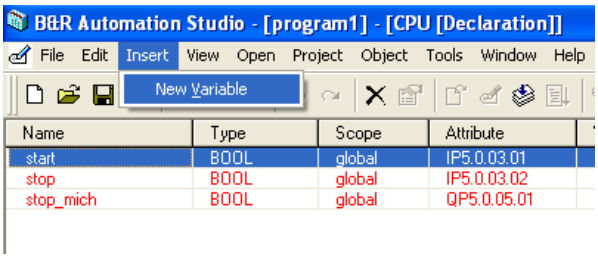
Vytváření proměnných, pojmenování digitálních a analogových I/O

Nové proměnné se vkládají v menu příkazem *Open →Declaration* nebo kliknutím na ikonu  (obr.7.3).



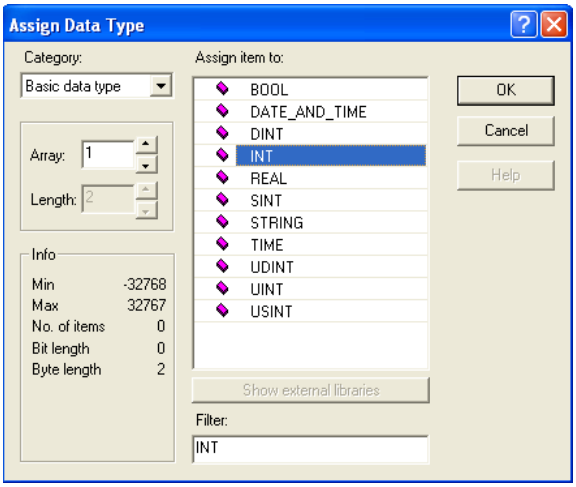
Obr.7.3 : Deklarace proměnných.

Kliknutím v menu *Insert* → *New Variable* (obr.7.4) se vloží nová proměnná.



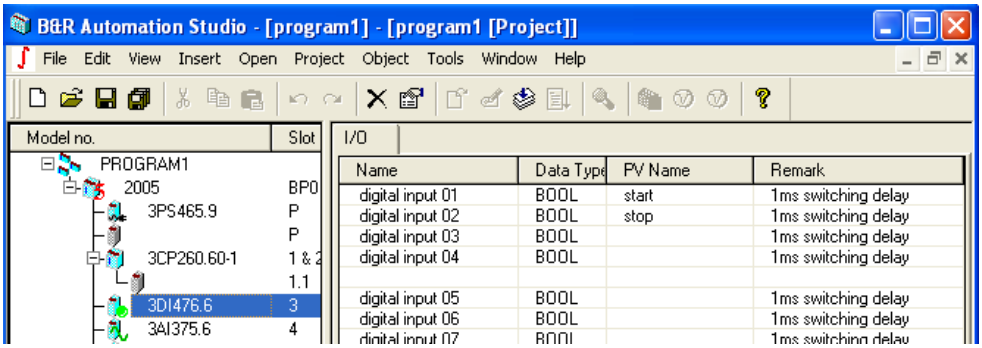
Obr.7.4 : Vytvoření nové proměnné.

Do nového řádku se napíše nová proměnná a v menu příkazem *Edit* → *Type…* se otevře okno *Assign Data Type*. Pomocí tohoto okna se vybere datový typ proměnné, jak ukazuje obr.7.5.



Obr. 7.5: Datové typy.

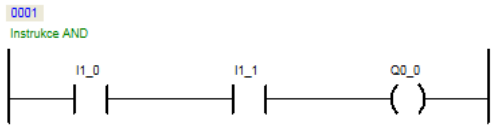
Když se provede upload hardwaru do PC, pak v levé části okna B&R Studia lze vidět jednotlivé moduly, které obsahuje systém. Přiřazením jména do sloupce *PV Name* k odpovídajícímu digitálnímu, analogovému vstupu/výstupu se provede pojmenování tohoto vstupu/výstupu např. start, stop, apod.



Obr. 7.6: Přiřazení proměnných pro digitální vstupy.

Binární instrukce

AND

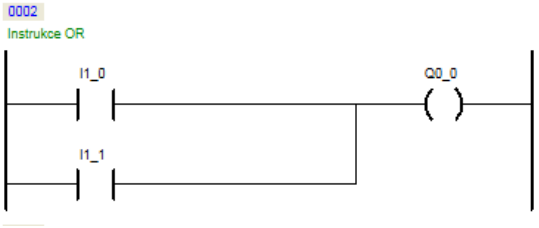


Obr. 7.7: Logický součin v LAD.

```
B&R Automation Basic : zakl_op
0001 (* cyclic program *)
0002 (* operace AND *)
0003 IF (I1_0=true) AND (I1_1=true)THEN
0004     Q0_0=1
0005 ELSE
0006     Q0_0=0
0007 ENDIF
```

Obr. 7.8: Logický součin v Automation Basicu.

OR

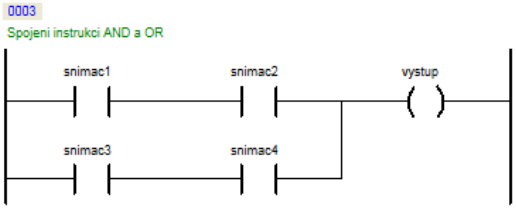


Obr. 7.9: Logický součet v LAD.

```
0009 (* operace OR *)
0010 IF (I1_0=true) OR (I1_1=true)THEN
0011     Q0_0=1
0012 ELSE
0013     Q0_0=0
0014 ENDIF
```

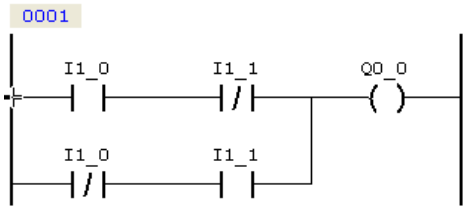
Obr. 7.10: Logický součet v Automation Basicu.

AND OR



Obr. 7.11: AND OR v LAD.

Exclusive OR

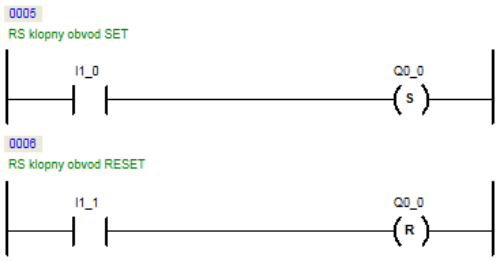


Obr. 7.12: XOR v LAD.

```
0009 (* operace XOR *)
0010 IF ((I1_0=true) AND (I1_1=false)) OR ((I1_0=false) AND (I1_1=true))THEN
0011     Q0_0=1
0012 ELSE
0013     Q0_0=0
0014 ENDIF
```

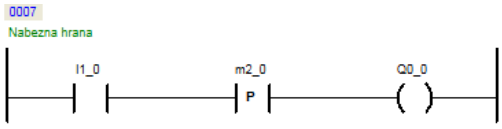
Obr. 7.13: XOR v Automation Basicu.

RS klopný obvod



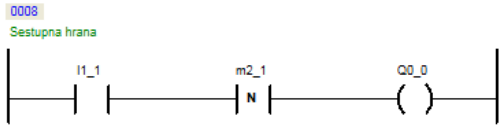
Obr. 7.14: RS klopný obvod v LAD.

Detekce náběžné hrany



Obr. 7.15: Detekce náběžné hrany v LAD.

Detekce sestupné hrany

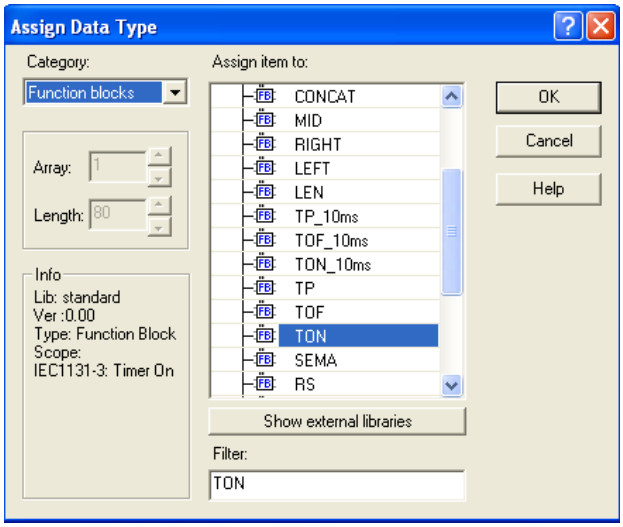


Obr. 7.16: Detekce sestupné hrany v LAD.

7.3 Časovače

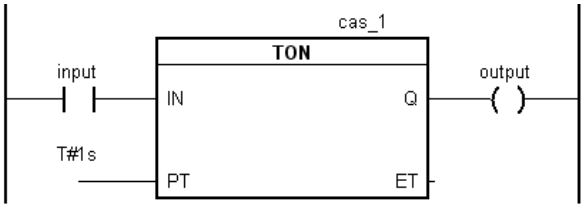
Časovače obsahuje knihovna Standard. Časovače jsou typu TON a TOF. Funkce těchto časovačů jsou stejné jako u PLC S7-300, takže je zde znovu nebudeme probírat, ale zaměříme se na vložení tohoto funkčního bloku do tasku.

Otevřete task a vyberte z menu *Insert* → *Function* a z okna *Assign Data Type* vyberte z knihovny *Standard* např. časovač TON a stiskněte OK (obr.7.17).



Obr.7.17: Výběr časovače TON.

Časovač je nutné bezprostředně po vložení do tasku pojmenovat např. cas_1. Časový interval zadejte např. pro 1s ve formátu T#1s.



Obr. 7.18: Časovač TON v LAD.

```
 (* call function block *)
TON_01.IN = input
TON_01.PT = T#1s
TON_01 FUB TON()

(* přiřazení výsledku do proměnné *)
output = TON_01.Q
```

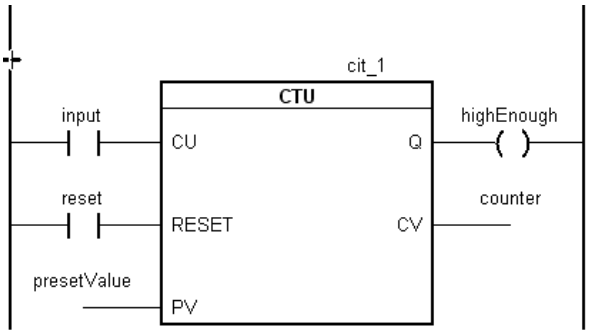
Obr. 7.19: Časovač TON v Automation Basicu.

7.4 Čítače

Čítače obsahuje knihovna Standard. Čítače mohou být opět vzestupné, sestupné, vzestupné/sestupné. Funkce těchto čítačů je stejná jako u PLC S7-300.

Pro více detailů lze využít nápovědu. Nejrychlejší vyvolání nápovědy je z Library Manager, výběr příslušného čítače a kliknutím na tlačítko Help.

Řešení programu v LAD



Obr. 7.20: Vzestupný čítač v LAD.

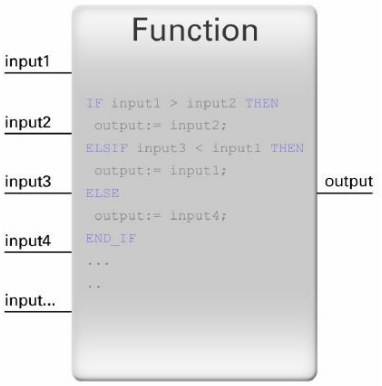
```
(* cyclic program *)

(* call function block *)
CTU_01.CU = input
CTU_01.RESET = reset
CTU_01.PV = presetValue
CTU_01 FUB CTU()

(* přiřazení výsledků do proměnných *)
counter = CTU_01.CV
highEnough = CTU_01.Q
```

Obr. 7.21: Vzestupný čítač v Automation Basicu.

7.5 Funkce, funkční bloky

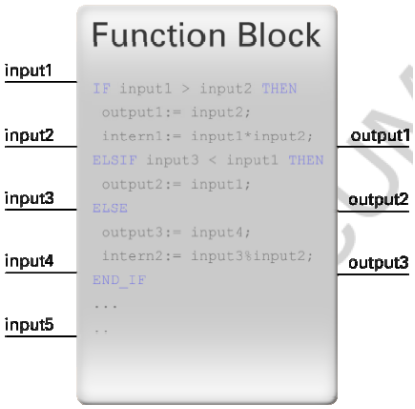
Funkce nebo funkční bloky umožňují programátorovi si zapsat svůj vlastní program např. výpočet úhlu apod. Většina nejčastěji používaných funkcí je již ale vytvořena a můžete si je do projektu pouze vložit. Všechny dostupné funkce, funkční bloky, které zahrnují knihovny obsahuje **Library Manager.**

Funkce slouží např. pro program, který bude vykonávat např. výpočet úhlu apod. Funkce vrací jednu hodnotu a funkce může být bez parametrů. Volání funkcí ve funkci není dovoleno.



Obr. 7.22: Příklad funkce v Automation Studiu.

Funkční blok je organizovaná jednotka, která navrací jeden nebo větší počet hodnot. Každý funkční blok může mít několik instancí. Ve funkčním bloku můžete volat také funkce.



Obr. 7.23: Příklad funkčního bloku v Automation Studiu.

Typy vstupně výstupních proměnných bloku popisuje tabulka 7.2.

Tab.7.2: Vstupně výstupní parametry funkce/funkčních bloků.

Scope	**Programovací jazyk**	**FBK – Funkční blok**	**FUN – funkce**	**Popis**
VAR_INPUT	Všechny	Ano	Ano	Vstupní parametr
VAR_OUTPUT	Všechny	Ano	Ano	Výstupní parametr
VAR	Všechny	Ano	Ano *	Statická vnitřní proměnná
VAR_DYNAMIC	B&R Automation Basic, ANSI-C	Ano	Ano *	Dynamická proměnná, která na očekává přiřazený pointer. Může být použita ve funkcích funkčních blocích v Automation Basic nebo ANSI C

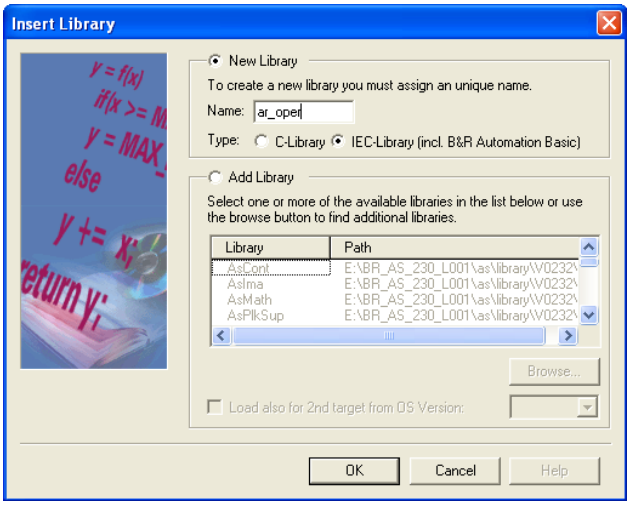
VAR_INPUT_DYNAMIC	B&R Automation Basic, ANSI-C	Ano	Ano	Ukazatel na proměnnou ve funkci/funkčním bloku použitelné pouze v Automation Basic nebo ANSI C

* kromě C funkcí

Příklad vytvoření vlastního funkčního bloku

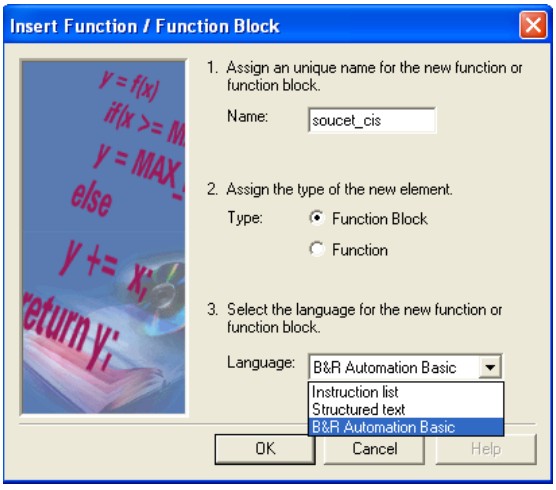
Protože je někdy zbytečné vypisovat neustále stejnou část programu, může si programátor vytvořit svůj vlastní funkční blok, který bude řešit určitou část programu. V této části je popsáno vytvoření jednoduchého funkčního bloku pro součet dvou proměnných.

Otevřením *Library Manageru* a příkazem v menu *Insert* → *Library* se spustí průvodce vytvořením nové knihovny. Je nutné zapsat nový název knihovny např. ar_oper a kliknout na *IEC-Library (incl. B&R Automation Basic)* obr.7.24.



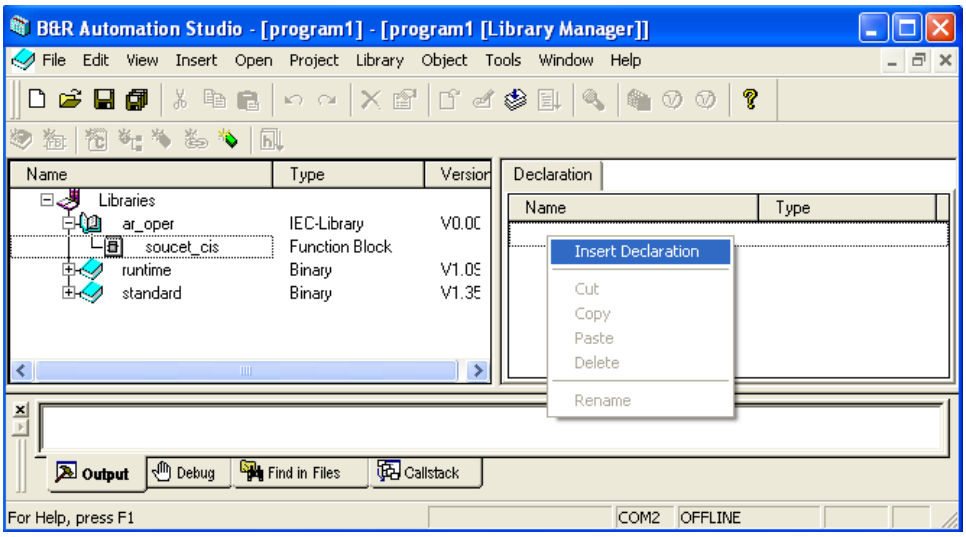
Obr.7.24: Vytvoření nové knihovny.

Volbou v menu *Insert* → *Function/Function Block* se spustí průvodce přidáním nové funkce/funkčního bloku. Dále se musí pojmenovat daný funkční blok/funkce. Vytváříme funkční blok a v posledním bodě se volí jazyk, ve které se bude psát tato funkce.



Obr.7.25: Průvodce vložením nového funkce/funkčního bloku.

Funkční blok bude mít dvě vstupní proměnné a jednu výstupní proměnnou tj. výsledek. Proměnné se vkládají kliknutím pravého tlačítka myši do pravé části obrazovky, jak ukazuje obr. 7.26.

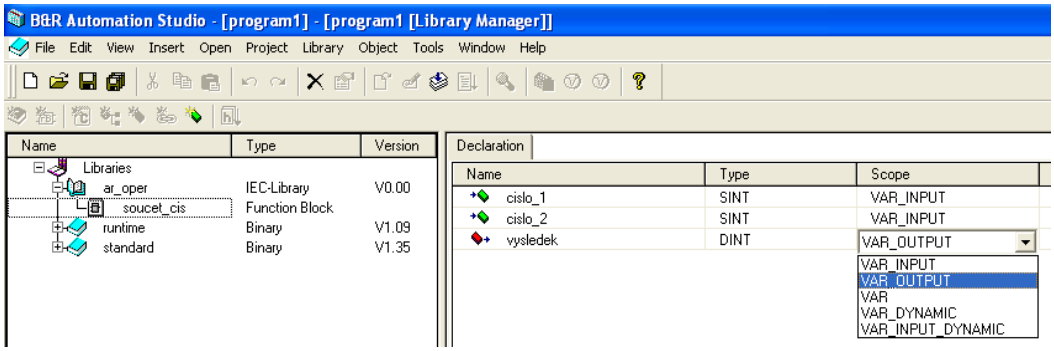


Obr.7.26: Vložení nové proměnné.

Do sloupce *Name* se napíše jméno nové proměnné např. cislo_1, cislo_2. Do sloupce *Type* se zapíše typ proměnné nebo je také možné tuto proměnnou si vybrat po dvojkliku levým tlačítkem myši.

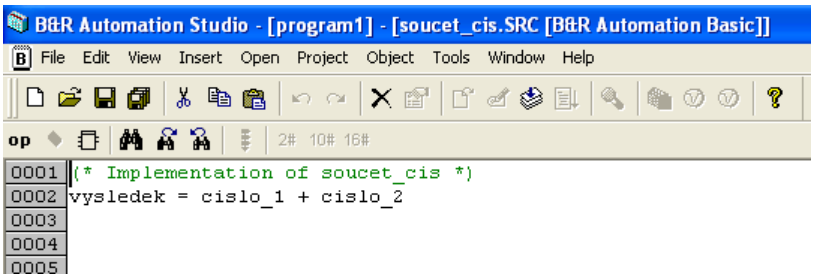
Do sloupce *Scope* se zapíše, jestli se jedná o vstupní proměnou výstupní proměnou apod. Jednotlivé možnosti uvádí následující tabulka 7.2.

Řešení může vypadat např. jak ukazuje obr. 7.28.



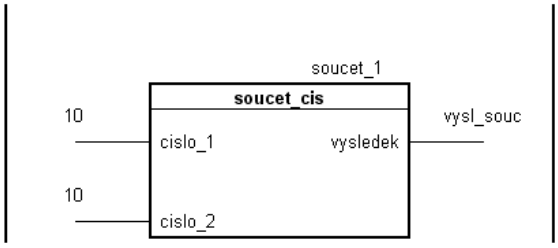
Obr.7.27: Řešení deklarace proměnných.

Součet čísel se zapíše, když se dvojklikem levým tlačítkem myši na daný funkční blok např. soucet_cis otevře příslušný jazyk, který byl zvolen při zakládání dané funkce/funkčního bloku např. B&R Automation Basic.



Obr.7.28: Zapsané řešení v B&R Automation Basicu.

Nový funkční blok – knihovna se musí uložit a pokud je potřeba jej v program využít, tak tuto knihovnu naleznete v Library Manageru. Bezprostředně po vložení daného FBK se musí také pojmenovat tento funkční blok např. soucet_1 a definovat vstupně výstupní parametry obr. 7.29.



Obr.7.29: Použití FBK v Ladder Diagramu.

7.6 Pole, struktury, dynamická proměnná

Pole

Automation Studio poskytuje také možnost vytvořit si datový typ pole. Opět i zde je číslování prvků pole od 0, kde maximální adresovatelný prvek pole je délka pole -1. Příklad pole je na následujícím obrázku.

Name	Type	Scope	Force	Value
Pressure	UINT[10]	local		
Pressure[0]	UINT			0
Pressure[1]	UINT			0
Pressure[2]	UINT			0
Pressure[3]	UINT			0
Pressure[4]	UINT			0
Pressure[5]	UINT			0
Pressure[6]	UINT			0
Pressure[7]	UINT			0
Pressure[8]	UINT			0
Pressure[9]	UINT			0

Obr. 7.30: Příklad pole v Automation Studiu.

Struktura

Struktura patří mezi uživatelsky definované datové typy, která může obsahovat různé proměnné různých datových typů. Jedna z možných struktur je zobrazena na následujícím obrázku

Name	Type	Scope	Force	Value
Bread	recipe_typ	global		
flour	SINT			120
water	UINT			12
salt	USINT			1
yeast	UDINT			2

Obr. 7.31: Příklad struktury v Automation Studiu.

Když např. potřebujete změnit obsah mouky je nutné v tasku zapsat např.

Bread.flour :=10;

Pole struktur

Automation Studio umožňuje vytvořit také pole struktur. Tento datový typ je zejména vhodný, když je potřeba vyrobit několik receptur, kde receptury jsou navzájem datovými typy stejné a mění se pouze hodnoty proměnných, jak ukazuje následující obrázek, který popisuje receptury pro výrobu chlebů. Pokud bude nutné změnit např. obsah vody v receptuře 2, je nutné v tasku zapsat:

Bread[1].water :=100;

Name	Type	Scope	Force	Value
Breads	recipe_typ[3]	global		
Breads[0]	recipe_typ			
flour	SINT			120
water	UINT			12
salt	USINT			1
yeast	UDINT			2
Breads[1]	recipe_typ			
flour	SINT			100
water	UINT			11
salt	USINT			1
yeast	UDINT			1
Breads[2]	recipe_typ			
flour	SINT			88
water	UINT			8
salt	USINT			0
yeast	UDINT			1

Obr. 7.32: Příklad pole struktur v Automation Studiu.

Pro výpočet délky struktury se používá metoda **sizeof.**

Dynamická proměnná

Často je vhodné než se neodkazovat přímo na proměnnou, využívat k tomu např. pointer. V Automation Studiu existuje dynamická proměnná. Využitím této dynamické proměnné získáte flexibilní řešení než s využitím statických proměnných. Využitím instrukce access se získává přístup k dynamické proměnné (např. pCounter). Dále je nezbytné se odkazovat přímo na adresu proměnné např. Counter (jak ukazuje následující zápis instrukce), proto se zde ještě využívá instrukce adr.

pCounter **access** adr (Counter);

7.7 Datové objekty

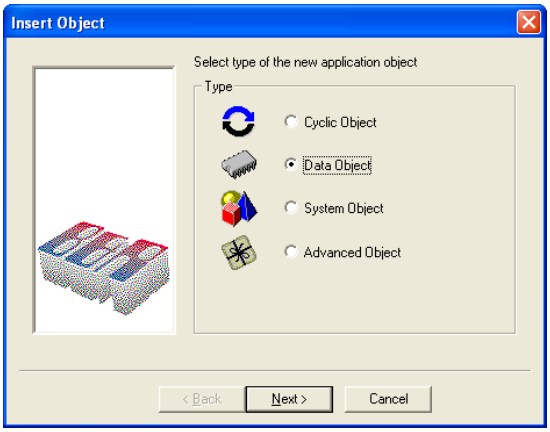
Použití datových objektů v Automation Studiu je stejné jako použití datových bloků v prostředí STEP7 Simatic Manageru. Pro práci s datovými objekty v AS se využívají knihovní funkce knihovny DataObj.

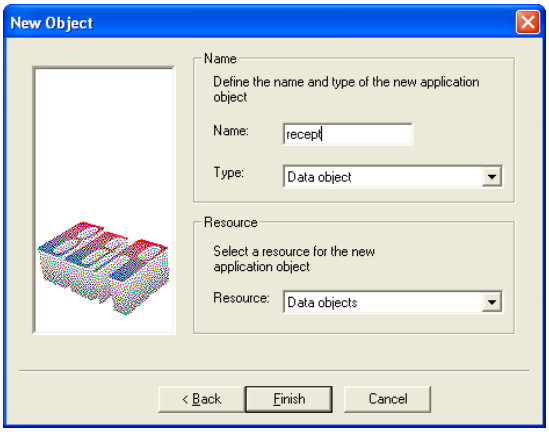
```
;-------------------------------------------------------------
;----------------------- RECIPE DATA -------------------------
;-------------------------------------------------------------
;
;       +---------------------------------------- Description
;       |                                          [STRING(12)]
;       |        +--------------------------------- Bread type
;       |        |                                  [USINT]
;       |        |      +---------------------------- Water
;       |        |      |                           [UINT]
;       |        |      |       +------------------- Flour
;       |        |      |       |                   [UDINT]
;       |        |      |       |      +------------ Yeast
;       |        |      |       |      |            [USINT]
;       |        |      |       |      |    +------- Filler Byte
;       |        |      |       |      |    |       [USINT]
;       |        |      |       |      |    |    +- Salt
;       |        |      |       |      |    |    |  [UINT]
;       |        |      |       |      |    |    |
"White Bread ", 001, 00200, 00035000, 010, 000, 00080;
"Brown Bread ", 002, 00300, 00078000, 025, 000, 00110;
"Grain Bread ", 003, 00350, 00125000, 030, 000, 00175;
```

Obr. 7.33: Příklad datové struktury.

Datový objekt je možné vytvořit dvěmi způsoby tj. vytvořit jej za běhu programu nebo vytvořit tento objekt stejným způsobem, jak se vkládá nový task obr. 7.34. Pokud se vytváří datový objekt za běhu programu, je nutné specifikovat, kde se má ve které části paměti založit tento datový blok tj. nejčastěji se využívá UserRom nebo UserRam.

Ukládat informace v systémech B&R generace SG4 je možné buď do souboru nebo do datových objektů. Výhodnější je využívat datové objekty, protože při zápise do souborů jsme omezeni počtem zápisů dáno použitou pamětí a také, že soubory nejsou kontrolovány Checksumem tj. nekontroluje se konzistence těchto souborů, zatímco datové objekty jsou kontrolovány pomocí checksumu při každém cyklu vykonávání programu. To je také důvod proč se např. tyto datové objekty využívají pro zápis nějakého nastavení, případně kopírování receptur apod.





Obr. 7.34: Vytvoření nového datového objektu. Obr. 7.35: Pojmenování datového objektu.

Aby bylo možné z datového objektu něco vyčíst, používá se stejná struktura jako u práce se soubory, tj. nejprve se musí zjistit, kde je datový objekt umístěn v paměti pomocí instrukce DataObjInfo. Obdržením tohoto identifikátoru se může následně přečíst pomocí knihovní funkce DataObjRead obsah datového objektu. Pro vyčtení se využívá dynamická proměnná. Více je uvedeno ve cvičení s podrobným popisem vytvoření datového objektu.

Pokud si tedy vytváří se datový objekt při vytváření programu pro PLC, musí se nejprve vytvořit datový objekt obr. 7.34, 7.35, následně vypsat jednotlivé položky do datového objektu obr. 7.33 a následně se musí také v *Open→Data Types* založit struktura, která bude mít zapsány položky ve stejném pořadí, jak jsou zapsány v datovém objektu.

Pokud se vytváří datový objekt za běhu programu, tak se nejprve musí vytvořit struktura, tj. co se bude ukládat do tohoto datového objektu, následně se musí tento datový objekt vytvořit tj. DataObjCreate, tím se také zjistí identifikátor, kde se vytvořil datový objekt a následně pomocí funkce DataObjWrite se zapíše obsah struktury na místo do paměti, na které se odkazuje proměnná Ident.

7.8 Modul CM 211- použití v řízení

CM 211 je univerzální I/O modul obsahující 8 DI/8DO/2AI/2AO a speciální funkce. Na učebně NK 317 je tento modul zařazen bezprostředně za základní jednotku CP 430.60-1 systém 2003.

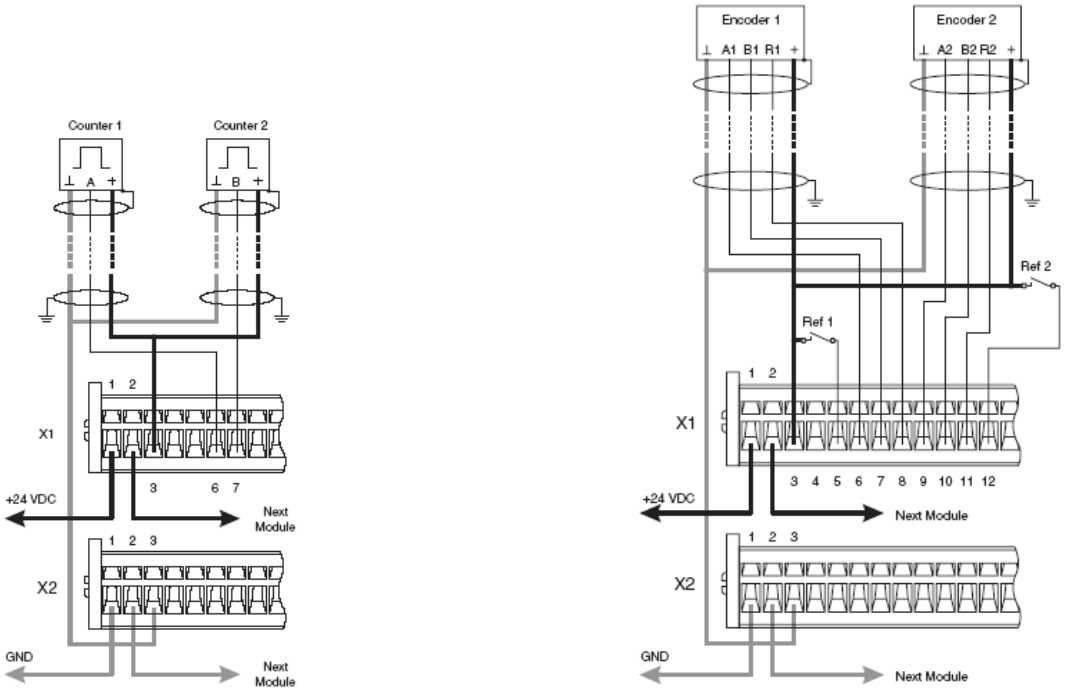
Obsahuje také speciální vstupy pro měření periody, frekvence a impulzů z inkrementálních snímačů. Jednotka obsahuje 8 digitálních vstupů z nichž některé mají výše popsané funkce. Příklad zapojení např. pro načítání impulzů ukazuje obr. 7.37.

Modul CM211 se skládá z modulu AF101 a modulu DM435. Modul AF101 obsahuje 4 sloty, SS1 má dva analogové vstupy, SS2 má dva analogové výstupy, SS3 je první čítač a SS4 je druhý čítač. Pro vyčítání hodnot z modulu slouží datové slovo 1 a datové slovo 2. Pokud využíváte rozšířené možnosti tohoto modulu jako čítač nebo měření frekvence musíte pro nastavení využít funkční blok IOC2003(). Funkční blok IOCGeneral slouží pro zápis a čtení konfiguračních slov pro B&R systémy řady 2010, 2005, ale i 2003. U systémů 2003 neumí tento funkční blok zapisovat na subsloty. K tomu je určen blok IOC2003. Každý ze subslotů obsahuje konfigurační slova. Každý subslot má většinou konfigurační slovo 14 např. u modulu CM211 je konfigurační slovo 14 pro SS1 pouze pro čtení a obsahuje označení modulu, ale konfiguračním slovem 14 na SS3 se konfiguruje první čítač. Aktuální hodnotu vyčítáte z datového slova 2. V základní konfiguraci, kde modul CM211 je zařazen za CP 430-60, jsou pro Vás platné ty řádky, u kterých stojí „BASE“. „BASE“ znamená, že modul je vedle CPU, RIO znamená, že je připojen přes sběrnici RIO (RIO je BR sběrnice staršího data), PLK znamená, že modul je na POWERLINKovém ostrůvku, CANIO znamená, že modul je na CANovském ostrůvku.

Obr. 7.36: Rozšiřovací modul CM 211.

Tab. 7.3: Rozsah a počet DI/DO/AI/AO

Počet DI/DO	8/8
Počet AI/AO	2/2 ±10V,0-20mA 12bit
Speciální funkce	3 jednokanálové nebo dvoukanálové čítače nebo 2 inkrementální snímače max. 20kHz



Obr. 7.37: Příklad zapojení pro načítání impulzů - čítač, enkodér.

Analogové vstupy a výstupy

Pomocí analogových vstupů se může z řízeného procesu získávat informace např. o teplotě, tlaku apod. Pomocí analogových výstupů se mohou řídit např. akční členy – ventily, frekvenční měniče apod. Nejčastější rozsahy, u B&R 2000 modulů analogových vstupů/výstupů, jsou 0-20mA a ±10V.

Tab. 7.4: Rozsahy AI/AO.

Analogové vstupy/ výstupy	**Rozsah**			**Datový typ**
Analogový vstup 1/2	Napětí	+10V	$7FFF	INT 16
		0V	$0000	
	Napětí	-10V	$8001	
	Proud	20mA	$7FFF	
		0mA	$0000	
Analogový výstup 1/2	Napětí	+10V	$7FFF	INT 16
		0V	$0000	
	Napětí	-10V	$8001	

Shrnutí pojmů

V programovacím prostředí Automation Studio je k dispozici **knihovna funkcí**, kde lze najít funkce a funkční bloky pro celou řadu úloh – od standardních funkcí nutných pro většinu aplikací (matematické funkce, čítače, časovače apod.), přes komunikační funkce až po funkce pro speciální použití. **Funkce** a **funkční bloky** se vytvářejí a spravují v Library Manageru v Automation Studiu. Pokud chceme v programu využít jakoukoliv funkci, je nutné importovat do projektu příslušnou knihovnu. Funkce a

funkční bloky mají své **vstupně/výstupní parametry**, pomocí kterých jim lze předávat hodnoty a ovlivňovat tak jejich provádění.

Základní **logické instrukce** jsou k dispozici přímo v Automation Studiu a je možné je volně využít v programu. Všechny ostatní instrukce jsou k dispozici jako funkce v příslušných knihovnách – jedná se o **čítače**, **časovače**, aritmetické instrukce a další.

Kromě základních datových typů popsaných v kapitole 6, umožňují automaty Bernecker&Rainer pracovat i s **poli**, **strukturami** a **dynamickou proměnnou**. Rovněž je možné vytvářet datové objekty, které umožňují uchovávat větší počty dat nebo parametrů.

Modul CM211 je rozšiřující modul pro připojení vstupně/výstupních signálů. Používá se pro PLC řady 2003. Obsahuje jak kanály pro DI/DO, tak kanály pro připojení analogových veličin.

Kontrolní otázky

1. Co je to Library Manager v Automation Studiu a k čemu slouží?
2. Co je to funkce u programovatelného automatu Bernecker&Rainer a k čemu slouží?
3. Co je to funkční blok u programovatelného automatu Bernecker&Rainer a k čemu slouží?
4. Jakým způsobem lze funkci nebo funkčnímu bloku předávat parametry?
5. Co je to knihovna funkcí a k čemu slouží?
6. Jak se u programovatelného automatu Bernecker&Rainer realizuje časovač?
7. Popište použití polí u programovatelného automatu Bernecker&Rainer.
8. Popište použití datových struktur u programovatelného automatu Bernecker&Rainer.
9. Co jsou to datové objekty u programovatelného automatu Bernecker&Rainer a k čemu slouží?
10. Jaké komunikační možnosti nabízí programovatelný automat Bernecker&Rainer?

Další zdroje

7-1. B&R training document: The Basics of Automation Studio TM210.

7-2. B&R training document: Automation Studio Online Communication TM211.

7-3. B&R training document: Automation Runtime TM213.

7-4. B&R training document: The Service Technician on the Job TM220.

7-5. B&R training document: Automation Studio Diagnostics TM223.

7-6. B&R training document: Ladder diagram (LD) TM240.

7-7. B&R training document: Automation Basic (AB) TM247.

7-8. B&R training document: Memory Management and Data Storage TM250.

7-9. B&R training document: Automation Studio Libraries I TM260.

7-10. B&R training document: Closed Loop Control with LOOPCONR TM261.

7-11. B&R Automation Studio 2.5.2.21, Help.

7-12. www.br-automation.com

Řešená úloha 7.1.

Napište program, který po přivedení impulsu na první digitální vstup vytvoří na prvním digitálním výstupu impuls o délce 7s.

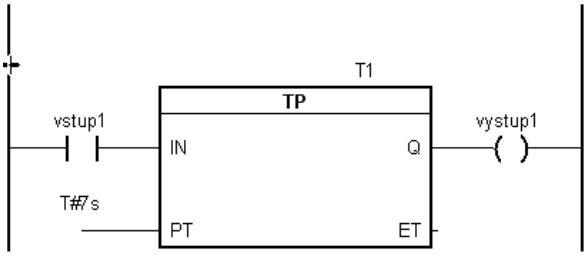
I/O

Name	Data Type	PV Name	Remark
SS1 DW 00 in	INT		±10 V or 20 mA
SS1 DW 01 in	INT		±10 V or 20 mA
SS2 DW 00 out	INT		±10 V
SS2 DW 01 out	INT		±10 V
digital input 01	BOOL	vstup1	24 VDC
digital input 02	BOOL	vstup2	24 VDC
digital input 03	BOOL		24 VDC
digital input 04	BOOL		24 VDC
digital input 05	BOOL		24 VDC
digital input 06	BOOL		24 VDC
digital input 07	BOOL		24 VDC
digital input 08	BOOL		24 VDC
digital output 01	BOOL	vystup1	0.5 A, 24 VDC
digital output 02	BOOL	vystup2	0.5 A, 24 VDC
digital output 03	BOOL	vystup3	0.5 A, 24 VDC
digital output 04	BOOL	vystup4	0.5 A, 24 VDC
digital output 05	BOOL		0.5 A, 24 VDC

Řešení programu v LAD

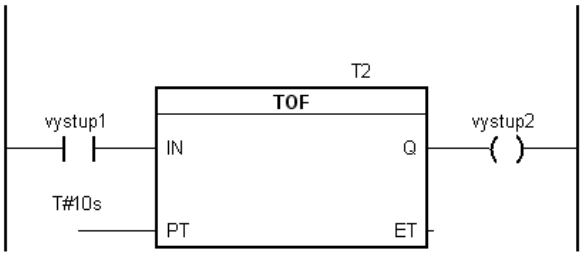
0001

Program, který po přivedení impulsu na první digitální vstup vytvoří na prvním digitálním výstupu impuls o délce 7s.



0002

Doplňte program tak, aby na třetím výstupu byl impuls delší o 10s než impuls na prvním digitálním výstupu.



Řešení programu v Automation Basicu

```
B&R Automation Basic : cas_bas
0001 (* Program, který po přivedení impulsu na první digitální
0002  vstup vytvoří na prvním digitálním výstupu impuls o délce 7s.  *)
0003 cas1.IN = vstup2
0004 cas1.PT = T#7s
0005 cas1 FUB TP()
0006 vystup2 = cas1.Q
0007
0008 cas4.IN=vystup2
0009 cas4.PT= T#10s
0010 cas4 FUB TOF()
0011 vystup4 = cas4.Q
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 7\pr_7_1.pgd.zip

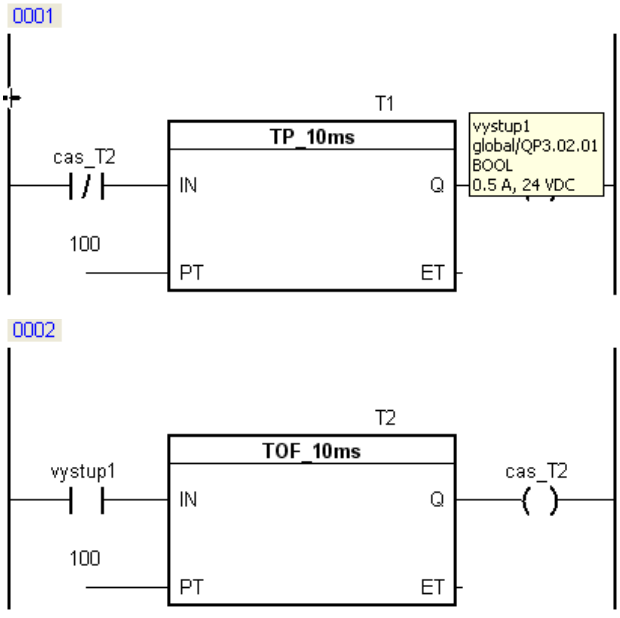
DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 8\ anim3.avi.

Řešená úloha 7.2.

Vytvořte program tak, aby na prvním digitálním výstupu vytvořil impulsní signál s délkou periody 2s a střídou 1/1.

Řešení v LAD



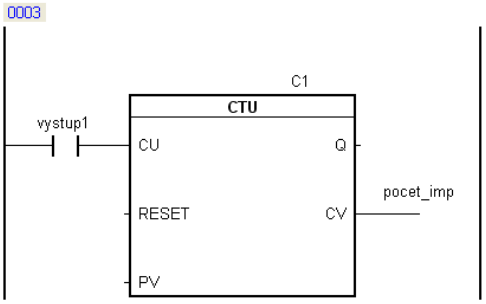
DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 7\pr_7_2.pgd.zip

Řešená úloha 7.3.

Doplňte program 7.2 čítačem tak, aby čítač čítal impulsy z prvního digitálního výstupu a jejich počet ukládal do proměnné, která bude typu UINT.

Řešení v LAD



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 7\pr_7_3.pgd.zip

8. TESTOVACÍ NÁSTROJE VE STEP7 A B&R AUTOMATION STUDIU

Čas ke studiu: 2 hodiny

Cíl:

Kapitola popisuje nástroje pro ladění a testování uživatelských programů v programovacích prostředích pro automaty Siemens Simatic a pro automaty B&R. V prostředí Step7 bude student po zvládnutí této kapitoly schopen využít **datové reference**, **monitorování stavu programu** a **tabulku proměnných.** Rovněž bude schopen ladit svůj program v **simulátoru PLCSIM**. Při odstraňování chyb v aplikaci bude schopen použít **diagnostické nástroje** a sledovat **systémové informace**, naučí se rovněž reagovat na různé druhy chyb.

U programovatelných automatů B&R se v prostředí Automation Studio student seznámí s použitím nástrojem **Logbook**, s nástrojem pro **monitorování stavu programu a proměnných**. Dále je zde demonstrována práce s nástrojem **Trace** pro grafické zobrazení sledovaných veličin, nástrojem **Profiler** pro sledování časového provádění tasků a nástrojem pro **zavádění operačního systému** do programovatelného automatu.

Výklad

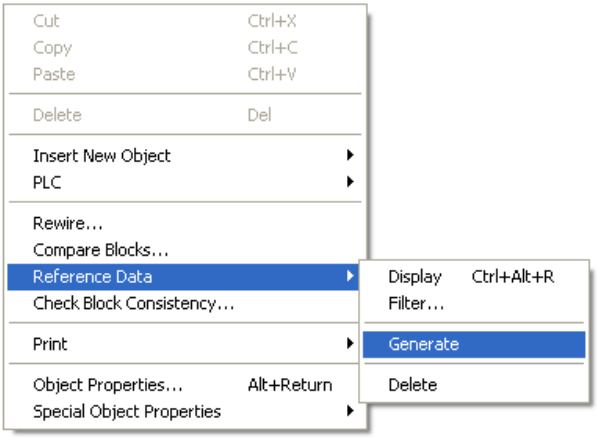
8.1 Testovací nástroje v prostředí STEP7

Při vývoji programu se mohou také i vyskytnout nežádoucí chyby, které lze důkladným testováním odhalit. STEP 7 nabízí v tomto směru programátorovi pomoc již při vytváření tohoto programu. V níže uvedeném textu se setkáte s popisem jednotlivých funkcí STEP7.

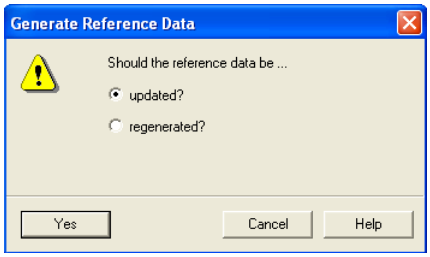
8.1.1 Reference dat

Pro snadnější práci v prostředí STEP7 zejména u rozsáhlých projektů, je vhodné si vytvořit také tzv. reference dat, protože usnadňují vyhledávání již použitých proměnných, generování struktury programu apod.

Reference dat se vytvoří kliknutím pravého tlačítka myší do pravé části okna (výpis FC, FB, DB, OB). Volbou v menu podle obrázku obr.8.1 se vygenerují reference. Dále budete vyzváni pro potvrzení, co chcete udělat obr. 8.2.



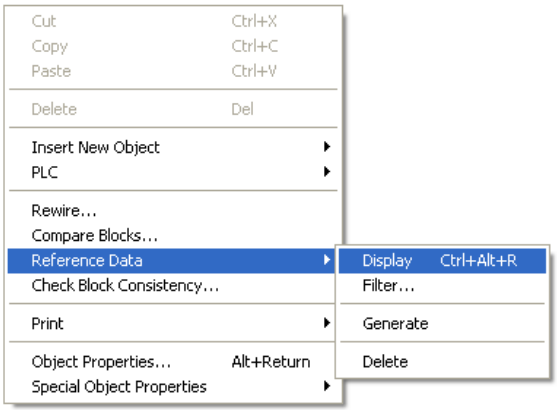
Obr.8.1: Generování odkazů na data.



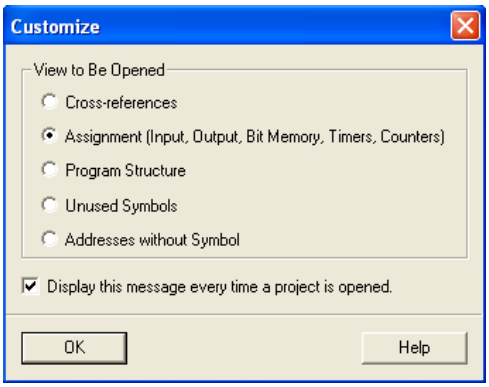
Obr. 8.2: Potvrzení provedení.

Obr.8.3: Informace o vytvoření odkazů.

Pokud byly reference dat vytvořeny, kliknutím pravého tlačítka myši do pravé části obrazovky lze tyto reference dat zobrazit. Volbu ukazuje obrázek 8.4.



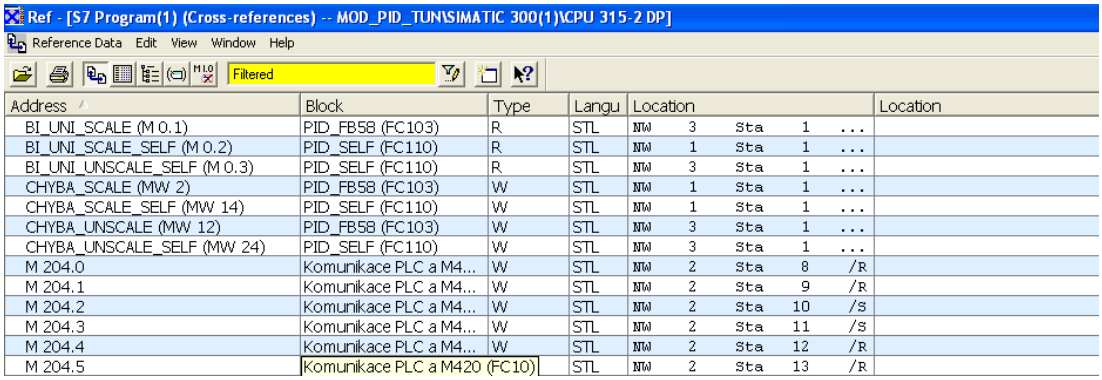
Obr. 8.4: Vyvolání odkazů na data.



Obr.8.5: Přehled jednotlivých možností.

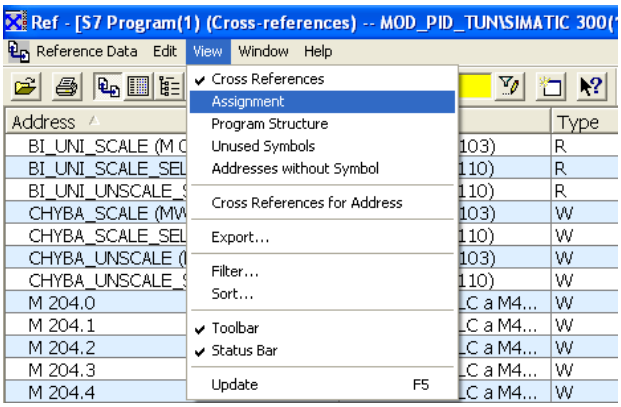
Je-li potřeba vyhledat nějakou proměnnou v projektu, lze využít vlastností Cross-referencí.

Volbou *Cross-references* v okně *Customize* lze obdržet výpis jednotlivých proměnných. Samozřejmě se dají také tyto informace filtrovat tzn. mohou být vybrány jenom memory bity apod. Filtrování dat umožňuje programátorovi rychleji najít jednotlivé proměnné v programu, jak ukazuje následující obrázek 8.6. Kliknutím na vybranou proměnnou se přesuneme na místo jejího výskytu v programu.

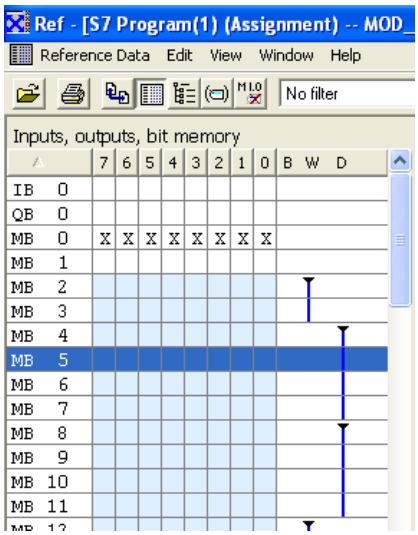


Obr. 8.6: Výpis proměnných.

Volbou v menu *View* → *Assignment* (obr 8.7.) získáme detailní výpis jednotlivých použitých proměnných (obr. 6.11).

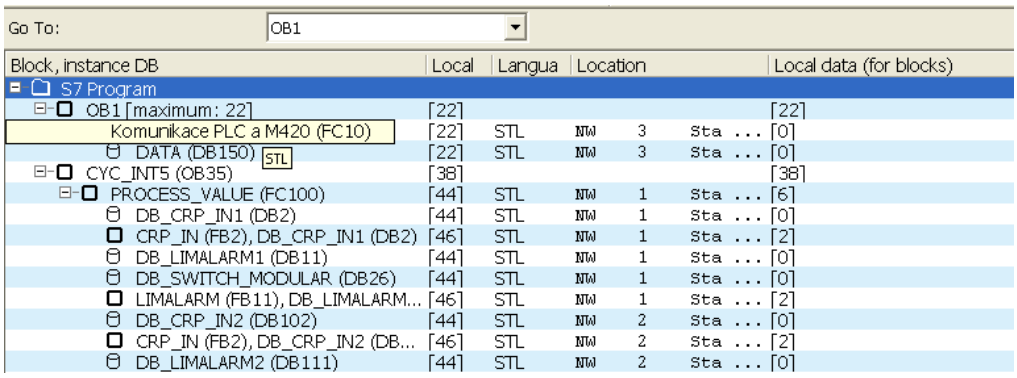


Obr. 8.7: Volba v menu.



Obr.8.8: Výpis proměnných.

Volbou *View*→ *Program Structure* se vygeneruje struktura volání funkcí, funkčních bloků, apod. v programu ve STEP7 (obr. 8.9).



Obr.8.9: Výpis struktury programu.

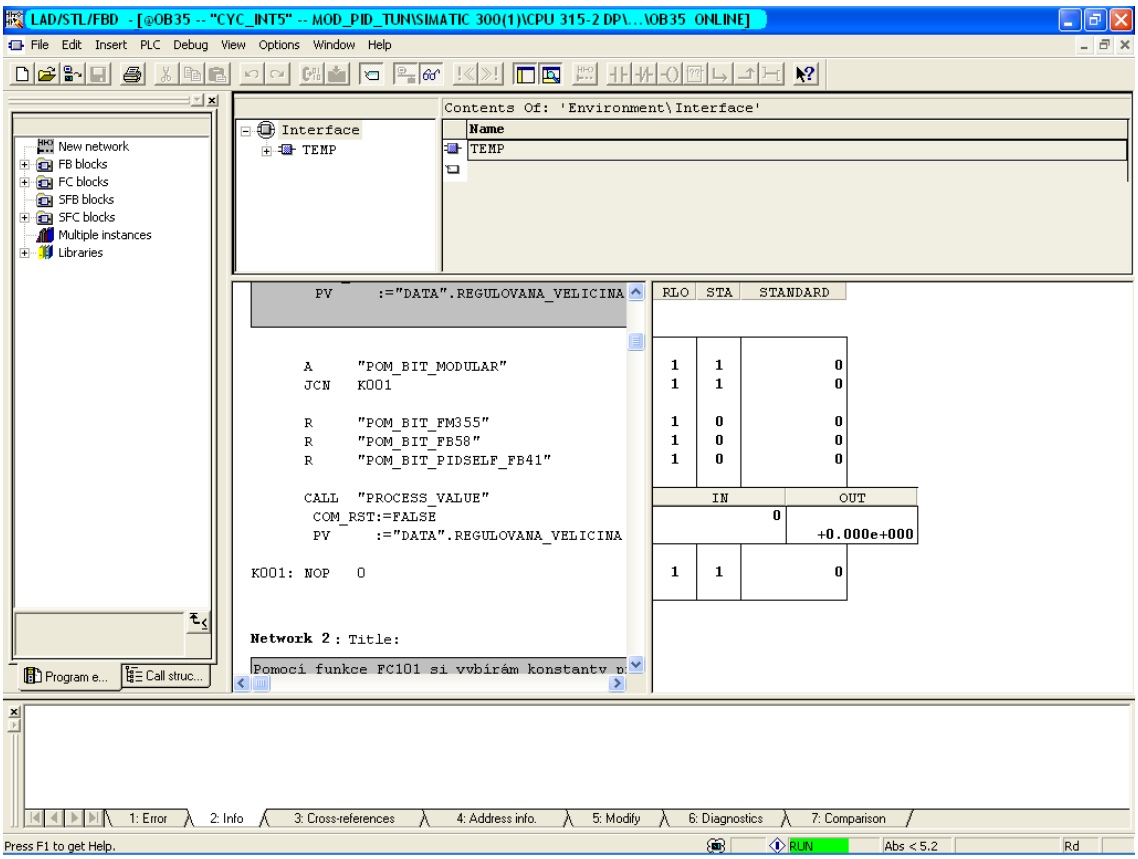
8.1.2 Použití LAD/STL editoru pro sledování stavu proměnných

Pomocí LAD/STL editoru lze sledovat stav programu a změny signálů za běhu programu.

LAD – Zobrazuje se tok signálu mezi jednotlivými prvky a slova na vstupech a výstupech bloků.

STL – Lze sledovat důležité registry a operandy.

Volbou v menu *Debug* → *Monitor* se zahájí monitorování stavu dané funkce, funkčního bloku apod.



Obr. 8.10: On-line sledování stavu programu.

8.1.3 Tabulka proměnných

Pro sledování stavu proměnných, změnu hodnoty proměnné, apod. nabízí STEP7 možnost vepsat si tyto proměnné do VAT tabulky. VAT tabulka nabízí možnost zobrazit si následující typy proměnných: vstupy, výstupy, memory bit, časovače, čítače, obsahy datových bloků, stavy vzdálených vstupů a výstupů.

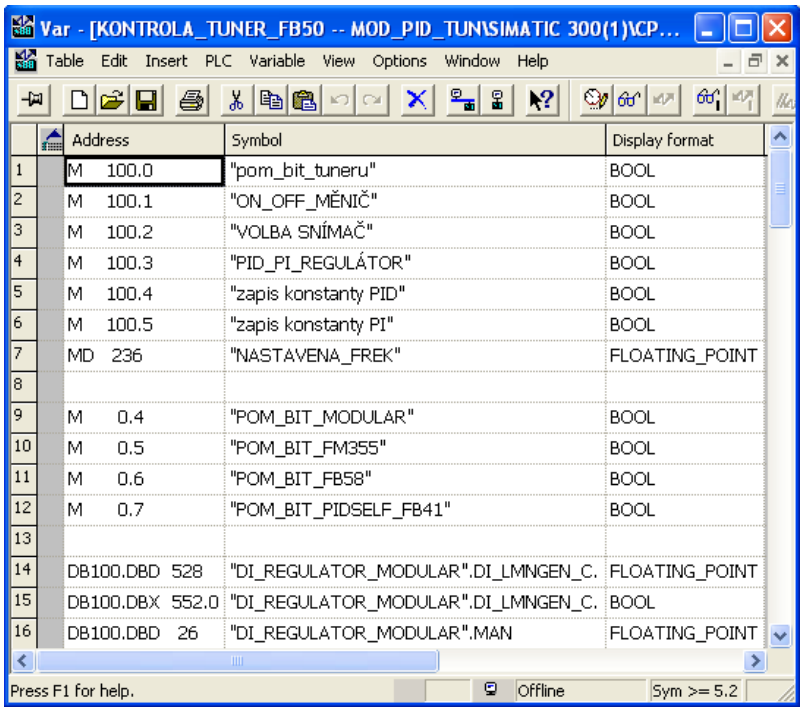
VAT tabulka se vloží do projektu z menu STEP7 *Insert* →*S7 Block*→ *Variable Table*

Proměnné lze do VAT tabulky vkládat buď symbolickým vyjádřením do sloupce *Symbol*, nebo se také proměnné mohou vkládat podle jejich adresy do sloupce *Address*.

Volbou v menu *Insert*→*Symbol* může být proměnná vybrána přímo ze Symbolické tabulky. Tímto způsobem si ušetříte bezchybnost převodu proměnné.

Sledování stavu proměnné lze zahájit výběrem v menu *Variable* → *Monitor*. Zapsáním nové hodnoty proměnné do sloupce *Modify value* a volbou *Variable* → *Modify* bude aktuální hodnota proměnné přepsána.

Maximální velikost VAT tabulky je 1024 řádků. Lze také vkládat komentáře do řádku pomocí volby v menu *Insert → Comment Line*.



Obr. 8.11: Příklad VAT tabulky.

8.1.4 Práce s S7-PLCSIM

S7-PLCSIM dovoluje testovat program bez nutnosti se připojit k reálnému PLC. S S7-PLCSIM můžeme testovat a ladit programy pro S7-300 a S7-400.

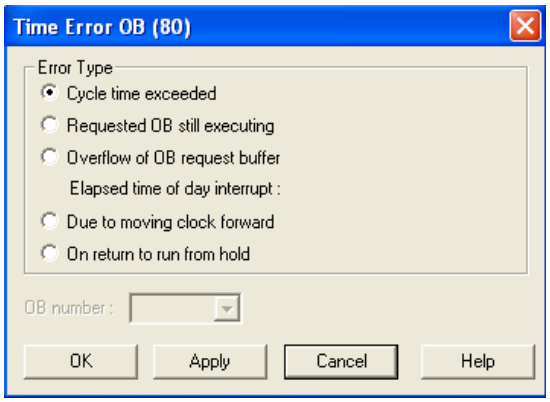
S7-PLCSIM může být také využíván ve spojení s VAT tabulkou. Maximální využití paměťové oblasti S7-PLCSIM ukazuje tabulka 8.1.

Tab. 8.1: Paměťový prostor S7-PLCSIM.

Paměťová oblast	**Popis**
Časovače	T0- T2047
Memory bity	131 072 bitů (16kB M paměti)
Celková adresovatelná I/O paměť	131 072 bitů (16kB I/O paměti)
Process image	Maximum 131 072bitů
Lokální data	Maximum 64kB
Logické bloky a datové bloky	2048 funkčních bloků a funkcí 4095 datových bloků
Organizační bloky	OB1, OB10-OB17, OB20-OB23, OB30-OB38, OB40-OB47, OB55, OB56, OB57, OB61-64, OB70, OB72, OB73, OB80, OB 82, OB83, OB84, OB 85, OB86, OB87, OB88, OB90 OB100, OB101, OB121, OB122

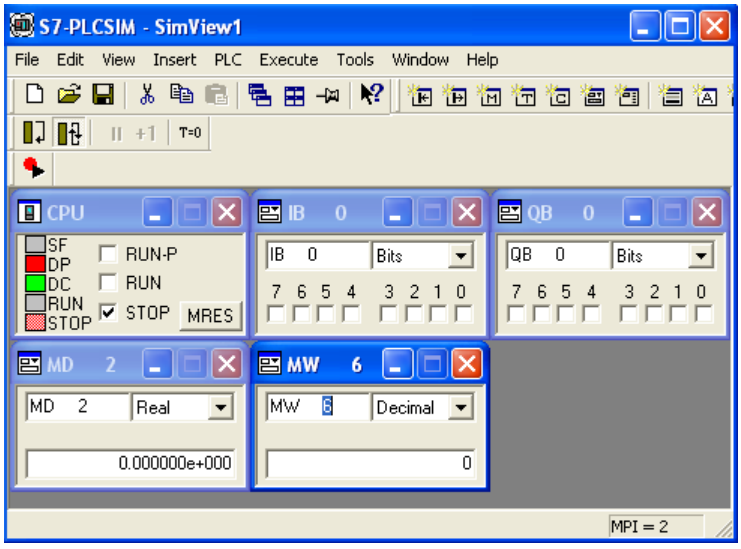
Jednotlivé paměťové oblasti se vkládají z menu *Insert* → *Inputs*, … S7-PLCSIM podporuje také následující typy organizačních bloků: OB40-47, OB70, OB 72, OB73, OB80, OB82, OB 83, OB85, OB86.

Jednotlivé OB se vkládají z menu *Execute* → *Trigger Error OB*. Jednotlivé organizační bloky vyžadují také doplňující informace, jak ukazuje obr 8.12.



Obr. 8.12: Dotaz na doplňující informace OB.

Poté již stačí také provést download programu do S7-PLCSIM a můžete využívat i tyto funkce s OB.

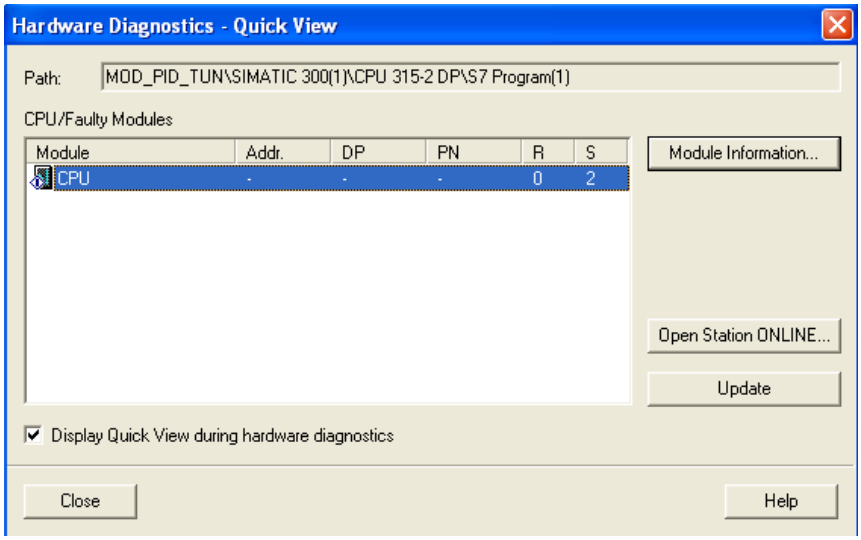


Obr. 8.13: Příklad S7-PLCSIM.

8.1.5 Informace o systému

Pokud nastanou nějaké problémy na straně hardwaru (chyba modulu, záměna adres na modulu, špatně ošetřená vzniklá chyba), existuje ve STEP7 následující funkce pro rychlý přehled aktuálního stavu systému.

Volbou v menu *PLC* → *Diagnostics/Settings* → *Diagnose Hardware* se zobrazí okno s diagnostikou hardwaru. Programátorovi se tímto vytvoří rychlý pohled na situaci, která nastala.



Obr. 8.14: Quick View.

Podle symbolů umístěných u jednotlivých CPU a FM se může rychle určit, zda je PLC nebo FM v režimu RUN nebo STOP apod.

Označením modulu (např. CPU, SM, FM) a kliknutím na tlačítko Module Information se získá výpis chyby viz bod 8.3.

Kliknutím na tlačítko *Open Station ONLINE* se otevře okno s hardwarovou konfigurací. Konfigurace obsahuje aktuální stavy jednotlivých částí systému.

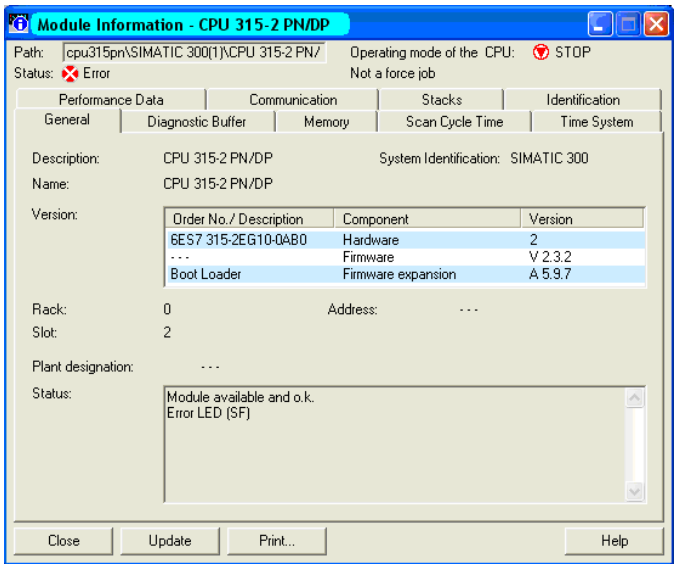
Tab. 8.2: Provozní stavy PLC.

Symbol	**Operační mód PLC/FM**
	Modul má chybu
	STARTUP
	STOP
	STOP přepnutí přepínačem na PLC
	RUN
	Diagnóza není možná. Je přerušeno on-line spojení apod.

8.1.6 Systémové informace

Systémová informace poskytuje pohled na PLC, pro které se vyvíjí program. Jednotlivé PLC se také liší od sebe např. verzí firmwaru. Verze firmwaru je důležitá pro využívání rozšířených funkcí programovatelných automatů. Např. programovatelný automat CPU 315PN/DP s firmwarem 2.3 nemá možnost využívat funkce WebServeru. Pokud se provede upgrade firmwaru na PLC firmwarem 2.5 můžete využívat také funkce WebServeru. Jednotlivé PLC se také liší velikostí paměti, výkonem CPU apod. Všechny tyto výše uvedené vlastnosti PLC lze také zobrazit v prostředí STEP7.

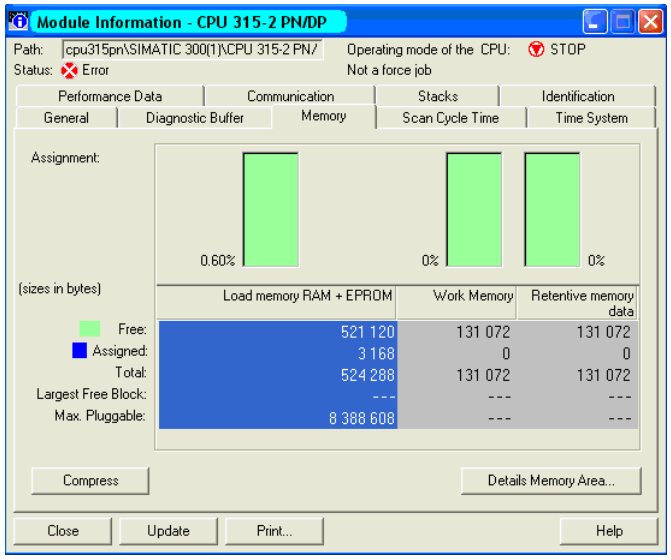
Volbou *PLC→ Diagnostic/Setting→ Module Information* se v záložce *General* zobrazí základní informace o PLC, jak ukazuje obr. 8.15.



Obr. 8.15: Základní informace o PLC.

Obsazení paměti

Informaci o aktuální obsazení paměti v PLC obsahuje záložka *Memory* v okně Module Information



Obr. 8.16: Aktuální využití paměti PLC.

8.1.7 Diagnostika

Diagnostikou se rozumí rozpoznávací funkce a funkce pro poznámky CPU S7-300. Oblast, ve které budou zobrazeny informace o chybách, se nazývá diagnostický archiv (*Diagnostic Buffer*). CPU provádí diagnostiku při každém cyklickém dotazu.

Objeví-li se chyba nebo nějaká událost, například změna provozního stavu, stane se následující:

- V diagnostickém archivu se zaznamená hlášení spolu s datem a časem. Poslední přišlé hlášení se zaznamená na začátek archivu. Je-li archiv plný, jsou dřívější zápisy smazány.

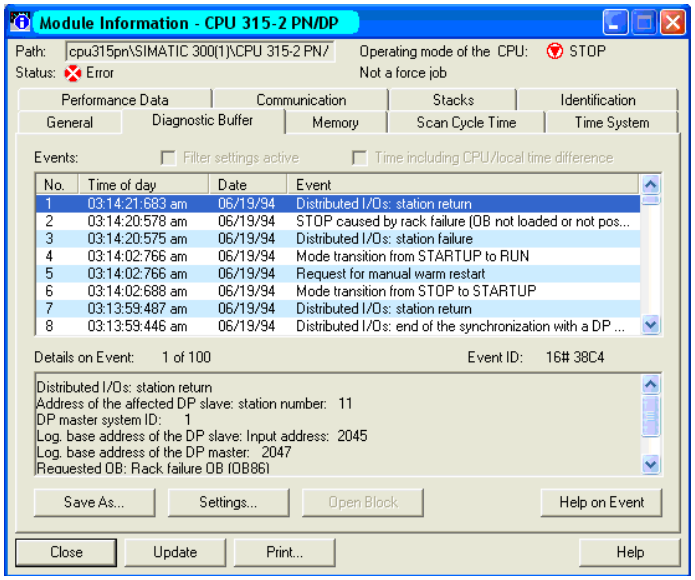
- Archiv zobrazuje popis diagnostikované události.
- Událost popřípadě aktivuje příslušný chybový OB.

Pomocí diagnostiky CPU lze rozpoznat následující chyby

- Systémová chyba CPU.
- Chyba některého modulu.
- Chyba programu v CPU.

Diagnostic Buffer

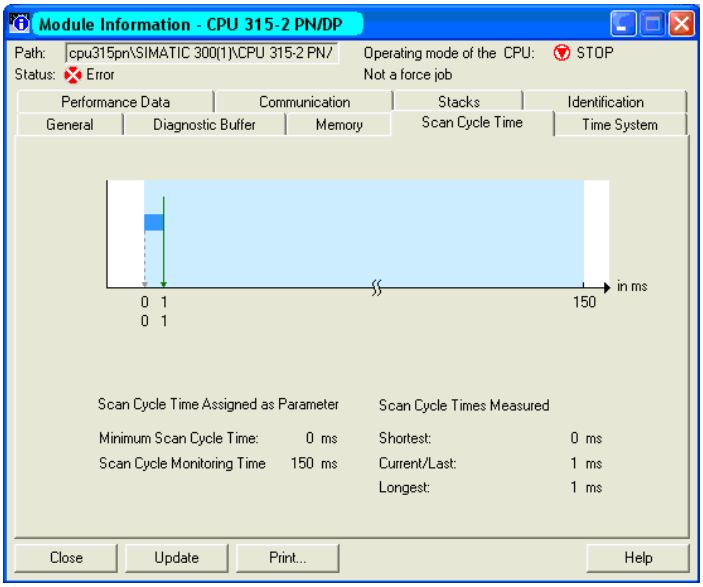
Diagnostický archiv je organizován jako kruhový zásobník, který se nedá vymazat. Obsahuje všechny diagnostikované události v řadě za sebou tak, jak přišly. Na obrazovce PG lze zobrazit všechny přišlé zprávy.



Obr. 8.17: Výpis diagnostického archivu.

Zobrazení doby cyklu

Čas, který používá CPU k aktualizaci vašeho I/O image, k provedení uživatelského programu, k provedení všech diagnostických funkcí a ke komunikaci s programovacím zařízením, se nazývá doba cyklu.



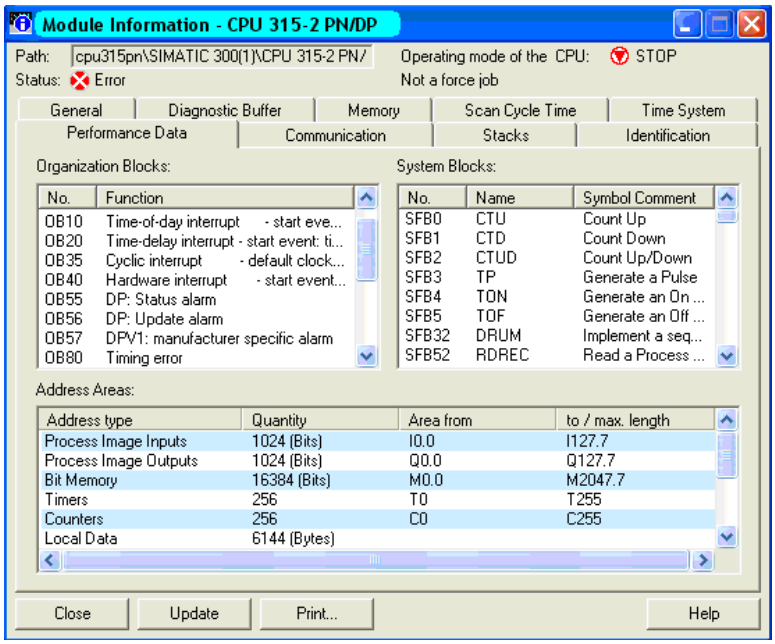
Obr. 8.18: Doba cyklu.

Zobrazení podporovaných funkcí, funkčních bloků a OB CPU

Pomocí Performance Data se dá zobrazit přehled informací o CPU. Přehled dat CPU (online) obsahuje následující informace:

- Počet a adresový rozsah vstupů, výstupů, časovačů, čítačů a merkerů.
- Oblast dynamických lokálních dat, s kterými může CPU pracovat.
- Počet OB, FC, FB, DB, SFB, SFC, které mohou být v uživatelském programu použity.

Pomocí tohoto můžete zajistit kompatibilitu uživatelského programu, který má být nahrán do CPU.



Obr.8.19: Zobrazení podporovaný funkcí, funkčních bloků a OB CPU.

Tab.8.3: Výpis organizačních bloků podporovaných PLC S7-300.

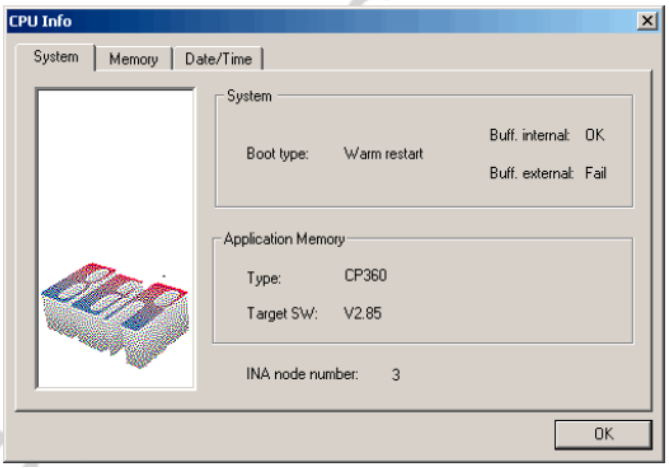
Organizační bloky	**Vyvolávací události (Hexadecimal Values)**
Cycle	
OB1	1101_H OB1 starting event 1103_H Running OB1 start event
Time - of - day interrupt:	
OB 10	1111_H Time-of-day interrupt event
Delay Interrupt:	
OB 20	1121_H Delay interrupt event
OB 21 (only CPU 317, CPU 319)	1122_H Delay interrupt event
Cyclic Interrupt:	
OB 32 (only CPU 317, CPU 319)	1133_H Cyclic interrupt event
OB 33 (only CPU 317, CPU 319)	1134_H Cyclic interrupt event
OB 34 (only CPU 317, CPU 319)	1135_H Cyclic interrupt event
OB 35	1136_H Cyclic interrupt event
Process interrupt:	
OB40	1141_H Process interrupt
DPV1-Interrupt (only DP-CPUs)	
OB 55 (except CPU 312)	1155_H Status interrupt
OB 56 (except CPU 312)	1156_H Update-interrupt
OB57 (except CPU 312)	1155_H Manufacture-specific interrupt
Synchronous cycle interrupt:	
OB 61 (only CPU 319)	1164_H Synchronous cycle interrupt
Technology synchronous interrupt (only Technology CPU)	
OB 65 (only CPU 315T, CPU 317T)	$116A_H$ Technology synchronous interrupt
Error responses:	
OB 80	3501_H Cycle time violation 3502_H OB or FB request error 3505_H Time-of-day interrupt elapsed due to time jump 3507_H Multiple OB reguest error caused start info buffer overflow
Diagnostic interrupt:	
OB 82	3842_H Module o.k. 3942_H Module fault

Organizační bloky	**Vyvolávací události (Hexadecimal Values)**
OB 83 (only 315PN, 317PN, 319PN)	3854_H PROFINET IO-Submodule plugged in and is proportional to a parameterized submodule 3855_H PROFINET IO-Submodule plugged in and is not proportional to a parameterized submodule 3861_H Module is inserted 3851_H Pull out PROFINET IO-Module 3861_H Module is removed
OB 85	$35A1_H$ No OB or FB $35A3_H$ Error during access of a block by the operating system $39B1_H$ I/O access error during process image updating of the inputs (during each access) $39B2_H$ I/O access error during transfer of the process image to the output modules (during each access) $38B3_H$ I/O access error during process image updating of the inputs (outgoing event) $38B4_H$ I/O access error during transfer of the process image to the output modules (outgoing event) $39B4_H$ I/O access error during transfer of the process image to the output modules (incoming event)
OB 86 (only DP, PN IO)	$38C4_H$ Distributed I/O: station failed, outgoing $38CB_H$ PROFINET I/O: Station restart $39C4_H$ Distributed I/O: station failed, incoming $39CB_H$ PROFINET I/O: Station failure
OB 87	$35E1_H$ Incorrect frame identifier in GD $35E2_H$ $35E2_H$ GD packet status cannot be entered in DB $35E6_H$ GD whole status cannot be entered in DB
Restart:	
OB 100	1381_H Manual restart requests 1382_H Automatic restart requests
Synchronous error responses:	
OB 121	2521_H BCD conversion error 2522_H Range length error during reading 2523_H Range length error during writing 2524_H Range error during reading 2525_H Range error during writing

Organizační bloky	Vyvolávací události (Hexadecimal Values)
	2526_H Timer number error 2527_H Counter number error 2528_H Alignment error during reading 2529_H Alignment error during writing 2530_H Write error during access to DB 2531_H Write error during access to DI 2532_H Block number error opening a DB 2533_H Block number error opening a DI 2534_H Block number error at FC call 2535_H Block number error at FB call $253A_H$ DB not loaded $253C_H$ FC not loaded $253E_H$ FB not loaded
OB122	2944_H I/O access error at nth read access (n > 1) 2945_H I/O access error at nth write access (n > 1)

8.2 Ladění programu v B&R Automation Studiu

Nejjednodušší způsob jak získat informace z PLC je kliknutím na jednotku pravým tlačítkem myši a vybrat položku *Online info….*Zde získáte základní přehled o stavu PCC. V tomto okně můžete také nastavit PCC aktuální čas, výběrem záložky *Date/time* a kliknutím na tlačítko na *Get PC Time*



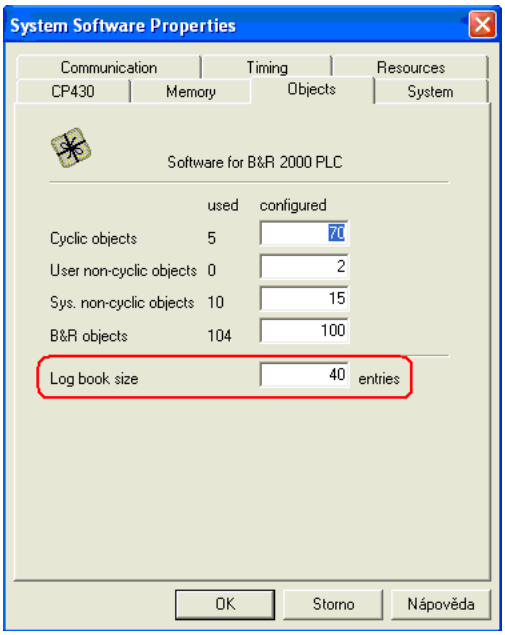
Obr. 8.20: On-line informace o CPU.

Další možnosti získat informaci o stavu PLC lze pomocí Logbooku nebo Loggeru (AS 2.5.2 nebo vyšší).

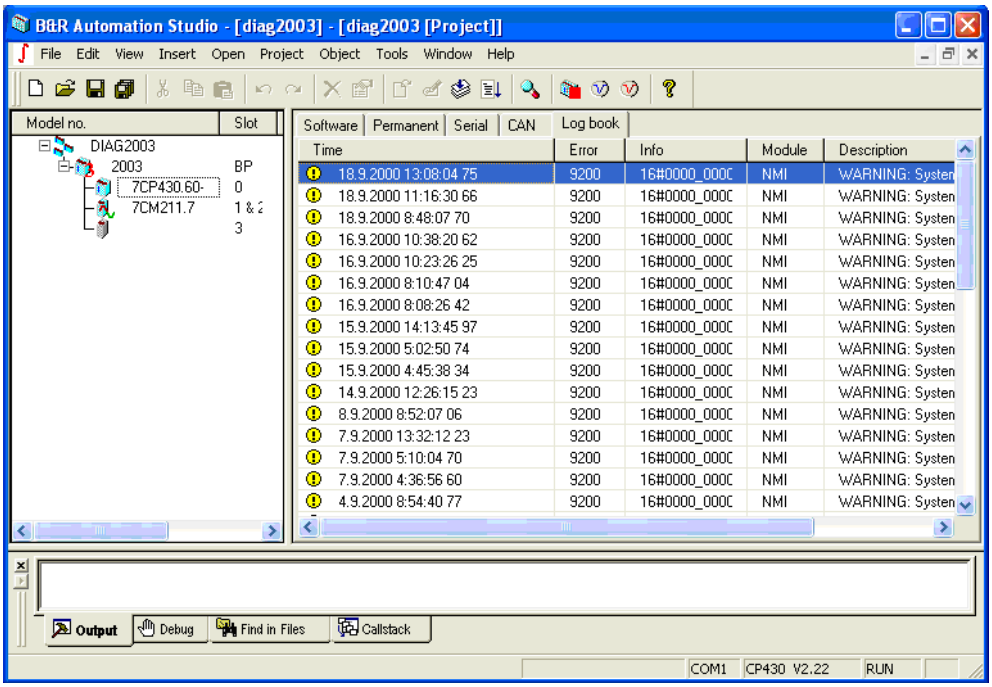
8.2.1 Log Book

Všechny druhy chyb, které nastanou během běhu aplikace (chyby softwaru v programovatelném automatu, nedodržení doby cyklu, apod.) jsou ukládány operačním systémem. Všechny chyby jsou ukládány do log booku a mohou být zobrazeny kliknutím na záložku LogBook. Kliknutím levým tlačítkem myši na danou chybu získáte popis chyby. Je možné nastavit, kolik chyb se má ukládat do Logbooku. Standardně je nastaveno 20, ale může toto číslo být zvětšeno. Kliknutím pravým tlačítkem myši na CPU a volbou Properties v dialogovém okně se zobrazí následující okno, jak ukazuje obr.8.21 , kde v záložce Objects lze nastavit ukládání do Logbooku.

Výpis chyb lze také uložit do souboru formátu XML.



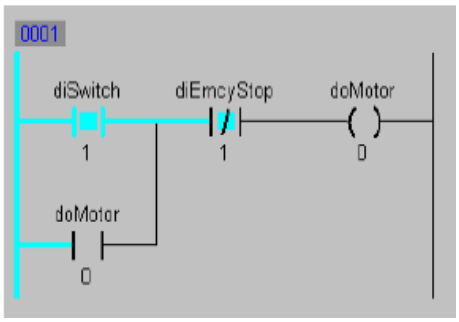
Obr.8.21: Nastavení počtu ukládaných informací do Logbooku.



Obr.8.22: Logbook.

8.2.2 Monitor

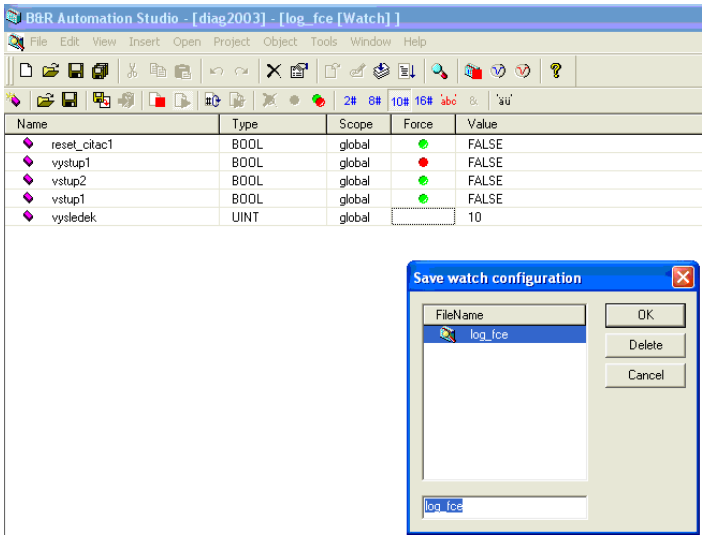
Pro monitorování běhu programu, hardwarové konfigurace slouží monitor. Monitor spustíte kliknutím na ikonu  . Můžete také monitor používat v jednotlivých taskových úlohách, kde kliknutím na ikonu  si můžete nechat zobrazit tok zpracování instrukcí. Lze také i jednotlivé stavy proměnných měnit jak v programovacím jazyku LAD, tak v Automation Basic. Musí být ale dodržena aktuální syntaxe, především velká a malá písmena.



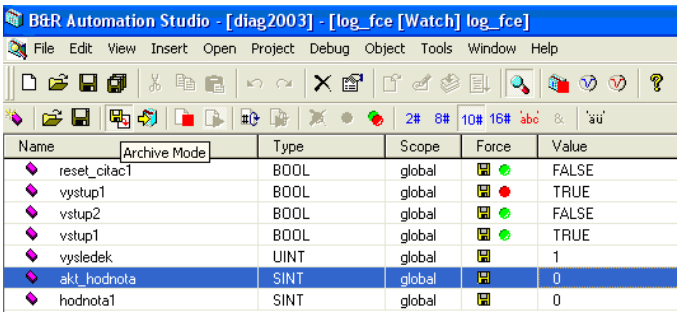
Obr. 8.23: Monitor v LAD.

8.2.3 Watch – monitor proměnných

Watch se používá pro zobrazení aktuálního stavu proměnných a jejich hodnot, které nabývají za běhu programu. Hodnoty mohou být také ukládány pro pozdější znovu obnovení testování. Kliknutím pravým tlačítkem myši na nějaký task a výběrem v dialogovém menu Watch, spustíte tento monitor proměnných. Pokud chcete měnit parametry musíte poté kliknout na ikonu  , aby jste mohli být spojeni s automatem online. Watch umožňuje ukládat a načítat proměnné, archivovat tyto proměnné obr. 8.24. Zapisovat data do PLC spouštět a zastavovat tasky, vypínat a zapínat funkci force.



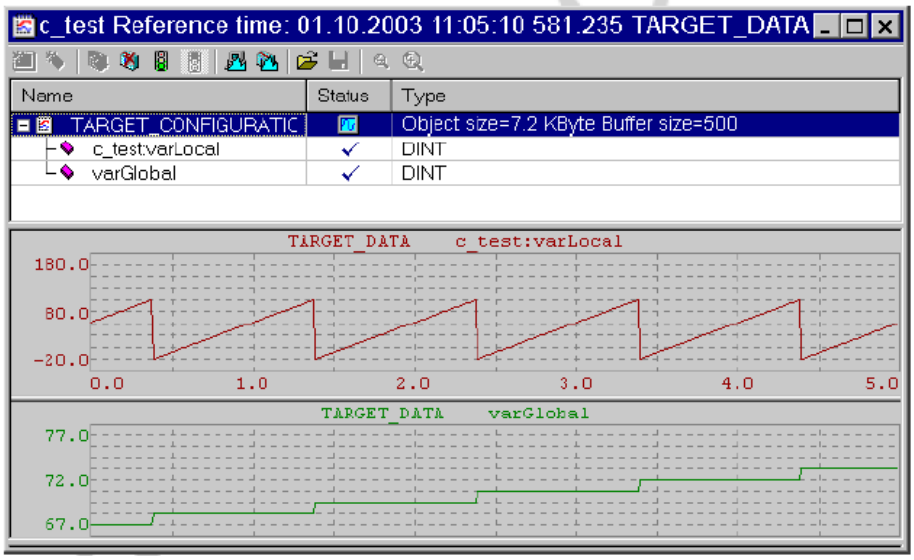
Obr. 8.24: Archivace Watch.



Obr.8.25: Příklad archivování proměnných.

8.2.4 Trace

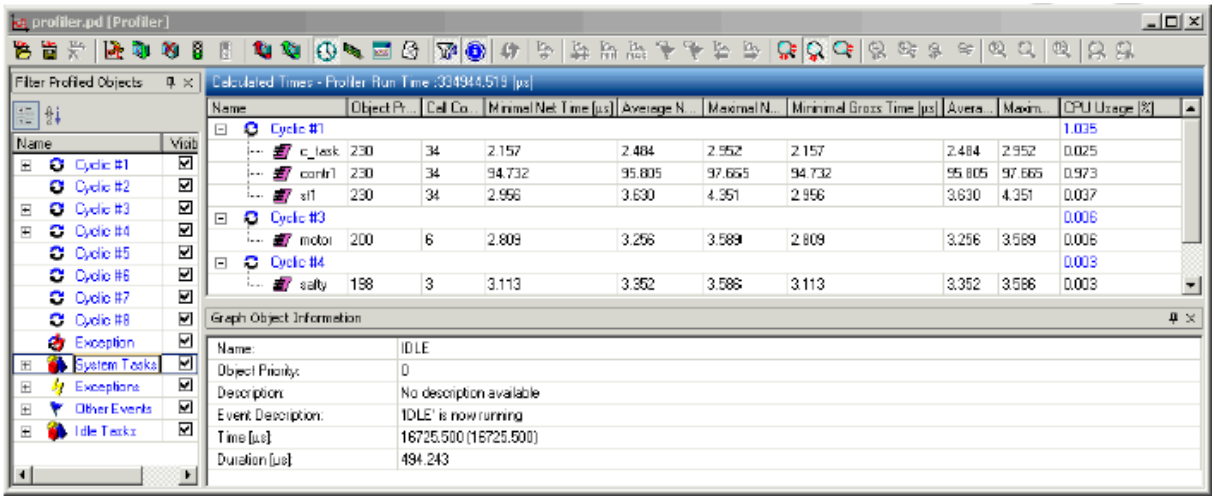
Opět kliknutím pravým tlačítkem myši na task a výběrem v dialogovém menu Trace můžete zahájit monitorování stavu proměnných v čase . Stav proměnných bude zobrazen ve formě grafu. Novou konfiguraci Trace spustíte kliknutím na ikonu  . Proměnné vkládáte kliknutím na ikonu  . Trace musí být také konfigurován do PCC pomocí ikony  . Maximálně se může vyčítat až 8 proměnných, ale záleží to také na velikosti bufferu.Trace se spouští ikonou  a zastavuje ikonou  . Pokud Trace zastavíte můžete kliknutím na ikonu  načíst data z PCC.



Obr. 8.26: Tracer.

8.2.5 Profiler

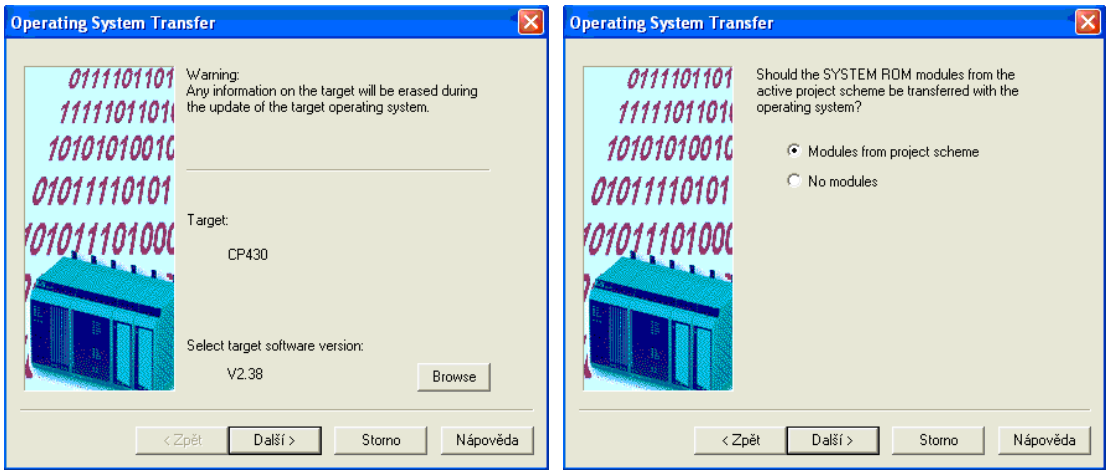
Profiler nabízí možnost sledování doby vykonávání taskových tříd a také použít zásobníku a systémové paměti. Profiler spustíte výběrem v menu Open→ Profiler. Profiler musí být také konfigurován do PCC pomocí ikony  . Maximálně se může vyčítat až 8 proměnných, ale záleží to také na velikosti bufferu. Trace se spouští ikonou  a zastavuje ikonou  . Pokud Trace zastavíte můžete kliknutím na ikonu  načíst data z PCC.



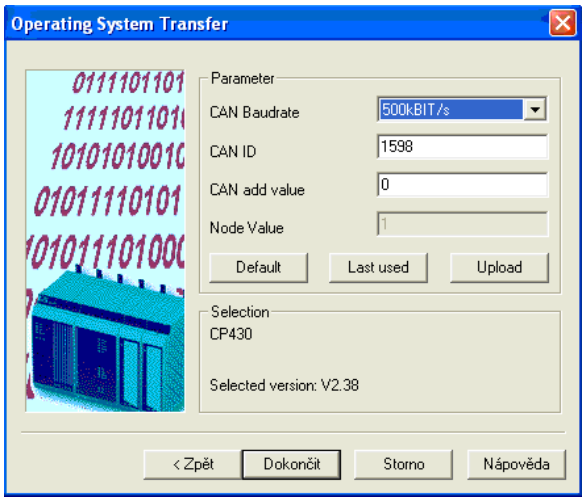
Obr. 8.27: Profiler.

8.2.6 Zápis operačního systému do PLC

Každý programovatelný automat obsahuje operační systém, který řídí vykonávání jednotlivých taskových tříd, čtení I/O, komunikaci apod. Tento operační systém se může také v průběhu času vyvíjet a výrobce může dodávat aktuální verzi firmwaru. Pokud měníte operační systém v programovatelné automatu musíte jej přepnout do režimu BOOT a poté postupovat podle následujících obrázků.



Obr. 8.28: Výběr nového operačního systému.



Obr.8.29: Nastavení komunikace .

Shrnutí pojmů

Nástroje pro testování a ladění uživatelského programu jsou velice důležité pro efektivní práci programátora. Prostředí Step7 nabízí celou řadu těchto nástrojů, z nichž nejdůležitější jsou tyto:

- **Datové reference** – slouží pro kontrolu využití jednotlivých datových oblastí, umožňuje zjistit, kde v programu jsou jednotlivá data využívána, umožňuje zobrazit strukturu programu apod.
- **Monitorování stavu programu** – umožňuje sledovat stav programu a změny signálů za běhu programu.
- **Tabulku proměnných** – umožňuje sledovat a měnit data.
- **Simulátoru PLCSIM** – umožňuje ladit program bez nutnosti mít k dispozici programovatelný automat.
- **Diagnostické nástroje** – nástroje, které umožňují najít a odstranit chybu v programu, v hardware apod.
- **Systémové informace** – nástroj poskytující informace o využití paměti automatu, době cyklu programu, o podporovaných funkcích automatu apod.

Prostředí Automation Studio nabízí zejména následující nástroje:

- **Logbook** – nástroj, který archivuje všechny chyby, které v automatu nastaly a umožňuje tak jejich detekci a odstranění.
- **Monitorování stavu programu** – umožňuje sledovat stav programu a změny signálů za běhu programu.
- **Monitorování stavu proměnných** – umožňuje sledovat a měnit data.
- **Trace** – nástroj pro grafické zobrazení sledovaných veličin.
- **Profiler** – nástroj pro sledování časového provádění tasků.
- **Nástroj pro zavádění operačního systému**.

Kontrolní otázky

1. K čemu slouží nástroj Datové reference v prostředí Step7?
2. Jakým způsobem je možné monitorovat stav programu ve Step7?
3. Jakým způsobem je možné monitorovat a měnit data ve Step7?
4. K čemu slouží nástroj PLCSIM?
5. Jaké nástroje jsou ve Step7 k dispozici pro diagnostiku chyb program a hardware?
6. K čemu slouží Logbook u programovatelných automatů B&R?
7. Jak lze v B&R Automation Studiu monitorovat program?
8. Jak lze v B&R Automation Studiu monitorovat a měnit data?
9. K čemu slouží v B&R Automation Studiu nástroj Profiler?
10. K čemu slouží v B&R Automation Studiu nástroj Trace?

Další zdroje

8-1. B&R training document: The Basics of Automation Studio TM210.

8-2. B&R training document: Automation Studio Online Communication TM211.

8-3. B&R training document: Automation Runtime TM213.

8-4. B&R training document: The Service Technician on the Job TM220.

8-5. B&R training document: Automation Studio Diagnostics TM223.

8-6. B&R training document: Ladder diagram (LD) TM240.

8-7. B&R training document: Automation Basic (AB) TM247.

8-8. B&R training document: Memory Management and Data Storage TM250.

8-9. B&R training document: Automation Studio Libraries I TM260.

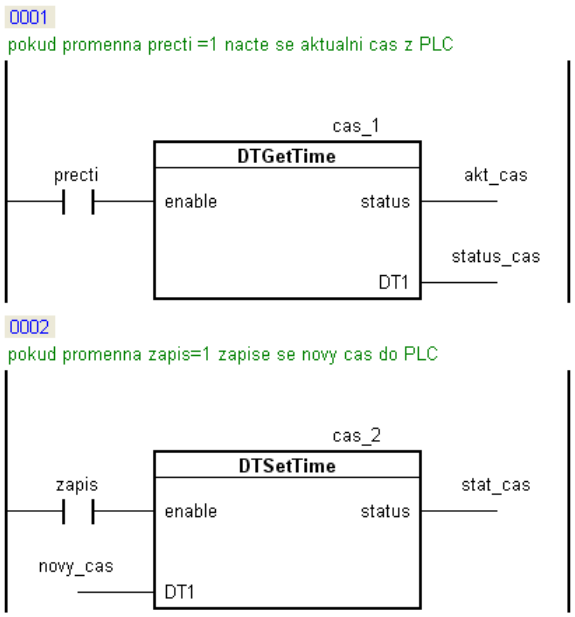
8-10. B&R training document: Closed Loop Control with LOOPCONR TM261.

8-11. B&R Automation Studio 2.5.2.21, Help.

Řešená úloha 8.1.

Vytvořte task, který zjistí aktuální reálný čas v PLC. Pokud bude odlišný od aktuálního času, proveďte nastavení správného času v PLC. Najděte vhodnou knihovnu pro práci s čase v programovatelném automatu.

Řešení v LAD



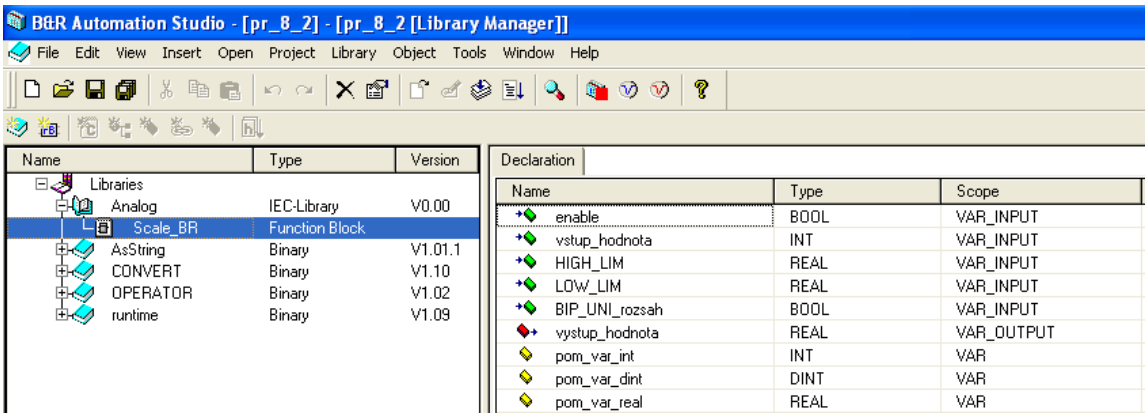
DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 8\pr_8_1.pgd.zip

Řešená úloha 8.2.

Vytvořte vlastní funkční blok, který bude načítat analogovou hodnotu a převede ji na reálné číslo. Výstupní hodnota reálného čísla bude v rozsahu, který bude na vstupu funkčního bloku možné změnit. Projekt bude vytvořen pro zařízení SG4. Knihovna se bude jmenovat Analog a funkční blok se bude jmenovat SCALE_BR.

Pozn. SG4, protože se bude počítat v REAL viz přednáška č.6

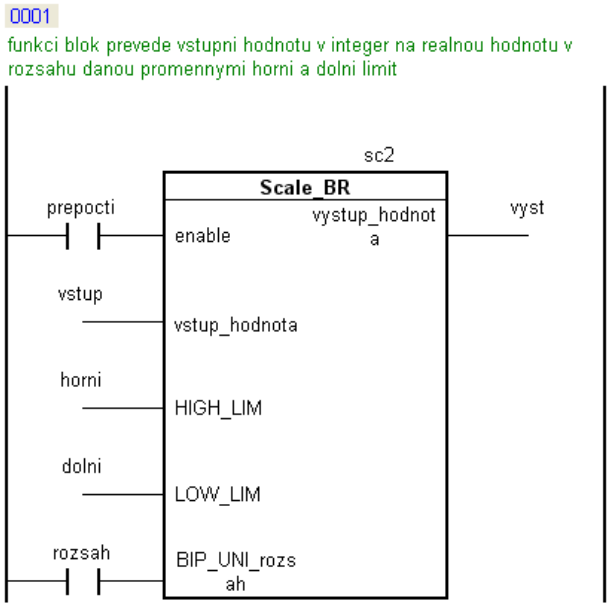


B&R Automation Basic : Scale_BR

```
0001 (* Implementation of Scale_BR *)
0002 (*jestlize bude povolen prepocet*)
0003
0004 IF enable = 1 then
0005
0006 (*nacti hodnotu v int*)
0007 pom_var_int = vstup_hodnota
0008
0009 (*preved hodnotu z int na dint*)
0010 pom_var_dint = DINT(pom_var_int)
0011
0012 (*preved hodnotu z dint na real*)
0013 pom_var_real = REAL(pom_var_dint)
0014
0015 (*pokud bude promenna BIP_UNI_rozsah = 1 pocita se unipolarni rozsah*)
0016 IF BIP_UNI_rozsah =1 THEN
0017
0018 (*vydel maximalnim kladnym rozsahem int *)
0019 pom_var_real = pom_var_real/32767
0020 ENDIF
0021
0022 (*pokud bude promenna BIP_UNI_rozsah = 0 pocita se bipolarni rozsah*)
0023 IF BIP_UNI_rozsah =0 THEN
0024
0025 (*vydel maximalnim rozsahem int*)
0026 pom_var_real = pom_var_real/65535
0027 ENDIF
0028
0029 (*zohledni horni mez*)
0030 pom_var_real = pom_var_real * (HIGH_LIM - LOW_LIM)
0031
0032 (*zohledni dolni mez a zapis vysledek*)
0033 (*vystup_hodnota je v rozsahu HIGH_LIM LOW_LIM*)
0034 vystup_hodnota = pom_var_real + LOW_LIM
0035
0036 ENDIF
```

Kód funkčního bloku SCALE_BR

Řešení programu v LAD



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 8\pr_8_2.pgd.zip

DVD-ROM

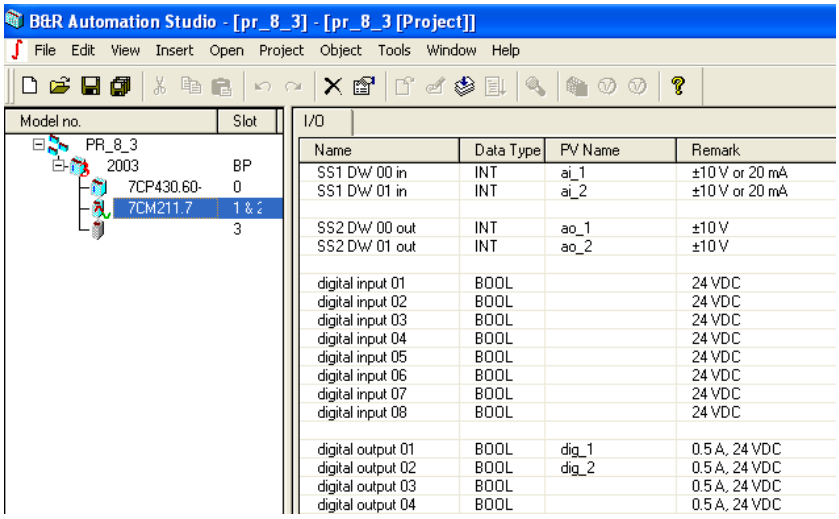
K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 8\ anim4.avi.

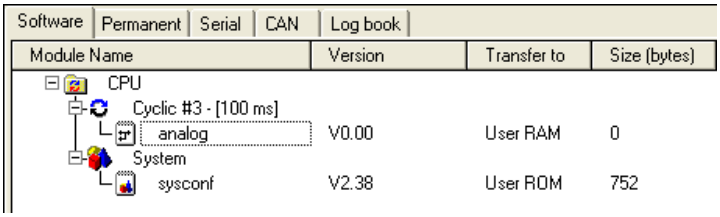
DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 8\ anim5.avi.

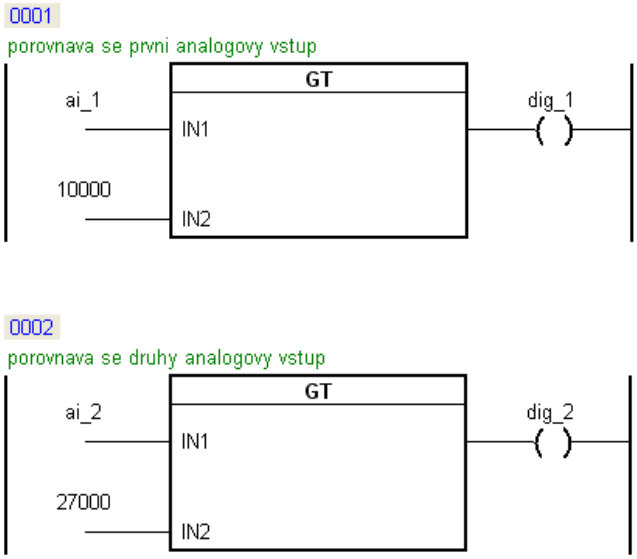
Řešená úloha 8.3.

Vytvořte program, který bude v pravidelných 100ms intervalech vyčítat hodnotu z prvního a druhého analogového výstupu a porovnávat je s hodnotou 10000 a 27000. Pokud tyto hodnoty se překročí sepne se digitální výstup 1 (analogový vstup 1) a digitální výstup 2 (analogový výstup 2).





Řešení programu v LAD



DVD-ROM

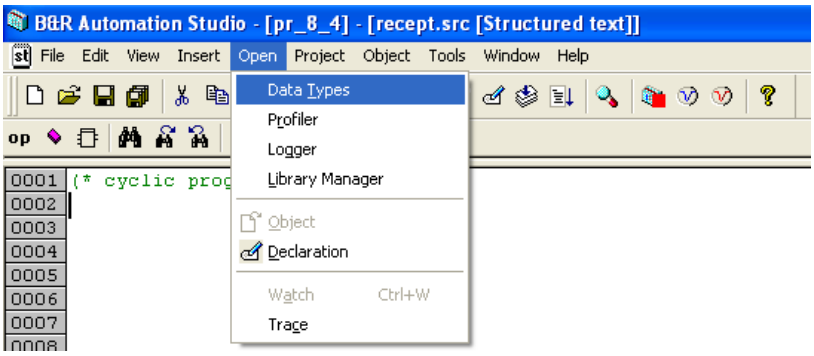
Řešený příklad naleznete na DVD: cvičení\cvičení 8\pr_8_3.pgd.zip

Řešená úloha 8.4.

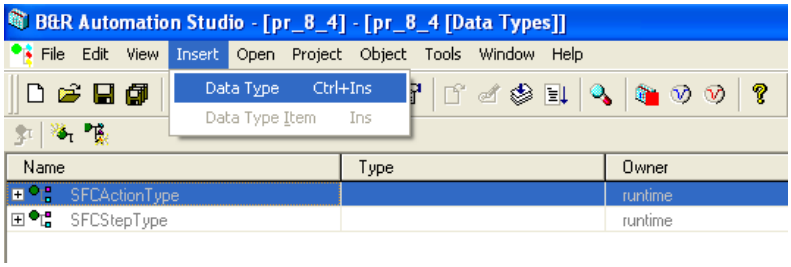
Vytvořte jednoduchou recepturu pro výrobu chleba. Předpokládejte, že se bude vyrábět 5 druhů chleba. Dalším požadavkem bude program sestavit tak, aby v případě napojení na vizualizaci, bylo na obrazovce operátorského panelu vždy vidět pouze jedna receptura a pomocí tlačítek (nahoru a dolu) si prohlížet zbývající receptury, případně také měnit tyto hodnoty.

Pro toto zadání je výhodné využít struktury v Automation Studiu, protože tím vytvořím recepturu chleba s jednotlivými ingrediencemi. Protože mám vytvořit program, který bude mít možnost vybírat si mezi 5 druhy receptur, využiji pole, do kterého vložím tuto strukturu. Protože musím na OP zobrazovat vždy jenom jednu recepturu využiji pro toto dynamickou proměnnou.

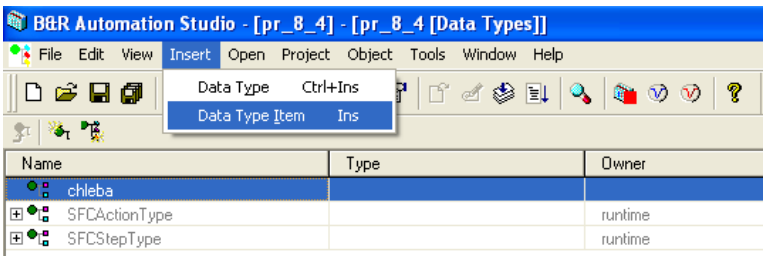
1. Otevření datových typů – struktur.



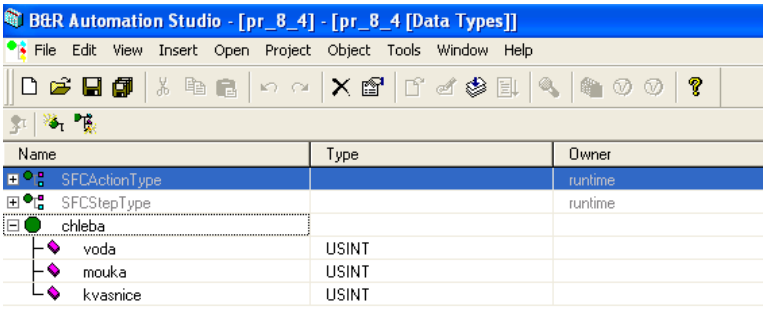
2. Vložení nového datového typu pojmenuji jako chleba.



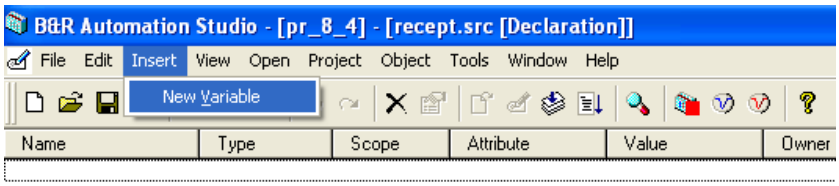
3. Vložím položky do struktury chleba – voda, mouka, kvasnice.

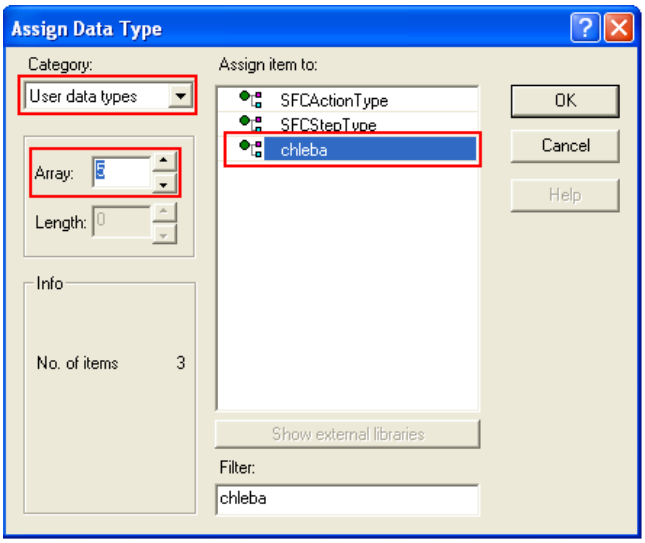


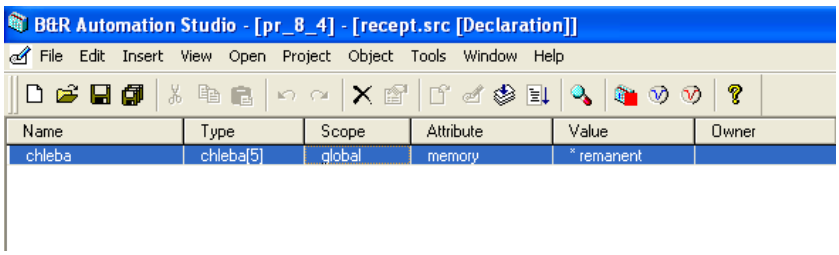
Nyní je založena struktura chleba.



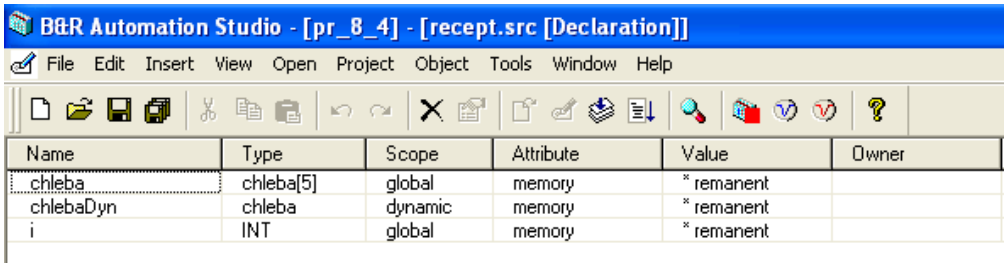
4. Nyní vytvořím pole struktur, protože potřebuji vyrábět 5 druhů chleba a také měnit receptury na OP pomocí tlačítek. Tuto změnu mohu elegantně provádět pomocí změny indexu pole i.







5. Nyní si nadefinuji dynamickou proměnnou chleba, která musí mít stejný datový typ.



6. Program je zapsán ve strukturovaném textu, kde v inicializační části naplním všechny položky nějakými hodnotami.

Řešení programu v ST

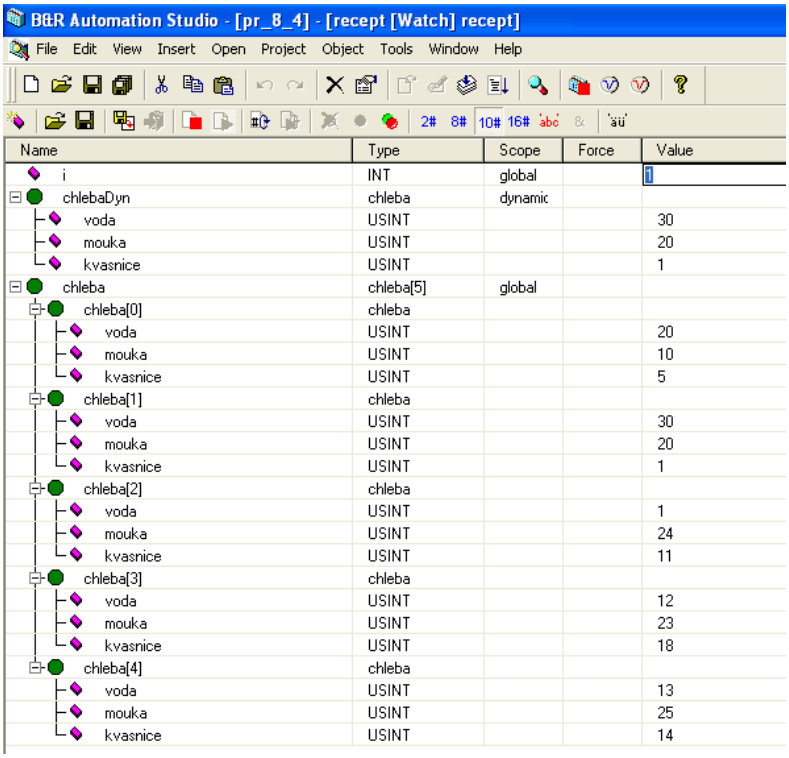
```
0001 (* init program *)
0002 chleba[0].voda:=20;
0003 chleba[0].mouka:=10;
0004 chleba[0].kvasnice:=5;
0005
0006 chleba[1].voda:=30;
0007 chleba[1].mouka:=20;
0008 chleba[1].kvasnice:=1;
0009
0010 chleba[2].voda:=1;
0011 chleba[2].mouka:=24;
0012 chleba[2].kvasnice:=11;
0013
0014 chleba[3].voda:=12;
0015 chleba[3].mouka:=23;
0016 chleba[3].kvasnice:=18;
0017
0018 chleba[4].voda:=13;
0019 chleba[4].mouka:=25;
0020 chleba[4].kvasnice:=14;
```

7. Do cyklické části tasku stačí zapsat tento příkaz.

Řešení programu v ST

```
0001 (* cyclic program *)
0002
0003 chlebaDyn ACCESS ADR (chleba[i]);
```

8. Pro sledování programu si otevřu Watch, kde změnou indexu i - proměnnou se mi mění také struktura dynamické proměnné tj. pokud napíši, že i = 1 obdržím v dynamické proměnné položky stavy proměnných jaké jsou definované ve struktuře chleba[1]. Kdybych potřeboval toto převést do vizualizace, tak zde by mi stačilo pouze vyčítat stavy dynamické proměnné a proměnnou i bych spojil s tlačítkem nahoru a dolu.



Name	Type	Scope	Force	Value
i	INT	global		1
chlebaDyn	chleba	dynamic		
voda	USINT			30
mouka	USINT			20
kvasnice	USINT			1
chleba	chleba[5]	global		
chleba[0]	chleba			
voda	USINT			20
mouka	USINT			10
kvasnice	USINT			5
chleba[1]	chleba			
voda	USINT			30
mouka	USINT			20
kvasnice	USINT			1
chleba[2]	chleba			
voda	USINT			1
mouka	USINT			24
kvasnice	USINT			11
chleba[3]	chleba			
voda	USINT			12
mouka	USINT			23
kvasnice	USINT			18
chleba[4]	chleba			
voda	USINT			13
mouka	USINT			25
kvasnice	USINT			14

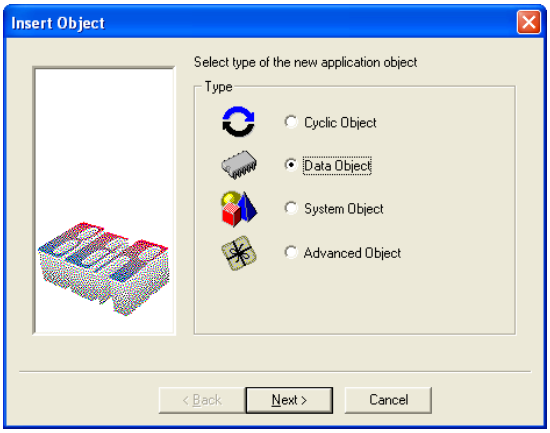
DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 8\pr_8_4.pgd.zip

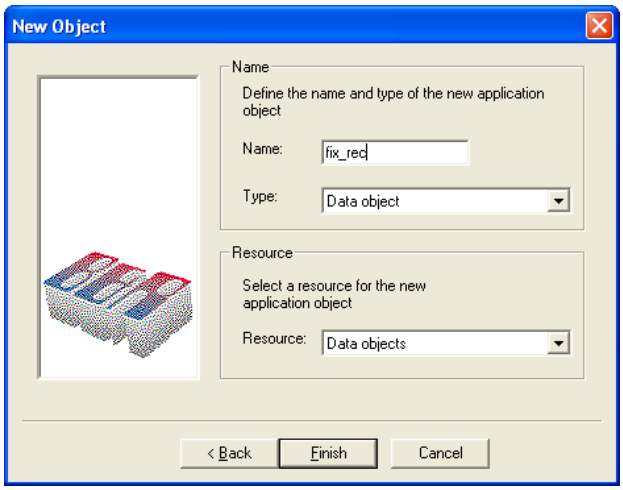
Řešená úloha 8.5.

Vytvořte jednoduchou recepturu pro výrobu chleba, kde receptury pro jednotlivé druhy chleba budou uloženy v datovém objektu. Vyrábět se budou 4 druhy chleba.

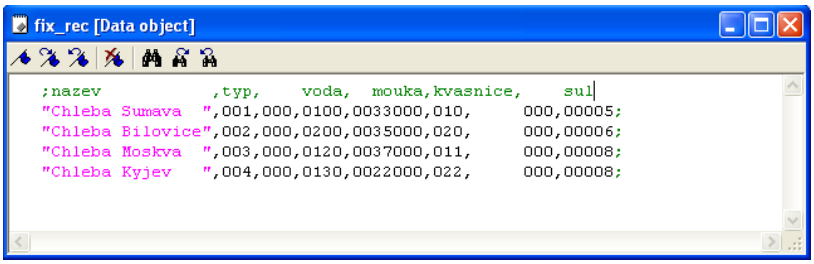
1. Založení nového datového objektu.



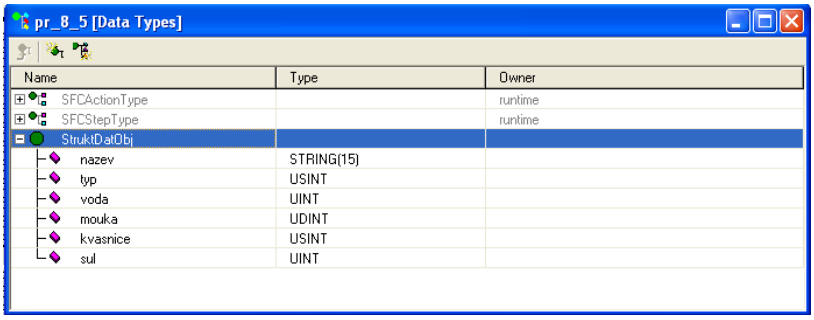
2. Pojmenuji datový objekt např. rec_chl.



3. Výpis receptur do datového objektu fix_rec.



4. Založím stejnou strukturu se stejnými datovými typy jako v receptuře.



5. Založím nový task *data*.

6. V inicializační části se musí nejprve načíst odkaz na datový objekt, kde je v paměti umístěn (proměnná dynModul je dynamická proměnná).

Řešení programu v ST

Structured text : rec_dat

```
0001 (* init program *)
0002 (*Nacteni identifikace datového objektu fix_rec*)
0003 DatObjInfo_0.enable:=1;
0004 DatObjInfo_0.pName:=ADR('fix_rec');
0005 DatObjInfo_0();
0006
0007 ident_fix_rec:=DatObjInfo_0.ident;
0008
0009 (*pouziti dynamicke promenne pro pristup k datovemu objektu v paměti PLC*
0010 dynModul ACCESS DatObjInfo_0.pDatObjMem;
0011
```

7. Cyklická část.

Řešení programu v ST

Structured text : rec_dat

```
0001 (* cyclic program *)
0002 (*cteni z datoveho modulu do promenne readData*)
0003 IF nacti = 1 THEN
0004     DatObjRead_0.enable:=1;
0005     DatObjRead_0.ident:=ident_fix_rec;
0006     DatObjRead_0.Offset:=SIZEOF(readData)*radek;
0007     DatObjRead_0.pDestination:=ADR(readData);
0008     DatObjRead_0.len:=SIZEOF(readData);
0009     DatObjRead_0();
0010
0011     IF DatObjRead_0.status = 0 THEN
0012         nacti:=0;
0013     END_IF
0014 END_IF
0016 (*zapis do datoveho modulu z promenne writeData*)
0017 IF zapis = 1 THEN
0018     DatObjWrite_0.enable:=1;
0019     DatObjWrite_0.ident:=ident_fix_rec;
0020     DatObjWrite_0.Offset:=SIZEOF(writeData)*writeRadek;
0021     DatObjWrite_0.pSource:=ADR(writeData);
0022     DatObjWrite_0.len:=SIZEOF(writeData);
0023     DatObjWrite_0();
0024
0025     IF DatObjWrite_0.status = 0 THEN
0026         zapis:= 0;
0027     END_IF
0028 END_IF
```

8. Výpis proměnných.

Declaration : rec_dat

Declaration

Name	Type	Scope	Attribute	Value	Remark
DatObjCopy_0	DatObjCopy	local	memory	* remanent	
DatObjInfo_0	DatObjInfo	local	memory	* remanent	
DatObjRead_0	DatObjRead	local	memory	* remanent	
DatObjWrite_0	DatObjWrite	local	memory	* remanent	
copyDatObj	BOOL	local	memory	* remanent	
copyData	STRING(20)	local	memory	* remanent	
doERR_DUPOBJECT	UINT	global	constant	20601	
doUSRRAM	USINT	global	constant	3	
dynModul	StruktDatOb	dynamic	memory	* remanent	
dynModulCopy	StruktDatOb	dynamic	memory	* remanent	
identDatObj	UDINT	local	memory	* remanent	
ident_fix_rec	UDINT	local	memory	* remanent	
nacti	BOOL	local	memory	* remanent	
pDatObjMem	UDINT	local	memory	* remanent	
radek	USINT	local	memory	* remanent	
readData	StruktDatOb	local	memory	* remanent	
writeData	StruktDatOb	local	memory	* remanent	
writeRadek	USINT	local	memory	* remanent	
zapis	BOOL	local	memory	* remanent	

9. V proměnné dynModul se zobrazí všechny receptury.

10. Pokud bude nacti = 1 potom se přečte receptura v prvním řádku v datovém objektu tj. radek=0. Změnou řadku se také změní řádek v receptury. Aktuální výpis receptury se bude zobrazovat do promenne readData.

11. Novou hodnotu např. vody mohu změnit pomocí readData.voda:=20;.

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 8\pr_8_5.pgd.zip

9. ZPĚTNOVAZEBNÍ ŘÍZENÍ U PROGRAMOVATELNÝCH AUTOMATŮ SIEMENS S7-300 A BERNECKER RAINER

Čas ke studiu: 2 hodiny

Cíl:

Kapitola se zabývá možnostmi zpětnovazebního řízení u programovatelných automatů Siemens S7-300 a Bernecker Rainer. Na začátku kapitoly je uveden výčet všech nástrojů pro **zpětnovazební řízení** u programovatelného automatu Simatic S7 300. Jedná se zejména o integrované funkce, **Standard PID Control**, **Modular PID Control** a řízení pomocí **funkčních modulů**. Rovněž jsou zde popsány možnosti **automatického nastavování konstant** PID regulátorů pomocí vestavěného bloku v PID Temperature Control a rovněž pomocí samostatného nástroje **PID SelfTuner**.

Další část kapitoly se zabývá zpětnovazebním řízením u automatů Bernecker Rainer a jsou zde popsány základní bloky knihoven **LoopCont** a **LoopConR**.

Výklad

9.1 PID regulátory pro programovatelné automaty SIMATIC S7-300

9.1.1 PID Continuous Control FB 41 “CONT_C“

Základní PID regulátor pro regulování spojitých veličin. Může být použit pro regulaci na konstantní hodnotu, ve více smyčkovém řízení jako kaskádní, poměrový, směsný a poměrový regulátor. Regulátor je založen na PID řídicím algoritmu pro diskrétní regulátor s vzorkovaným analogovým signálem. Regulátor může být doplněn pulzně modulovaným výstupem pro 2/3 stavové řízení proporcionálních řídicích akčních členů. [9-10]

9.1.2 PID Step Control FB 42 “CONT_S“

Regulátor se používá pro řízení procesů s digitální akční veličinou pro integrační akční členy. Používá se jako PI regulátor na konstantní hodnotu nebo v sekundární regulační smyčce jako kaskádní, směsný nebo poměrový regulátor. Regulátor je založen na PI algoritmu pro vzorkovaný signál a je doplněn funkcí pro generování digitálního výstupu z analogového akčního signálu. [9-10]

9.1.3 Pulse Generation FB 43 “PULSEGEN“

Funkční blok, který slouží pro převod analogového signálu na pulzně šířkovou modulaci. S použitím tohoto bloku je možné vytvářet PID dvou nebo tří stavové regulátory s pulzně šířkově modulovaným výstupem. Většinou se používá ve spojení s regulátorem CONT_C. Podle parametrů přiřazených generátorů pulzů, může být vytvořen PID regulátor s třístavovým výstupem nebo s bipolárním či unipolárním dvou-stavovým výstupem. [9-10]

9.1.4 Continuous Temperature Controller FB 58 "TCONT_CP"

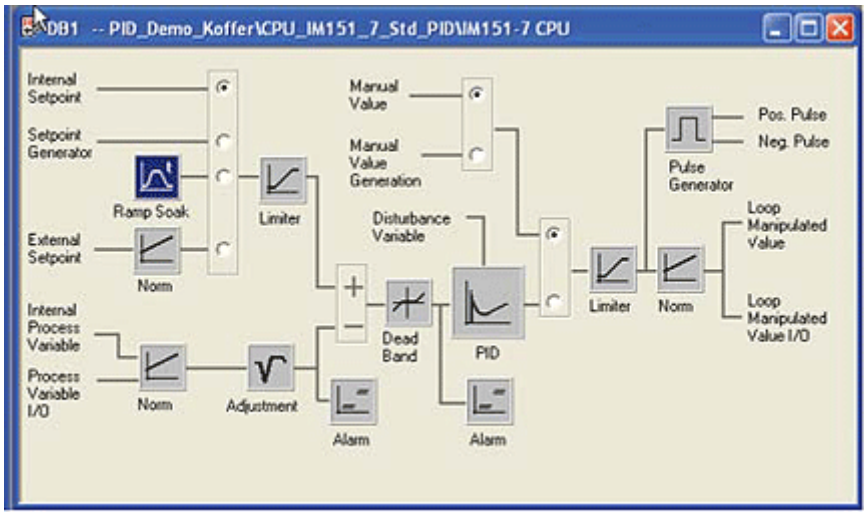
Regulátor je založen na PID algoritmu a je doplněn o přídavné funkce pro teplotní procesy. Regulátor může mít buď analogový výstup nebo pulzně modulovaný výstup. Regulátor může být použit pro regulaci ohřevu nebo chlazení. [9-9]

9.1.5 Temperature Step Controller FB 59 "TCONT_S"

Regulátor se používá pro řízení procesů s digitální akční veličinou pro integrační akční členy. Je založen na PI algoritmu, který je doplněn funkcí pro generování digitálního signálu z analogové hodnoty. [9-9]

9.1.6 Standard PID Control

Standard PID Control je softwarový balík, který se skládá z vlastních funkčních bloků a z parametrizačního nástroje. Balík obsahuje dva funkční bloky, které jsou vhodné pro řídicí úlohy se spojitou i digitální akční veličinou. [9-11]



Obr. 9.1: Instanční datový blok DB1.

9.1.7 Modular PID Control

Modular PID Control je druh výlučně softwarového řešení řídicího algoritmu.

Modular PID Control obsahuje funkční bloky a funkce, které obsahují algoritmy pro vytváření řídicích funkcí. Jedná se softwarový balík, který obsahuje startovací software, 27 standardních funkčních bloků a 12 aplikačních příkladů. Za pomocí těchto bloků a funkce dovoluje programátorovi vytvořit si pro svou aplikaci ten správný regulátor. Knihovna obsahuje různé druhy funkčních bloků a funkci, aby bylo možno vytvářet i rozvětvené regulační obvody. Dále tento software nabízí možnost práce s více smyčkami. Pro tuto možnost je zde loop scheduler, který dovoluje tyto rozsáhlé systémy jednoduše a názorně řídit. [9-12]

9.1.8 PID Self-Tuner

S PID Self- Tunerem lze optimalizovat nastavení regulátoru v systémech SIMATIC S7 a SIMATIC C7. Jednou z možností, kde lze uplatnit PID Self-Tuner je např. v softwarovém balíku Standard PID Control nebo Modular PID Control. Dále lze tento self-tunning uplatnit ve spojení s regulátorem PID Control, který je integrován v prostředí STEP7. Dále lze také tento software uplatnit ve spojení s modulem FM355. [9-13]

9.1.9 Funkční moduly FM 355/455

Pokud je požadavek na nezávislém běhu CPU a přitom regulovat teplotní proces, je možno využít funkční moduly FM 355C/S a FM 455C/S. Jedná se o čtyřkanálový zpětnovazební řídicí modul pro univerzální řídicí úlohy online automatickou optimalizaci. Používá se pro řízení teploty, tlaku, průtoku a hladiny. Významnou výhodou těchto modulů je to, že pokračují v činnosti i při zastavení PLC. [9-14]

9.1.10 Fuzzy Control ++

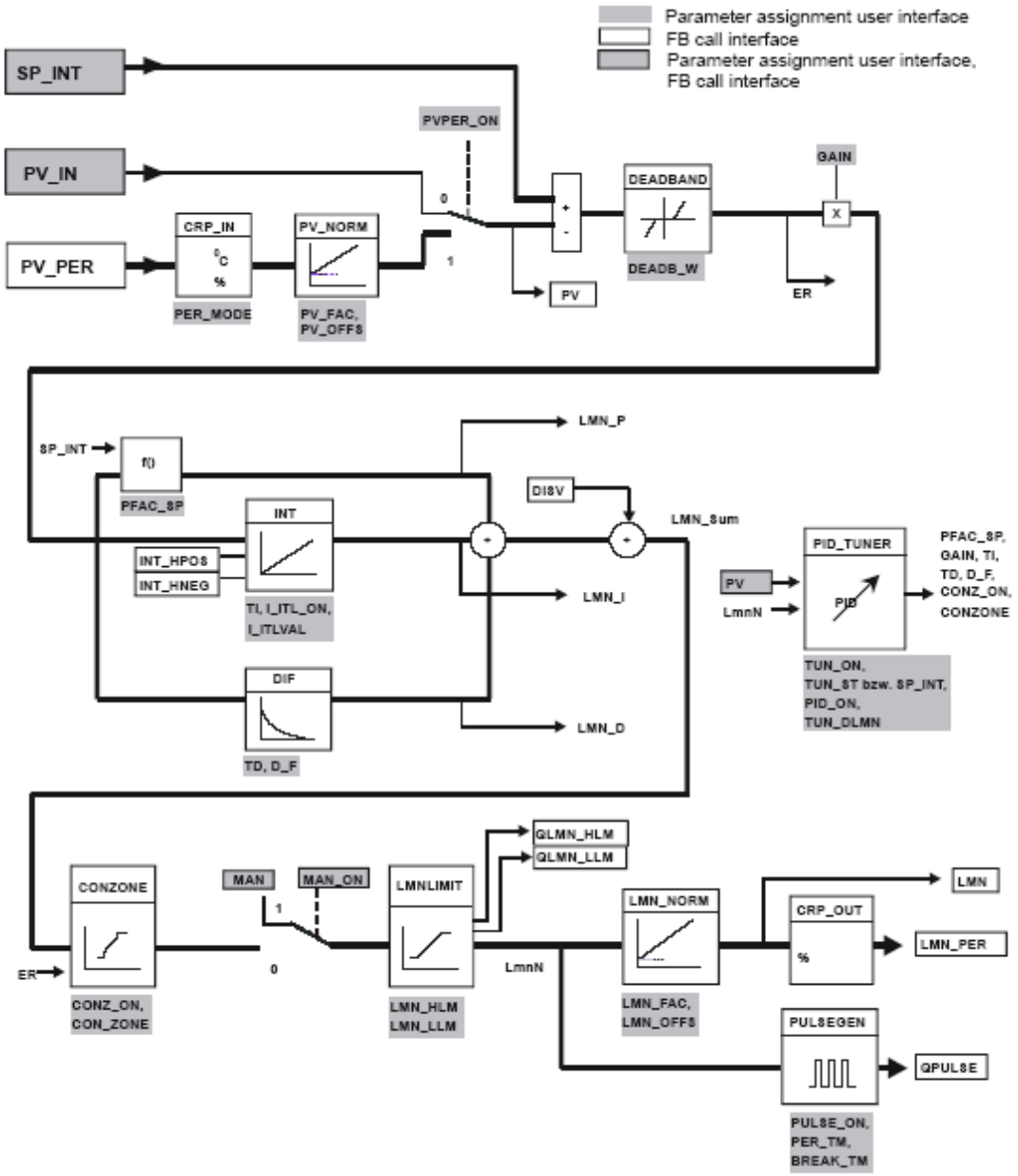
Softwarový nástroj, který umožňuje vytvářet fuzzy aplikace v SIMATICu S7 nebo ve WINCC. Je použitelný na všech úrovních automatizace od samostatného regulátoru až po optimalizaci výroby. Může být kombinován s klasickými PID regulátory a tím lze využít výhod obou systémů pro optimalizaci zpětnovazebního řízení.

9.1.11 NeuroSystems

Slouží pro vytváření a učení neuronových sítí. Je vhodný pro problémy jejich struktura a řešení je známá pouze částečně. Používají se při optimalizaci databází, identifikaci procesů, filtrování dat, při nelineárním jedno a více rozměrném zpětnovazebním řízení, rozpoznávání struktur, v diagnostice apod.

9.2 Continuous Temperature Controller FB 58 “TCONT_CP“

Regulátor je založen na PID algoritmu a je doplněn o přídavné funkce pro teplotní procesy. Regulátor může mít buď analogový výstup nebo pulzně modulovaný výstup. Regulátor může být použit pro regulaci ohřevu nebo chlazení. [16]



Obr.9.2: Blokové schéma Continuous Temperature Controller.

Popis k obrázku 9.2:

- SP_INT (setpoint) Prostřednictvím tohoto vstupu zadáváme regulátoru žádanou hodnotu
- PV_IN (process value) Na tento vstup je přivedena měřená hodnota z procesu . Přivedená hodnota musí být před přiřazením vyjádřena ve fyzikálních jednotkách
- PV_PER Pokud nemáme hodnotu upravenu na fyzikální rozsah, můžeme prostřednictvím tohoto vstupu, tuto hodnotu vhodně upravit za pomocí bloků CRP_IN, PV_NORM
- DISV (disturbance value) Pokud při regulaci na konstantní hodnotu zohledníme také poruchovou veličinu, musíme tuto poruchu přiřadit na tento vstup
- LMN (manipulated value) Na tento výstup připojíme akční člen.
- LMN_PER na tento výstup můžeme , pokud jsme pro načítání process value použili přímé čtení vstupů PIW 272, zapsat také tuto hodnoty přímo na výstup PQW272

Z obrázku je vidět, že jednotlivé větve regulátoru jsou zapojeny paralelně. Toto nám umožňuje vytvářet regulátory typu P, PI, PD, PID.

Pro správnou funkci regulátoru musíme zajistit pravidelné volání funkčního bloku. Toho docílíme použitím cyklického přerušení pomocí organizačního bloku OB35. Přerušení a interval mezi jednotlivými přerušeními, lze nastavit v hardwarové konfiguraci.

Algoritmus výpočtu konstant regulátoru je založen na metodě **Chiena, Hronese a Reswicka** (**CHR**). Tato metoda je upravená metoda Zieglera a Nicholse a je odvozena za předpokladu, že regulovaný systém je popsán přenosem prvního řádu s dopravním zpožděním

$$G_s(s) = \frac{K}{Ts+1} e^{-Tds}$$

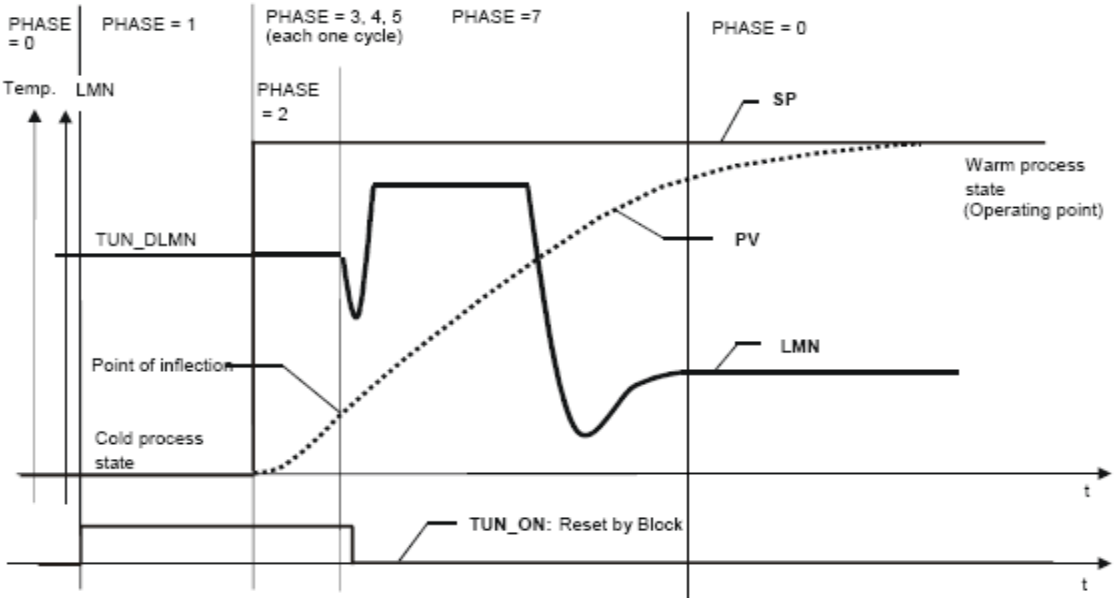
Metoda umožňuje výběr ze dvou variant regulačního pochodu: aperiodický nebo s překmitem 20% a také volbu, zda cílem regulace je sledování změn žádané hodnoty nebo potlačení poruch na vstupu soustavy.

Tab. 9.1: Vztahy pro výpočet nastavení regulátoru.

Regulátor	**Aperiodický regulační pochod**		**Překmit 20%**	
	Žádaná hodnota	**Porucha**	**Žádaná hodnota**	**Porucha**
P	$r_0 = \frac{0{,}3 \cdot T}{k \cdot Td}$	$r_0 = \frac{0{,}3 \cdot T}{k \cdot Td}$	$r_0 = \frac{0{,}7 \cdot T}{k \cdot Td}$	$r_0 = \frac{0{,}7 \cdot T}{k \cdot Td}$
PI	$r_0 = \frac{0{,}35 \cdot T}{k \cdot Td}$ $r_I = 1{,}2 \cdot T$	$r_0 = \frac{0{,}6 \cdot T}{k \cdot Td}$ $r_I = 4 \cdot Td$	$r_0 = \frac{0{,}6 \cdot T}{k \cdot Td}$ $r_I = T$	$r_0 = \frac{0{,}7 \cdot T}{k \cdot Td}$ $r_I = 2{,}3 \cdot T$
PID	$r_0 = \frac{0{,}6 \cdot T}{k \cdot Td}$ $r_I = T$ $r_D = 0{,}5 \cdot Td$	$r_0 = \frac{0{,}95 \cdot T}{k \cdot Td}$ $r_I = 2{,}4 \cdot Td$ $r_D = 0{,}42 \cdot Td$	$r_0 = \frac{0{,}95 \cdot T}{k \cdot Td}$ $r_I = 1{,}35 \cdot T$ $r_D = 0{,}47 \cdot Td$	$r_0 = \frac{1{,}25 \cdot T}{k \cdot Td}$ $r_I = 2 \cdot T$ $r_D = 0{,}42 \cdot Td$

Samočinné nastavení PID regulátoru

Možnost samočinného nastavení regulátoru je jenom u regulátoru typu PI, PID. Volbu regulátoru lze provést parametrem PID_ON. Samočinné nastavení regulátoru lze zahájit jak z manuálního, tak automatického režimu. Poté musíme změnit parametr **TUN_ON** na **TRUE**, touto volbou přejde regulátor z fáze 0 do fáze 1. Poté přibližně počkat minutu a po uplynutí této doby nastavit **TUN_ST** také na hodnotu **TRUE**, čímž regulátor přejde do fáze dva. Ve fázi 1 čekal automat na ustálení hodnoty PV. Pokud přepneme nastavení do fáze dva, automat začne hledat na přechodové charakteristice inflexní bod. Pokud jej najde, vypočte si doby náběhu a průtahu a na základě znalosti dob náběhu a průtahu, vypočte konstanty pro nastavení regulátoru. Tato fáze může být různě dlouhá, závisí to na druhu soustavy. Pokud vše proběhlo správně, měli bychom se dostat do fáze sedm, kde automat přejde do automatického režimu. Průběh samočinného nastavení zachycuje obr.9.3. Důležité jsou dva parametry, které vypovídají o tom, jak se nastavil regulátor. Jedná se o tyto parametry **STATUS_H, STATUS_D**. Pokud se nám regulátor ohodnotí na hodnotu 10000 (STATUS_H=10000), je tento regulátor optimálně nastaven. Ale pokud je STATUS_H roven 2xxxx,3xxxx je nutné samočinné nastavení opakovat. [9-9]



Obr.9.3: Průběh nastavení PID Temperature Control.

9.3 PID Self-Tuner

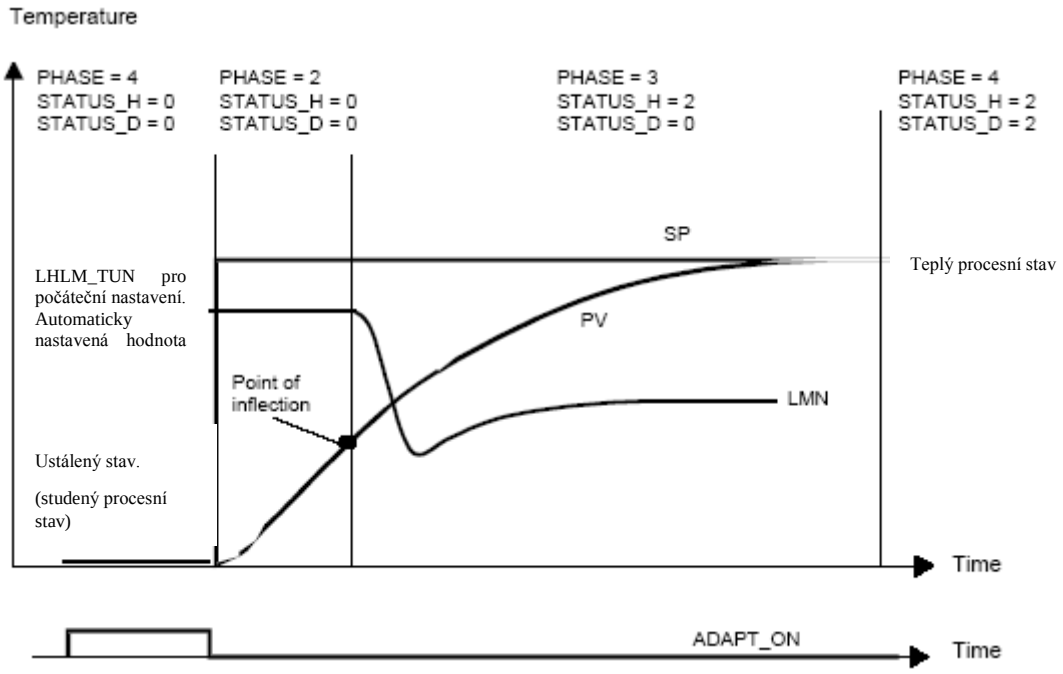
Software PID Self-Tuner nabízí možnost samočinného nastavení regulátoru. Obsahuje dva funkční bloky FB 50 ”TUN_EC“, FB 51 “TUN_ES“. PID Self-Tuner lze také využít pro nastavení konstant PID regulátoru ve spojení s funkčními bloky FB41, FB42, softwarovými balíky Standard PID Control, Modular PID Control a funkčním modulem FM 355C.

Pro možnost nastavení spojitého regulátoru se využívá funkční blok FB 50.

Samočinné nastavení

Samočinné nastavení regulátoru je možné zahájit jak z manuálního režimu, tak automatického. Optimalizace regulátorů se provádí pomocí parametrů **ADAPT_ON** nebo **ADAPT1ST**. Samočinné nastavení regulátoru se zahájí aktivací **ADAPT1ST = TRUE.** Self-Tuner pak nastaví **ADAPT_ON = TRUE**. Proběhne tzv. „první přiblížení“. Self-Tuner použije jako výchozí hodnotu akční veličiny nastavení z parametru **LHLM_TUN** pro skokovou změnu žádané hodnoty (parametr LHLM_TUN je obdoba parametru TUN_DLMN použitý v funkčním bloku FB 58). Pokud již proběhlo první nastavení regulátoru, je možné změnit konstanty on-line. Toho lze docílit aktivací **ADAPT_ON = TRUE** , **ADAPT1ST = FALSE.** Self-Tuner zjistí, jestli má informaci o tom, zdali proběhlo první nastavení a proběhlo-li správně, může pokračovat v on-line přizpůsobení. Pokud zrušíme samočinné nastavení **ADAPT_ON = FALSE** a pokud jsme ve fázi 2 nebo 3 musíme poté přepnout do manuálního řízení. Pokud probíhá samočinné nastavení a jsme např. ve fázi 2 nebo 3 a provedeme požadavek na přizpůsobení nastavením **ADAPT_ON = TRUE**, je samočinné nastavení zrušeno.

Průběh nastavení regulátoru a jednotlivé fáze zobrazuje následující obrázek 9.4.



Obr.9.4: Průběh nastavení Self-Tuneru.

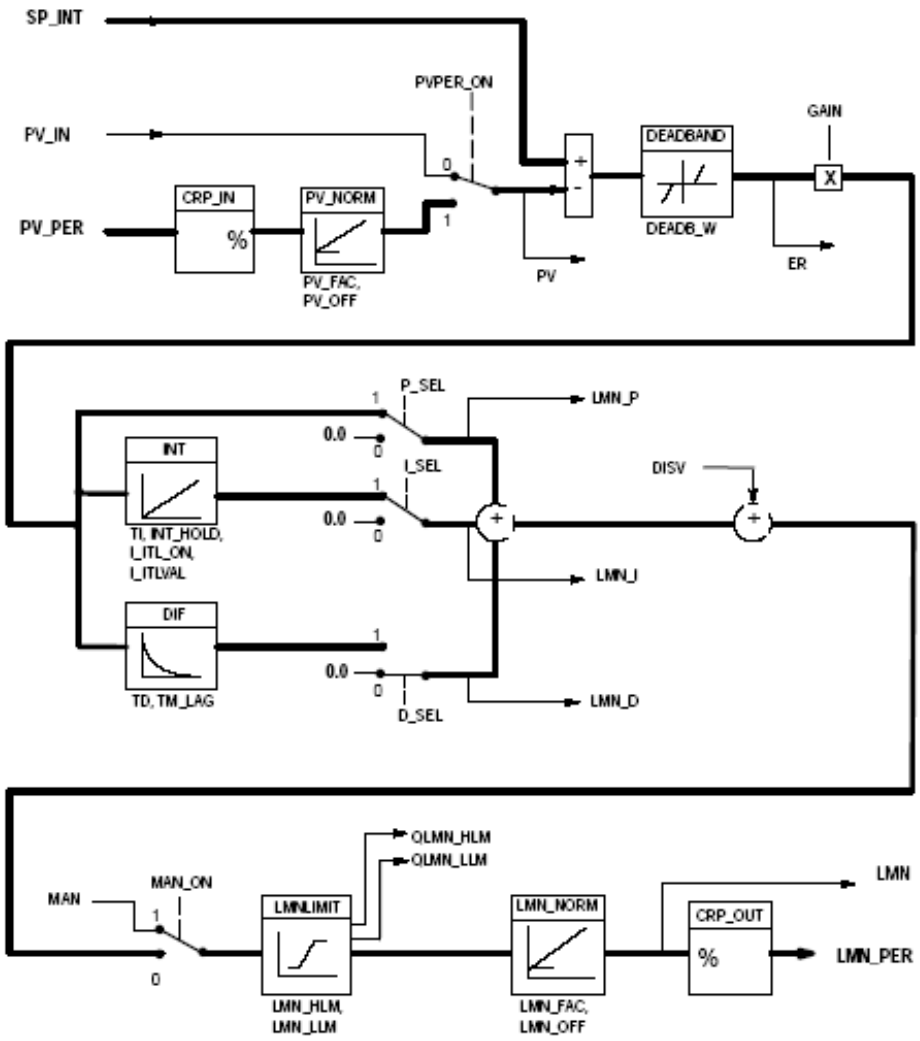
Průběh výpočtu konstant regulátoru:

Samočinné nastavení se může spouštět jak z manuálního tak automatického režimu. Pokud je zvolen manuální režim, změní se také fáze na **7.** Musí se opět počkat přibližně 1 minutu pro ustálení vstupu a poté nastavit žádanou hodnotu a přepnout PID Self-Tuner do automatického režimu nebo nastavit ADAPT_ON = FALSE a zadat žádanou hodnotu. Self-Tuner si jako výchozí hodnotu pro výstup převzal hodnotu z parametru LHLM_TUN. Self-Tuner zahájí hledání inflexního bodu na přechodové charakteristice. Pokud Self-Tuner nalezl inflexní bod, tak se přepne do fáze tři a provádí se výpočet konstant. Nejprve vypočte někdy např. velké zesílení v průběhu fáze 3, ale po dokončení výpočtu je zesílení v rozsahu od 0 do 20, dále se vypočítají časové konstanty regulátoru a tyto nové hodnoty se uloží do parametru PI_CON a staré konstanty se přesunou do PI_CON_OLD, pokud byl zvolen PI regulátor pro self-tunning. Pokud uživatel zadal volbu, že chce provést samočinné nastavení pro PID regulátor tj. **PID_ON = TRUE**, tak Self-Tuner stejně nejprve provádí identifikaci pro PI regulátor a po dokončení samočinného nastavení tj. fáze 4 je možno mezi těmito regulátory přepnout. Stejný postup je i u návrhu PID regulátoru. Regulátor při ukládání nastavení (SAVE_PAR= TRUE) porovná, jestli nyní vypočtené konstanty jsou lepší než ty minulé. Ukládání konstant je detekováno nastavením výstupu LOAD_PAR = TRUE. Pokud zjistí, že konstanty jsou lépe navrženy než při minulém self-tunningu, uloží tyto hodnoty do PI_CON_OLD. Pokud ale nyní vypočtené konstanty jsou horší než konstanty zapsané v parametrech PI_CON_OLD, načte si předešlé konstanty (UNDO_PAR=TRUE).

9.4 PID Control FB 41 “CONT_C“ + PID Self-Tuner FB50 “TUN_EC“

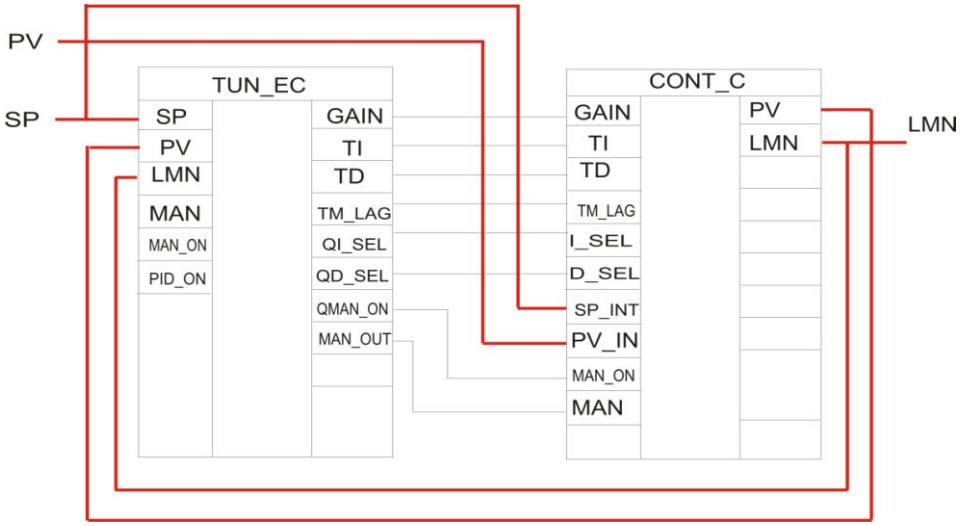
Funkční blok FB 41 “CONT_C“ neposkytuje možnost samočinného nastavení regulátoru, proto se může využít v kombinaci s funkčním blokem FB50, který obsahuje možnost samočinného nastavení regulátoru. Blokové schéma regulátoru ukazuje následující obrázek obr.9.5.

Z obrázku lze vidět, že jednotlivé větve regulátoru jsou zapojeny paralelně. Proto tento funkční blok dovoluje vytvářet regulátory typu P, PI, PD, PID.



Obr.9.5: Blokové schéma regulátoru PID Control.

Aby se zajistila správná funkce PID Control s PID Self-Tunerem, musí se provést spojení bloků podle obr. 9.6 , kde **"TUN_EC"** je funkční blok PID Self-Tuneru a **"CONT_C"** je funkční blok FB41 PID Control PID regulátoru v prostředí STEP7. Pro správnou činnost PID Self-Tuneru platí, že všechny vstupní hodnoty musí být přivedeny nejprve do Self-Tuneru a poté jsou předány přes výstupy PID Self-Tuneru funkčnímu bloku PID Control FB41. Pro načtení hodnoty z analogových vstupů, není možno využít bloku CRP_IN z funkčního bloku FB 41, protože regulovaná veličina musí být nejprve vedena na vstup do Self-Tuneru a až potom do FB 41. Pro možnost načtení můžeme využít funkce **SCALE**.

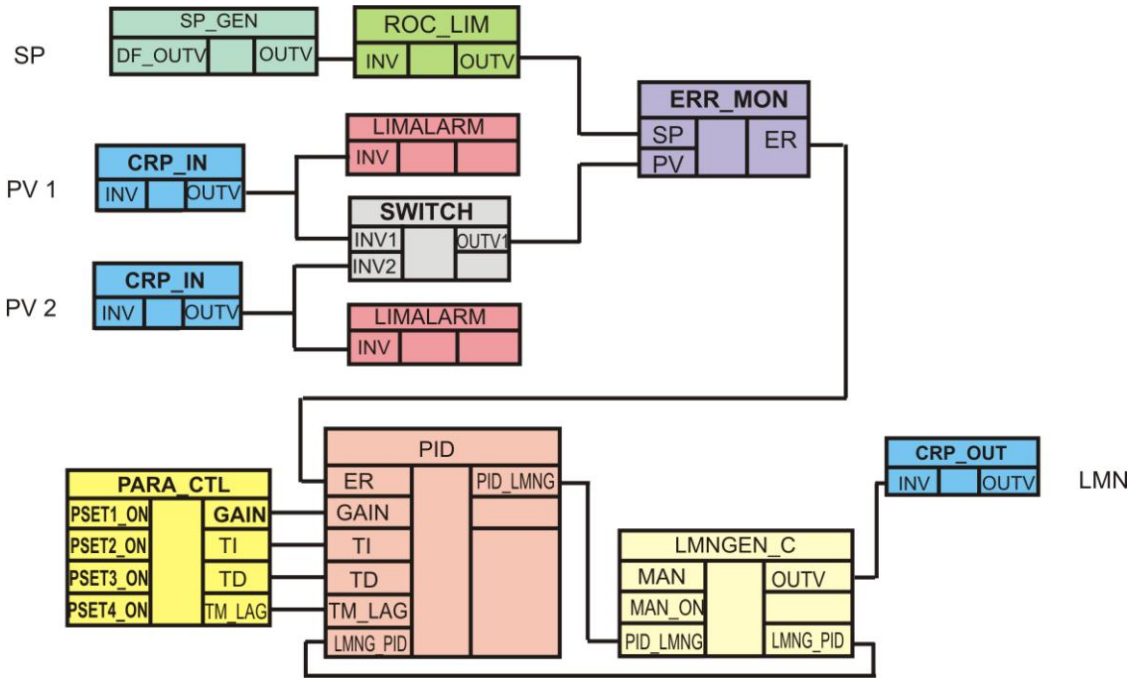


Obr.9.6: Propojení bloku FB 41 s PID Self-Tunerem.

Příchozí žádanou hodnotu je nutno předat jak funkčnímu bloku FB50, tak FB41. Regulovanou veličinu získanou jako reálnou hodnotu je nutno přivést na vstup **PV_IN** do bloku **CONT_C**. Z výstupu bloku **CONT_C** předáme tuto hodnotu na vstup do **TUN_EC** jako **PV**. Akční veličina je opět propojena z výstupu regulátoru **CONT_C** na vstup do **Self-Tuneru LMN**. Předání konstant mezi bloky **TUN_EC** a **CONT_C** je řešeno vzájemným propojením (**GAIN**, **TI, TD, TM_LAG**). Volba regulátoru, pro který se má provést self-tunning, se provádí vstupem **PID_ON** ve funkčním bloku **FB 50**. Proto musí dojít také ke spojení výstupu **QI_SEL, QD_SEL** se vstupem **I_SEL, D_SEL** na funkčním bloku FB41. Těmito výstupy PID Self-Tuner aktivuje popř. deaktivuje větve PID regulátoru **FB 41**. Režim regulátoru se volí vstupem **MAN_ON** na vstupu funkčního bloku **FB50**. Opět musí být spojen výstup **QMAN_ON**, **QMAN** se vstupem **MAN_ON, MAN** na FB41, abychom mohli měnit režim regulátoru z manuálního na automatický.

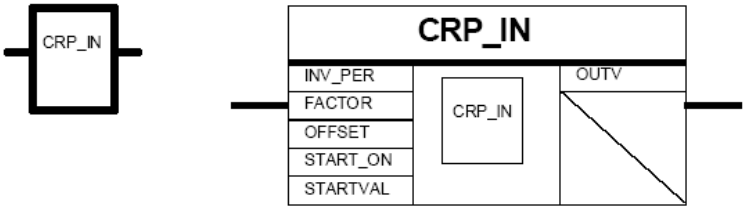
9.5 Modular PID Control

Softwarový balík se skládá z několika funkčních bloků, které je nutné podle požadavků vzájemně pospojovat. Příklad struktury PID regulátoru se dvěma vstupy může vypadat jak ukazuje obr. 9.7.



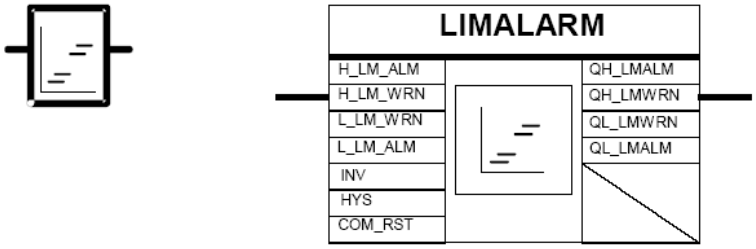
Obr. 9.7: Sestava regulátoru.

Pro načítání hodnoty z tepelné soustavy se může využívat funkční blok **FB 2 CRP_IN**. Funkční blok umožňuje převést vstupní regulovanou veličinu na reálnou hodnotu. Dále funkční blok **CRP_IN** umožňuje simulaci vstupní veličiny, např. když nemáte připojen snímač, si můžete hodnotu zapsat na vstup **STARTVAL** a aktivací vstupu **START_ON = TRUE** se provede simulace vstupu.



Obr. 9.8: Blokové schéma a symbolické vyjádření funkčního bloku **CRP_IN.**

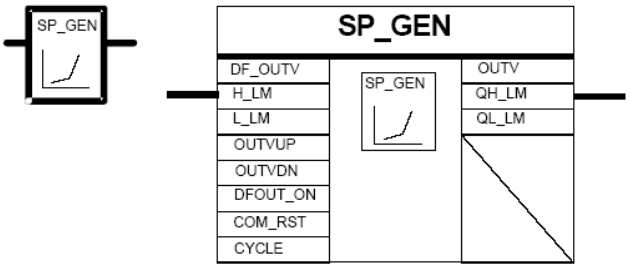
Pro možnost nastavení mezí, ve kterých se mi musí pohybovat regulovaná veličina, slouží funkční blok **FB 11 LIMALARM**. Funkční blok dovoluje nastavit hysterezi, horní a dolní limit alarmů, varování. Dosažení nastavených mezí je signalizováno nastavením výstupů na hodnotu TRUE.



Obr. 9.9: Blokové schéma a symbolické vyjádření funkčního bloku **LIMALARM.**

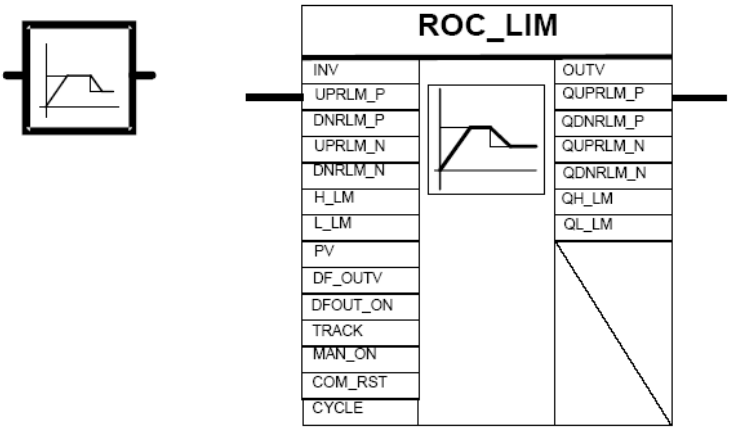
Pomocí funkčního bloku **FB 25 SP_GEN** si můžete vybrat mezi dvěmi možnostmi zadání žádané hodnoty. Žádanou hodnotu můžete přivést na vstup **DF_OUTV** a nastavením vstupu **DFOUT_ON** na

TRUE se provede skoková změna žádané hodnoty. Dále funkční blok **SP_GEN** umožňuje nastavit velikost žádané hodnoty na výstup plynule např. kdyby jste měli na panelu dvě tlačítka pro inkrementaci a dekrementaci hodnoty, tak by použití tohoto bloku bylo určitě výhodné. Musí se provést nastavení horního a dolního limitu žádané hodnoty např. budete požadovat žádanou hodnotu 60. Musíte nastavit horní limit **H_LM** na **60**, dolní limit **L_LM** na **0**. Na vstup **DF_OUTV** nastavíte např. **10.** Tím se provede výchozí nastavení od jaké hodnoty se bude provádět inkrementace popř. dekrementace hodnoty. Aktivací vstupu **DFOUT_ON** se tato hodnota zapíše na výstup. Poté tento vstup deaktivujte a nastavte vstup **OUTVUP** na **TRUE**. Tím se začne plynule měnit žádaná hodnota až na hodnotu 60, kde při dosažení horního limitu sepne výstup **QH_LM** signalizující dosažení horní hranice. S jakou rychlostí se nám mění výstup, ovlivňuje nastavení CYCLE.



Obr. 9.10: Blokové schéma a symbolické vyjádření funkčního bloku **SP_GEN.**

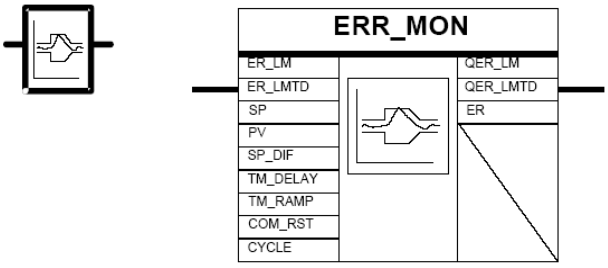
Pro monitorování mezí, ve kterých se musí žádaná hodnota pohybovat slouží funkční blok **FB 22 ROC_LIM**. Tento funkční blok poskytuje také možnost nastavení rampy.



Obr. 9.11: Blokové schéma a symbolické vyjádření funkčního bloku **ROC_LIM.**

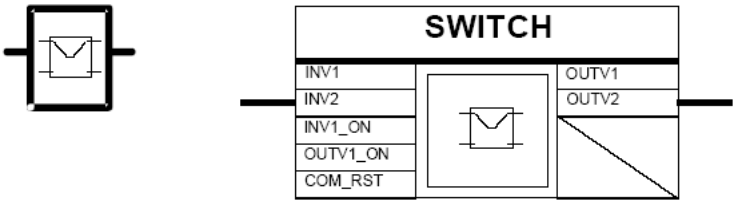
K výpočtu regulační odchylky **ER** můžete využít funkční blok **FB7 ERR_MON.**

Žádaná hodnota je přivedena na vstup **SP** a regulovaná veličina regulovaná veličina je připojena na vstup **PV**



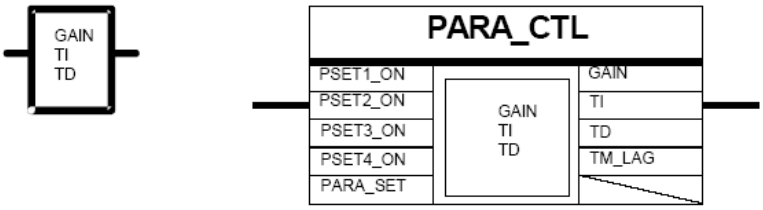
Obr. 9.12: Blokové schéma a symbolické vyjádření funkčního bloku **ERR_MON.**

Pro možnost výběru teplotního snímače můžete využít funkční blok **FB26 SWITCH**. Kombinací vstupů **INV1_ON** a **OUTV_ON** přepínáte vstupy a vstupní hodnota je přivedena např. na výstup **OUTV1**.



Obr. 9.12: Blokové schéma a symbolické vyjádření funkčního bloku **SWITCH.**

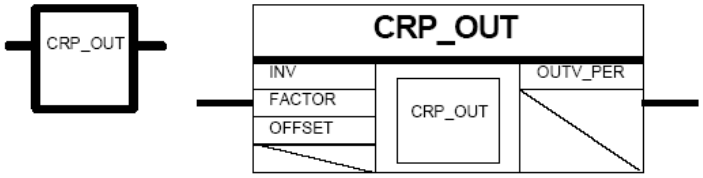
Pro uložení různých konstant PID regulátoru se používá funkční blok **FB 18 PARA_CTL: Parameter Control**.



Obr. 9.13: Blokové schéma a symbolické vyjádření funkčního bloku **PARA_CTL.**

Aktivací vstupů PSET1_ON až PSET4_ON vybíráte mezi jednotlivými konstantami regulátoru. Pokud dva nebo více vstupů jsou nastaveny na TRUE je vybrán vstup, který má nejnižší číslo a tím má největší prioritu. Pokud není žádný spínač sepnut jsou na výstup přesunuty parametry z PARASET1.

Pro zápis na výstup a převod reálné hodnoty na integer se využívá funkční blok **FB 3 CRP_OUT.**



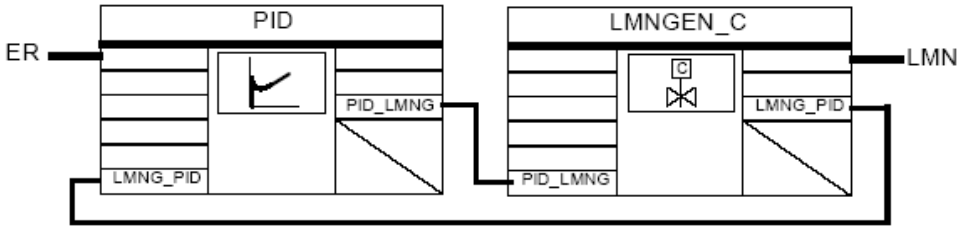
Obr. 9.14: Blokové schéma a symbolické vyjádření funkčního bloku **CRP_OUT.**

PID regulátor

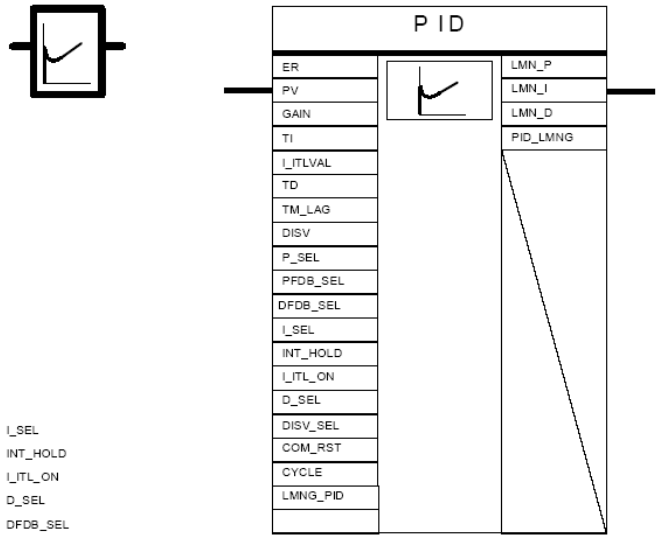
Regulátor obsahuje funkční blok **FB 19 PID** blokové schéma zobrazuje obr.9.16. Pomocí funkčního bloku lze vytvářet regulátory se spojitým výstupem v kombinaci **PID + LMNGEN_C**. Dále je možno sestavit regulátor s pulzním výstupem pro proporcionální akční členy ve spojení **PID + LMNGEN_C + PULSEGEN**. Pro krokové řízení se nabízí spojení **PID + LMNGEN_S**.

Funkční blok dovoluje vytvářet regulátory typu **P, PI, PD, PID. PID** regulátor nemá výstup pro připojení akčního členu, proto se musí vždy nacházet v nějakém spojení buď s funkčním blokem **LMNGEN_C**, nebo **LMNGEN_S**. Funkční blok PID regulátoru obsahuje pouze možnost výběru regulátoru, které je zvolen na základě kombinací vstupů **P_SEL, I_SEL, D_SEL**. Dále zde najdeme povolení monitorování poruchové veličiny. Funkční blok PID regulátoru neobsahuje na rozdíl od funkčního bloku FB 58 možnost výběru manuálního řízení, nastavení limitů akční veličiny a signalizaci jejich překročení. Dále neposkytuje možnost samočinného nastavení regulátoru. Pro možnost manuálního řízení a nastavení rozsahu akční veličiny musím využít funkční blok **FB 13**

LMNGEN_C. Aby si tyto dva funkční bloky předávaly informace o velikosti akční veličiny, je nutno vytvořit níže uvedené spojení podle obr.9.15.

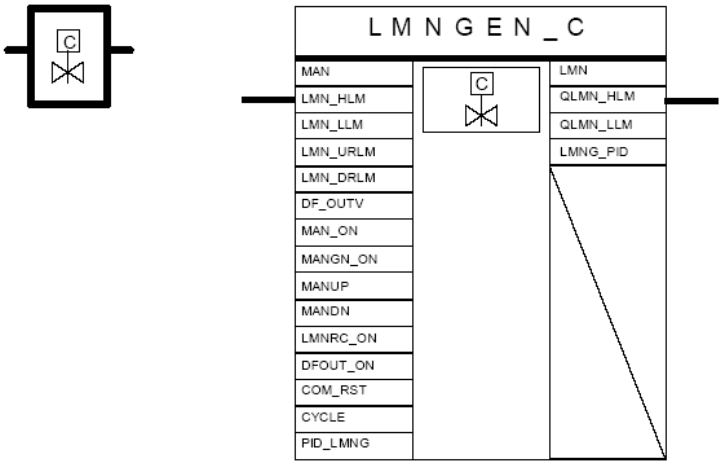


Obr. 9.15: Potřebné zapojení pro úspěšné předání parametrů mezi bloky PID a LMNGEN_C.



Obr.9.16: Blokové schéma a symbolické vyjádření funkčního bloku PID.

Funkční blok **FB13** obsahuje přepínač mezi automatickým a manuálním řízením. V manuálním řízení můžete zadávat velikost výstupní hodnoty. Tento blok je vždy ve spojení s PID regulátorem. Opět i zde můžete vložit manuálně zadané hodnoty skokem nebo plynule. Blokové schéma je znázorněno na níže uvedeném zapojení.



Obr. 9.17: Blokové schéma a symbolické vyjádření funkčního bloku LMNGEN_C.

9.6 PID regulátory v systémech Bernecker Rainer

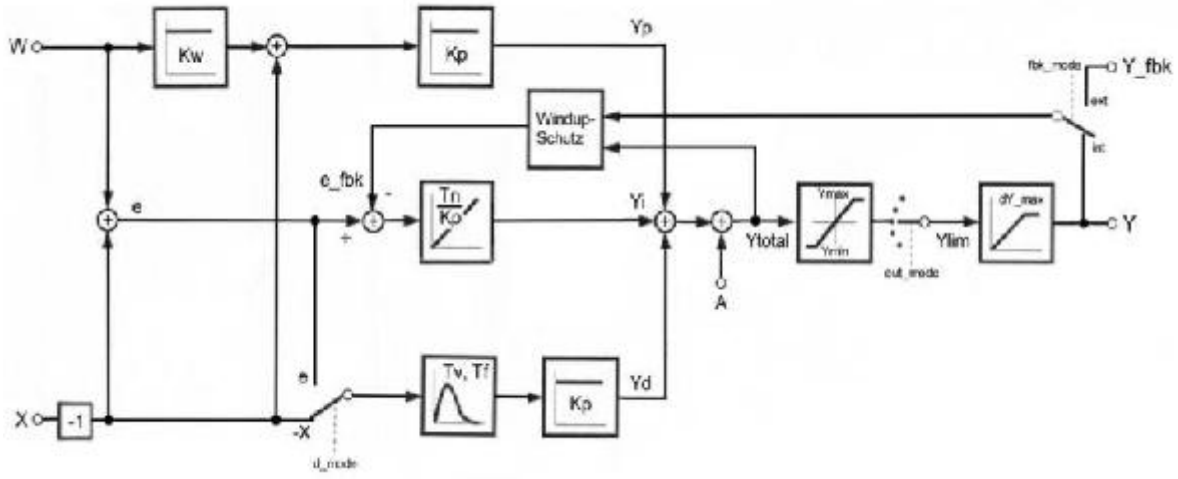
Pro zpětnovazební řízení jsou určeny knihovny LoopCont a LoopConR. Rozdíl mezi těmito dvěma knihovnami je ten, ve kterém systému budou použity. Knihovna LoopConR počítá přesněji ve FLOATech a je možné tuto knihovnu pouze využít v systémech generace SG4. Knihovna LoopCont počítá rychleji v INTegerech a proto se může pouze použít v systémech SG3 [9-7].

Protože na učebně NK 317 jsou zakoupeny B&R Systémy 2003 s CP 430-60-1, které patří do generace SG3, bude se dále přednáška zabývat použitím PID regulátoru, které pracují s datovými typy Integer. Pokud by jste pracovali s automatem s CP360, který patří do generace SG4 a využívá procesor Intel, mohli by jste také využívat knihovnu LCRPID().

Pro práci s PID regulátory slouží základní funkční bloky LCPID(), LCPIDpara(), LCPIDAutoTune().

- LCPIDpara() slouží pro nastavení mezí, konstant PID regulátoru, apod.
- LCPID() je základní funkční blok pro práci s PID regulátorem.
- LCPIDAutoTune() je funkční blok, který provádí samočinné nastavení PID regulátoru.

Blokové schéma PID regulátoru popisuje obr. 9.18. Vysvětlení jednotlivých symbolů v obrázku je vypsáno do tabulky 9.2.



Obr. 9.18.: Schéma zapojení PID regulátoru.

Tab 9.2: Vysvětlení jednotlivých symbolů z obrázku 9.18.

Symbol	Popis	Symbol	Popis
S		**Tf**	Algoritmus D členu obsahuje derivační zpoždění, které může být přiřazeno touto proměnnou.
e	Regulační odchylka	**Ymax**	Horní limit akční veličiny.
Y_fbk		**Ymin**	Dolní limit akční veličiny.
Yp	Tímto výstupem získáme proporcionální hodnotu akční	**dY_max**	Definice přírůstku pro funkci rampy.

	veličiny.		
Yi	Tímto výstupem získáme integrační hodnotu akčníveličiny	**d_mode**	Volba druhu regulace.
Yd	Tímto výstupem získáme derivační hodnotu akční veličiny	**out_mode**	Režim regulátoru.
Kw		**fbk_mode**	Řízení zpětnovazebního spojení.

PID regulátor dovoluje nastavit tzv. „bezpečnou hodnotu při startu“. Nastavením této hodnoty můžete nastavit při zavolání této funkce např. výstup na 100%. Tzn. že nejprve se nastaví výstup na 100% a až poté se provede regulace a výstup se může třeba nastavit na 10%. Použitím této funkce si musíte být jisti, že si to můžete dovolit. Proto je vhodné ve většině případů tuto hodnotu nastavit na 0.

Protože se také může na vstup X superponovat rušivý signál můžete jej filtrovat nastavením proměnné Tf.

Regulátor umožňuje také natavit meze, ve kterých se bude pohybovat akční veličina Y. Podle hodnoty výstupní proměnné status u PID regulátoru je možné signalizovat dosažení mezí akční veličiny. PID regulátor má integrovanou funkci rampy dY_max tzn. že např. výstup se nebude skokově měnit, ale bude zpožděn o dobu nastavenou na vstupním parametru dY_max. Rychlost přírůstku akční veličiny Y je nastavena na hodnotu 1/unit of time.

Regulátor může být také provozován v automatickém a manuálním režimu. Pro manuální řízení musí být také definována tato manuální hodnota Y_man.

Protože v programu může být těchto funkčních bloků LCPIDpara() použito více, musí se vytvořit spojení těchto funkčních bloků LCPIDpara() s LCPID(). K tomu slouží výstupní parametr ID u LCPIDpara() a vstupní parametr u LCPID().

Někdy může být výhodnější regulace na konstantní hodnotu žádané veličiny nebo regulace na konstantní hodnotu regulační odchylky. Pro toto slouží vstup d_mode u funkčního bloku LCPIDpara().

9.6.1 Funkční blok LCPIDpara()

Tento funkční blok (dále FBK) slouží pro nastavení hodnot daného PID regulátoru a musí být volán dříve než funkční blok LCPID(). Popis jednotlivých vstupů a výstupů popisuje následující tabulka 9.3.

Tab. 9.3.:Vstupně/výstupní parametry funkčního bloku LCPIDpara().

I/O	Parametr	Datový typ	Popis
I	Enable	BOOL	Povolení vykonání tohoto FBK
I	enter	BOOL	Povolení převzetí parametrů
I	Y_max	INT	Horní limit akční veličiny
I	Y_min	INT	Dolní limit akční veličiny
I	dY_max	REAL	Maximální velikost funkce rampy, pokud 0 rampa je deaktivována
I	Kp	REAL	Zesílení
I	Tn	REAL	Integrační složka
I	Tv	REAL	Derivační složka
I	Tf	REAL	Algoritmus D členu obsahuje derivační zpoždění, které může být

			přiřazeno touto proměnnou.
I	Kw	REAL	útlum
I	Kfbk	REAL	
I	Fbk_mode	USINT	Výběr druhu zpětné vazby
I	d_mode	USINT	Volba druhu regulace
I	calc_mode	USINT	Druh výpočtu
O	Status	UINT	Chyba funkčního bloku
O	ident	UDINT	ID pro LCPID()

Příklad zápisu v B&R Automation Basic

LCPIDpara_1.enable=

LCPIDpara_1.enter=

LCPIDpara_1.Y_max=

LCPIDpara_1.Y_min=

LCPIDpara_1.dY_max=

LCPIDpara_1.Kp=

LCPIDpara_1.Tn=

LCPIDpara_1.Tv=

LCPIDpara_1.Tf=

LCPIDpara_1.Kw=

LCPIDpara_1.Kfbk=

LCPIDpara_1.fbk_mode= LCPID_FBK_MODE_INTERN

LCPIDpara_1.d_mode= LCPID_D_MODE_E

LCPIDpara_1.calc_mode= LCPID_CALC_MODE_EXACT

LCPIDpara_1 FUB LCPIDpara()

id = LCPIDpara_1.ident

9.6.2 Funkční blok LCPID ()

Funkční blok LCPID() je základní blok pro práci s PID regulátory v B&R PLC.

Popis jednotlivých vstupů a výstupů popisuje následující tabulka 9.5.

Tab. 9.5: Vstupně/výstupní hodnoty funkčního bloku LCPID().

I/O	**Parametr**	**Datový typ**	**Popis**
I	enable	BOOL	Povolení vykonání tohoto bloku
I	ident	UDINT	ID pro LCPIDpara()
I	W	INT	Žádaná hodnota

I	X	INT	Regulovaná veličina
I	A	INT	Výchozí hodnota při startu FBK
I	Y_man	INT	Velikost akční veličiny při manuálním režimu
I	hold_I	BOOL	Režim HOLD
I	out_mode	USINT	Režim regulátoru
I	Basetime	UDINT	Hodnota je odvozena od použití FBK LCCouter()
O	Status	INT	Status FBK
O	e	DINT	Regulační odchylka
O	Y	INT	Akční veličina
O	Yp	DINT	Tímto výstupem získáme proporcionální hodnotu manipulované veličiny
O	Yi	DINT	Tímto výstupem získáme integrační hodnotu manipulované veličiny
O	Yd	DINT	Tímto výstupem získáme derivační hodnotu manipulované veličiny

Příklad zápisu v B&R Automation Basic

LCPID_0.enable=

LCPID_0.ident = id

LCPID_0.W=

LCPID_0.X=

LCPID_0.A=

LCPID_0.Y_man=

LCPID_0.Y_fbk=

LCPID_0.hold_I=

LCPID_0.out_mode = LCPID_OUT_MODE_AUTO

LCPID_0.basetime = BaseTime

LCPID_0 FUB LCPID()

vyst = LCPID_0.Y

PIDstatus=LCPID_0.status

Shrnutí pojmů

Zpětnovazební řízení je u programovatelných automatů velice často využívané. Používají se zejména PID regulátory a většina výrobců nabízí několik variant těchto regulátorů pro své programovatelné automatu. U automatu Siemens Simatic S7 300 je možné využít následující základní možnosti:

- **Integrované funkce** z knihoven PID Control a PID Temperature Control
- **Standard PID Control**

- **Modular PID Control**
- **Funkční moduly** – hardwarové řešení PID regulátoru pomocí přídavného modulu.

Pro správné nastavení konstant regulátorů jsou k dispozici nástroje pro **automatické ladění – self-tunning**. Jedná se zejména o nástroj PID Self Tuner.

U programovatelných automatů B&R jsou k dispozici funkce pro realizaci zpětnovazebního PID řízení v **knihovnách LoopCont a LoopConR**. Základní funkce z těchto knihoven pro realizaci PID řízení jsou LCPID(), LCPIDpara(), LCPIDAutoTune().

Kontrolní otázky

1. Popište možnosti realizace PID zpětnovazebního řízení u programovatelných automatů Simatic.
2. Popište realizaci PID řízení pomocí integrovaných funkcí z knihovny PID Control.
3. Popište realizaci PID řízení pomocí integrovaných funkcí z knihovny PID Temperature Control.
4. Popište možnosti automatického nastavení konstant u funkcí z knihovny PID Temperature Control.
5. Popište nástroje Standard PID Control a Modular PID Control.
6. Popište základní vlastnosti nástroje PID Self Tuner.
7. Co jsou to funkční moduly, jak je lze využít pro zpětnovazební řízení?
8. Jaké jsou možnosti realizace PID zpětnovazebního řízení u programovatelných automatů B&R?
9. Popište funkce LCPID(), LCPIDpara() a LCPIDAutoTune().

Další zdroje

9-1. B&R training document: The Basics of Automation Studio TM210.

9-2. B&R training document: Automation Studio Online Communication TM211.

9-3. B&R training document: Automation Runtime TM213.

9-4. B&R training document: The Service Technician on the Job TM220.

9-5. B&R training document: Memory Management and Data Storage TM250.

9-6. B&R training document: Automation Studio Libraries I TM260.

9-7. B&R training document: Closed Loop Control with LOOPCONR TM261.

9-8. B&R Automation Studio 2.5.2.21, Help.

9-9. Siemens: SIMATIC PID Temperature Control, 12/2003, A5E00125039-02.

9-10. Siemens: SIMATIC Standard Software for S7-300 and S7-400 PID Control, C79000-G7076-C516-01.

9-11. Siemens: SIMATIC Standard PID Control, 03/2003, A5E00204510-02.

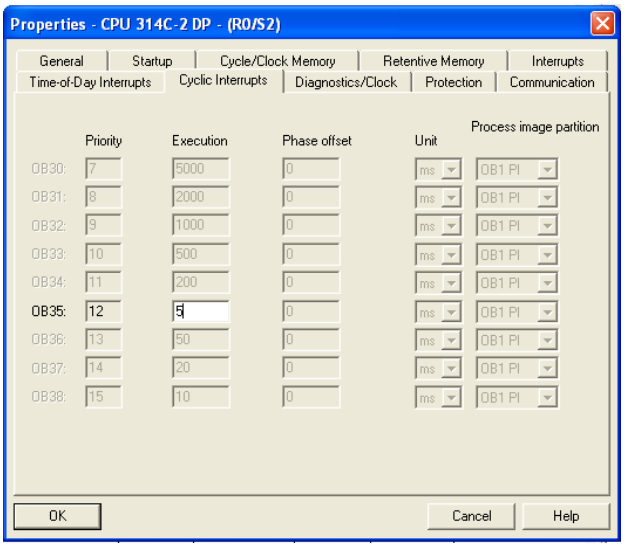
9-12. Siemens: SIMATIC Modular PID Control, 11/2003, A5E00275589-01.

9-13. Siemens:SIMATIC PID Self-Tuner V5, 12/99, C79000-G7076-C825, Edition 3.

9-14. Siemens: SIMATIC Controller Module FM 355, 02/2000, A5E00059344, Edition 02.

Řešená úloha 9.1.

Vytvořte program, který bude generovat signály o frekvenci 10Hz, 50Hz, 100Hz. Výstup Q124.0 bude blikat s frekvencí 100Hz, výstup Q124.1 bude blikat s frekvencí 50Hz, výstup Q 124.2 bude blikat s frekvencí 10Hz. Pro generování signálů využijte organizační blok OB35 – časové přerušení.



Řešení programu v LAD

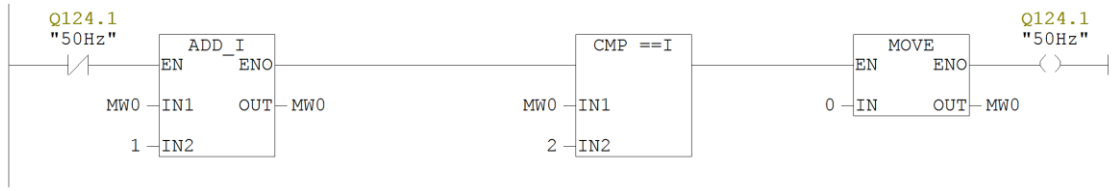
Block: OB35 "Cyclic Interrupt"

Network: 1

zde se generuje signal o frekvenci 100Hz

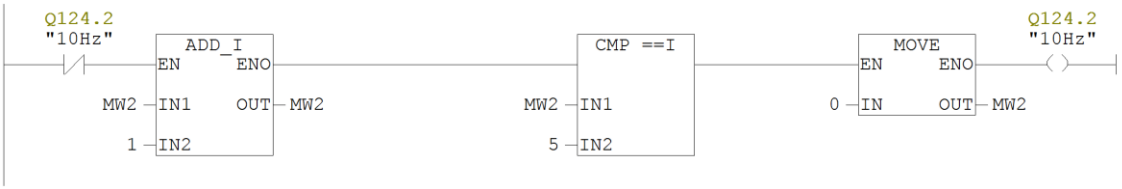
Network: 2

zde se generuje signal o frekvenci 50Hz



Network: 3

zde se generuje signal o frekvenci 10Hz



Řešení programu v STL

```
Network: 1

zde se generuje signal o frekvenci 100Hz

      AN    "100Hz"  Q124.0
      =     "100Hz"  Q124.0

Network: 2

zde se generuje signal o frekvenci 50Hz

      A(
      AN    "50Hz"    Q124.1
      JNB   _001
      L     MW     0
      L     1
      +I
      T     MW     0
      AN    OV
      SAVE
      CLR
_001: A     BR
      )
      A(
           L     MW     0
           L     2
           ==I
           )
           JNB   _002
           L     0
           T     MW     0
           SET
           SAVE
           CLR
     _002: A     BR
           =     "50Hz"    Q124.1

Network: 3

zde se generuje signal o frekvenci 10Hz

      A(
      AN    "10Hz"    Q124.2
      JNB   _003
      L     MW     2
      L     1
      +I
      T     MW     2
      AN    OV
      SAVE
      CLR
_003: A     BR
      )
      A(
      L     MW     2
      L     5
      ==I
      )
      JNB   _004
      L     0
      T     MW     2
      SET
      SAVE
      CLR
_004: A     BR
      =     "10Hz"    Q124.2
```

DVD-ROM

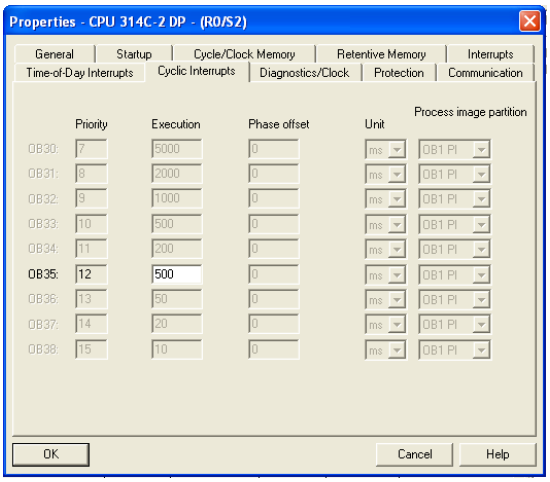
Řešený příklad naleznete na DVD: cvičení\cvičení 9\pr_9_1.zip

Řešená úloha 9.2.

Vytvořte program s PID regulátor FB41 „CONT_C“, který bude načítat data z prvního analogového vstupu analogové karty, která je součástí CPU 314C-2DP a bude zapisovat analogovou hodnotu na první analogový výstup. Vstupní analogovou hodnotu upravte pomocí bloků CRP_IN a PV_NORM, které jsou součástí FB41. Zajistěte periodické volání (500ms) tohoto PID regulátoru. Nastavte optimálně konstanty PID regulátoru.

Slot	Module	...	O...	Fi...	M...	I address	Q address	Comment
1								
2	CPU 314C-2 DP		6ES7	V2.0	2			
X2	DP					1023*		
2.2	DI24/DO16					124...126	124...125	
2.3	AI5/AO2					752...761	752...755	
2.4	Count					768...783	768...783	
2.5	Position					784...799	784...799	

Hardwarová konfigurace



Nastavení vzorkovací periody

Řešení programu v LAD

```
Block: OB100    "Complete Restart"

Network: 1      complete restart
Nastaveni bitu v DB, ktery resetuje regulator

                                        DB41.DBX0.
                                            0
                                         complete
                                         restart
                                        "DP_PID".
   M0.0                                  COM_RST
  ----|/|-------------------------------------(S)----
```

Network: 2 process variable peripherie on

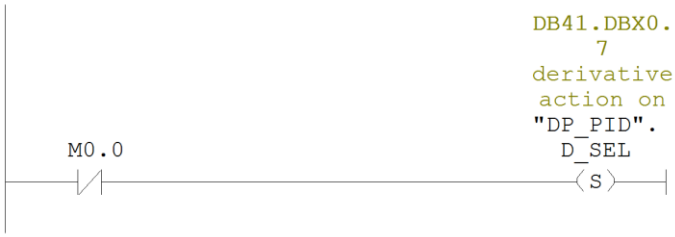
Nastaveni bitu PVPER_ON = 1, protoze bude nacitat ze vstupu PV_PER viz blokove schema PID regulatoru

Network: 3 manual value on

Nastaveni automatickeho rezimu

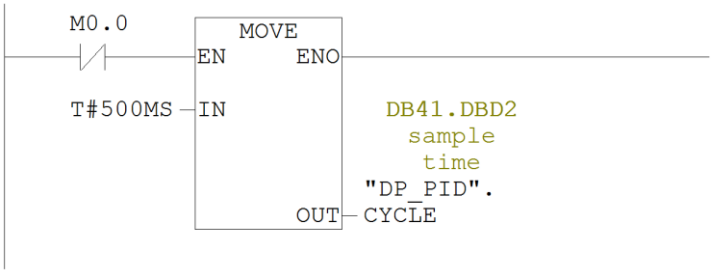
Network: 4 derivative action on

aktivace derivacni slozky protoze to ma byt PID
D slozka neni defaultne nastavena

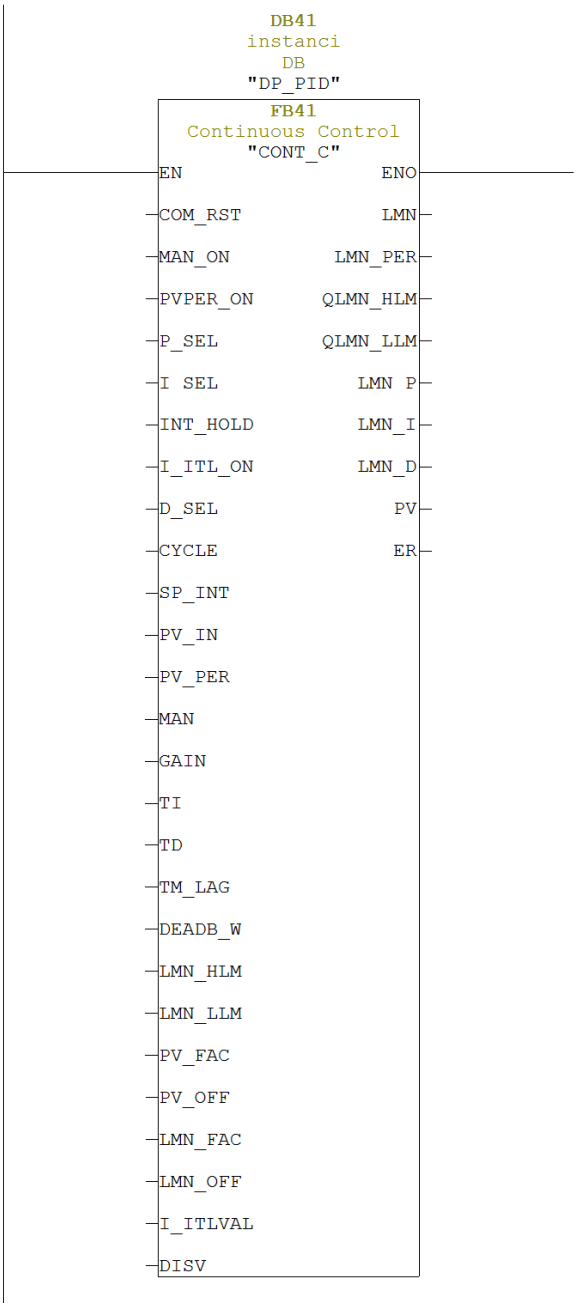


Network: 5

Nastaveni doby cyklu PID regulatoru



```
Network: 6

Zavolani FB PID regulatoru pro zapsani zmeny v nastaveni
nyni se uplatni restart regulatoru
                nastaveni automatickeho rizeni
                nastaveni casove zakladny
                D-slozky
```

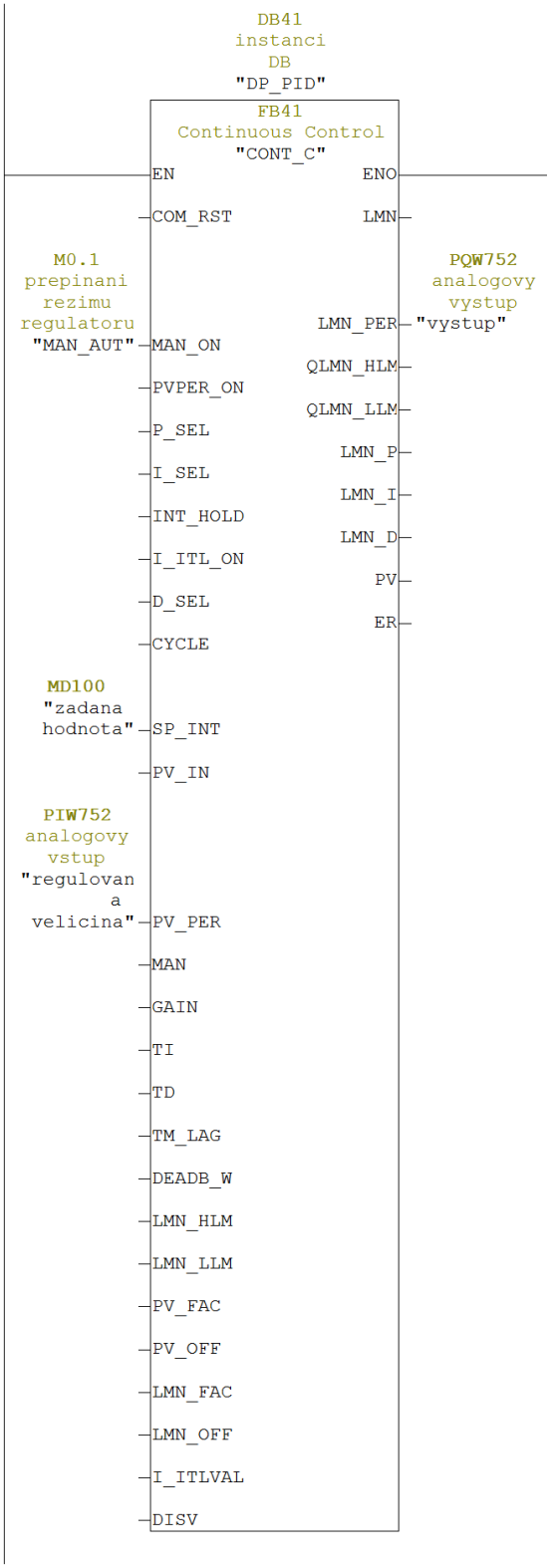



```
Network: 7      complete restart

zruseni restartu PID regulatoru
```

Block: OB35 "Cyclic Interrupt"

Network: 1

funkci blok PID regulatoru



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 9\pr_9_2.zip

DVD-ROM

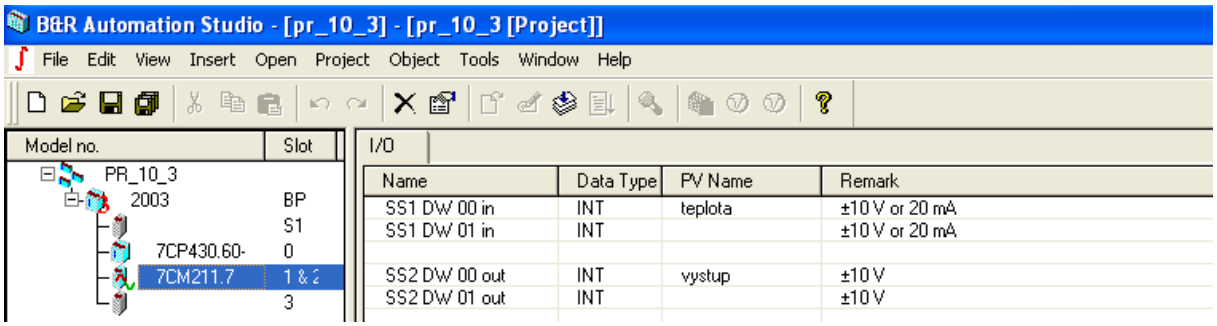
K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 9\ anim1.avi.

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 9\ anim2.avi.

Řešená úloha 9.3.

Vytvořte PID regulátor pro B&R 2003. Analogová hodnota je čtena z prvního analogového napěťového vstupu 0-10V. Akční veličina bude zapisována na první analogový výstup. Program umožňuje měnit žádanou hodnotu a přepínat režim regulátoru manuální a automatický.



Řešení programu v Automation Basicu

B&R Automation Basic : pid1

```
0001 (* init program *)
0002
0003 (*LCPIDpara nastaví rozsirene hodnoty PID regulatoru*)
0004 LCPIDpara_0.enable=true
0005 LCPIDpara_0.enter=true
0006 LCPIDpara_0.Y_max=32760
0007 LCPIDpara_0.Y_min=0
0008 LCPIDpara_0.dY_max=0
0009 LCPIDpara_0.Kp=8
0010 LCPIDpara_0.Tn=50
0011 LCPIDpara_0.Tv=0
0012 LCPIDpara_0.Tf=0
0013 LCPIDpara_0.Kw=1
0014 LCPIDpara_0.Kfbk=0
0015 LCPIDpara_0.fbk_mode= LCPID_FBK_MODE_INTERN
0016 LCPIDpara_0.d_mode=LCPID_D_MODE_E
0017 LCPIDpara_0.calc_mode=LCPID_CALC_MODE_FAST
0018 LCPIDpara_0 FUB LCPIDpara()
0019 id = LCPIDpara_0.ident
0020 statusPara = LCPIDpara_0.status
```

B&R Automation Basic : pid1

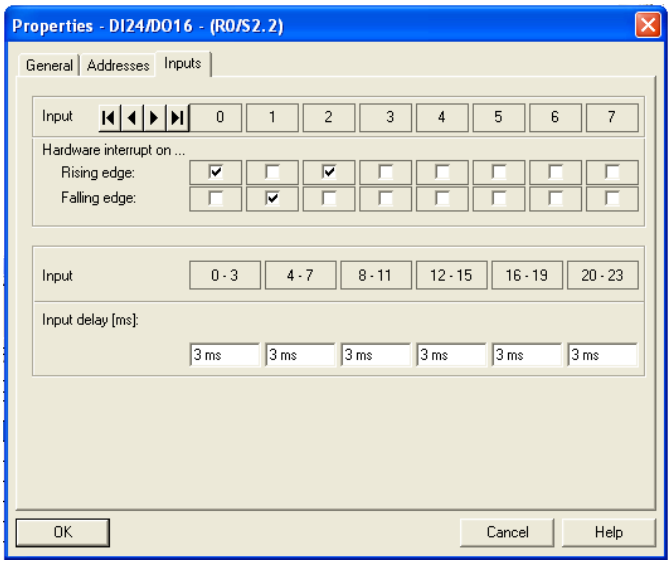
```
0001 (* cyclic program *)
0002 (*nastaveni casove zakladny*)
0003 LCCounter_0 FUB LCCounter()
0004 BaseTime = LCCounter_0.mscnt;
0005
0006 (*volba rezimu regulatoru manualni automaticky*)
0007 if man_on = true then
0008 LCPID_0.out_mode=LCPID_OUT_MODE_MAN
0009 else
0010 LCPID_0.out_mode=LCPID_OUT_MODE_AUTO
0011 endif
0012
0013 (*blok PID regulatoru*)
0014 LCPID_0.enable=true
0015 LCPID_0.ident=id
0016 LCPID_0.W=SP_INT
0017 LCPID_0.X=teplota
0018 LCPID_0.A=0
0019 LCPID_0.Y_man=27000
0020 LCPID_0.Y_fbk=1
0021 LCPID_0.hold_I=false
0022 LCPID_0.basetime=BaseTime
0023 LCPID_0 FUB LCPID()
0024 vystup = LCPID_0.Y
0025 PIDstatus = LCPID_0.status
```

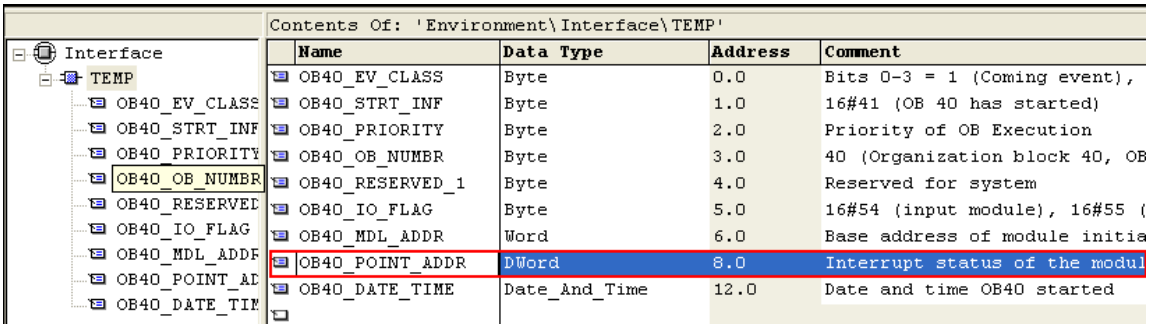
DVD-ROM

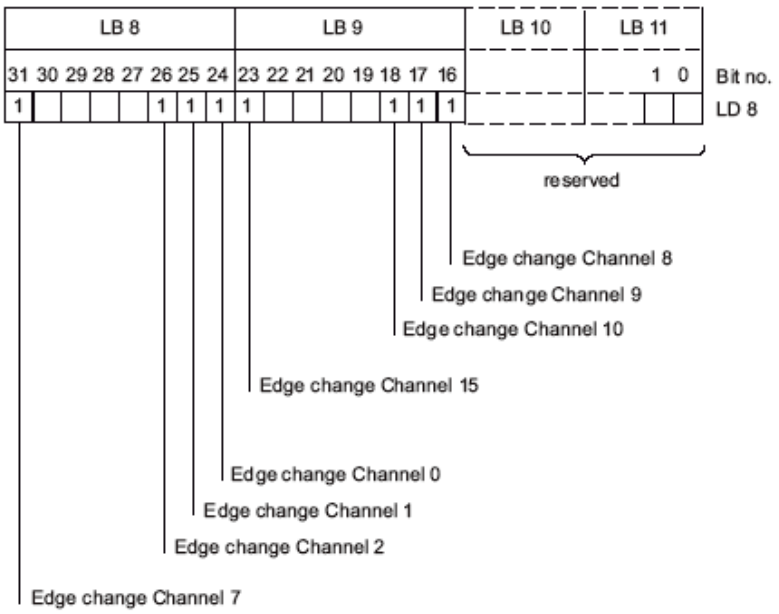
Řešený příklad naleznete na DVD: cvičení\cvičení 9\pr_9_3.pgd.zip

Řešená úloha 9.4.

Vytvořte program, který přerušovacím vstupem I124.0, reagujícím na náběžnou hranu signálu, na kompaktním PLC 314C-2PtP vyvolá přerušení, ve kterém se zvýší hodnota proměnné MW100 o 1. Pokud nastavené přerušení na vstupu I124.1 reagující na sestupnou hranu signálu proveďte dekrementaci hodnoty MW102 o 2 a pokud nastane přerušení na vstupu I124.2 (náběžnou hranou) nastavte bit Q124.0 na hodnotu 1.



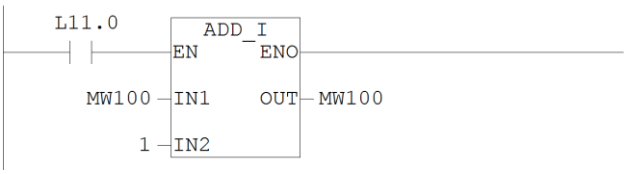




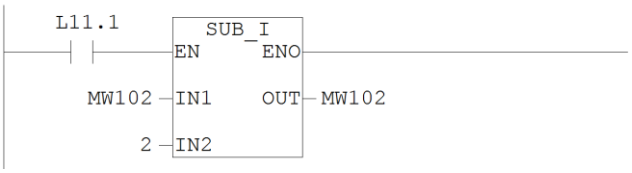
Řešení programu v LAD

```
Block: OB40    "Hardware Interrupt"
```

```
Network: 1
Pokud nastane preruseni od I 124.0 zvysi se hodnota MW100 o 1
```




```
Network: 2
Pokud nastane preruseni od I124.1 snizi se hodnota MW 102 o 2
```




```
Network: 3
Pokud nastane preruseni od I 124.2 nastavi se digitalni vystup Q 124.0 do 1
```

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 9\pr_9_4.zip

10. METODIKA NÁVRHU SYSTÉMŮ AUTOMATICKÉHO ŘÍZENÍ, ANALÝZA ŘÍZENÉHO SYSTÉMU POMOCÍ PETRIHO SÍTÍ

Čas ke studiu: 2 hodiny

Cíl:

Kapitola se zabývá metodikou návrhu řídicích systémů. Správný postup při návrhu má zásadní význam pro výslednou kvalitu řídicí aplikace. Pro návrh systémů reálného času je vyvinuta řada metod, pro systémy s programovatelnými automaty jsou však nejvhodnější **metody strukturovaného návrhu.** V kapitole je popsán postup strukturovaného návrhu vycházející z norem **IEC 61499** a **IEC 61131**, který pokrývá všechny kroky návrhu řídicí aplikace. Postup spočívá v tvorbě jednotlivých **modelů**, které popisují navrhovaný systém z různých pohledů.

Další část kapitoly se věnuje **metodě Petriho sítí**, která je vhodná pro provedení systémové analýzy zejména u systémů se sekvenčním charakterem.

Výklad

10.1 Modely a koncepce návrhu řídicích systémů

Pro celkový popis řídicího systému je potřeba více pohledů na daný systém a tedy je vhodné sestavit několik modelů. Tyto modely vedou vývojáře od základního hardwarového popisu systému až k modelu řídicí aplikace, jehož základem budou funkční bloky [10-12, 10-13].

Při návrhu systému je vhodné vycházet z doporučení mezinárodních norem IEC 61499 a IEC 61131. Základem návrhové sekvence je pětice modelů, vycházejících z uvedených norem. Jedná se o:

Funkční model.

Systémový model.

Model zařízení.

Model zdroje.

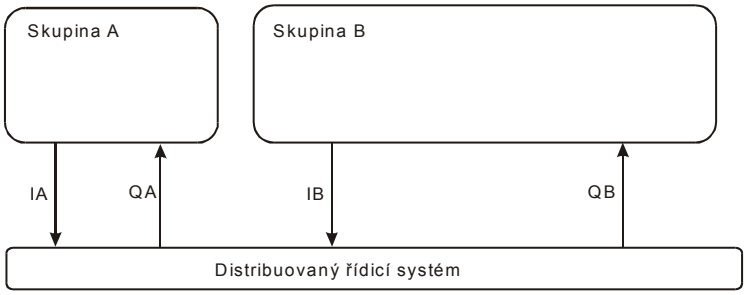
Aplikační model.

Pro popis distribuovaného řídicího systému je možné použít i další modely, ale uvedených pět je základem nutným pro celkový popis systému.

10.2 Funkční návrh

Cílem funkčního návrhu je provést analýzu systému tak, aby mohl být řízen, popsat jeho strukturu, dekomponovat jej na sub-systémy nebo procesy, které mohou být posléze reprezentovány softwarovými komponentami. Výsledkem funkčního návrhu je funkční model. Tento model má

grafickou reprezentaci a jedná se o orientovaný graf skládající se z bloků a orientovaných hran. Bloky reprezentují části systému a hrany vztahy mezi nimi. Základní komponenty obecného frameworku jsou pro tento model dostatečné a žádná speciální notace není nutná. Dobře vytvořený funkční model je nejlepším základem pro další návrh řídicího systému. Na obrázku 10.1 je znázorněn vzhled funkčního modelu.



Obr. 10.1: Principiální vzhled funkčního modelu.

Základní prvky funkčního modelu

Skupina

Prvek skupina reprezentuje samostatnou funkční část řízeného systému. Skupinu je možné dále dělit na subsystémy. Jednotlivé subsystémy ve skupině mají společné to, že se podílejí na celkové funkci skupiny. To však neznamená, že jsou ve fyzické struktuře řízeného systému umístěny ve stejné lokalitě. Na úrovni řídicího software odpovídá skupině aplikace a subsystému subaplikace. Aplikace zajišťuje řízení celé skupiny. Pokud je skupina rozdělena do subsystémů, může aplikace obsahovat několik dílčích subaplikací. Tyto subaplikace mohou být distribuovány do několika zařízení v rámci distribuovaného řídicího systému.

Subsystém

Je to prvek, který reprezentuje dílčí komponentu určité funkční části řízeného systému, definované jako skupina. Subsystém je ve fyzické struktuře řízeného systému umístěn lokálně, jeho řízení pak provádí určitá subaplikace běžící na jednom zařízení.

Vazba na distribuovaný řídicí systém

Prvek vazba na distribuovaný řídicí systém, znázorňuje tok informací z daného subsystému ve skupině k systému řízení. Je znázorněn orientovanou hranou, takže je patrné, jestli informace vycházejí ze subsystému do řízení nebo naopak. Konkrétně tato vazba reprezentuje technologické I/O signály distribuovaného systému řízení.

Vazba mezi subsystémy

Prvek vazba mezi subsystémy znázorňuje vzájemný vztah mezi jednotlivými subsystémy, který je daný konstrukcí řízené technologie. Tyto vazby umožňují lepší pochopení funkce řízeného systému. Tyto vazby se však netýkají řídicího systému.

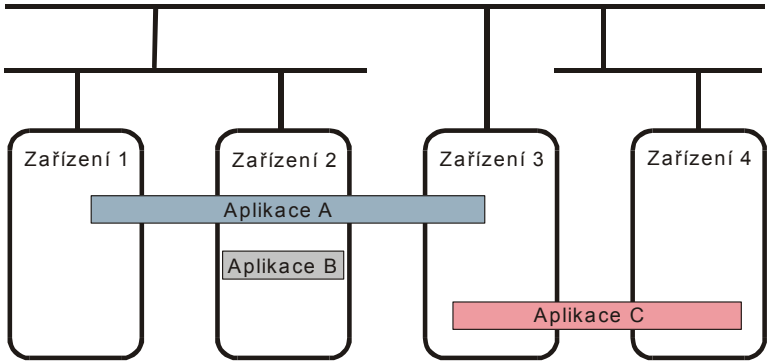
10.3 Systémový model

Systémový model popisuje fyzickou strukturu systému řízení. Má následující funkce:

Definice vztahů mezi komunikujícími zařízeními a aplikacemi.

Definice způsobu a vlastností komunikace mezi zařízeními.

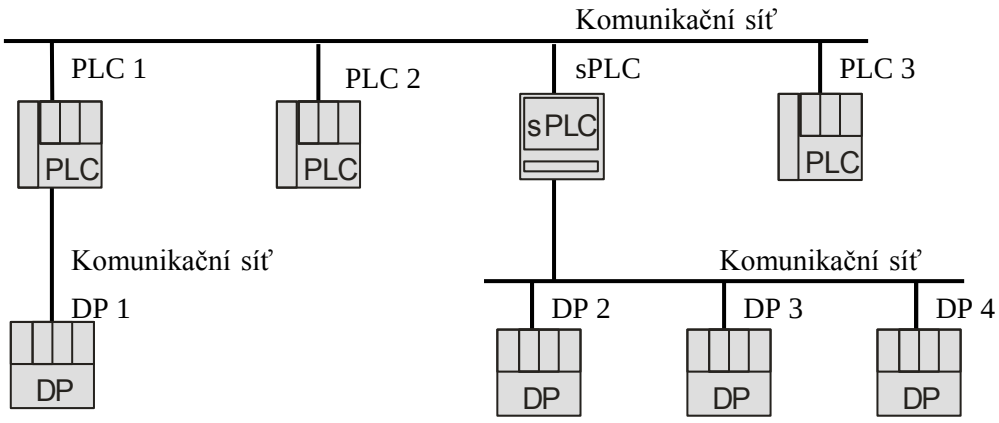
Popis rozložení (distribuci) aplikace mezi jednotlivá zařízení propojená komunikační sběrnicí. Aplikace může celá být vykonávána jedním zařízením, zdrojem, nebo může být distribuována mezi několik zařízení a zdrojů (obr. 10.2).



Obr. 10.2: Principiální vzhled systémového modelu.

Základními prvky systémového modelu jsou programovatelný automat, softPLC, distribuovaná periferie, komunikační spojení apod.

Na obrázku 10.3 je znázorněn příklad grafu systémového modelu.



Obr. 10.3: Příklad systémového modelu.

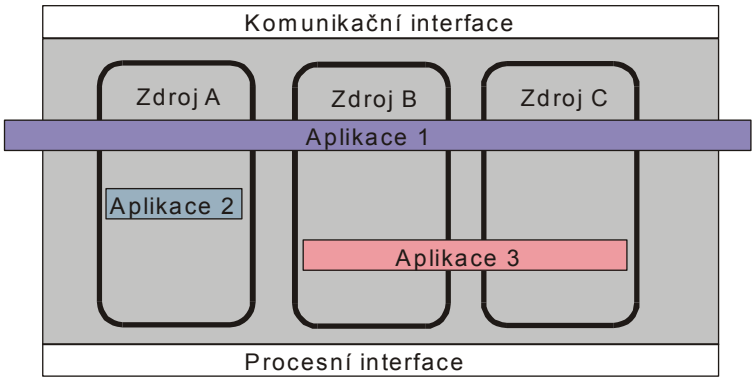
Sestavený systémový model popisuje hardwarovou strukturu distribuovaného řídicího systému. Je zřejmé, pomocí jakých komunikačních spojení jsou jednotlivé řídicí prvky propojeny a jak jsou definovány vlastnosti těchto spojení. V modelu je rovněž určeno, jaký počet aplikací bude systém obsahovat a jak budou distribuovány do jednotlivých řídicích prvků. Je zřejmé, že systémový model se nezabývá dílčími detaily, ale poskytuje základní strukturální pohled na distribuovaný řídicí systém.

Dalším krokem návrhu bude podrobná specifikace hardwarových prostředků jednotlivých prvků systému. Pro každý prvek bude sestaven vlastní model – model zařízení.

10.4 Model zařízení

Model zařízení zobrazuje, jaké zdroje jsou zde k dispozici a kterými aplikacemi jsou zdroje využívány. Zdroj definovaný podle IEC 61499 je velice podobný zdroji definovanému v IEC 61131-3. Zdroj poskytuje nezávislé provádění a řízení struktury funkčních bloků. Model zařízení má procesní rozhraní, které poskytuje služby pro získávání a zápis procesních vstupů/výstupů. Obsahuje rovněž komunikační rozhraní, které poskytuje řadu služeb pro podporu vykonávání funkčních bloků uvnitř zdrojů.

Model zařízení definuje hardwarovou konfiguraci zařízení a vstupně/výstupní signály, které jsou k zařízení připojeny. Principiální vzhled modelu zařízení je znázorněn na obrázku 10.4.



Obr. 10.4: Principiální vzhled modelu zařízení.

Prvky modelu zařízení jsou v podstatě jednotlivé komponenty, ze kterých jsou sestavena zařízení distribuovaného systému. Těchto komponent je značné množství, mohou mít řadu vlastností, které jsou u jednotlivých výrobců různé. Proto zde bude uvedena jen základní skupina komponent bez vztahu ke konkrétnímu výrobci. Hardwarová konfigurace, kterou lze v podstatě považovat za model zařízení, se určitým způsobem vytváří v každém nástroji pro programování programovatelných automatů. Každý výrobce má pro tuto úlohu vlastní software, který vychází z jeho sortimentu komponent. Pokud by si vytvářená aplikace obecného komponentního frameworku kladla za cíl profesionální využití tohoto nástroje, bylo by nezbytné použít detailní data komponentů jednotlivých výrobců. Jelikož je však v této fázi řešení cílem vytvoření základního nástroje a vytýčení směru dalšího vývoje, bude plně dostačující využít pouze základní skupinu komponent.

Procesor

Prvek procesor je procesorovou jednotkou, která je srdcem programovatelného automatu. Dokáže vykonávat program, zpracovávat vstupně/výstupní signály a řídit komunikaci. Procesorová jednotka může obsahovat rozhraní pro některou komunikační sběrnici, pomocí které je připojena k ostatní prvkům distribuovaného systému.

S prvkem procesoru úzce souvisí prvek zdroj (ve smyslu IEC 61131-3 nebo IEC 61499). Jak již bylo uvedeno, zdroj poskytuje podporu pro všechny vlastnosti potřebné pro vykonávání programu. V softwarové terminologii lze zdroj považovat za rozhraní virtuálního stroje, který je schopný provádět IEC program. Jeho hlavní funkcí je poskytovat podporu systému pro spouštění programu. U současných programovatelných automatů se s pojmem zdroj příliš často nesetkáváme a dá se říct, že je chápán jako procesorová jednotka. Zjednodušeně řečeno, běžné PLC s jedním procesorem poskytuje jeden zdroj, víceprocesorové PLC poskytuje více zdrojů, jejichž počet odpovídá počtu procesorů. V následujícím textu bude tedy předpokládáno, že jedna procesorová jednotka poskytuje jeden zdroj.

V tabulce 5.5 jsou uvedeny základní vlastnosti prvku procesor, které jsou definovány v modelu zařízení. Počet vlastností je závislý na typu procesoru. Parametry jsou důležité pro způsob vykonávání programu a upřesňují charakteristiku prostředků poskytovaných zdrojem.

Komunikační procesor

Komunikační procesor je prvek, který slouží pro připojení programovatelného automatu ke komunikační síti.

V tabulce 5.6 jsou uvedeny základní vlastnosti prvku komunikační procesor, které jsou definovány v modelu zařízení. Počet vlastností je závislý na typu komunikačního procesoru.

I/O modul

Prvek I/O modul (vstupně/výstupní modul) slouží pro připojení technologických signálů k PLC. V rámci modelu zařízení jsou definovány signály, které jsou k PLC přivedeny.

Zadáním konkrétního typu I/O modulu je určen počet a druh signálů, pak je jim možné doplnit názvy, symboly a určit, ve které aplikaci budou využity.

SoftPLC

Prvek softPLC je obdoba procesoru, který k realizovaný pomocí počítače typu PC. Jeho důležitým parametrem je typ a vlastnosti rozhraní, pomocí kterého komunikuje s distribuovanými periferiemi. Ze systémového modelu je patrné, které distribuované periferie softPLC využívá. Tabulka vlastností je podobná jako u procesorové jednotky.

Kompakt

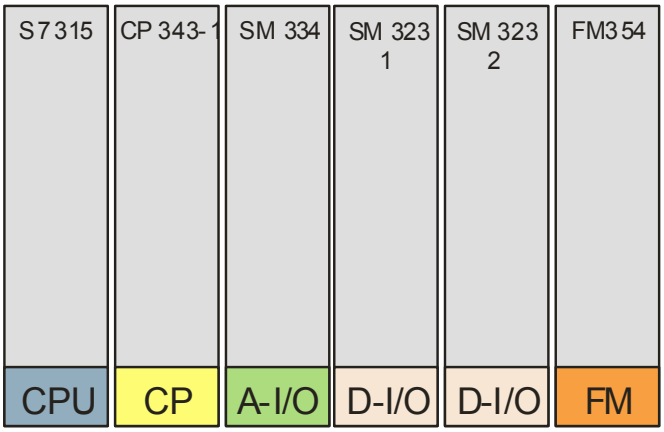
Prvek kompak je programovatelný automat kompaktního typu, který obsahuje procesor i vstupně/výstupní jednotky. Kombinují se zde vlastnosti dvou výše uvedených prvků – procesorové jednotky a I/O modulů.

Model zařízení nemá klasický charakter grafu, kde jsou uzly propojeny hranami. Jedná se v podstatě pouze o nastavení vlastností konkrétního objektu použitého v systémovém modelu. Prvky zachycené v modelu zařízení lze považovat za uzly, u kterých nastavujeme vlastnosti. Tyto uzly jsou součástí jedné platformy, jsou propojeny automaticky, toto propojení nemá možnost tvůrce distribuovaného systému ovlivnit a proto není nutné jej zakreslovat hranami.

Tvorba modelu zařízení je výběrem jednotlivých jednotek a vyplněním jím přiřazených tabulek s příslušnými vlastnostmi.

Na obrázku 10.5 je znázorněn příklad modelu zařízení.

Vytvořením systémového modelu a modelu zařízení jsou definovány hardwarové charakteristiky distribuovaného řídicího systému. Další postup návrhu už se bude týkat převážně oblasti software, tedy řídicí aplikace. Prostředky pro vykonávání programu poskytuje zdroj. Dalším bodem návrhu je tedy definování vlastností jednotlivých zdrojů, které máme v distribuovaném systému k dispozici.



Obr. 10.5: Příklad modelu zařízení.

10.5 Model zdroje

Zdroje podporují vykonávání částí aplikací. Základními prvky aplikací jsou funkční bloky, které jsou alokovány ve zdrojích vzájemně propojených zařízení. Zdroje poskytují aplikacím rozhraní ke komunikačním systémům a ke specifickým procesům zařízení (jako např. I/O subsystém).

Důležitou vlastností zdrojů je to, že poskytují nezávislou činnost. Zdroj může být nahrán, konfigurován, spuštěn nebo zastaven aniž by to mělo vliv na ostatní zdroje v zařízení nebo v síti.

Model zdroje popisuje, které aplikace případně části aplikací jsou zdrojem vykonávány. V případě aplikací distribuovaných do několika zdrojů je zde definován způsob komunikace mezi zdroji. Způsob komunikace je závislý na použitém komunikačním spojení, ale v podstatě jsou zde voleny možnosti popsané v normě IEC 61131-5.

Základní prvky modelu zdroje

Aplikace

Prvek aplikace je softwarové řešení řídicího problému. Je to úplné softwarové řešení, které slouží pro řízení některé části řízeného systému. Pokud zdroj obsahuje aplikaci znamená to, že tato aplikace je lokální a že není distribuována do ostatních zařízení. Využívá pak lokální technologické signály, které jsou k dispozici v daném zařízení. Aplikace je tvořena sítí funkčních bloků a může být rovněž dekomponována na subaplikace. Aplikace je v grafu modelu znázorněna jako uzel.

Subaplikace

Prvek subaplikace je část aplikace. Pokud zdroj obsahuje subaplikaci znamená to, že obsahuje část distribuované aplikace. Distribuovaná aplikace je rozložena (distribuována) do několika zdrojů a jednotlivé její části jsou propojeny pomocí komunikačního spojení. Subaplikace je v grafu modelu znázorněna jako uzel.

Cíl komunikace

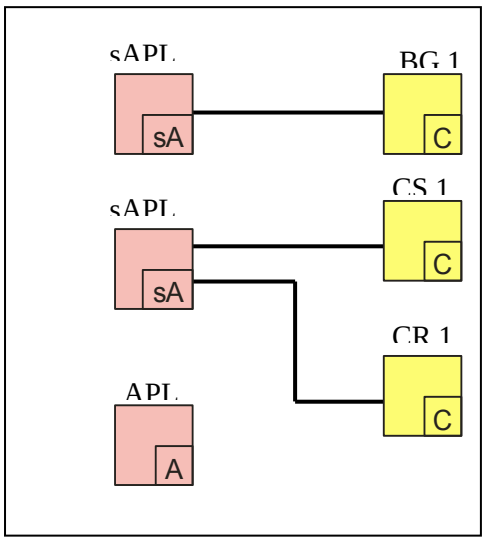
Prvek cíl komunikace definuje, se kterým vzdáleným prvkem subaplikace komunikuje (se kterou vzdálenou subaplikací, ve kterém vzdáleném zdroji). Z předchozích modelů je zřejmé, které komunikační rozhraní má zdroj k dispozici, jaké jsou vlastnosti komunikace a použité protokoly a jak jsou distribuované aplikace rozloženy mezi příslušná zařízení. Pomocí vlastností prvku cíl komunikace je třeba specifikovat, jakým způsobem se bude se vzdáleným bodem komunikovat a tedy jaké konkrétní komunikační funkce budou využity. Výčet funkcí je závislý na druhu komunikačního spojení a protokolu, ale obecně se jedná o:

Čtení dat – čtení dat bez nutnosti, aby vzdálený systém musel komunikaci nějakým způsobem řídit (zpravidla funkce GET nebo READ).

Zápis dat – zápis dat bez nutnosti, aby vzdálený systém musel komunikaci nějakým způsobem řídit (zpravidla funkce PUT nebo WRITE).

Svázaná komunikace – poskytuje řídicí transakci, kdy lokální PLC požaduje, aby vzdálené PLC provedlo určitou operaci a potom výsledek poslalo zpět (funkce SEND a RECEIVE).

Na obrázku 10.6 je uveden příklad modelu zdroje.



Obr. 10.6: Příklad modelu zdroje.

Identifikátory jednotlivých prvků jsou jejich názvy, které určuje tvůrce modelu. V příkladu na obrázku 5.6 byla zvolena následující metodika tvorby názvu – BG 1 = funkce GET č. 1 pro aplikaci B, CS 1 = funkce SEND č. 1 pro aplikaci C apod.

Pomocí modelu zdroje je definováno, které aplikace a které konkrétní subaplikace zdroj provádí. Je rovněž upřesněno, jaké komunikace jsou pro jednotlivé subaplikace požadovány. Nyní je nutné v rámci návrhu distribuovaného systému definovat činnost a parametry jednotlivých aplikací. K tomu je nutné vytvořit aplikační modely pro jednotlivé aplikace, které budou mít charakter sítě funkčních bloků. Funkční bloky budou základními stavebními komponentami pro vytvářené aplikace.

10.6 Model aplikace

Aplikace je definována jako struktura funkcí vzájemně propojených pomocí událostí a dat. Aplikace může být distribuována do několika zdrojů. Může být rovněž dekomponována na subaplikace. Subaplikace mají vnější vlastnosti podobné funkčnímu bloku, ale mohou obsahovat strukturu funkčních bloků, které mohou být rovněž distribuovány do dalších zdrojů.

Aplikace popisuje vztahy mezi událostmi a datovými toky, které jsou požadovány mezi jednotlivými funkčními bloky. Zdroje, do kterých jsou bloky distribuovány, musí zajistit, aby události svázané s časovanými algoritmy v rámci bloku byly k dispozici ve správném čase a pořadí.

Základní členění (dekompozici) aplikace lze provést pomocí subaplikací. Subaplikace může zajišťovat řízení určité části systému. Lze je vytvářet s ohledem na fyzickou nebo logickou strukturu systému, na

funkčnost jednotlivých částí apod. Neméně podstatným důvodem použití subaplikací je zpřehlednění aplikace a zlepšení její čitelnosti. Jednotlivé subaplikace (aplikace) jsou pak tvořeny sítí funkcí a funkčních bloků. Funkce a funkční bloky jsou základními stavebními prvky aplikačního modelu a také vlastní aplikace. Při tvorbě aplikačního modelu propojujeme jednotlivé funkční bloky a funkce pomocí řídicích a datových toků. Pro jejich výběr jsou k dispozici knihovny, které buď většina výrobců ke svým PLC dodává, nebo které si programátor může vytvářet sám, případně které získá z jiných zdrojů. Za předpokladu, že jsou k dispozici všechny funkce a funkční bloky potřebné pro vytvářenou aplikaci, pak tvorba aplikačního modelu je v podstatě tvorba aplikace. Taková aplikace by mohla být přímo zavedena do PLC. Někteří výrobci v současné době dodávají nástroje, které umožňují vytvářet aplikaci pomocí propojování funkčních bloků podobným způsobem, jak se vytváří model aplikace. Příkladem takových nástrojů jsou Siemens Simatic CFC (Control Flow Chart) nebo Simatic iMap.

Pokud však nejsou k dispozici všechny funkce a funkční bloky, jsou dvě možnosti:

Je možné si potřebné bloky naprogramovat a pak vytvořit kompletní model aplikace, jak bylo uvedeno výše. Vytvoření úplného modelu aplikace má výhodu v tom, že se takto vytvoří aplikace, které může být přímo nahrána do PLC. Přestože jsou v současnosti k dispozici často rozsáhlé knihovny funkcí a funkčních bloků, je velice pravděpodobné, že si programátor bude muset při tvorbě aplikace vytvářet i své vlastní bloky. To se provádí pomocí standardních jazyků podle IEC 61131-3.

Nebo lze vytvořit takový aplikační model, který nebude popisovat aplikaci do detailů, ale navrhne pouze základní strukturu. Z takového modelu pak lze vygenerovat základní strukturu a části kódu v některém ze standardních jazyků (podle IEC 61131-3). Tento kód pak může být doplněn podle požadavků programátora. Tato varianta je rovněž velice výhodná. Postupný návrh řízení od systémového modelu k aplikačnímu zajistí, že vygenerovaný základ kódu programu bude kvalitní a poskytne dobrý výchozí stav pro jeho bezproblémové dokončení.

Základní prvky modelu aplikace

Subaplikace

Prvek subaplikace představuje část aplikace. Aplikace může být tvořena několika subaplikacemi, které mohou být distribuovány do několika zdrojů. Subaplikace mohou být dále dekomponovány do dalších subaplikací. Subaplikace mají vnější vlastnosti podobné funkčnímu bloku, ale mohou obsahovat strukturu funkčních bloků a subaplikací, které mohou být rovněž distribuovány do dalších zdrojů.

Funkční blok

Prvek funkční blok je základním stavebním prvkem aplikace. Má následující vlastnosti:

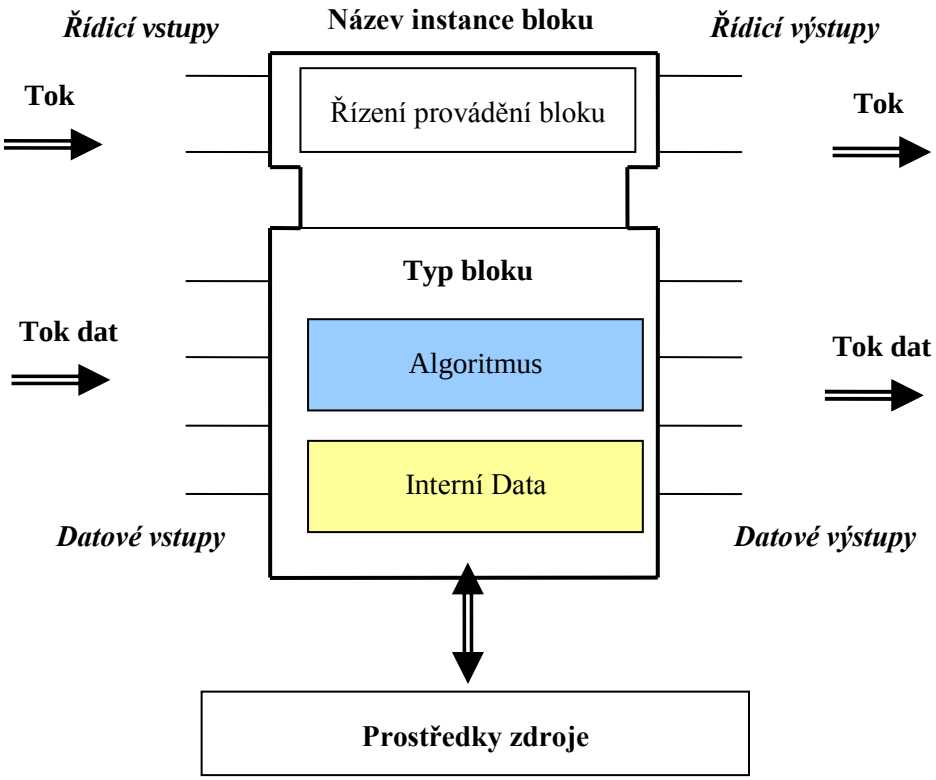
Každý funkční blok má název typu a název instance.

Každý blok má skupinu řídicích vstupů a jeden nebo více událostních výstupů, pomocí kterých je možné jej propojit s dalšími bloky.

Každý blok má skupinu datových vstupů a jeden nebo více datových výstupů, pomocí kterých si předává data s ostatními bloky.

Každý blok má množinu vnitřních proměnných, které se používají pro uchovávání dat mezi jednotlivými voláními bloku.

Na obrázku 10.7 je znázorněna obecná struktura funkčního bloku, jak ji definuje norma IEC 61499.



Obr. 10.7: Funkční blok.

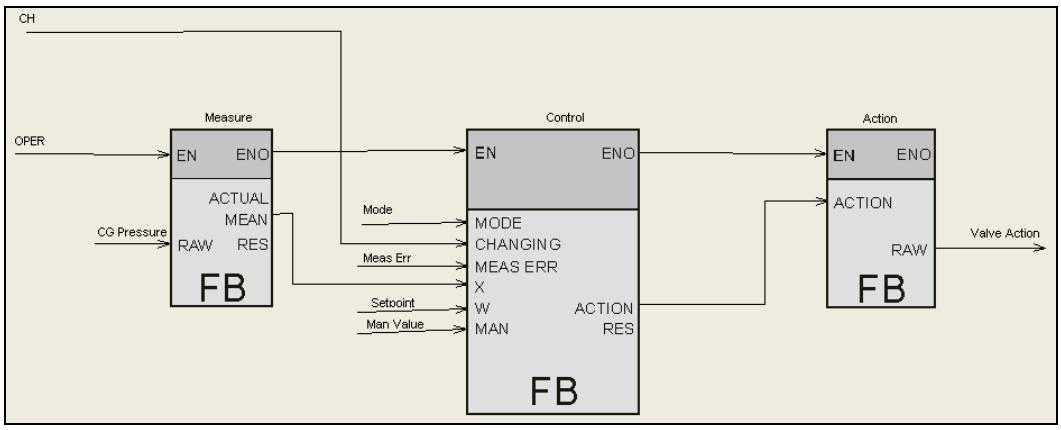
Funkční bloky lze rozdělit do několika typů:

Základní funkční blok – jeho chování je definováno algoritmy, které jsou spouštěny signály na vstupní události.

Složený funkční blok – chování bloku je definováno strukturou instancí funkčních bloků, které jsou navzájem propojeny datovými toky a událostmi.

Servisní funkční blok – poskytují rozhraní mezi předchozími typy funkčních bloků a externími zařízeními, například při komunikaci se vzdáleným zařízením.

Na obrázku 10.8 je příklad části aplikačního modelu, ve kterém je znázorněna struktura funkčních bloků.



Obr. 10.8: Příklad struktury funkčních bloků v aplikačním modelu.

10.7 Analýza řízeného systému pomocí Petriho sítě

Petriho sítě jsou formální a grafický jazyk, který je vhodný pro modelování systémů, zejména sekvenčních systémů a systémů s paralelismem. Petriho sítě vznikly v šedesátých letech, kdy C.A.Petri definoval tento jazyk a bylo to poprvé, kdy vznikla obecná teorie pro diskrétní paralelní systémy. Petriho sítě nemají nic společného s komunikačními sítěmi, nicméně mohou být použity pro popis protokolů používaných v komunikačních sítích.

Při tvorbě řídicích aplikací se Petriho sítě používají zejména jako grafický nástroj pro systémovou analýzu řízeného systému a zejména v případech kdy se jedná o sekvenční proces.

Pojem Petriho sítě byl postupně obohacován a zobecňován tak, aby jeho modelovací schopnost vyhověla praktickým potřebám. V této úvodní kapitole neformálním způsobem probereme několik typů Petriho sítí a na příkladech vysvětlíme jak fungují a k čemu mohou sloužit. Postupovat budeme historicky od dřívějšího k pozdějšímu, od jednoduchého k složitějšímu, od speciálního k obecnějšímu. Postupně projdeme tyto typy Petriho sítí:

- C/E (Condition/Event) Petriho sítě.
- P/T (Place/Transitions) Petriho sítě.
- P/T Petriho sítě s inhibičními hranami.
- P/T Petriho sítě s prioritami.
- TPN Časované (Timed) Petriho sítě.
- CPN Barevné (Coloured, barvené) Petriho sítě.
- HPN Hierarchické (Hierarchical) Petriho sítě.
- OOPN Objektové (Object Oriented) Petriho sítě.

10.7.1 P/T Petriho sítě (Place/Transitions PN)

P/T Petriho síť je tvořena následujícími objekty:

- **Místy** (places), graficky reprezentovanými kružnicemi.
- **Přechody** (transitions), graficky reprezentovanými obdélníky.
- Orientovanými **hranami** (arcs), graficky reprezentovanými šipkami směřujícími od míst k přechodům nebo od přechodů k místům.
- Udáním **kapacity** (capacity indications) pro každé místo sítě, tj. přirozeného čísla udávajícího maximální počet tokenů, který se může v místě nacházet.
- Udáním **váhy** (weights) pro každou hranu sítě, tj. přirozeného čísla udávajícího násobnost hrany.
- Udáním **počátečního značení** (initial marking), udávajícího počet tokenů pro každé místo sítě.

Místa a přechody jsou propojeny orientovanými hranami. Místa a přechody se v průběhu cesty vytvořené hranami střídají. V každém čase t, může místo P obsahovat celé číslo M(P) značek. M(P) se nazývá značením místa P. Značení sítě M je definováno vektorem značení každého místa:

M = (M(P1) - M(Pn)), kde P1...Pn jsou místa představující stavy.

Počáteční značení M0 popisuje počáteční stav modelovaného systému. Podle tohoto počátečního značení a spouštěním přechodů lze získat množinu dosažitelných značení M*0. Každé M z M* 0

odpovídá jednomu stavu modelovaného systému. Vývoj systému je reprezentován počtem značek v síti na základě aktivace přechodů.

Ke každému přechodu náleží jedna nebo více vstupních hran a jedna nebo více výstupních. Tyto hrany umožňují modelovat paralelní sekvence a jejich synchronizaci.

Změny stavů (značení) P/T Petriho sítě jsou charakterizovány následujícími pravidly:

- Stav **sítě** je určen značením, tj. počtem tokenů v každém místě,
- Místo p patří do **vstupní množiny** (pre-set) přechodu t, jestliže z místa p vede hrana do přechodu t a Místo p patří do **výstupní množiny** (post-set) přechodu t, jestliže z přechodu t vede hrana do místa p.
- Přechod t je **proveditelný** (enabled, activated), jestliže:
  - Pro každé místo p vstupní množiny přechodu t platí, že obsahuje alespoň tolik tokenů, kolik činí násobnost hrany vedoucí z místa p do přechodu t.
  - Pro každé místo p výstupní množiny přechodu t platí, že počet tokenů obsažených v místě p zvětšený p o násobnost hrany, mířící z přechodu t do místa p, nepřevyšuje kapacitu místa p.
- Při **provedení** (firing) proveditelného přechodu t se změní stav (značení, marking) sítě takto:
  - Počet tokenů v každém vstupním místě p přechodu t se zmenší o násobnost hrany spojující toto místo s tímto přechodem.
  - Počet tokenů v každém výstupním místě p přechodu t se větší o násobnost hrany spojující toto místo s tímto přechodem.

1. V P/T Petriho sítích místa zpravidla označují stavy modelovaného systému a přechody změny stavu. Stav je charakterizován celým nezáporným číslem daným značením daného místa (počtem tokenů v daném místě). Při modelování počítačových dějů jedná se např. o počty jednotek volné nebo obsazené paměti (např. bufferu), o počty jednotek disponibilních nebo využívaných zdrojů různého typu a pod.

2. Implicitně (defaultně, tj. když není výslovně uvedeno jinak) předpokládáme násobnost hrany 1 a kapacitu místa nekonečnou. Násobnost jednoduchých hran (w=1) a kapacitu kapacitně neomezených míst (K=nekonečno) na grafech Petriho sítí nemusíme uvádět a kvůli větší přehlednosti obrázku také neuvádíme.

3. Zavedení více násobných hran a omezených kapacit míst nikterak nezvyšuje modelovací možnosti (expresivitu) P/T Petriho sítí, ale pouze umožňuje jejich jednodušší zápis. Každou P/T Petriho síť lze převést na ekvivalentní síť bez násobných hran a bez kapacitně omezených míst a přitom modelující stejný reálný problém.

4. Speciálním případem P/T Petriho sítí jsou:

- C/E Petriho sítě, což jsou P/T Petriho sítě ve kterých je kapacita každého místa a násobnost každé hrany rovna 1.
- Obyčejné Petriho sítě (ordinary Petri nets) jsou P/T Petriho sítě ve kterých je kapacita každého místa nekonečná (tj. žádné místo není kapacitně omezené) a násobnost každé hrany je rovna 1. V souladu s výše uvedenou konvencí - viz poznámka 2. - se v obyčejných sítích nevyskytují inskripce kapacit a násobností.
- Elementární obyčejné Petriho sítě jsou obyčejné Petriho sítě pro které platí následující podmínky
  - Každý přechod má alespoň jedno vstupní místo (přechod bez vstupních míst by umožňoval nekontrolovaný vstup libovolného počtu tokenů do sítě).

- Každý přechod má alespoň jedno výstupní místo (přechod bez výstupních míst by umožňoval nekontrolovaný výstup libovolného počtu tokenů ze sítě).
- Žádný přechod nemá žádné místo, která by pro něj bylo současně vstupním i výstupním.

Shrnutí pojmů

Návrh řídicí aplikace je proces, který má několik kroků. Každý z těchto kroků má vliv na kvalitu výsledné řídicí aplikace. Pro návrh řídicí aplikace systémů reálného času existuje řada metod, pro systémy s programovatelnými automaty jsou vhodné zejména **metody strukturovaného návrhu**. Kapitola popisuje postup vycházející z norem IEC61499 a IEC61131. Návrh je zde řešen jako **soubor modelů**, od funkčního modelu (systémové analýzy) až po aplikační model, který je základem řídicí aplikace.

Pro tvorbu systémové analýzy se často používá metoda **Petriho sítí**, což je grafická metoda, pomocí které je možno popsat stavy systému a podmínky pro přechod mezi těmito stavy. Grafy Petriho sítí se skládají ze tří prvků – **míst** (které vyjadřují stavy systému), **přechodů** (které reprezentují podmínky pro přechod mezi stavy) a **orientovaných hran** (které místa a přechody propojují).

Kontrolní otázky

1. Popište postup při návrhu řídicích aplikací.
2. Co je to systémová analýza ?
3. Jaké jsou prostředky systémové analýzy?
4. Jaké modely je možné využít při návrhu řídicí aplikace?
5. Co jsou to Petriho sítě?
6. Z jakých prvku se skládají P/T Petriho sítě.

Další zdroje

10-1. ČSN EN 61131-3: Kopie normy ČSN EN 61131-3. Český normalizační institut, Praha, 1996.

10-2. David R., Alla H.: Petri Nets and Grafcet: Tools for Modelling Discrete Systems. Prentice Hall, 1992. Cambridge, Great Britain. ISBN: 0-13-327537-X.

10-3. Demel J.: Grafy a jejich aplikace. Academia, 2002. Praha. ISBN: 80-200-0990-6.

10-4. Douglass P. B.: Doing Hard Time – Developing Real-Time Systems with UML, Objects, Frameworks and Pattrens. Addison-Wesley Longman, Massachusetts, USA, 1999. ISBN: 0-201-49837-5.

10-5. Hanzálek Z.: Tutoriál – Petriho sítě I. – III. Automatizace 7(8)-10/2001. ISSN: 0005-125X.

10-6. Jensen K., Rozenberg G.: High-Level Petri Nets. Springer-Verlag, Berlin 1991. ISBN: 3-540-54125-X

10-7. Jensen K.: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1. Springer – Verlag, Berlin, 1996. ISBN: 3-540-60943-1.

10-8. John K. H., Tiegelkamp M.: IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems. Springer Verlag, 2001. ISBN: 3540677526.

10-9. IEC/PAS 61499-1: Function blocks for industrial-process measurement and control systems – Architecture. International Electrotechnical Commision, 2000.

10-10. IEC/PAS 61499-2: Function blocks for industrial-process measurement and control systems – Software tools requirements. International Electrotechnical Commision, 2001.

10-11. IEC/PAS 61499-4: Function blocks for industrial-process measurement and control systems – Rules for compliance profiles. International Electrotechnical Commision, 2002.

10-12. Lewis R. W.: Modelling Control Systems Using IEC 61499: Applying Function Blocks to Distributed Systems (IEE Control Series, 59). The Institution of Electrical Engineers, 2001. ISBN: 0852967969.

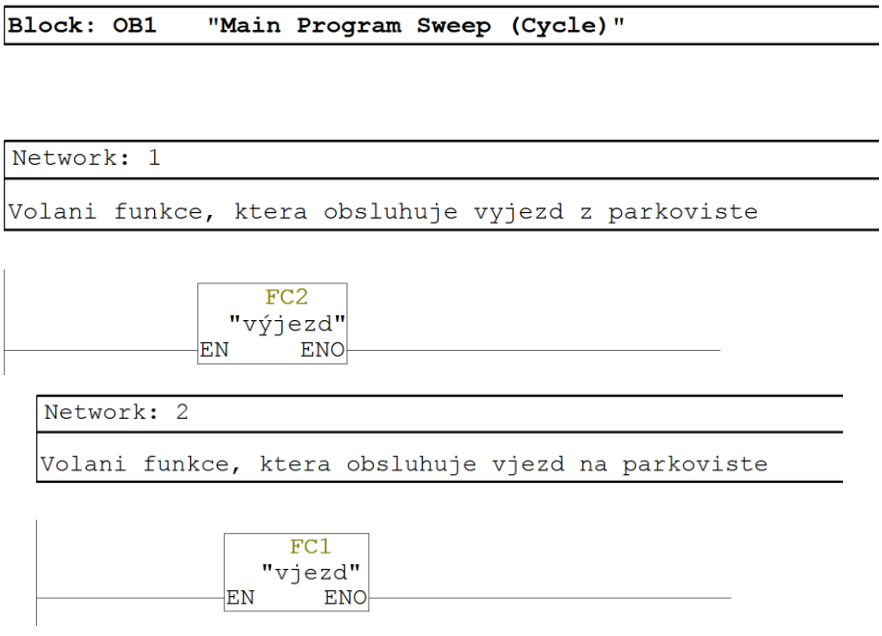
10-13. Lewis R. W.: Programming Industrial Control Systems Using IEC 1131-3. The Institution of Electrical Engineers, London, UK, 1998. ISBN: 0852969503.

Řešená úloha 10.1.

Vytvořte řídicí program pro automaticky řízené parkoviště.

Parkoviště má vstupní a výstupní závoru. Po příjezdu na parkoviště stiskne řidič tlačítko pro vyžádání lístku (VYŽÁDÁNÍ LÍSTKU) a systém vydá lístek aktivací podavače (PODAVAČ LÍSTKU). Jakmile si řidič lístek odebere (ČIDLO PŘÍTOMNOSTI LÍŠKU 1), otevře se na 10s vstupní závora (ZÁVORA1). Zavření závory je blokováno přítomností vozidla v prostoru závory (ČIDLO PŘÍTOMNOSTI VOZIDLA POD ZÁVOROU). Během otevření závory bliká u vjezdu výstražné světlo s frekvencí 1s (INDIKÁTOR1). Při výjezdu vloží řidič lístek a minci (ČIDLO PŘÍTOMNOSTI LÍSTKU 2, ČIDLO PŘÍTOMNOSTI MINCE), poté se na 10s otevře výstupní závora (ZÁVORA 2). Zavření závory je blokováno přítomností vozidla v prostoru závory (ČIDLO PŘÍTOMNOSTI VOZIDLA POD ZÁVOROU 2). Během otevření závory bliká u výjezdu výstražné světlo s frekvencí 1s (INDIKÁTOR 2). Počet aut na parkovišti je počítán, jeho hodnota je uložena v paměťové buňce.

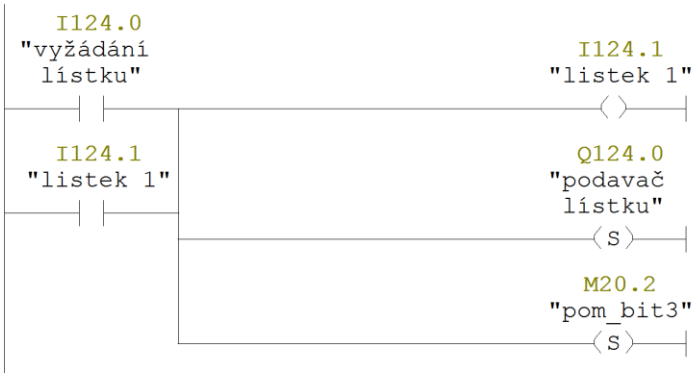
Řešení programu v LAD



Block: FC1

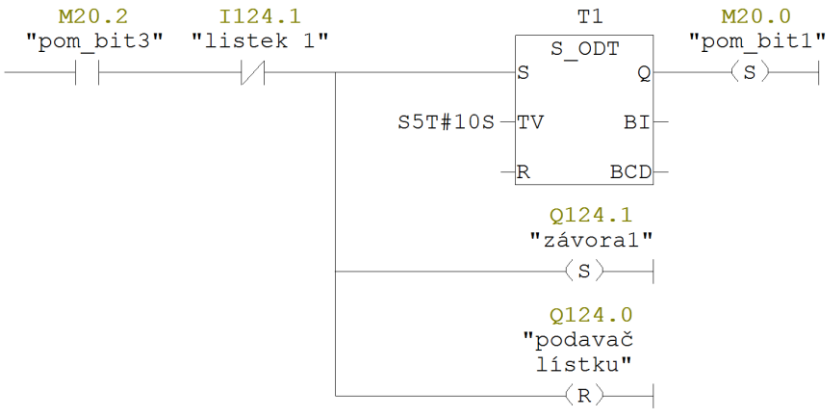
Network: 1

pokud je vyzadan listek, spust podavac



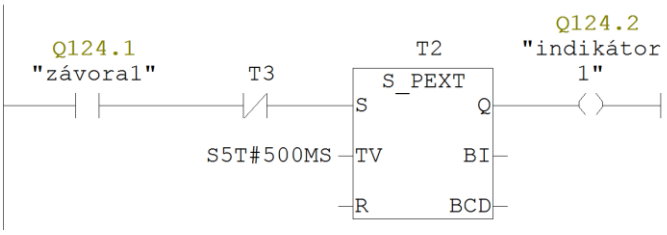
Network: 2

pokud podavac spusten a listek byl odebran, otevri zavoru pro prijezd na parkoviste



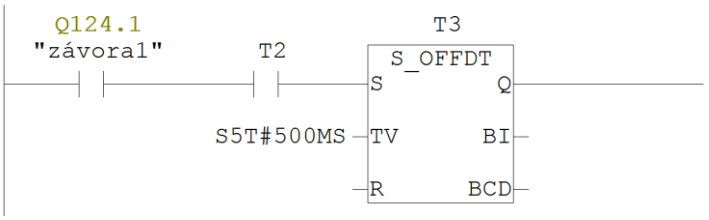
Network: 3

pokud je zavora otevrena blikej s frekvenci 1s



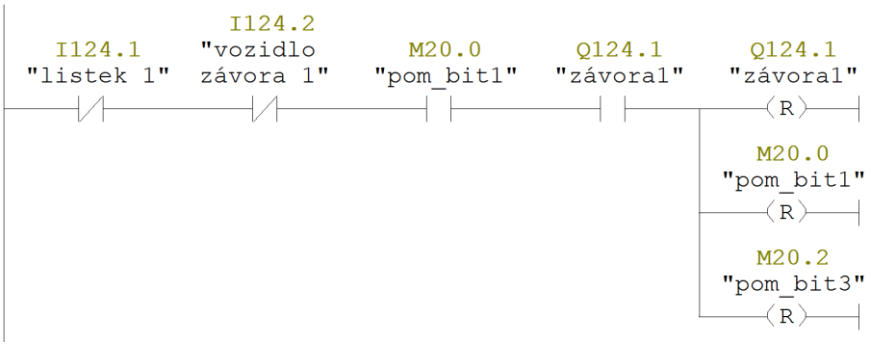
Network: 4

pokud je zavora otevrena blikej s frekvenci 1s



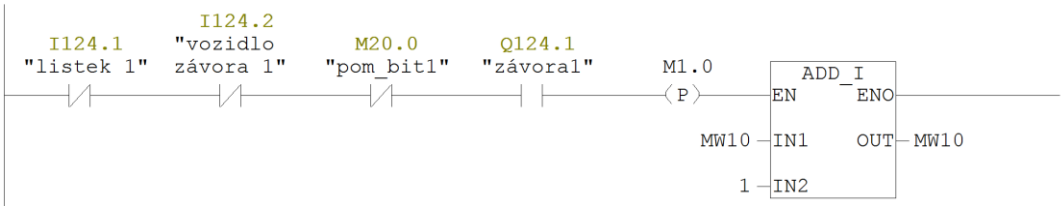
Network: 5

pokud vozidlo neni pod zavorou, zavri zavoru a prestan blikat



Network: 6

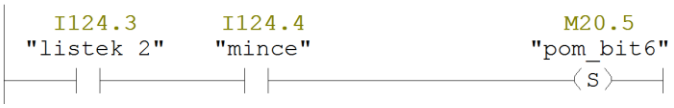
pokud se zavora otevrela a vozidlo projelo zvys pocitadlo



Block: FC2

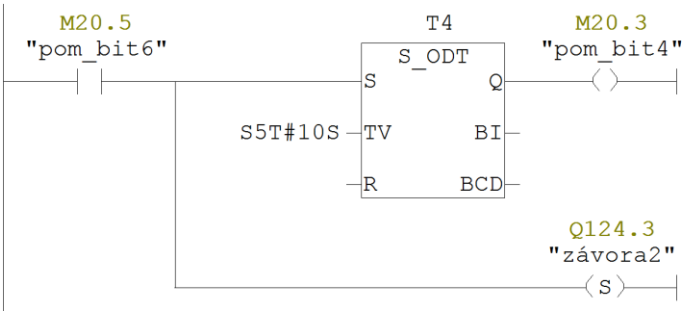
Network: 1

pokud ridic vlozil listek a vlozil take minci otevri zavoru



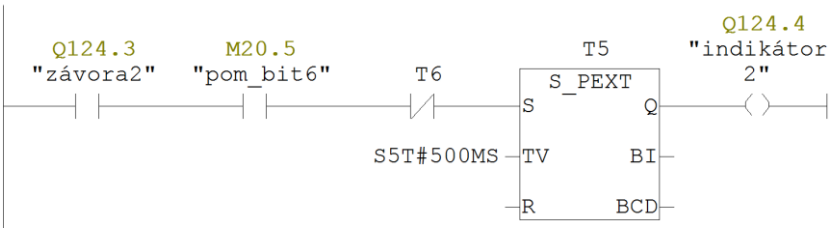
Network: 2

zavora bude otevrena 10s



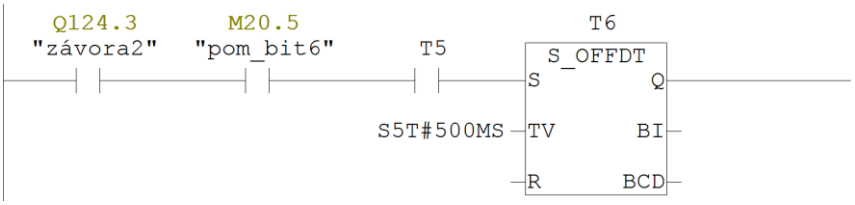
Network: 3

pri otevreni zavory 2 blikej s frekvenci 1s



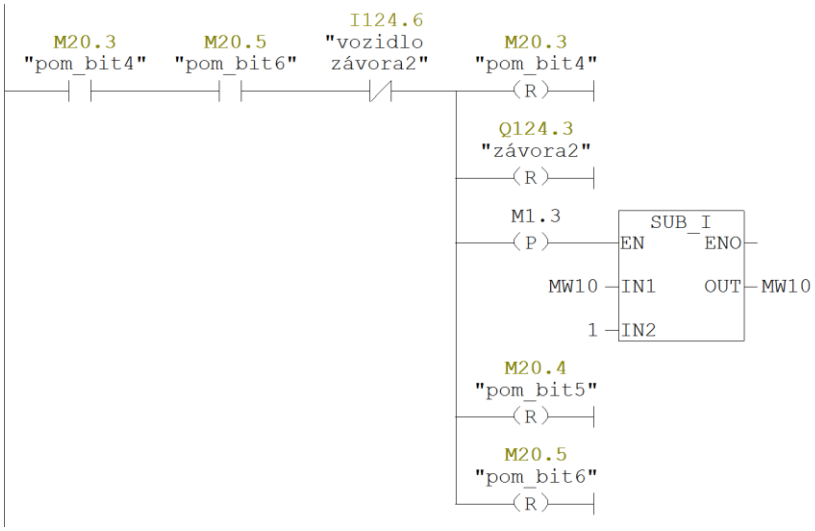
Network: 4

pri otevreni zavory 2 blikej s frekvenci 1s



Network: 5

pokud neni auto pod zavorou, zavri zavoru a odecti 1 tzn. je uvolneno misto



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 10\pr_10_1.zip

11. NORMA IEC61131 A JEJÍ ČÁSTI

Čas ke studiu: 2 hodiny

Cíl:

Kapitola přináší informace o standardizační normě **IEC 61131**. Je zde popsán význam této normy a její jednotlivé části. Další část textu obsahuje popis **IEC softwarového modelu** a společných prvků třetí části uvedené normy. Jsou zde rovněž představeny jazyky popsané ve třetí části normy **IEC 61131-3**. Po prostudování kapitoly bude student znát význam uvedené standardizační normy, její jednotlivé části a rovněž podrobněji **společné prvky programovacích jazyků** uvedené ve třetí části této normy. Dále bude mít student přehled o jednotlivých **standardizovaných jazycích** a bude schopen rozhodnout, pro jaký druh řídicích úloh je určitý jazyk vhodný.

Výklad

11.1 Úvodní informace

Programovatelné automaty se běžně používají pro řízení technologických procesů už řadu let. Během této doby se podstatně změnila vlastní technika PLC, zvýšil se výkon a rozšířily se možnosti automatů. Pro psaní řídicích aplikací bylo používáno široké spektrum programovacích technik a programovacích jazyků. Dá se říci, každý výrobce programovatelných automatů má svůj programovací jazyk. Tyto jazyky se často podstatně liší. Mohlo by se zdát, že existuje několik programovacích technik (jako například Ladder Logic nebo Fuction Block Diagram), které jsou podporovány většinou výrobců. Navíc si tyto techniky získaly velkou oblibu, protože jsou lehce intuitivně pochopitelné. Bohužel implementace těchto metod jednotlivými výrobci je často rozdílná a má svá specifika. Takže jediná věc, která je společná pro všechny programovací jazyky pro PLC je ta, že je každý jiný. Tento stav však přináší řadu problémů pro programátory, techniky, údržbu a další pracovníky využívající tuto techniku. Pokud má pracovník používat několik různých programovatelných automatů, je nucen znát řadu programovacích jazyků. Důsledkem je neefektivní využívání času těchto pracovníků a tedy peněz. Existence takového počtu jazyků může vést k nedorozuměním, které mohou mít v některých případech vážné následky.

Vznikl tedy požadavek vytvořit nový standard pro programovatelné automaty. V roce 1979 byla při IEC (International Electro-technical Commission) založena skupina, která měla sledovat vývoj programovatelných automatů, včetně návrhu hardware, instalace, testování, programování a komunikace. Výsledkem práce skupiny byl nový IEC PLC standard 1131. Jeho jednotlivé části byly poprvé publikován v letech 1992 - 1998. Části jsou uvedeny v následující tabulce.

Tabulka 11.1: Části standardu IEC 1131.

Část	Název	Obsah	Publikována
Část 1	Základní informace.	Definice základní terminologie a pojmů.	1992
Část 2	Požadavky na techniku a její testování.	Elektronická a mechanická konstrukce, ověřovací testy.	1992

Část	Název	Obsah	Publikována
Část 3	Programovací jazyky.	Struktura PLC software, jazyky a vykonávání programu.	1993
Část 4	Uživatelské pokyny.	Návody pro výběr, instalaci a údržbu programovatelných automatů.	1995
Část 5	Specifikace zpráv.	Softwarové prostředky pro komunikaci mezi zařízeními založené na MAP (Manufacturing Messaging Services).	1998
Část 6	Komunikace prostřednictvím fiedbusů.	Softwarové prostředky PLC pro komunikaci s použitím IEC fieldbus.	Podle dokončení fieldbus standardu.
Část 7	Programování fuzzy control.	Softwarové prostředky, zahrnující standardní funkční bloky pro práci s fuzzy logikou v PLC.	1997
Část 8	Směrnice pro implementaci jazyků pro PLC.	Aplikační a implementační směrnice pro IEC 1131-3 jazyky.	1998

Tato kapitola se věnuje třetí části uvedené normy, programovacím jazykům. Většina výrobců programovatelných automatů v současné době přizpůsobuje své produkty této normě, což usnadňuje programátorům a dalším pracovníkům přechod mezi systémy různých výrobců. Předpokládá se, že využití normy IEC 1131-3 bude v budoucnu ještě větší.

Norma IEC1131-3 versus konvenční liniové programování

Určitý programovací jazyk na bázi liniových schémat podporuje většina výrobců programovatelných automatů. Tyto konvenční liniové jazyky se však vyznačují řadou nevýhod:

Symboly a možnosti těchto liniových jazyků se liší u jednotlivých typů PLC.

Nedostatečné možnosti pro strukturovanou a hierarchickou dekompozici programu.

Omezené možnosti znovupoužití programu.

Nedostatečné možnosti pro adresaci a manipulaci s datovými strukturami.

Omezené možnosti pro vytváření komplexních sekvencí.

Omezená kontrola nad prováděním programu.

Používání aritmetických operací je nešikovné.

Některé z těchto nevýhod mohou být minimalizovány použitím kvalitních programovacích prostředí, využívající např. generátory kódu, off-line dokumentační nástroje, informační technologie s databázemi apod. Obecně však není možné uvedená omezení úplně ignorovat.

Standardizační norma IEC 1131-3 se snaží uvedené problémy řešit. Vychází z řady technik a jazyků používaných výrobci PLC a poskytuje robustní a užitečný standard. Hlavní výhody, které norma nabízí jsou tyto:

1. Standard podporuje dobře strukturovaný vývoj programu – shora-dolů nebo zdola-nahoru. Umožňuje rozdělit program na dílčí funkční elementy, které se označují jako programové organizační jednotky. Navíc každý z těchto elementů může být hierarchicky vytvořen z jiných elementů.

2. Standard požaduje dodržování datových typů. To znamená, že programovací software je schopen poznat, kdy programátor použil chybný datový typ pro zápis dat do proměnné. To odstraní velký zdroj chyb při konvenčním liniovém programování.

3. Poskytuje nástroje pro spouštění různých částí programu v různých časech, s různou četností nebo paralelně, tzn. poskytuje podporu pro plnou kontrolu vykonávání programu.

4. Je zde plná podpora pro popis sekvencí, takže komplexní sekvenční úlohy lze jednoduše rozložit s použitím přehledného grafického programovacího jazyka sekvenčních funkčních grafů.

5. Standard podporuje definování datových struktur.

6. Standard nabízí flexibilní výběr jazyka – poskytuje soubor tří grafických a dvou textových jazyků pro zápis jednotlivých částí aplikace. Programátor si může volně vybrat ten jazyk, který se nejlépe hodí pro danou část programu.

7. IEC 1131-3 poskytuje standardizované jazyky a metody provádění programu, takže řada technologických problémů může být naprogramována jako software nezávislý na výrobci. Jinými slovy, velká část programů napsaná pro PLC podporující IEC 1131-3 může být přenesena a spouštěna na automatech různých výrobců.

Norma obsahuje následující programovací jazyky – viz. tabulka 11.2. Tyto jazyky budou dále v textu podrobněji popsány.

Bylo řečeno, že IEC 1131-3 umožňuje vývoj dobře strukturovaného software, který může být vyvíjen ve směru shora-dolů nebo zdola-nahoru. Jedním z hlavních prostředků standardu tvorbu strukturovaného software jsou funkční bloky.

Funkční bloky jsou části řídicího programu, které jsou určitým způsobem "zapouzdřeny", takže mohou být snadno použity vícenásobně - buď v rámci jednoho programu nebo i v různých programech. Funkční blok může řešit jednoduchý problém, jako je např. vytvoření časovače, ale může také zajišťovat řízení velké části technologie, např. řízení tlakové nádoby. Funkční blok představuje určitý algoritmus, ale také má schopnost uchovávat data. To je velké výhoda v porovnání s podobnými strukturami z vyšších programovacích jazyků, jako jsou subrutiny nebo funkce.

Tabulka 11.2: Programovací jazyky podporované normou IEC 1131-3.

Strukturovaný text (Structured Text - ST)	Vyšší programovací jazyk, který podporuje strukturované programování. Je podobný Pascalu.	TEXTOVÉ JAZYKY
Výpis instrukcí (Instruction List – IL)	Jazyk nižší úrovně, podobný assembleru nebo zápisu STL u PLC Siemens Simatic S5, S7.	
Liniové schéma (Ladder Diagram – LD)	Jazyk založený na liniových schématech reléové logiky.	GRAFICKÉ JAZYKY
Diagram funkčních bloků (Function Block Diagram – FBD)	Jazyk znázorňující signálové a datové toky mezi funkčními bloky. Je podobný hradlovým schématům.	
Sekvenční funkční graf (Sequential Function Chart – SFC)	Jazyk popisující sekvenční chování řízeného systému.	

Standard poskytuje dobře definované rozhraní pro funkční bloky - formální definice vstupně/výstupních parametrů. Potom funkční bloky, vytvořené různými programátory, lze bez

problémů propojovat, protože každý vstupní a výstupní parametr má přesně definovaný datový typ odpovídající standardu. Funkční bloky je vhodné využívat jako základní stavební kameny programu.

Jejich používáním při tvorbě řídicích aplikací se začínají vytvářet knihovny těchto bloků pro velké množství různých průmyslových aplikací.

Naskýtá se srovnání funkčních bloků a objektů z objektově orientovaného programování. Funkční bloky obsahují, podobně jako objekty, zapouzdřená data a mají metody pro práci s těmito daty (algoritmus, řídicí strategie). Další výhody objektově orientovaných jazyků, jako dědičnost a polymorfismus nejsou u funkčních bloků podporovány.

Přenositelnost programů

Hlavní výhoda pro koncové uživatele pracující s produkty vyhovující normě IEC 1131-3 je možnost použít řídicí software pro rozdílné typy PLC. Software vyvinutý pro jeden typ PLC může být teoreticky provozován na jiném PLC vyhovující normě. Tato možnost může značně otevřít trh s PLC. Takový vysoký stupeň přenositelnosti, označovaný jako inter-systém portability, však může v praxi narazit na problémy. Norma definuje řadu vlastností programovacích jazyků a záleží na tom, zda se výrobce rozhodne tyto vlastnosti implementovat. Jinými slovy, standard od výrobce nevyžaduje a neočekává, že všechny vlastnosti implementuje. Tedy šance, že dva výrobci implementují stejnou množinu vlastností není příliš velká. A bez toho jsou možnosti přenositelnosti malé.

Jelikož IEC standard není jednoznačný z hlediska požadavků kompatibility, někteří výrobci PLC označují své produkty jako IEC 1131-3 kompatibilní už při jednoduché implementaci jedné nebo dvou jazykových vlastností. Kvůli nízké kompatibilitě a malým možnostem přenositelnosti programů, byla založena PLCopen asociace, která je tvořena řadou výrobců PLC. PLCopen aktivně pracuje na definování různých úrovní kompatibility s cílem, aby produkty se stejnou úrovní kompatibility měly známou úroveň softwarové přenositelnosti.

IEC 1131-3 standard bývá často kritizován pro definování mnoha volitelných vlastností, které mohou být implementovány podle uvážení výrobce. Úrovně kompatibility, které vyžadují implementaci dobře definovaných množin vlastností určených asociací PLCopen, by měly výrazně pomoci řešit tento problém.

Mezijazyková přenositelnost

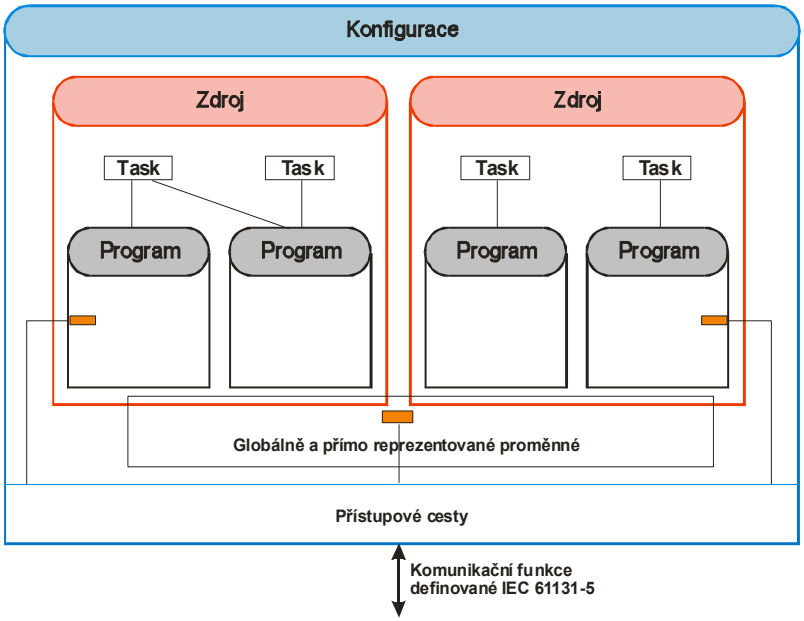
Standard poskytuje několik jazyků pro tvorbu řídicího software, takže programátor si může pro konkrétní úlohu vybrat nejvhodnější z nich. Obecně je možné převádět softwarové elementy z jednoho jazyku do druhého, ikdyž to nikdy nebylo základním cílem tvůrců normy. Chceme-li zapsat stejnou řídicí úlohu různými jazyky, algoritmus se může mírně lišit, což může při konverzi působit určité problémy. Přesto však řada výrobců dodává systémy, které umožňují automatickou konverzi mezi jednotlivými jazyky. Problémy, které zatím brání plné mezijazykové přenositelnosti budou řešeny v příštích revizích standardu.

Základní pojmy normy IEC 1131-3

Snad nejdůležitější aspekt jakéhokoliv programovacího systému je možnost dekompozice software do menších, přehlednějších částí. V následujícím textu budou popsány možnosti, jak rozložit program do dílčích softwarových elementů, které mají mezi sebou jasně definované rozhraní.

11.2 IEC softwarový model

IEC softwarový model je vrstvový – každá vrstva, úroveň skrývá mnoho vlastností nižších vrstev.



Obr. 11.1: IEC 1131-3 softwarový model.

Konfigurace (Configurations)

Software pro určitý řídicí problém je označen jako konfigurace. Je to nejvyšší úroveň softwarového modelu. Konfigurace je definována jako jazykový element odpovídající programovatelnému řídicímu systému. Obvykle je konfigurace chápána jako software pro jedno PLC. Konfigurace je schopna komunikovat s dalšími IEC konfiguracemi různých PLC systémů přes definované rozhraní, které musí být formálně definováno s použitím standardních jazykových elementů.

Zdroje (Resources)

V každé konfiguraci může být jeden nebo více zdrojů. Zdroje poskytují podporu pro všechny vlastnosti potřebné pro vykonávání programu. V softwarové terminologii se lze na zdroj dívat jako na rozhraní virtuálního stroje, který je schopný provádět IEC program. Jeho hlavní funkcí je poskytovat podporu systému pro spouštění programu.

IEC program nemůže fungovat, dokud není nahrán do zdroje. Obvykle je zdroj v PLC systému, ale může být také v jiném systému, např. může být simulován v PC.

Zajímavou vlastností zdroje je, že částí software a může také odrážet fyzickou strukturu PLC. Například software pro PLC s několika procesory může být navržen tak, že každý zdroj bude pro jeden procesor. Zdroj v konfiguraci je schopen nezávislé činnosti – není ovlivňován jinými zdroji. Jednou z hlavních funkcí zdroje je to, že poskytuje rozhraní mezi programem a fyzickými I/O PLC.

Zajímavé je, že standard umožňuje, aby konfigurace obsahovala několik zdrojů a zdroj může být schopen podporovat více než jeden program. To dává PLC zajímavou možnost nahrát, startovat a provádět množství naprosto nezávislých programů.

Programy

IEC program může být vytvořen z množství různých softwarových elementů, které jsou zapsány v jednom z několika IEC jazyků. Typicky program obsahuje řadu propojených funkčních bloků, které jsou schopny si předávat data prostřednictvím softwarových spojení. Program může číst a zapisovat I/O proměnné a komunikovat s jinými programy. Vykonávání různých částí programu je řízeno pomocí tasků.

Tasky

Task může být nakonfigurován k řízení skupiny programů, funkčních bloků, buď provádění periodicky nebo podle výskytu nějakého spouštěče (triggeru). V IEC modelu nejsou žádné skryté nebo vnitřní mechanizmy pro provádění programu. Jinými slovy, program nebo funkční blok bude nevyužitý, dokud nebude přiřazen některému tasku a ten nebude nakonfigurován k periodickému spouštění nebo svázán s triggerem. Na druhou stranu, i když není funkční blok přiřazen tasku, spouští se v rámci programu, kde je použit.

Funkční bloky

Koncept funkčních bloků je jedním z nejdůležitějších vlastností IEC 1131-3 standardu, umožňující hierarchický návrh programů. Funkční bloky umožňují rozdělit program do menších částí. Funkční blok má dvě základní vlastnosti:

- Definuje data jako množinu vstupních a výstupních parametrů, které mohou být použitý pro softwarové propojení s jinými bloky a má vnitřní proměnné.
- Má algoritmus, kus kódu, který se vykoná vždy, když je funkční blok volán.

Algoritmus pracuje s aktuálními parametry a vnitřními proměnnými a výsledkem jeho vykonání je množina výstupních parametrů. Vnitřní parametry mohou být lokální (definované ve funkčním bloku) nebo globální (definované na úrovni programu, zdroje nebo konfigurace).

Jelikož funkční bloky mohou uchovávat hodnotu, mají definovaný stav. To je důležitá vlastnost, která dovoluje jejich využití pro řadu řídicích problémů - čítače, časovače, PID algoritmus, fuzzy algoritmus, hodiny reálného času apod.

Funkce

Funkce jsou softwarové elementy, které při vyvolání s určitými vstupními parametry, vytvoří jeden hlavní výsledek. Mohou to být například matematické funkce – Sin(), Cos() apod. Funkce mohou mít rovněž vstupně-výstupní parametry. To dovoluje, aby funkce při vykonání modifikovala hodnotu takového parametru.

Hlavní rozdíl mezi funkcí a funkčním blokem – funkce nemá vnitřní paměť. Pro stejnou kombinaci vstupních hodnot vždy vyprodukuje stejný výsledek. Funkční bloky mohou mít více výstupů – množinu výstupů.

Lokální a globální proměnné

Lokální proměnné – přístupné v rámci softwarového elementu, kde jsou definovány (konfigurace, zdroj, program, funkční blok, funkce).

Globální proměnné – jsou to proměnné definované v programu, které jsou viditelné ve všech softwarových elementech tohoto programu. Podobně, je-li globální proměnná definovaná ve zdroji (konfiguraci), je viditelná ve všech softwarových elementech tohoto zdroje (konfigurace). Globální proměnné jsou typickým mechanizmem předávání dat mezi programy, nebo mezi funkčními bloky umístěnými v různých programech.

Přímé proměnné

Paměť PLC může být adresována pomocí přímých proměnných. Je to běžný typ proměnných používaný v řadě PLC – např. %IW100, %Q75 apod. Přímé proměnné umožňují číst a zapisovat data do známé oblasti paměti. Používání takových proměnných je ve standardu omezeno - smí být

deklarovány a používány pouze v programech, ne ve funkčních blocích. Jejich používání ve funkčních blocích by přinášelo problémy při opakovaném používání těchto bloků.

Přístupové cesty

Poslední vlastností IEC softwarového modelu jsou přístupové cesty. Umožňují přenos dat mezi jednotlivými konfiguracemi. Každá konfigurace může mít množství určených proměnných, ke kterým může být přistupováno z jiných, vzdálených konfigurací. Přístupové cesty jsou určeny pomocí speciální proměnné deklarované s použitím konstrukce známé jako VAR_ACCESS.

Tvůrce systému může tedy specifikovat množinu proměnných, které budou tvořit rozhraní konfigurace ke všem dalším vzdáleným konfiguracím.

Řídicí tok

Obecně jsou konfigurace a zdroje zodpovědné za řízení a provádění softwarových elementů v nich. Start konfigurace vyvolá posloupnost reakcí, jejichž výsledkem je start všech softwarových elementů v ní.

Nastartování konfigurace

- Když je konfigurace nastartována, všechny globální proměnné se inicializují a pak se nastartují všechny zdroje.
- Když jsou nastartovány všechny zdroje, inicializují se všechny jejich proměnné a pak jsou povoleny všechny tasky.
- Jakmile jsou povoleny všechny tasky, začnou se provádět všechny programy a funkční bloky k nim přiřazené.

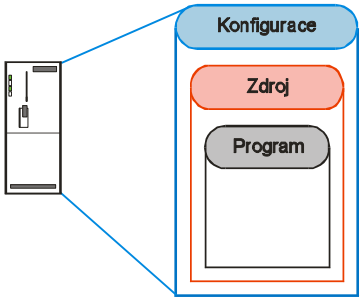
Ukončení konfigurace

- Když je konfigurace zastavena, všechny její zdroje jsou zastaveny.
- Když jsou zastaveny zdroje, jsou zakázány všechny tasky s výsledkem, se kterým programy a funkce ukončily činnost.

Mapování softwarového modelu na reálný systém

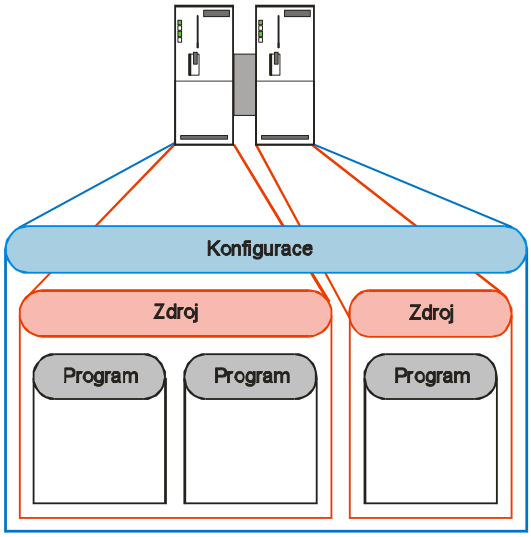
Mapování různých částí softwarového modelu na reálný systém je hlavně implementační otázka.

V malých PLC systémech s jedním procesorem se obvykle softwarový model zjednoduší na jednu konfiguraci, zdroj a program (obr. 11.2) . Konfigurace obsahuje celý software, který určuje chování PLC systému. Zdroj reprezentuje schopnosti procesorové desky a program reprezentuje část softwaru zpracovávající signály – běžně reprezentovanou ladder programem.



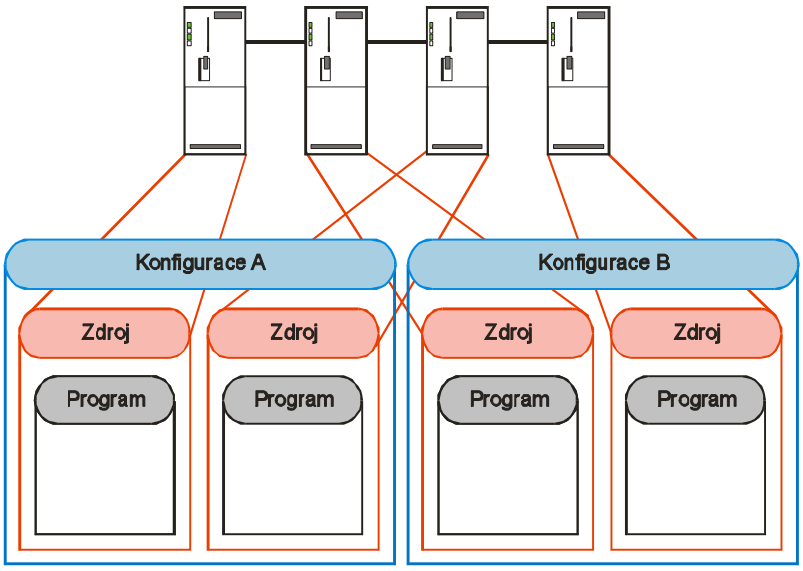
Obr. 11.2: Malé PLC s jedním procesorem.

U větších systémů s více procesory je mapování komplexnější. Většina PLC však může být stále považována za jednu konfiguraci. Každá procesorová deska obvykle reprezentuje zdroj. Každý zdroj pak může podporovat jeden nebo více programů – podle multitaskových možností procesoru (obr. 11.3).



Obr. 11.3: Multiprocesorové PLC.

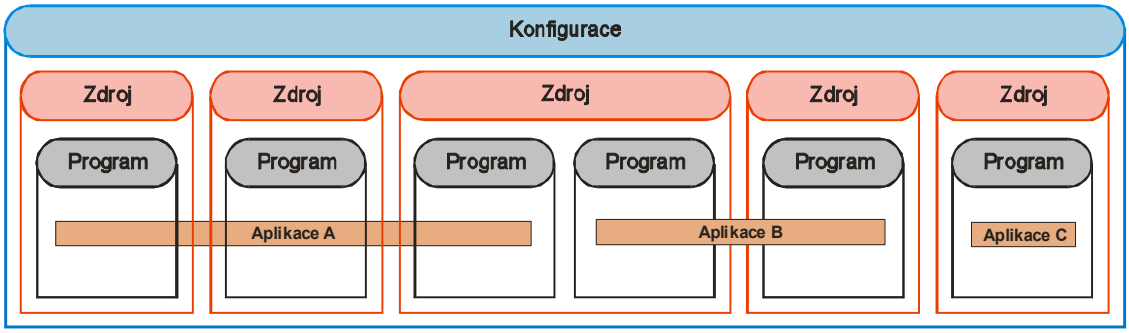
Ve velkých distribuovaných systémech, s mnoha procesory propojenými rychlými sítěmi, mohou být různé skupiny řídicích prvků považovány jako jedna konfigurace nebo jako několik konfigurací (obr. 11.4).



Obr. 11.4: Distribuovaný systém.

Vidíme, že IEC umožňuje vytvářet komplexní konfigurace, které obsahují množství zdrojů a programů. Pojem aplikace však není ve standardu definován.

Řešení řídicího problému, neboli aplikace, musí zajišťovat všechny řídicí požadavky od startu, přes běh, po vypnutí stroje nebo technologie. Takové řízení obsahuje zpravidla práci se vstupy a výstupy, regulační smyčky, komunikace apod. Aby aplikace splnila všechny tyto požadavky, může vyžadovat spolupráci několika IEC programů nahraných do jednoho nebo více zdrojů. Jelikož IEC dovoluje nahrávat a startovat zdroje nezávisle, není důvod proč by velký PLC systém nemohl provádět několik nezávislých aplikací současně. PLC systém může například provádět řadu aplikací, které nezávisle řídí několik reaktorů, parních generátorů a kompresorů, zatímco monitoruje kritické podmínky okolního prostředí. Každá aplikace může být startována nezávisle, pokud je nahrána do několika zdrojů.



Obr. 11.5: Mapování aplikací do zdrojů.

Nejmenší aplikace budou vždy vyžadovat jeden program.

Programové organizační jednotky (POU)

IEC standard označuje programy, funkční bloky a funkce jako programové organizační jednotky. Tyto softwarové elementy mají důležitou vlastnost, že mohou být použity opakovaně v různých částech aplikace. Chování a struktura programové organizační jednotky, jako např. funkčního bloku, je definována pomocí typové deklarace. Kopie funkčních bloků jsou označovány jako instance funkčního bloku. Programové organizační jednotky umožňují snadnější znovupoužití software, ať na vyšší úrovni

programů nebo na nízké úrovni funkcí a funkčních bloků. Typová definice POU může být považována za prototyp, ze kterého lze vytvořit řadu instancí s totožným chováním.

11.3 Společné prvky

Společné programovací elementy

IEC standard definuje řadu společných vlastností a elementu, které mohou být použity ve všech programovacích jazycích. Činnost programových organizačních jednotek může být popsána pomocí jednoho z pěti programovacích jazyků. Bez ohledu na použitý jazyk lze popsat proměnné a datové typy používané programovými organizačními jednotkami pomocí společných programových elementů.

Znaková sada

Aby bylo zajištěno, že IEC programy budou čitelné na různých systémech, všechny textové informace by měly používat omezený soubor písmen, čísel a znaků. Standard vyžaduje použití pouze znaků podle normy ISO 646.

Identifikátory

Identifikátory se používají pro označování různých elementů v IEC jazycích, jako např. proměnných, datových typů, funkčních bloků apod. Identifikátor může být řetězec znaků, číslic a podtržítek takový, že první znak není číslice a nejsou tam více než dvě podtržítka za sebou. Standard poskytuje podporu pro identifikátory s malými písmeny a s prvním znakem podtržítkem.

Klíčová slova

Klíčové slovo je speciální slovo, které se používá v IEC jazycích k definování různých konstrukcí nebo začátku a konce určitého softwarového elementu – např. FUNCTION a END_FUNCTION.

Hlavním pravidlem je to, že je třeba se vyvarovat používání identifikátorů, které by mohly být zaměněny za klíčová slova – některé kompilátory jsou však schopny je automaticky odlišit podle jejich pozice v programu. Je třeba se vyvarovat použití identifikátorů, jako TYPE, TRUE, PROGRAM apod. nebo identifikátory se stejným názvem jako mají standardní funkční bloky a funkce, např. TON, RS, SIN ... .

Komentáře

V kterémkoli z IEC jazyků může být vložen komentář. Délka může být od krátkých textů do několikařádkových komentářů. Komentáře jsou označeny znaky (*.....*), např. (*Zapnutí motoru*)

Komentáře mohou být umístěny kdekoliv, kde je akceptovatelná jedna nebo více mezer, pouze u jazyka IL jsou určitá omezení.

Datové typy

IEC standard poskytuje řadu standardních elementárních datových typů pro práci s hodnotami typických průmyslových proměnných. Jsou mezi nimi čísla s pohyblivou čárkou vhodné pro aritmetické výpočty, čísla integer pro čítání, boolean pro logické operace, hodnoty časů a dat, stringy pro textové řetězce a bity, byty a wordy pro operace na nižších úrovních. Datové typy:

- **Integer** – může být SINT(8b.), INT(16b.), DINT(32b.) a LINT(64b.). Tyto typy mají rozsahy odpovídající jejich bitové reprezentaci. Jsou to znaménkové typy a tedy mají rozsah od záporných do kladných hodnot. Mohou však být i bez znaménka a ty se pak označují jako USINT, UINT, IDINT, ULINT. Čísla integer jsou reprezentována např. jako –132, 0, +456 apod.
- **Real** – může být buď REAL(32b., rozsah $\pm 10^{38}$) nebo LREAL(64b. , rozsah $\pm 10^{308}$). Formát je definován normou IEC 559 a je totožný s obecně používaným IEEE formátem pro plovoucí desetinnou čárku. Typické použití čísel real je uchovávání hodnot analogových vstupů, v PID algoritmu, u analogových výstupů. Zapisují se jako např. 10.123, -0.00123, 0.9E-10, 0.369e+12 apod.
- **Doba** – TIME – slouží pro ukládání časových údajů. Bitová délka závisí na konkrétní implementaci. Zapisují se jako např. T#12d3h3s, T#3s25ms, TIME#12d_3h_3s, T#10.12s apod.
- **Datum a denní čas** – DATE, TIME_OF_DAY (TOD), DATE_AND_TIME (DT) – pro ukládání data a času. Bitová délka závisí na konkrétní implementaci. Zápis může být např. D#1994-06-10, DATE#1994-06-10, TOD#10:31:25, TIME_OF_DAY#10:31:25, DT#1999-05-01-15:36:20.56 apod.
- **Řetězce** – STRING – určují řadu tisknutelných i netisknutelných znaků. Všechny řetězce musí být uzavřeny v jednoduchých uvozovkách – '.....'. Netisknutelné znaky mohou být vloženy pomocí prefixu $ a hexadecimálního kódu znaku. Existuje rovněž řada řídicích znaků, které umožňují např. přesun na nový řádek apod. (Lewis, 1998). Zápis stringů může být např. 'Motor M1', 'Konec zprávy', '$01$10' apod.
- **Bitové řetězce** – slouží ukládání binárních informací, lze je využít jako stavové bity. Mohou být typu BOOL (1b.), BYTE (8b.), WORD (16b.), DWORD (32b.), LWORD (64b.).

Počáteční hodnoty základních datových typů

Důležitým principem IEC je to, že všechny proměnné mají známou počáteční hodnotu. U všech proměnných je základní počáteční hodnotou nula, u stringů je to prázdný řetězec, u data je to D#0001-01-01. Základní počáteční hodnoty jsou přepsány, když je definován konkrétní datový typ.

Odvozené datové typy

Ze základních datových typů mohou být vytvořeny odvozené datové typy, které mohou pomoci lepší čitelnosti programu. Nový datový typ je zaveden pomocí deklarace TYPE ... END_TYPE.

Např. TYPE

Tlak: REAL;

END_TYPE;

Strukturované datové typy – deklarace STRUCT...END_STRUCT. Tento kombinovaný typ se vytvoří jako struktura pomocí existujících datových typů. Používá se tam, kde data pro určitý objekt (např. senzor) obsahují několik různých informací (např. tlak, stav snímače, kalibrační hodnotu a maximální hodnotu).

Výčtové datové typy – dovolují, aby jednotlivé hodnoty proměnné byly označeny různými názvy.

Např.: TYPE MOD:

(INICIALIZACE, BEH, STOP, CHYBA)

END_TYPE

Podmnožinové typy – používají se tehdy, když je třeba omezit rozsah hodnot, které mohou být přiřazeny proměnné Integer.

Např: TYPE

MOTOR_U: INT(-6..+12);

END_TYPE;

Pole – používají se tehdy, když je nutné ukládat pole hodnot. Jsou často označovány jako multi-elementové proměnné.

Např: TYPE TLAK_DATA:

TLAK[1..20] OF TLAK;

END_TYPE;

Základní počáteční hodnoty odvozených datových typů

Každému odvozenému datovému typu může být přiřazena nová počáteční hodnota, která nahradí základní počáteční hodnoty použitých základních typů. Počáteční hodnota se definuje zároveň s typem.

Např.: TYPE TLAK: REAL:=1.0; (*počáteční hodnota 1 Bar*)

END_TYPE

Proměnné

Proměnné se definují na začátku každé definice jednotlivých programových organizačních jednotek (programu, funkčních bloků a funkcí. Existují různé kategorie proměnných:

- Vstupní – slouží jako vstupní parametr do POU a je ji přiřazena hodnota z externího zdroje. Konstrukce VAR_INPUT.
- Výstupní - slouží jako výstupní parametr do POU a poskytuje hodnotu externí proměnné. Konstrukce VAR_OUTPUT.
- Vstupně/výstupní – vstupně/výstupní parametr POU, nabývá hodnotu z externího zdroje, ale POU ji může modifikovat. Konstrukce VAR_IN_OUT.
- Interní proměnná POU – neboli lokální – je použitelná uvnitř POU, kde je definována. Konstrukce VAR.
- Globální proměnné jsou proměnné, které jsou deklarovány v programech, zdrojích nebo konfiguracích. Každá POU existující v dané konfiguraci, zdroji nebo programu k ní může přistupovat pomocí externích proměnných. Konstrukce VAR_GLOBAL.
- Externí proměnné - poskytují přístup ke globálním proměnným definovaným v programech, zdrojích nebo konfiguracích. Konstrukce VAR_EXTERNAL.
- Dočasné proměnné – IEC umožňuje používání dočasných proměnných, které jsou deklarovány v POU. Konstrukce VAR_TEMP.

Proměnné mohou mít následující parametry:

- Retain – tento parametr říká, že daná proměnná bude zachována i při výpadku napájení, když proběhne restart PLC.
- Constant – parametr říká, že se jedná o konstantu a že její hodnotu nelze měnit.

- At – tento parametr slouží pro umožnění pevného umístění proměnné v paměti PLC (lze zadat, na které paměťové buňce bude proměnná uložena).

Inicializace proměnných

Proměnným může být při jejich deklaraci přiřazena počáteční hodnota. Ta nahradí základní počáteční hodnotu definovanou pro daný typ.

Funkce

Funkce jsou vícenásobně použitelné softwarové elementy, které když jsou vykonány s určitou skupinou vstupních hodnot, dají jako výsledek jednu hlavní hodnotu. Jako příklad lze uvést funkce SIN, COS, SQRT apod. Funkce mohou být použity v kterémkoliv z programovacích jazyků.

Důležitou vlastností všech funkcí je to, že si nemohou ukládat interní proměnné – narozdíl od funkčních bloků, které mohou ukládat data do interních a výstupních proměnných.

Existuje celá řada standardních funkcí. Navíc si může uživatel definovat své funkce. Standardní a uživatelské funkce pak mohou být dále používány při tvorbě dalších funkcí, funkčních bloků a programů.

Deklarace funkčních typů

Nový funkční typ se definuje pomocí konstrukce FUNCTION ... END_FUNCTION.

Např.:

```
FUNCTION PRUMER:          REAL
        VAR_INPUT
                INPUT1, INPUT2:       REAL;
        END_VAR
        PRUMER = (INPUT1+INPUT2)/2;
END_FUNCTION
```

Funkce pro konverzi datových typů

IEC 1131-3 vyžaduje přísnou typovou kontrolu dat. Programovací stanice nebo jazykový kompilátor musí vždy zkontrolovat, zda jsou datové typy používány korektně. Pro případ, že je nutné provést výpočet, operaci apod. s různými datovými typy, poskytuje standard řadu konverzních funkcí. Jsou to funkce typu:

<Vstupní datový typ>_TO_< Výstupní datový typ >

Ostatní funkce

Standard definuje celou řadu běžně použitelných funkcí. Jsou to:

- Numerické funkce (např. SIN, LOG, EXP ...).
- Aritmetické funkce (např. ADD, SUB, MUL ...).
- Funkce s bitovými řetězci (např. SHL, SHR, ROL ...).
- Booleovské funkce (např. AND, OR ...).

- Funkce výběru (např. MAX, MIN, SEL ...).
- Porovnávací funkce (např. GT, EQ, LT ...).
- Funkce se znakovými řetězci (např. LEFT, RIGHT, INSERT ...).
- Funkce pro práci s datem a časem.
- Funkce pro převody časových údajů.
- Funkce pro výčtové datové typy.

Provádění funkcí

Pokud je funkce použita v některém z grafických programovacích jazyků, provedení funkce se zajistí pomocí speciálního vstupu EN. Pokud se na vstupu EN objeví hodnota TRUE, funkce se vykoná. Je-li na EN FALSE, funkce je neaktivní a na výstupu není přiřazena hodnota. Funkce má rovněž podobný speciální výstup ENO, na kterém se objeví TRUE, pokud je funkce úspěšně provedena. Pokud při vykonávání funkce vznikne chyba, ENO zůstává FALSE. Když se definuje nový funkční typ, EN a ENO jsou implicitně dostupné. Není je třeba zvlášť definovat.

Funkční bloky

Funkční bloky jsou programovými organizačními jednotkami, které umožňují aplikovat určitý algoritmus nebo skupinu akcí na vstupní proměnné a vytvořit tak množinu výstupních proměnných. Používají se pro realizaci PID algoritmů, čítačů, filtrů apod. Funkční blok má definovanou množinu vstupních, výstupních, interních a dočasných proměnných. Obsahuje také příslušný algoritmus pro práci s těmito daty. Funkční bloky mají jednu hlavní výhodu oproti funkcím a to je uchovávání dat.

Typ funkčního bloku je specifikace datové struktury a algoritmu. Instance funkčního bloku je určitá množina datových hodnot uložených ve strukturách bloku a zpracovávaných jeho algoritmem.

Použití funkčního bloku

Při používání funkčních bloků platí následující pravidla:

- Externě lze přistupovat pouze k vstupním a výstupním parametrům bloku. Interní proměnné nejsou přístupné.
- Instance funkčního bloku je vyvolána, pokud je součástí grafického programu propojených bloků tvořících POU, nebo pokud je volána z textového jazyku (ST, IL).
- Instance určitého funkčního bloku může být použita v jiném funkčním bloku nebo programu.
- Instance funkčních bloků jsou vždy deklarovány pomocí deklarací proměnných v POU (v programu, ve funkčním bloku). Je-li deklarace proměnných lokální, instance funkčního bloku je viditelná pouze v daném POU. Instance funkčního bloku může být definována jako globální, pak je blok přístupný z jakéhokoliv programu nebo funkčního bloku ve zdroji nebo konfiguraci, kde je globální proměnná definována.
- Je možné pokládat instanci funkčního bloku jako vstup do jiného programu, funkce nebo funkčního bloku.
- V textových jazycích může být k okamžitým hodnotám vstupů a výstupů funkčního bloku přistupováno stejně, jako k datům v datové struktuře.
- Instance funkčního bloku může být rovněž reprezentována v grafických jazycích (LD a FBD).

Typ funkčního bloku se definuje pomocí konstrukce FUNCTION_BLOCK...END_ FUNCTION_BLOCK. Začíná se definováním proměnných a následuje software, který specifikuje algoritmus. Tento software může být zapsán některým z IEC jazyků. Definice ve funkčním bloku obsahují rovněž instance dalších funkčních bloků.

Instance funkčních bloků se deklarují stejným způsobem jako proměnné. Instance funkčních bloků mohou být deklarovány pouze v rámci definice programu nebo funkčního bloku.

Programy

Program je nejvyšší forma programové organizační jednotky a může být deklarován na úrovni zdroje. Program je v principu velice podobný funkčnímu bloku a tvoří veliký opakovaně použitelný softwarový komponent. Je definován pomocí programového typu, který obsahuje podobně jako funkční blok, deklarace vstupních, výstupních a interních proměnných a tělo, které definuje činnost programu.

Program se obvykle považuje ze velký stavební blok software. Typický PLC projekt může používat rozsáhlé knihovny funkčních bloků, ale pouze několik programů. Program může být například použit pro řízení hlavní části technologie, jako například parní turbíny apod.

Mezi programy a funkčními bloky jsou následující rozdíly:

- PLC programy mohou obsahovat deklarace proměnných přímo reprezentovaných proměnných, které dovolují přístup pomocí hierarchických paměťových adres.
- Program může obsahovat deklarace globálních proměnných. Ty mohou být využity funkčními bloky pomocí externích proměnných.
- Program může obsahovat přístupové proměnné, které umožňují pojmenovaným proměnným, aby byly využívány vzdáleným zařízením pomocí komunikačních služeb.
- Program nesmí obsahovat instance dalších programů.
- Program může obsahovat instance funkčních bloků, které mohou být případně prováděny prostřednictvím jiného tasku.
- Instance programů mohou být deklarovány pouze ve zdrojích.

Programový typ se definuje pomocí konstrukce PROGRAM...END_PROGRAM. Začíná se definováním proměnných a následuje software, který specifikuje algoritmus. Tento software může být zapsán některým z IEC jazyků.

Instance programů se deklarují pomocí klíčového slova PROGRAM, za kterým následuje identifikátor a typ programu.

Zdroje

Zdroj je v podstatě procesní služba, která je schopná provádět IEC programy. Zdroj je definován klíčovým slovem RESOURCE, za kterým následuje identifikátor a typ procesoru, na kterém bude zdroj nahrán. Definiční oblast zdroje obsahuje deklarace proměnných, tasků a programů. Definice zdroje je ukončena klíčovým slovem END_RESOURCE.

Zdroj může obsahovat deklarace:

- Globálních proměnných.
- Přístupové cesty, tzn. proměnné, které umožňují vzdálený přístup k pojmenované proměnné.
- Deklarace proměnných včetně externích proměnných, pomocí kterých se lze odkazovat na globální proměnné z úrovně konfigurace.

- Deklarace programů.
- Definice tasků.

Tasky

Při vyvíjení real-time systému potřebuje vývojář flexibilní a konfigurovatelné řízení spouštěcích dob jednotlivých částí programů. Obvykle mohou být jednotlivé části programu prováděny v intervalech závislých na požadované odezvě systému a na potřebě optimalizovat využití výpočetní kapacity PLC.

Tasky jsou asociovány s určitým zdrojem, který řídí jeho vykonávání. Standard dovoluje přiřadit označené funkční bloky a programy určitému tasku. Task je plánovací nástroj, který se spouští periodicky v daném intervalu nebo podle změny stavu nějaké boolovské proměnné.

Když je definováno několik tasků, jsou jim obvykle přiřazeny různé intervaly a priority. Moment, kdy je daný task proveden závisí na plánovacím režimu, prioritě, intervalu provádění ostatních tasků a také na tom, jak dlouho ostatní tasky jsou připraveny a čekají na spuštění. PLC může poskytnou dvě metody plánování – non-preemptivní (pokud je task spuštěn, nemůže být přerušen, doku se celý nevykoná) a preemptivní (task může být přerušen taskem s vyšší prioritou – nutné pro deterministické systémy).

Deklarace tasku začíná klíčovým slovem TASK, pak následuje identifikátor a volitelné hodnoty následujících parametrů – SINGLE (binární proměnná, jejíž náběžná hrana způsobí spuštění tasku), INTERVAL (perioda vykonávání tasku), PRIORITY (priorita tasku). Programy a funkční bloky se taskům přiřazují pomocí klíčového slova WITH.

Např.: PROGRAM TURBINA1 WITH PROG_TASK: TURBINA (PROM1:=A, PROM2:=B)

Konfigurace

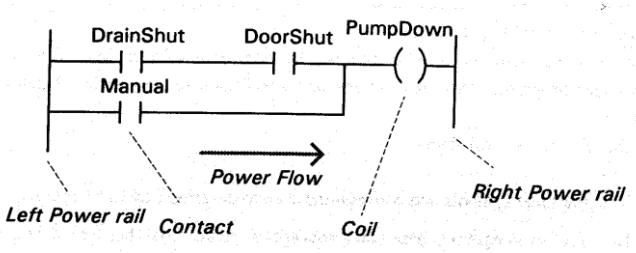
Konfigurace definuje software pro celé PLC nebo programovatelný řídicí systém a bude vždy obsahovat alespoň jeden zdroj. Konfigurace je specifická pro určitý typ PLC a určité sestavení jeho hardware. Může být použita k vytvoření software pro jiné PLC pouze v případě, pokud je toto PLC identické s původním. Obecně odpovídá konfigurace konkrétnímu sestavení hardware PLC, které bude obsahovat specifické:

- Procesní zdroje (CPU).
- Paměťové adresy pro vstupní a výstupní kanály.
- Systémové vlastnosti.

Konfigurace se deklaruje pomocí konstrukce CONFIGURATION ... END_ CONFIGURATION.

11.4 Ladder diagram

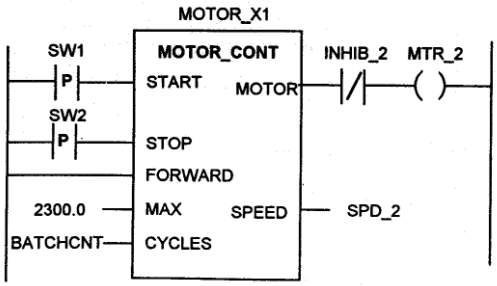
Ladder diagram je grafický jazyk, který lze použít jak pro psaní programů, tak pro definování podmínek v Sequential Function Chartu. LD byl IEC vyvinut s ohledem na většinu obecně používaných symbolů a terminologie u běžných PLC. LD je založen na technice návrhu reléových obvodů. LD má vždy na levé straně vertikální linii, která představuje napájecí vodič, který napájí horizontální linie. Každý kontakt reprezentuje stav binární proměnné. Pokud jsou všechny kontakty v horizontální linii spojeny, napájecí napětí projde a provede "coil" na pravé straně linie. Spínací kontakty (normally open) dávají hodnotu "true", když je kontakt sepnut, rozpínací kontakty (normally closed) naopak. Kombinací těchto kontaktů lze realizovat jakoukoliv logickou funkci. Kontakt realizuje read-only přístup k proměnné a nemůže být použit ke změně stavu proměnné. Na druhou stranu "coil" poskytuje write-only přístup k proměnné.



Obr. 11.6: Koncept LD.

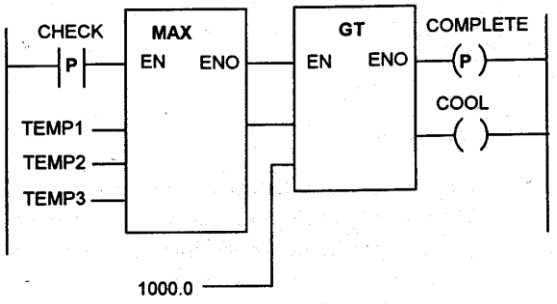
Norma umožňuje, aby pravá napájecí linie nebyla zobrazena.

V LD lze běžně používat funkční bloky.



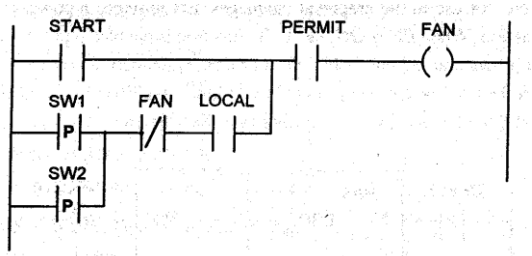
Obr. 11.7: Použití funkčních bloků v LD.

Funkce v LD mají doplňkový vstup EN a výstup ENO. EN přivádí tok signálu do funkce. Pokud je EN "1", funkce je povolena a může zpravovat své vstupy a nastavit výstupy. Když je provádění funkce dokončeno, může být ENO použito k zavedení toku signálu do další funkce nebo "coil".



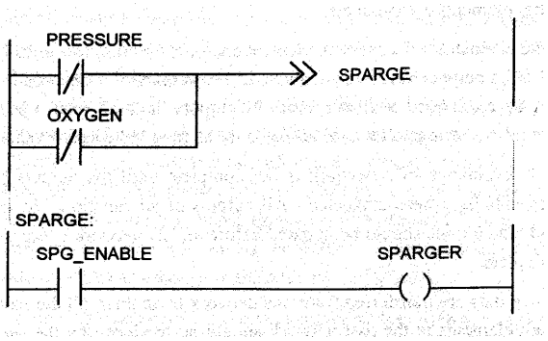
Obr. 11.8: Využití ENO.

Je možné vytvářet LD schémata, které obsahují zpětnou vazbu, tzn. jedna nebo více hodnot použitých jako kontakty nebo vstupy funkcí a funkčních bloků, je brána z proměnné, která je tímto schématem nastavována.



Obr. 11.9: Příklad zpětné vazby.

Pro přenesení řízení programu jinam lze v LD využít skok na návěští.



Obr. 11.10: Skok v LD.

Pravidla provádění LD:

- Žádný prvek networku není vyčíslen, dokud nebyly vyčísleny stavy všech jeho vstupů.
- Vyčíslení elementu networku není dokončeno, dokud nebyly vyčísleny stavy všech jeho výstupů.
- Vyčíslení networku není kompletní, dokud nejsou vyčísleny všechny výstupy všech jeho elementů.
- Jsou-li přenášena data z jednoho networku do druhého, všechny hodnoty procházející z prvního networku by měly být vyčísleny ve stejném průchodu programu networkem. Další network nezačne vyčíslování, dokud nejsou dostupné všechny hodnoty z prvního networku.

Programy zapsané v LD nebo FBD nejsou vždy přenositelné do dalších IEC jazyků. Jednoduché networky, které obsahují logické funkce, lze ve většině případů přenést. Při převodu networků, které obsahují funkce, bývá problém se signály EN a ENO, které nemají v ST odpovídající reprezentaci. To je určitým způsobem vyřešeno u rozšíření normy z roku 1993, kde je umožněno, aby funkce reprezentované v ST měly více výstupů. Převod z ST do LD také není vždy možný. Konstrukce jako IF...THEN, CASE, FOR, WHILE, REPEAT nelze přímo vyjádřit v LD. Rovněž nelze používat odkazy na pole nebo datové struktury.

LD je velice efektivní pro popis binární logiky. V tomto případě je jednoduchý na programování a dobře čitelný. Naopak použití LD v aplikacích s aritmetickými výpočty, zpětnovazebním řízením apod. může vést k velice složitým schématům, které mohou být obtížné na programování a špatně čitelné. Proto je doporučeno, aby LD sekce v programu byly dostatečně krátké. Ideální použití LD je pro zápis kombinační logiky.

11.5 Instruction List

Jazyk IL je jazykem nižší úrovně, podobný assembleru. IEC vyvinul IL ne základě řady jazyků nižší úrovně používaných výrobci programovatelných automatů. Základní struktura jazyka je velice jednoduchá a nenáročná na zvládnutí. IL je ideální pro řešení malých řídicích úloh, kde je několik rozhodovacích podmínek a omezený počet změn řídicího toku programu. Řada výrobců dává přednost IL před ST. Výhodou IL je to, že je jednodušší na implementaci. Často není ani potřeba jej před nahráním do PLC kompilovat.

Dalším příkladem, kde je vhodné použít IL je programování kritických sekcí programu - IL umožňuje psát úsporný, optimalizovaný kód.

Struktura IL jazyka

IL je jazyk, který se skládá ze série instrukcí, kdy každá z nich je na jednom řádku. Za instrukcí následuje jeden nebo více operandů. Operandy jsou reprezentovány proměnnými nebo konstantami definovanými jako "Obecné elementy". Každá instrukce může používat nebo měnit hodnotu uloženou v jednom registru. Standard označuje tento registr jako "výsledek" instrukce. "Výsledek" je někdy označován jako akumulátor. IL také poskytuje srovnávací operace, kdy se srovnává hodnota operandu s akumulátorem. Různé typy instrukcí skoku mohou testovat hodnotu v akumulátoru a podle toho skočit. IL instrukce má následující strukturu:

Návěští	Operátor	Operand	Komentář
SKOK1:	LD	1000	(*Načtení konstanty*)

Komentáře se píšou ve stejném tvaru jako u ST. Sémantiku většiny instrukcí IL lze popsat následujícím výrazem:

NovýVýsledek := AktuálníVýsledek	Operator	Operand	----> např.
NovýVýsledek := AktuálníVýsledek	SUB	10	

Některé instrukce mohou mít tzv. modifikátor. Například modifikátor "N" je použit pro negování hodnoty operandu.

IL poskytuje tři různé formáty instrukce CALL pro volání funkčních bloků a dva formáty pro volání funkcí. Syntaxe volání funkcí a funkčních bloků není stejná.

- Volání funkčního bloku s použitím formálního volání se vstupními parametry - za instrukcí CALL následuje seznam parametrů s jejich hodnotami.
- Volání funkčního bloku s použitím neformálního volání - hodnoty vstupních parametrů jsou nastaveny před voláním FB.
- Volání funkčního bloku pomocí jejich vstupních operátorů - formát používaný jen u standardních FB.
- Volání funkce pomocí formálního volání.
- Volání funkce pomocí neformálního volání.

IL umožňuje uživatel také definovat funkční bloky a funkce. Pokud definujeme funkci, výsledná hodnota bude po provedení funkce uložena v akumulátoru. U funkčního bloku se jeho vstupy a výstupy definují v deklaraci bloku (VAR_INPUT, VAR_OUTPUT).

Konverze IL do ostatních IEC jazyků je problematická. Převoditelnost by mohla být možná, pokud by byla používána jen omezená množina IL operátorů a instrukce by byly psány ve striktním formátu. Naopak konverze ostatních IEC jazyků do IL je jednodušší, ikdyž není vždy přímočará.

Jelikož IL je v podstatě programovací jazyk nižší úrovně, je nutné při jeho používání dbát na přehledné psaní kódu, aby byla zajištěna jeho dobrá čitelnost a údržba. Je důležité, aby program byl dobře okomentován. Instrukce skoku by měly být používány jen výjimečně.

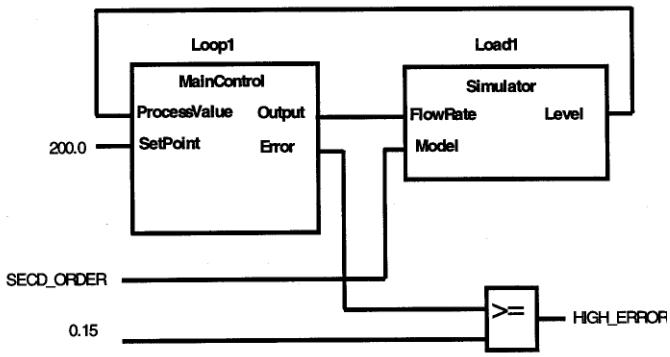
11.6 Function block diagram

Programovací jazyk, který používá pro popis chování funkcí, funkčních bloků a programů množinu vzájemně propojených grafických bloků. Může být rovněž použit u pro k vyjádření chování kroků, akcí a přechodů. V FBD popisujeme systém jako tok signálů mezi jednotlivými elementy schématu. To je velice podobné signálovým spojením v elektronických schématech. Standard definuje typy linek a propojení pro všechny typy IEC grafických jazyků.

FBD by mělo být použito tam, kde problém představuje tok signálů mezi jednotlivými bloky. Typické použití FBD je zpětnovazební řízení a binární logické funkce. V FBD platí určité grafické konvence. Funkční bloky jsou kresleny jako obdélníky se vstupy přicházejícími zleva a výstupy vycházejícími zprava. Typ funkčního bloku je vždy uvnitř, zatímco jeho jméno je vždy nad blokem. Formální jména operandů jsou uvnitř bloku.

Tok signálu

Výstupy z funkčních bloků jsou vstupy pro další funkční bloky. Výstupy FB jsou aktualizovány prostřednictvím provedení bloku. Změny stavů signálů a hodnot se proto přirozeně vyvíjí zleva doprava. Negaci binárního signálu provést pomocí kroužku ve vstupním bodu.



Obr. 11.11: Zpětná vazba v FBD.

Standard umožňuje vztažení signálu zpět z výstupu jednoho FB do vstupu jiného FB (zleva doprava). Tato zpětná vazba znamená, že hodnota ve zpětné cestě je po vyčíslení FBD networku podržena a použita jako vstupní hodnota pro následující vyčíslování networku.

Vykonávání funkcí ve FBD networku je odvozeno podle pozice funkce v networku. Provádění funkcí může být určeno pomocí povolovacích vstupů EN. Je rovněž možné přesunout provádění programu z jedné části FBD do jiné části pomocí grafických skokových možností. Sekce FBD networků mohou být označeny návěštími. Přiřazením binárního signálu identifikátoru návěští je možné přesunout provádění programu do jiné části networku, který je tímto návěštím označen.

Pro FBD platí následující základní pravidla:

- Žádný element v networku se neprovede, dokud nemá hodnoty na všech svých vstupech.
- Provedení elementu není kompletní, dokud nejsou vyčísleny všechny jeho výstupy.
- Provedení networku není kompletní, dokud nejsou vyčísleny výstupy všech jeho elementů.
- Když jsou přenášena data z jednoho networku do jiného, všechny hodnoty přicházející od elementů v prvním networku musí být vyčísleny ve stejném průchodu tímto networkem. Druhý network nesmí zahájit své provádění, dokud nemá dostupné všechny hodnoty z prvního networku.

Většina konstrukcí používaných v FBD networcích lze převést do ST. Problém nastává při převodu funkcí, kde je využito ENO.

Opačný převod - z ST do FBD - je mnohem problematičtější. Konstrukce jako IF, CASE, FOR, WHILE, REPEAT nelze přímo převést do grafické podoby. Podobně vznikají problémy s odkazy na prvky polí nebo strukturovaných typů.

11.7 Structured Text

Structured text je vyšší programovací jazyk se syntaxí podobnou Pascalu. I přes určitou podobnost s Pascalem je ST samostatný programovací jazyk vyvinutý pro průmyslové řídicí aplikace. Může být použit k vyjádření chování funkcí, funkčních bloků a programů. Rovněž může být použit v SFC k vyjádření kroků, akcí a přechodů.

Jednotlivé řádky programu mohou být psány v libovolném stylu. Mezi klíčová slova a identifikátory mohou být vkládány tabelátory, znaky konce řádků a komentáře. Jazyk je dobře čitelný a přehledný. Poskytuje celý rozsah konstrukcí pro přiřazování hodnot proměnným, volání funkcí a funkčních bloků, pro vytváření matematických výrazů, pro podmíněné provádění vybraných řádků, pro cykly.

Základní pravidla ST:

Přiřazení - x:=y;.

- Výrazy - používají se pro vyčíslení hodnot odvozených z jiných proměnných a konstant. Výrazy se mohou skládat z vnořených jednodušších výrazů.
- Operátory - ST obsahuje standardní operátory pro aritmetické a logické operace.
- Vyčíslování výrazů - výrazy jsou vyčíslovány s ohledem na přednost jednotlivých operátorů před ostatními - viz. tab.
- Volání funkčních bloků - instance funkčního bloku může být aktivována voláním jména instance funkčního bloku s hodnotami příslušných parametrů. Pokud nejsou hodnoty parametrů poskytnuty, jsou použity hodnoty z posledního volání.
- Podmíněné provádění programu - konstrukce IF...THEN...ELSE, CASE.
- Cyklické provádění programu - FOR...DO, WHILE...DO, REPEAT...UNTIL, EXIT, RETURN

Shrnutí pojmů

Norma IEC 61131 zavádí standard do techniky programovatelných automatů. Obsahuje několik částí. Ve třetí části této normy jsou definovány **standardní programovací jazyky** pro programovatelné automaty. Tato část rovněž popisuje **IEC softwarový model**, **komunikační model** a **společné prvky** programovacích jazyků.

Třetí část normy IEC 61131 definuje 5 standardních programovacích jazyků pro programovatelné automaty. Jsou mezi nimi základní nižší jazyky – **IL**, **LAD**, **FBD**, vyšší programovací jazyk **ST** a grafický programovací jazyk **SFC.** Každý z jazyků je vhodný pro specifický typ úloh.

Kontrolní otázky

1. Co je to norma IEC 61131 a k čemu slouží?
2. Jaké programovací jazyky jsou definovány ve třetí části této normy?
3. Jak je členěn řídicí program podle IEC 61131-3?
4. Co jsou to funkční bloky? Kdy se používají? Čím se liší od funkcí?
5. Jaké možnosti komunikace IEC 61131 definuje?
6. Co je to Ladder Diagram a pro jaké úlohy je vhodný?
7. Co je to Instruction List a pro jaké úlohy je vhodný?
8. Co je to Function Block Diagram a pro jaké úlohy je vhodný?
9. Co je to Structured Text a pro jaké úlohy je vhodný?

Další zdroje

11-1. ČSN EN 61131-3: Kopie normy ČSN EN 61131-3. Český normalizační institut, Praha, 1996.

11-2. John K. H., Tiegelkamp M.: IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems. Springer Verlag, 2001. ISBN: 3540677526.

11-3. Lewis R. W.: Programming Industrial Control Systems Using IEC 1131-3. The Institution of Electrical Engineers, London, UK, 1998. ISBN: 0852969503.

Řešená úloha 11.1.

ŘEŠENÍ SEMESTRÁLNÍHO PROJEKTU

12. KOMUNIKAČNÍ MOŽNOSTI PLC, PRŮMYSLOVÉ SBĚRNICE, DISTRIBUOVANÉ SYSTÉMY ŘÍZENÍ

Čas ke studiu: 2 hodiny

Cíl:

Kapitola se zabývá základními pojmy z oblasti **distribuovaných systémů řízení**. Dále je zde popsán **ISO-OSI model** a možnosti propojování otevřených systémů. Po prostudování kapitoly bude student schopen posoudit vlastnosti, výhody a nevýhody centrálních a distribuovaných systémů řízení. Seznámí se rovněž s ISO-OSI modelem a jeho jednotlivými vrstvami.

Druhá část kapitoly je věnována **fyzické** a **linkové vrstvě** ISO-OSI modelu. V rámci popisu fyzické vrstvy jsou popsány jednotlivé typy přenosových médií, které se v současné době používají a metody kódování bitů. Následuje popis linkové vrstvy, zejména **rozdělení kapacity přenosového kanálu**, **režimy přenosu**, **metody synchronizace** a **metody přístupu ke komunikačnímu médiu**.

Výklad

12.1 Komunikační možnosti programovatelných automatů

Komunikace je u současných programovatelných automatů jednou z jejich základních funkcí. Je to dáno stále vyšší potřebou přenášet informace mezi zařízeními v rámci řídicího systému i mimo něj. Programovatelné automaty nabízejí zejména tyto komunikační možnosti:

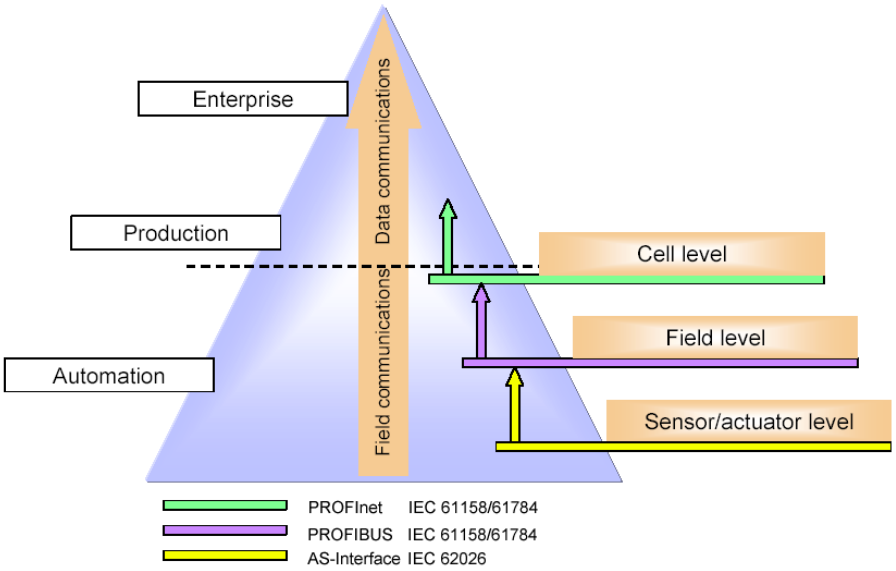
- Komunikace se senzory a akčními členy.
- Komunikace se vzdálenými jednotkami vstupů a výstupů.
- Komunikace se systémy HMI (Human Machine Interface – operátorské panely a vizualizační počítače).
- Vzájemná komunikace programovatelných automatů mezi sebou.
- Otevřená komunikace dalšími prvky používanými v rámci řídicího systému.
- Komunikace v rámci Intranetu/Internetu.
- Komunikace s programovacími a diagnostickými nástroji.

Komunikační systémy používané v průmyslové komunikaci lze rozdělit na několik úrovní (viz. Obr.12.1):

- Actuator/Senzor Level – úroveň snímačů a akčních členů. Používají se zde sběrnice pro přenos binárních signálů z výše uvedených prvků. Klade se důraz na spolehlivost, jednoduchost a nízké náklady. Příkladem je ASI (Actuator Senzor Interface).
- Field Level – „polní“ úroveň. Komunikace mezi komponentami ř.s. v rámci jedné technologické buňky – mezi PLC, PLC-DP, PLC-HMI apod. Používají se dostatečně výkonné

sběrnice, vhodné pro přímé řízení, real-time komunikační systémy. Příkladem je Profibus, Interbus apod.

- Cell Level – komunikace v nebo mezi technologickými buňkami. Používají se zde výkonné komunikační sběrnice, umožňující rychlý přenos velkých objemů dat. Komunikace mezi PLC, PLC-HMI, HMI-vyšší úroveň apod. Nejběžnějším příkladem je Ethernet a jeho modifikace.



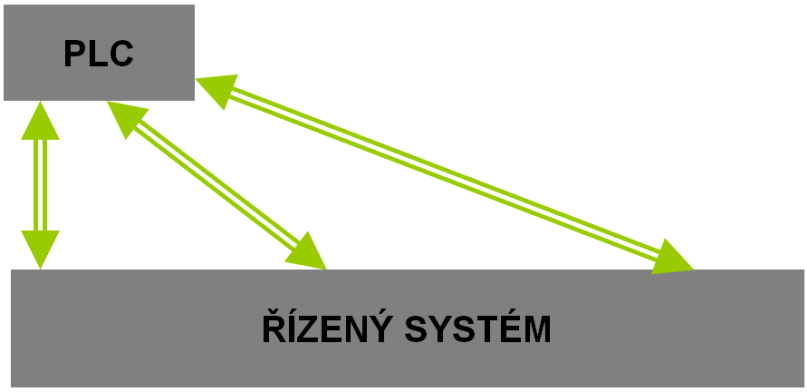
Obr. 12.1 : Úrovně průmyslové komunikace.

12.2 Centralizované x distribuované řídicí systémy

V řídicích systémech v průmyslu i jiných technických oblastech lze v současné době pozorovat dva základní přístupy:

- Centralizované řídicí systémy.
- Distribuované řídicí systémy.

Centralizovaný řídicí systém se vyznačuje tím, že je zde jeden výkonný řídicí prvek, který zajišťuje všechny řídicí aktivity. K tomuto prvku jsou přivedeny všechny potřebné technologické signály. Bloková struktura takového systému je znázorněna na obrázku 12.2.



Obr. 12.2 : Struktura centralizovaného řídicího systému.

Výhody:

- Účinný, vhodný, pokud není složitý.
- Jednodušší údržba.
- Jednodušší návrh aplikačních programů.

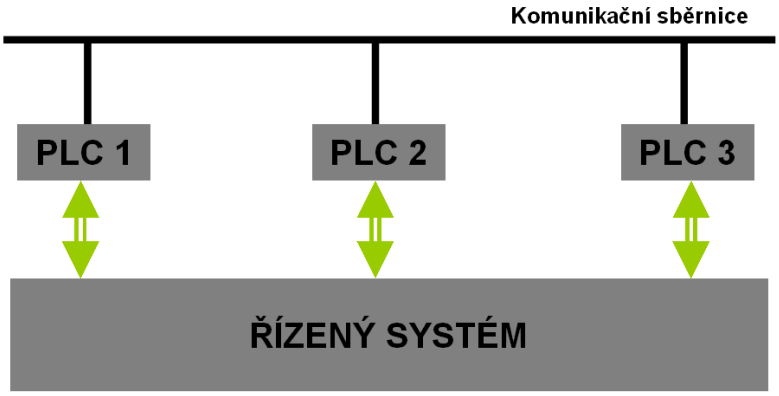
Nevýhody:

- Omezená kapacita – řídicí jednotku lze rozšiřovat jen do maximálního počtu modulů pro daný typ, je tedy většinou nutné, aby jednotka měla určitou rezervu pro případné rozšíření systému.
- Při poruše havaruje vše.
- Rozsáhlá kabeláž pro přivedení technologických signálů.

Distribuovaný řídicí systém je složen z několika dílčích systémů, které jsou mezi sebou propojeny komunikační sběrnicí a které se společně podílejí na řízení. Tento přístup se začal v automatizaci rychle rozvíjet a nasazovat v době, kdy vznikly výkonné a spolehlivé komunikační sběrnice, které bylo možné k tomuto účelu využít a rovněž kdy cena jednotlivých zařízení s procesorem byla čím dál nižší (a nebyl tedy tak výrazný cenový rozdíl, pokud se v řídicím systému použil jeden výkonný prvek nebo několik menších prvků). Komunikace je tedy základním předpokladem pro možnost realizace distribuovaných řídicích systémů.

Vzhledem k tomu, že řídicí systém obsahuje několik řídicích prvků, lze se na řídicí aplikaci dívat jako na lokální či globální, podle toho, zda běží pouze na jednom zařízení nebo zda využívá několika zařízení.

Na obrázku 12.3 je znázorněna struktura distribuované řídicí aplikace.



Obr. 12.3: Struktura centralizovaného řídicího systému.

Výhody:

- Vyšší výkonnostní možnosti.
- Výstavba složitých systémů.
- Úspora kabeláže pro přivedení technologických signálů.
- Jednodušší odlaďování nové aplikace (jelikož aplikace jako celek je rozdělena mezi několik PLC a může být laděna po částech).
- Postupné rozšiřování – řídicí systém je možné postupně rozšiřovat přidáváním dalších zařízení na komunikační sběrnici.

Nevýhody:

- Nepříznivý vliv prostředí – řídicí prvky jsou umísťovány přímo tam, kde je potřeba něco řídit a kde jsou potřebné signály.
- Komunikace mezi zařízeními různých výrobců.
- Vzájemná součinnost.

12.3 ISO-OSI model

Když se uskutečnily první pokusy s přenosem číslicových údajů prostřednictvím existující telefonní sítě, pozornost se soustředila na nejnižší úroveň komunikace, fyzickou linku. Programování se provádělo v assembleru a programátoři pracovali na úrovni bitů a proto nebyla potřebná abstraktnější reprezentace. Současná technologie nabízí prostředky pro levný přenos velkých objemů dat a pozornost se soustředí na různé aplikace – databáze, řízení procesů, počítačově řízené výrobní systémy apod. Propojení musí být zabezpečené na několika úrovních. Od úrovně bitů až po údaje a funkce, které reprezentují. Cestou k tomu je používání všeobecně přijatých standardů.

Aby se předešlo problémům souvisejícím s používáním velkého množství vzájemně nekompatibilních standardů, mezinárodní organizace ISO (International Organization for Standartization) definovala model pro komunikaci otevřených systémů – OSI (Open System Interconection). OSI není komunikačním standardem, ale poskytuje návod jak identifikovat a oddělit koncepčně odlišné části komunikačního procesu. V praxi to znamená, že OSI neurčuje napěťové úrovně, přenosové rychlosti anebo které protokoly se mají používat, aby se dosáhla vzájemná kompatibilita. Hovoří o tom, že má existovat kompatibilita jak v napěťových úrovních, rychlostech, protokolech, tak v dalších faktorech. Praktickým cílem OSI je optimální propojení sítě, ve které je možné přenášet údaje mezi různými místy bez toho, aby se zbytečně vynakládaly prostředky na konverze, s čímž souvisí časové zpoždění a poruchovost. Model OSI byl poprvé publikovaný v roce 1984 v souboru dokumentů nazvaných ISO 7498.

12.3.1 Zásady propojování otevřených systémů

OSI zavedla koncepční model pro komunikaci na rozdílných úrovních operačních systémů, na kterých se pracuje s rozdílnou úrovní abstrakce – ta se pohybuje od strojového kódu až po jazyky vyšší úrovně a aplikace. Základem modelu je vrstvová architektura. Při tvorbě modelu se zformulovalo celkem 13 principů, které je možné shrnout do následujících myšlenek:

- Samostatná vrstva by měla vzniknout všude tam, kde je potřebný jiný stupeň abstrakce.
- Každá vrstva by měla zajišťovat přesně vymezené funkce. Tyto funkce by měly být volené tak, aby pro jejich realizaci mohly být vytvořené standardizované protokoly s mezinárodní působností.
- Rozhraní mezi vrstvami by měla být volená tak, aby byl minimalizovaný tok údajů přes tato rozhraní
- Počet vrstev by měl být tak velká, aby vzájemně odlišné funkce nemusely být zahrnuty do stejné vrstvy a zároveň tak malý, aby celá architektura zůstala dostatečně přehledná.

V OSI modelu je 7 funkčních vrstev. Každá vrstva komunikuje přímo jen se sousední vrstvou, která leží nad ní nebo pod ní. Služby požaduje od nejbližší nižší vrstvy nebo je poskytuje nejbližší vyšší vrstvě. Volání služeb OSI jsou podobné voláním operačního systému – žádající vrstva posílá údaje a parametry do nejbližší nižší vrstvy a očekává odpověď, přičemž se nezabývá podrobnostmi, jak se požadavek vyřizuje. Moduly umístěné ve stejné vrstvě, ale v jiném uzlu sítě, se označují jako peer

(rovnocenné) a navzájem komunikují pomocí protokolu, který definuje formáty zpráv a pravidla pro výměnu dat.

Model OSI definuje v každé vrstvě služby, které tato vrstva musí mít k dispozici pro vyšší vrstvu. Služby jsou striktně odlišené od protokolu. Vzájemná propojitelnost je postavená na skutečnosti, že rozdílné protokoly jsou strukturované na základě podobných služeb a že v rovnocenných vrstvách jsou totožné protokoly. Základní filosofii modelu OSI lze shrnout do věty – nekombinovat entity, které navzájem nesouvisí a ať spolu související entity spolu komunikují.

12.3.2 Vrstvy modelu OSI

Fyzická vrstva - Úlohou této vrstvy je zajistit přenos jednotlivých bitů mezi příjemcem a odesílatelem prostřednictvím fyzické přenosové cesty, kterou tato vrstva bezprostředně ovládá. Fyzická vrstva přenášené bity žádným způsobem neinterpretuje. Obsahuje elektrické, mechanické a optické rozhraní spolu s potřebnými softwarovými ovladači portů. Na této úrovni se realizují všechny detaily týkající se přenosového média, úrovní signálů, frekvencí, kabelů, konektorů apod. Fyzická vrstva je ve skutečnosti jen reálným spojením mezi dvěma komunikujícími místy – uzly, nodes.

Linková vrstva - Linková vrstva se označuje také jako spojová vrstva. Tato vrstva má za úkol zajistit přenos celých bloků údajů, označovaných jako rámce, frames, jen mezi dvěma uzly. Linková vrstva má správně rozpoznat začátek a konec rámce a jeho jednotlivé části. Tyto části se označují jako pakety. Paket je část přenášeného datového souboru, kterou lze přenést najednou. Linková vrstva zajišťuje verifikaci správnosti přenosu posloupnosti bitů. Pokud například vznikla v důsledku šumu chyba přenosu, může tato vrstva požádat o opakované zaslání sekvence bitů. Linková vrstva poskytuje pro vyšší úroveň bezchybnou datovou linku mezi dvěma uzly.

Síťová vrstva - Zřizuje kompletní cestu a dohlíží na to, aby cestou od zdroje k cíli prošly všechny zprávy i v případě, že tato cesta je sestavena z odlišných větví procházející několika uzly. Musí zajistit potřebné směrování, routing (tzn. volbu vhodné trasy) přenášených rámců a zajistit postupné odevzdávání jednotlivých paketů po této trase od odesílatele k příjemci. Síťová vrstva si tedy musí uvědomovat konkrétní topologii sítě – tzn. způsob vzájemného propojení uzlů mezi sebou.

Transportní vrstva - Tato vrstva zajišťuje koncové řízení komunikace (tj. mezi původním odesílatelem a konečným příjemcem) a funguje jako rozhraní mezi aplikačním softwarem, který požaduje komunikaci a externí sítí. Tato vrstva má prostřednictvím síťové vrstvy vytvořenou iluzi, že každý uzel sítě má přímé spojení s kterýmkoliv uzlem sítě. Díky tomu se už věnuje jenom komunikaci koncových účastníků. Při odesílání dat zajišťuje sestavování jednotlivých paketů (samostatně přenášených bloků zprávy), do kterých rozděluje přenášené údaje a při přijímání je zase z paketů odebírá a skládá do původního tvaru. Dokáže zajistit přenos libovolně velkých zpráv, i když jednotlivé pakety mají jen omezenou velikost. Je odpovědná za ověřování toho, že data byla z jednoho zařízení správně vyslané a že byla v druhém zařízení správně přijata.

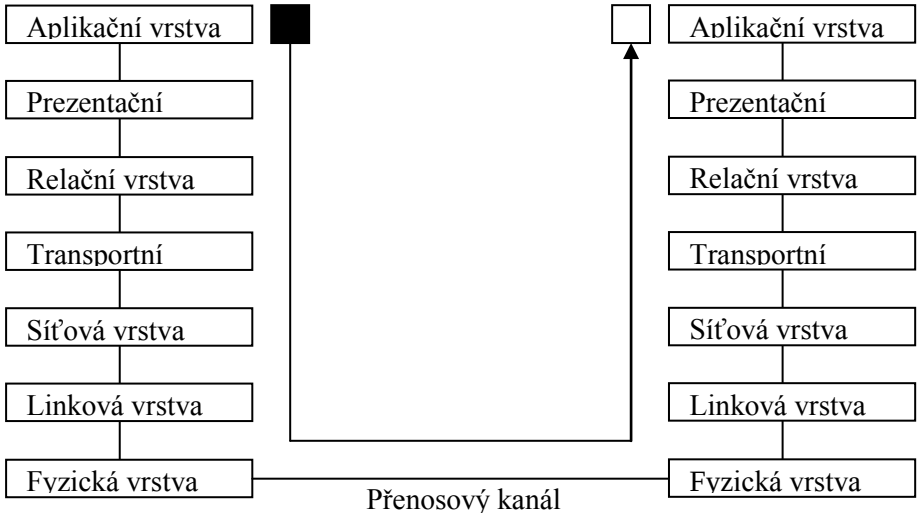
Relační vrstva - Její úlohou je navazovat, udržovat a rušit relace, sessions mezi koncovými účastníky. V rámci navazovaní relace si tato vrstva vyžádá od transportní vrstvy vytvoření spojení, prostřednictvím kterého pak probíhá komunikace mezi oběma účastníky relace. Pokud je třeba tuto komunikaci nějakým způsobem řídit (např. určovat, kdo má kdy vysílat, pokud to nemohou dělat oba účastníci současně), zajišťuje to právě tato vrstva. Má také na starosti všechno, co je třeba k ukončení relace a zrušení existujícího spojení. Vylepšuje tedy transportní vrstvu prostřednictvím dodatečných služeb, které podporují úplnou relaci mezi odlišnými zařízeními. Jedním z příkladů je spojení se vzdáleným počítačem přes síť.

Prezentační vrstva - Jednotlivé počítače mohou používat navzájem odlišnou vnitřní reprezentaci dat (např. kódy ASCII, EBCDIC apod.). Tato vrstva teda zabezpečuje potřebné kódování dat a konverzi, pomocí kterých jsou binární informace transformované na to, co vlastně samy o sobě znamenají – zprávy, texty, obrázky apod. V rámci této vrstvy bývá též realizovaná případná komprese dat, eventuálně jejich šifrování.

Aplikační vrstva - Koncoví uživatele používají počítačové sítě prostřednictvím nejrůznějších síťových aplikací – systémů elektronické pošty, přenosů souborů, vzdáleného připojování apod.

Začleňovat všechny tyto různé aplikace přímo do vrstvy by pro jejich velkou různorodost nebylo rozumné. Proto se do aplikační vrstvy zahrnují jen ty části aplikací, které realizují společné, resp. všeobecně použitelné mechanizmy. Významným prvkem je tu realizace virtuálního zařízení. Prostředky pro práci s virtuálním zařízením jsou přitom součástí aplikační vrstvy (protože jsou všude stejné), zatímco prostředky pro jeho přizpůsobení konkrétnímu zařízení už součástí aplikační vrstvy nejsou. Tato nejvyšší vrstva se zabývá úlohami aplikačního systému, jako je přenos datových souborů, činnost distribuovaných databází apod.

Konkrétní vzhled má jen fyzická vrstva. Všechny ostatní jsou soubory pravidel a popisů funkčních volání. Proto jsou implementované softwarově nebo firmwarově. Nejnižší tri vrstvy jsou síťové nebo komunikační. Tři nejvyšší jsou implementované aplikačním softwarem. Čtvrtá – transportní vrstva je spojením mezi komunikačně a aplikačně orientovanými vrstvami.



Obr. 12.4: Vrstvový model OSI.

Základní principy činnosti OSI

Dvě rovnocenné entity jsou propojené pomocí virtuálního (logického) spojení. Z hlediska rovnocenných entit (peers) se virtuální spojení jeví jako reálné, i když pouze ve vrstvě 1 je virtuální a fyzická linka totožná. Moduly peer si vyměňují data podle protokolu pro danou úroveň. V praxi to znamená, že požadují službu od vrstvy bezprostředně nižší prostřednictvím volání procedury. Podrobnosti protokolu nejsou pro entitu, která službu požaduje, viditelné a je možné je kdykoli změnit bez toho, aby to vnější entity zpozorovaly. Mezi moduly běžícími na jednom stroji neexistuje přímé spojení na vzdálenost více než jedna úroveň – ani reálně, ani virtuálně. Podobně je tomu mezi moduly běžícími na různých strojích. Např. modul, který na daném stroji beží na úrovni 4 může komunikovat pouze s úrovní 3 a 5 na tomto stroji a s úrovní 4 na vzdáleném stroji.

Ten, kdo v určité vrstvě modelu vykonává nějakou činnost, se označuje jako entita. Bývá to nejčastěji objekt programové povahy, např. určitý proces, v nižších vrstvách to může být např. hardwarový celek, např. I/O řadič. Na úrovni aplikační vrstvy jde o aplikační entity, na úrovni prezentační vrstvy jde o prezentační entity apod. Entity na stejné úrovni se označují jako peer entity.

Entity ve vrstvě N implementují služby, které jsou využívané entitami vrstvy N+1. Vrstva N tedy vystupuje jako poskytovatel služby a vrstva N+1 je v úloze uživatele služby. Vrstva N je však současně v pozici uživatele služeb k vrstvě N-1, protože používá její služby.

Spojované a nespojované služby

Spojovaná služba (Connection-oriented Service) – podobný princip jako telefonní spojení – nejdříve je nutné navázat spojení - vytočení čísla na jedné straně a zvednutí telefonu na druhé, a pak je možné

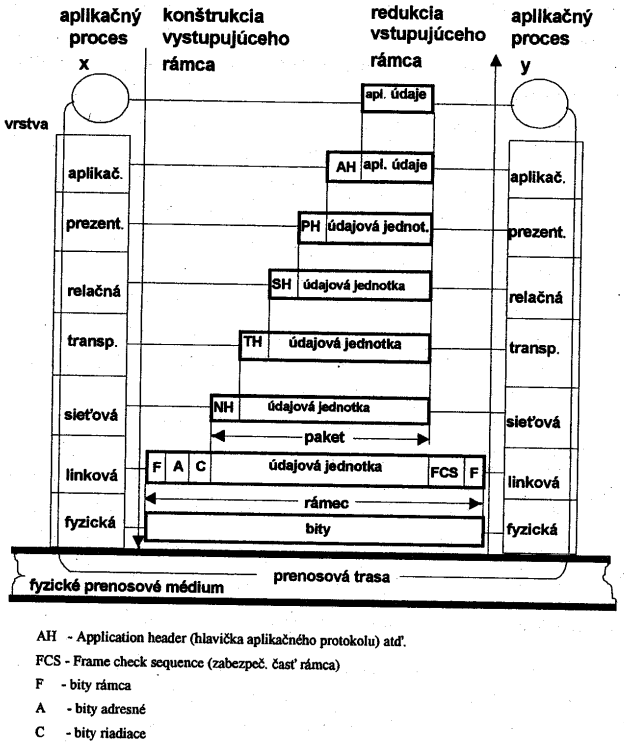
komunikovat. Stejným způsobem spolu komunikují dvě entity na stejných úrovních. Nejdříve musí navázat spojení a pak mohou spolu komunikovat. Vznikne něco jako potrubí, do něhož jsou na jedné straně data vkládána a na druhé straně ve stejném pořadí odebírána. Příklad – telekomunikační systémy. Jsou vhodné pro přenos větších objemů dat, mají větší jednorázovou režii na navázání spojení.

Nespojovaná služba (Connectionless Service) – lze ji přirovnat k dopisní poště. Jednotlivé části zpráv se považují za samostatné celky, které jsou opatřené adresou příjemce. Mohou být tedy doručeny různými cestami a jejich pořadí nemusí být zachováno. Příklad – lokální sítě. Jsou vhodné pro přenos menších objemů dat, nemají žádnou jednorázovou režii pro navázání spojení, mají však vyšší režii pro vlastní přenos.

Spolehlivé a nespolehlivé služby

Spolehlivá služba (Reliable Service) – taková, která niky neztrácí žádné informace. Obvykle je realizovaná pomocí vhodného potvrzovacího mechanizmu. S tím je však spojená určitá režie, která nemusí být vždy žádoucí, např. z hlediska zpoždění přenosu.

Nespolehlivé služby (Unreliable Service) – nebo spíše služby, které mají vysokou míru spolehlivosti, ale neposkytují záruku úspěšnosti přenosu. Jsou však rychlejší než obdobné spolehlivé služby.



Obr. 12.5: Struktura údajů v protokolech vrstvové komunikace.

12.4 Základní informace o fyzické vrstvě ISO-OSI modelu

Fyzická vrstva přenášené bity žádným způsobem neinterpretuje. Obsahuje elektrické, mechanické a optické rozhraní spolu s potřebnými softwarovými ovladači portů. Na této úrovni se realizují všechny detaily týkající se přenosového média, úrovní signálů, frekvencí, kabelů, konektorů apod. [11-1].

12.4.1 Přenosové média

Kvalitativní parametry přenosových médií:

- **Přenosová rychlost**, kterou je možné na daném kabelu dosáhnout.
- **Útlum (attenuation)** – představuje zeslabení přenášeného signálu. Je způsobený odporem, který kabel klade přenášenému signálu – bývá větší při vyšších frekvencích a rovněž roste se zmenšováním průměru kabelu. Hodnota útlumu je přímo úměrná délce kabelu a je jedním z rozhodujících faktorů určující maximální použitelnou délku souvislého úseku.
- **Odolnost proti rušení** – v okolí kabelu může docházet k rušivým jevům, které mají nepříznivý vliv na přenášený signál. Každý druh média má jinou odolnost proti rušení.

Jednou ze specifických forem rušení je tzv. přeslech (Crosstalk), kde signál přenášený jedním kabelem ovlivňuje signál přenášený jiným kabelem. Projevuje se hlavně u souběžných vedení. V důsledku útlumu a rušení dochází ke zkreslení přenášeného signálu, které se při datových přenosech projevuje chybami.

12.4.2 Elektrické vodiče

Běžnými typy vodičů používaných pro komunikaci jsou kroucená dvojlinka a koaxiální kabel. Oba typy jsou málo citlivé vůči rušení. Nejlepší ochranou je stínění kabelu. Šířka pásma kroucené dvojlinky je limitovaná několika Mhz a nelze ji tedy použít pro rychlosti větší než několik Mbit/s. Je však jednoduchá a levná. Vyšší odolnost proti rušení lze dosáhnout stíněním – stíněná kroucená dvojlinka (jeden ze standardů Ethernet ji používá pro rychlosti 10 Mbit/s).

Koaxiální kabel umožňuje šířku pásma do 500Mhz a běžně se používá pro přenos na vysokých frekvencích a TV signálů. Lze jimi přenášet mnohem rychleji než u kroucené dvojlinky.

12.4.3 Optické kabely

Viditelné světlo má frekvenci přibližně 10^8 Mhz. Přenášená číslicová data lze reprezentovat pomocí světelných impulzů.

Způsob, jakým optické vlákno vede světelný paprsek záleží na tom, jak se mění optické vlastnosti (index lomu) na přechodu mezi vláknem a jeho pláštěm. Pokud se mění skokově a pokud je průměr vlákna dostatečně velký (50-100μm), jde o vlákno schopné vést různé vlny světelných paprsků – tzv. vidy (modes). Jde tedy o mnohovidové vlákno, v tomto případě se stupňovitým indexem lomu. Pokud se na přechodu mezi vláknem a pláštěm index lomu mění plynule, jde o mnohovidové vlákno s gradientovým indexem lomu. Výhodou mnohovidových vláken je relativně nízká cena, jednodušší spojování velká numerická apertura (rozsah úhlů, pod kterými může paprsek dopadat na optické vlákno tak, aby byl vláknem vedený) a možnost buzení luminiscenční diodou.

Nejvyšší přenosové rychlosti (až Gbit/s na vzdálenosti 1km) lze dosáhnou pomocí jednovidových vláken, které přenášejí jediný vid. Schopnost přenášet jediný vid bez odrazů a ohybů se dosahuje buď velmi malým průměrem vlákna (jednotky μm) nebo malým poměrným rozdílem indexů lomu vlákna a pláště. Lze je použít i na dlouhé vzdálenosti (100km bez opakovače), vyžadují buzení laserovou diodou.

Optické vlákna jsou velmi citlivé na mechanické namáhání a ohyby. Jejich ochranu musí zajišťovat svým konstrukčním řešením optický kabel, který kromě optických vláken obsahuje také vhodnou výplň zajišťující potřebnou mechanickou odolnost. Dalšími výhodami, kromě rychlosti, je úplná necitlivost vůči elektromagnetickému rušení, velká bezpečnost proti odposlechu, malá hmotnost kabelů. Problematičtější je spojování vláken.

Optické vlákna jsou atraktivní pro počítačové sítě hlavně pro svou velkou přenosovou rychlost, kterou lze dosahovat s relativně nízkými náklady.

12.4.4 Kódování bitů

Pro přenos bitů na fyzické lince jsou dvě metody[11-1]:

- Přenos v základním pásmu – vysílání bitů přímo na linku, příp. s určitým typem kódování.
- Využití modulace - Pokud chceme využít pro přenos frekvenční pásmo, které neobsahuje základní harmonické přenášeného signálu, je potřeba použít modulaci. Moduluje se nosná frekvence a tento modulovaný signál se vysílá.

12.4.5 Přenos v základním pásmu

Přímé kódování - signál se na kabel vyšle přímo - úroveň log. 0 – nízká úroveň napětí, např. 0V, úroveň log. 1 – vysoká úroveň napětí, např. 10V.

Inverzní kódování – log. 0 – vysoká úroveň napětí, log. 1 – nízká úroveň napětí

Polární kódování – log. 0 a 1 mají oproti referenční úrovni navzájem opačné úrovně.

Přímé, nepřímé a polární kódování se označují jako NRZ (non-return to zero), protože nemají zpětný přechod na nulovou napěťovou úroveň signálu. Posloupnost jednotek udržuje potenciál linky na konstantní úrovni podle toho, jak je kódování zvolené.

1. Kódování NRZ je jednoduché, ale dost citlivé na šum a zkreslení. Pro kompenzaci útlumu linky a zkreslení je možné na straně přijímače určit práh citlivosti, takže např. signál s úrovní menší než 2V se interpretuje jako log. 0 a úrovně vyšší než 5V se interpretují jako log. 1. U NRZ kódování se objevuje ještě jeden problém. Při jednoduchém vysílání bitů na linku není schopný přijímač detekovat začátek a konec jednotlivých bitů. Jinými slovy, nedá se odlišit stav „žádná zpráva“ od posloupnosti několika log. 0. Rovněž se na začátku vysílání neví , pokud je možných více přenosových rychlostí, o kterou rychlost jde. To lze řešit tím, že každá zpráva bude mít hlavičku skládající se ze sekvence log. 0 a 1, z které by se dalo určit správné časování. Stále to však hrozí ztráta synchronizace během přenosu a tedy ztráta dat.

2. Výše uvedené problémy lze řešit RZ (Return to Zero) kódováním, kdy je původní údaj kombinovaný se synchronizačním signálem. Při RZ kódování jsou definovány dvě úrovně napětí s opačnými znaménky, vztáhnuté k nulové referenční hodnotě. Každý bit začíná vysokou nebo nízkou úrovní napětí a uprostřed každého impulsu přechází do nulové úrovně. Tato hrana se používá pro synchronizaci přijímače. Kódování vyžaduje dvojnásobnou šířku pásma oproti NRZ a má komplikovanější elektroniku rozhraní. Výhody jsou zřejmé.

3. Kódování NRZI (Non-return to Zero Inversion) – je charakteristické změnou polarity na začátku bitového intervalu log. 0, přičemž na začátku bitového intervalu log. 1 se polarita nemění.

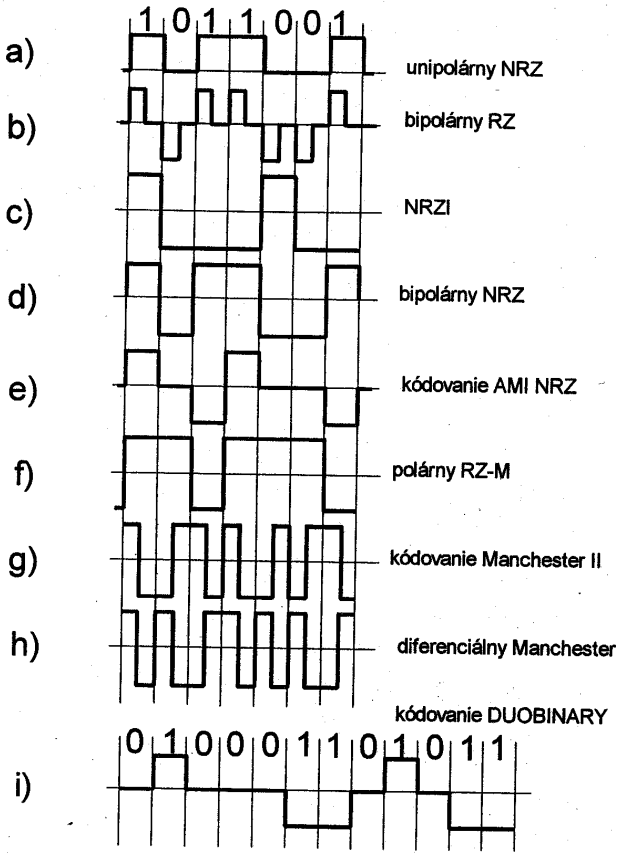
4. AMI NZR (Alternativ Mark Inversion) – má v log. 0 nulovou úroveň a v bitových intervalech log. 1 mění střídavě polaritu.

5. Přímé kódování Manchester – dvojfázové kódování. Při kódování Manchester se každý bit kóduje dvěma napěťovými úrovněmi a přechodem uprostřed každého impulsu (bitového intervalu). Log. 0 je reprezentovaná přechodem z nižší úrovně do vyšší a log. 1 z vyšší do nižší.

6. Diferenciální kódování Manchester – je podobné, ale opačné než předchozí.

Kódování Manchester obsahují synchronizační referenci a proti RZ kódování mají lepší šumovou imunitu. Požadují rovněž dvojnásobnou šířku pásma vzhledem k NRZ. Diferenciální Manchester se používá v aplikacích lokálních počítačových sítí podle doporučení IEEE 802.

Kódování duobinary – používá se v sítích podle doporučení IEEE 802.4. V log. 0 má nulovou úroveň signálu, v intervalu log. 1 mění střídavě úroveň napětí podle polarity předešlého impulsu a podle počtu nul mezi impulsy.



Obr. 12.6: Způsoby kódování dat.

12.4.6 Modulace nosné

Amplitudová modulace.

Frekvenční modulace.

Fázová modulace.

QAM (Quadratic Amplitude Modulation) – kombinuje amplitudovou a fázovou modulaci, používá se v telefonních modemech.

12.5 Základní informace o linkové vrstvě ISO-OSI modelu

Sestavení fungující fyzické vrstvy je pouze prvním krokem při budování spolehlivé komunikace. Druhá vrstva OSI zajišťuje řízení datové linky, což je soubor procedur a protokolů potřebných k tomu, aby se zpráva dostala neporušená do uzlu přijímače. Ve fyzické vrstvě máme jen omezené možnosti jak zabránit poškození signálu šumem a jak poškozené údaje znovu obnovit. Za ověřování správnosti přijatých zodpovídají vrstvy ležící nad fyzickou vrstvou, přičemž nejdůležitější je vrstva linková. V používaných metodách se data posílají podle určitých protokolů a spolu s přídavnými informacemi umožňují verifikaci celistvosti údajů.

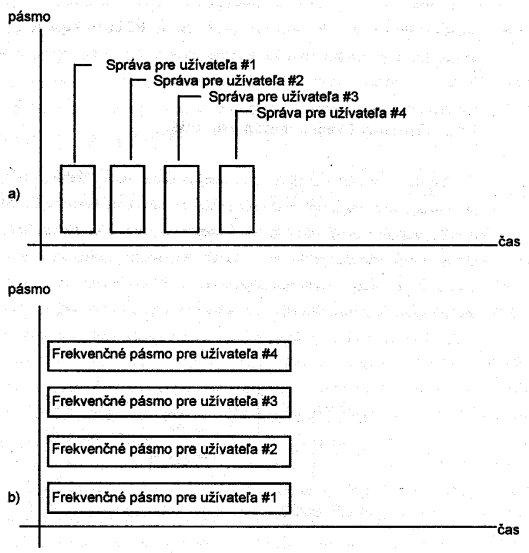
Komunikační protokol - soubor pravidel nutných pro vzájemnou komunikaci dvou programových entit – kdy a jak si data předávat, v jakém tvaru, jaký význam tomu budou přisuzovat, jak reagovat na jednotlivé stavy a situace vzniklé při přenosu.

12.5.1 Rozdělení kapacity kanálu mezi jednotlivé uživatele

Rozdělení fyzického kanálu mezi více uživatelů se označuje jako multiplexování kanálu (viz. obr. 12.7). Multiplexování je pro uživatele transparentní, protože uživatelé se detailně nevěnují způsobu, jakým je kanál organizovaný. Každý uživatel vidí svůj virtuální kanál a částí kapacity původního fyzického kanálu. Multiplexování lze uskutečnit v časové i frekvenční oblasti. Při časovém multiplexu (TDM – Time Division Multiplexing) je kanál rozdělený na periodické časové úseky (sloty) a každý uživatel má přístup pouze na jemu přidělené sloty. Při frekvenčním multiplexu (FDM – Frequency Division Multiplexing) je šířka pásma kanálu rozdělená na užší frekvenční pásma, z nichž každé je přiděleno jednomu virtuálnímu kanálu. Oba typy vyžadují přídavné zpracování dat na obou koncích kanálu. TDM však má oproti FDM dvě výhody:

Celé zpracování je číslicové a není tedy nutné instalovat vysokofrekvenční zařízení.

Je možné řídit přidělování slotů uživatelům, kteří o ně žádají. Pokud uživatel data v daném momentě nevysílá, je možné slot přidělit dalšímu uživateli, který kapacitu potřebuje.



Obr.12.7: Způsoby multiplexování kanálu.

Časový multiplex je obecně účinnější než FDM, protože součet šířek subkanálů se více blíží teoretické horní hranici, tj. celkové šířce pásma.

Za předpokladu, že část uživatelů nebude v daném časovém okamžiku žádat o přístup na kanál, je možné sdílet kanál více uživateli, kteří jsou na něj fyzicky připojeni. Tento způsob se nazývá statistické multiplexování. Kanál je přidělený jen té jednotce, která jej v daném okamžiku požaduje. Pokud však požaduje přístup více jednotek, než kanál zvládne, musí některá z jednotek čekat. Výsledkem je to, že je kanál využívaný více entitami než v případě, kdy byl jeden slot přidělen jedné entitě.

12.5.2 Režimy přenosu

Simplex - vysílač a přijímač je umístěný na opačných koncích kanálu a vysílač vysílá údaje k přijímači.

Pokud by měly jednotky na obou stranách kanálu pracovat jako vysílač i přijímač, musí být kanál multiplexovaný pro komunikaci v obou směrech.

Poloviční-duplex – v daném časovém okamžiku disponuje kanálem jen jedna jednotka, to je ekvivalentní časovému multiplexu. Umožňuje obousměrný provoz, ale ne současně.

Plný-duplex – obě jednotky komunikují po celou dobu v obou směrech, umožňuje obousměrný provoz současně. Plný duplex se realizuje pomocí frekvenčního multiplexu, přičemž každá strana má přidělená vlastní kanál. Pro zprávy jdoucí opačnými směry jsou přidělené jiné frekvenční pásma.

Multiplexování není jedinou metodou rozdělení kapacity kanálu. Obecnějším problémem je přístup na médium, který je zvlášť důležitý v lokálních sítích, kde kanál sdílejí až stovky jednotek, které si vyměňují data oběma směry.

12.5.3 Detekce a korekce chyb

Samotná fyzická linka sice umožňuje přenos dat mezi různými místy, ale nemůže zaručit, že data došly v přesně stejném tvaru, v jakém byly vyslané. Šum na lince může zprávy poškodit.

Pro zabezpečení dat proti poruchám zle využít dva způsoby korekce chyb a oba vyžadují aktivní účast přijímače.

Detekce chyb (error detection) – k původní zprávě jsou přidané určité informace, na základě kterých přijímač zjistí, zda při komunikaci nastaly poruchy.

Oprava chyb (error correction) – realizuje se na základě toho, že k původní informaci se přidá dostatek informací, aby přijímač mohl na základě přijatých údajů poškozenou zprávu rekonstruovat.

V praxi se poruchy vyskytují spíše v hlucích než jednotlivě. Je pravděpodobnější, že v důsledku poruchy změní svojí hodnotu několik sousedních bitů a ne pouze jeden. Zdroje diskrétního šumu často generují impulzy široké několik ms, což při používaných přenosových rychlostech znamená ovlivnění řádově desítek bitů.

Na zjišťování poruch při přenosu byly definované detekční a kontrolní metody.

- Metoda kontrolního součtu (checksum, bcc) – vysílací strana spočítá kontrolní součet, který má délku až několik desítek bitů a přidá jej k bloku dat. Přijímací strana rovněž spočítá součet podle stejného algoritmu a pokud se součty shodují, zprávu akceptuje. Pokud se neshodují přijímač požádá o opakované zaslání bloku dat.
- Cyklické kódy – název CRC (Cyclic Redundancy Check) nebo také kontrola rámce FCS (Frame Check Sequence), je odvozený ze skutečnosti, že výpočet CRC lze realizovat posouváním bitů přicházejícího bloku dat přes posuvný registr. Registr musí uskutečňovat operaci EX-OR s pevně nastavenou maskou. Hodnota masky je určená tzv. generujícím polynomem, na kterém jsou příjemce a odesílatel předem dohodnuti. Původní řetězec přenášených údajů je takto vlastně dělená jedno nebo dvou bytovým binárním číslem, které je vyjádřené mnohočlenem: $x^n + x^{n+1} + \ldots + x^2 + x^1 + 1$. Generující polynom je obvykle o jeden bit delší, než výsledný CRC polynom. Např. standardní CRC (CCITTV.41 a CCITTX.25) polynom je: $x^{16} + x^{12} + x^5 + 1$, což je 1000100000010001. Zabezpečovací část je doplňkem podílu zbytku původního bloku dat generujícím polynomem. Aby podíl dával smysl, musí být původní blok dat delší než generující polynom. Pokud se zabezpečovací část připojí k původním údajům, je výsledná posloupnost bitů celočíselným násobkem generujícího polynomu. Pokud se celá posloupnost znovu vydělí, výsledek (asi zbytek) musí být roven nule. Pokud tomu tak není, je potřebné nové vyslání zprávy. Účinnost zabezpečovacího algoritmu z hlediska opravy chyb klesá s délkou zprávy. Pomocí takovéhoto zabezpečení je možné detekovat všechny shluky chyb kratší než CRC a více než 99,99% delších shluků.

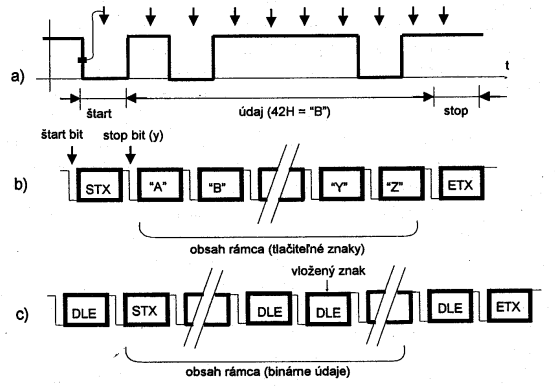
Samoopravné kódy – s původními údaji se zde přenáší i nadbytečná informace. Přijímač k tomu, aby sestavil původní blok údajů, používá celou přijatou informaci. Potřebné redundantní údaje tvoří obyčejně 10-20% délky původního bloku. Obecně je ekonomičtější použít jednodušší detekční metodu a požadovat opakovaný přenos údajů. Samoopravné procedury jsou důležité u jednosměrné komunikace, nebo tam kde není uskutečnitelné plně duplexní vzájemné potvrzování (hanshaking) (družicová služba, rádiové sítě).

12.5.4 Synchronizace přenosu v linkové vrstvě

Fyzická vrstva ISO/OSI modelu zajišťuje přenos jednotlivých bitů mezi dvěma uzlovými počítači, mezi kterými existuje přímé spojení. Linková vrstva pak využívá tyto prostředky pro přesnost větších bloků údajů (rámce) a přenos těchto rámců potom sama poskytuje bezprostřední vyšší vrstvě – síťové. Fyzická vrstva nijak nerozlišuje jednotlivé bity, které přenáší. Proto je na linkové vrstvě, aby zajistila jejich správnou reprezentaci.

12.5.5 Asynchronní sériový přenos

Při tomto přenosu mohou být jednotlivé znaky přenášené s libovolnými časovými odstupy mezi sebou (viz. obr. 12.8). Příjemce však nemůže dopředu vědět, kde začíná další znak a proto musí být schopný jeho příchod podle vhodného příznaku rozpoznat. Tím příznakem je tzv. startovací bit. Tento bit rovněž umožní nastavit měřítko času (časovou základnu). To je důležité pro to, aby příjemce správně určil časové okamžiky, kdy má vyhodnocovat stav jednotlivých údajových bitů, které za start bitem následují.



Obr.12.8: Asynchronní přenos.

Za vlastními datovými bity může být tzv. paritní bit a nakonec stop bit, jehož délka odpovídá jeden, jeden a půl nebo dva údajové bity. Stop bit nenese informaci, jeho úkolem je zajistit minimální odstup mezi jednotlivými znaky. Hodnota paritního bitu záleží na uvažovaných parametrech komunikace. Při sudé paritě (even) musí být sudý počet jedniček včetně paritního bitu, při liché paritě musí být počet jedniček lichý. Pokud se paritní bit nevyužívá, je jeho hodnota nepodstatná. Pokud je detekovaná chybná parita, přijímač to oznámí vyšší vrstvě (která např. zajistí nové vyslání poškozeného znaku). Nejčastěji se používá kombinace 8N1(8bitů, žádná parita, 1 stopbit). Vysílání bez parity se nejčastěji používá při vysílání 7 b. znaků.

V případě asynchronního přenosu se tedy přenášejí data členěné na znaky (stejně dlouhé skupiny bitů). Ty jsou pro přenos „obalené“ tzv. start a stop bity, které umožňují správně detekovat začátek a konec znaku. Pokud potřebujeme posílat data tvořené posloupnostmi běžných znaků (ASCII), je nejjednodušší metodou vložit celý blok znaků mezi dvojici speciálních netisknutelných znaků – STX (Start of Text) a ETX (End of Text), které se označují jako řídicí znaky přenosu. Tím se dostane potřebná synchronizace na úroveň rámců (frame synchronization), protože STX a ETX umožní příjemci rozpoznat začátek a konec zprávy.

Výše uvedený způsob však nelze použít, pokud se přenáší všeobecné binární údaje. Ty je sice možné také rozdělit na skupiny bitů, ale může se stát, že se v přenášených datech objeví i některý z řídicích znaků. Pak je nutné zajistit tzv. transparentnost údajů, tedy umožnit, aby mezi přenášenými znaky mohly být i řídicí znaky a aby byly reprezentované jako data. Používá se k tomu technika vkládání znaků, kdy je před řídicí znaky STX a ETX vložený ještě jiný řídicí znak DLE (Data Link Escape – změna významu následujícího znaku). Ten se však může také vyskytovat mezi přenášenými znaky a proto se každý jeho výskyt zdvojuje. Pokud příjemce přijme znak DLE, rozhoduje se podle dalšího

znaku – je-li to zase DLE, následující znak považuje ze data, pokud je to STX nebo ETX, považuje je ze řídicí znaky.

Jednou z nevýhod asynchronního přenosu je snížení efektivních přenosových rychlostí v důsledku vkládání řídicích znaků. Pro rychlejší přenosy se používá synchronní přenos, který je však náročnější na realizaci.

12.5.6 Synchronní sériový přenos

Při synchronním přenosu jsou obvykle přenášené celé bloky znaků. Datové bity jednotlivých znaků přitom následují těsně po sobě bez jakýchkoliv časových odstupů a nejsou do nich vkládané žádné start nebo stop bity. Začátek bloku je indikovaný jedním nebo několika speciálními synchronizačními znaky (SYN), jejichž hlavním smyslem je zajistit potřebnou časovou synchronizaci, tj. pomoci příjemci přesně stanovit časové okamžiky, ve kterých má vyhodnocovat jednotlivé datové bity. Blok je opět zakončený synchronizačními znaky, které mohou (ale nemusí) být nepřetržitě vysílané až do začátku dalšího bloku dat.

Jinými slovy, synchronní přenos si lze představit jako spojitý proud znaků, ve kterém musí příjemce správně rozpoznávat jednotlivé znaky – to představuje synchronizaci na úrovni znaků.

Synchronizace na úrovni rámců se při synchronní přenosu realizuje podobně jako při asynchronním – pomocí řídicích znaků. Pak jde o znakově orientovaný přenos.

Rozdělení protokolů podle způsobu zajištění kódové transparentnosti, tzn. jak se zajistí, aby mezi přenášenými daty mohly být i řídicí znaky bez toho, aby užitečné údaje byly považovány za řídicí:

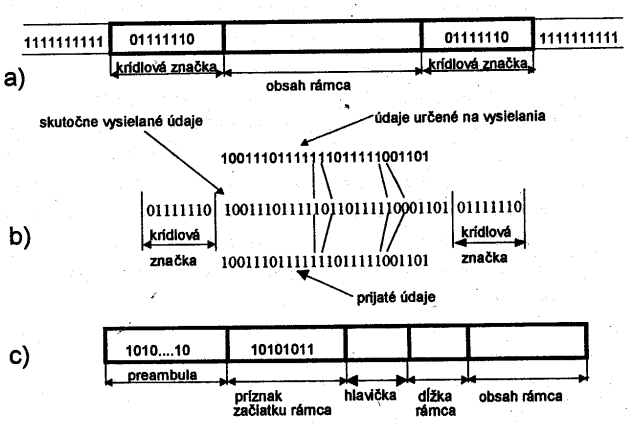
V znakově orientovaných protokolech se kódová transparentnost zabezpečuje pomocí znaků DLE.

V blokově orientovaných protokolech se transparence zabezpečí pomocí vložení informačního pole do rámce, které specifikuje délku datového pole, resp. počet následujících bytů.

Vkládání celých znaků do přenášených dat a jejich potřebné zdvojování však opět přináší snížení efektivní rychlosti přenosu. Proto se dnes stále více uplatňuje bitově orientovaný přenos. Je založený na myšlence indikovat začátek a konec rámců ne řídicím znakem, ale skupinou bitů. Přenášená data jsou vyhodnocována bit za bitem, dokud není nalezena hledaná skupina bitů znamenající začátek nebo konec rámce.

Jednou z možností pro bitově orientovaný přenos je použití stejné skupiny bitů pro začátek a konec rámce (rámcová synchronizace) – této skupině bitů se říká flag (např. 01111110). Tento flag se však nesmí vyskytovat v přenášených datech, což lze zajistit vkládáním bitů, při kterém je za každých 5 po sobě jdoucích bitů vložena log.0.

Další možností rámcové synchronizace je uvedení celého rámce příznakem začátku rámce (po preambuli neboli synchronizačním poli). Za příznakem následuje hlavička předem daného formátu a údaj o délce rámce. Tato varianta se uplatňuje hlavně v lokálních sítích. Preambule zabezpečuje dostatečný počet bitových změn pro bitovou synchronizaci.



Obr. 12.9: Synchronní přenos.

12.5.7 Metody potvrzování

Služby, které linková vrstva poskytuje síťové, mohou mít charakter spolehlivých nebo nespolehlivých služeb. Pro realizaci spolehlivých služeb musí mít potom linková vrstva k dispozici mechanizmy pro zajištění přijetí skutečně všech vyslaných rámců bez chyb.

Simplexní spoje neumožňují zpětnou vazbu mezi vysílajícím a příjemcem, příjemce si tedy nemá možnost vyžádat nové vyslání špatně přijatých rámců. Příjemce si musí s případnými chybami poradit sám, např pomocí použití samoopravných kódů. Podobná situace nastává u spojů, které sice nejsou simplexní, ale pracují s dlouhými dobami přenosu, takže se linkové vrstvě nevyplatí čekat na zpětnou vazbu od příjemce dat.

V případě poloduplexních a duplexních spojů je možné vystačit se zabezpečením dat pomocí detekčních kódů. Nejúčinnější jsou cyklické kódy, které je možné použít k zabezpečení rámce jako celku. Při odesílání rámce se přidá krátký zabezpečovací údaj (typicky 16b), podle kterého je příjemce schopen s velkou pravděpodobností určit, zda přijal rámec bez chyby. Pokud ne, může využít zpětnou vazbu a vyžádat si nové vyslání celého chybně přijatého rámce. Tento algoritmus se obvykle implementuje ve formě potvrzování, neboli potvrzovací zpětné vazby, která předpokládá, že příjemce zkontroluje bezchybnost každého rámce a o výsledku informuje vysílajícího.

Potvrzování se realizuje řadou způsobů a je možné je rozdělit na dvě skupiny:

- Jednotlivé potvrzování – vysílající odešle rámec a čeká na reakci příjemce. Další rámec vyšle až po potvrzení úspěšného přijetí rámce. Jinak jej vyšle znovu(když je signalizováno neúspěšné přijetí nebo pokud do časového limitu nedostane odpověď). Existují zde metody s čistě pozitivním potvrzováním, s čistě negativním potvrzováním nebo jejich kombinace. Nevýhodou jednotlivého potvrzování je nutnost čekat na reakci protistrany.
- Kontinuální potvrzování – používá se u delších dob přenosu. Vysílající vysílá nové rámce bez toho, že by dostal potvrzení o úspěšném přijetí předešlých rámců. Potvrzení dostává se zpožděním a reaguje na ně až v okamžiku, kdy je skutečně dostane.

12.5.8 Řízení toku dat

Další úlohou linkové vrstvy je zajistit, aby vysílající nezahltil příjemce daty. Linková vrstva se tedy musí zabývat také řízením toku dat a tedy musí zajistit, aby vysílající vysílal skutečně jen tehdy, když přijímající je schopen přijímat.

Nejjednodušší je dočasně pozastavovat vysílání celých rámců. U jednotlivého potvrzování stačí, aby příjemce nepotvrdil poslední přijatý rámec.

V případě kontinuálního potvrzování může vysílající vysílat dopředu jen určitý maximální počet rámců.Vzniká tím okénko odeslaných a zatím nepotvrzených rámců, které prostřednictvím svých

potvrzování posouvá příjemce (metoda sliding window). Díky tomu má také možnost podle svých potřeb dočasně pozastavit vysílání.

12.5.9 Řízení přístupu na médium

Pro činnost linkové vrstvy je velice důležitý i konkrétné způsob propojení jednotlivých uzlů, mezi kterými má přenos rámců realizovat:

Dvoubodové spojení (point-to-point)– mezi uzly existuje alespoň během přenosu přímý přenosový kanál, protokol linkové vrstvy pak zajišťuje přímou komunikaci koncových účastníků.

Vícebodové spojení (multipoint connection) – propojuje vzájemně více uzlů, používá se hlavně na kratší vzdálenosti. Umožňují přenos mezi kterýmikoliv uzly na síti. Dokonce i přenos z jednoho uzlu do více uzlů současně (broadcast channel).

Mnohobodové spojení je sdílený prostředek, kde v roli vysílajícího může být pouze jeden uzel. Pokud dojde k situaci, že se o získání tohoto sdíleného prostředku pokusí více uzlů současně, musí existovat mechanizmus, který umožní vybrat jednoho žadatele. Opět existuje několik možností.

Metody přístupu na médium lze rozdělit na dvě základní skupiny:

Deterministický přístup ke sdílenému médiu

Centralizované řízení.

Distribuované řízení.

Náhodný přístup ke sdílenému médiu

Přístupové metody zajišťují korektní přístup ke sdílenému vícebodovému spojení a musí být implementovány nad fyzickou vrstvu, protože využívají její služby. Linková vrstva, zajišťující přenos rámců však musí mít potřebný přístup k médiu. Přístupové metody by tedy měly být implementovány mezi fyzickou a linkovou vrstvu ISO/OSI modelu. Situace se vyřešila tak, že linková vrstva se rozděluje na dvě podvrstvy:

Nižší – MAC (Media Access Control) – podvrstva řízeného přístupu k médiu, je v ní implementovaná příslušná přístupová metoda.

Vyšší – LLC (Link Layer Control) – podvrstva řízení logického spojení, která zajišťuje úlohy linkové vrstvy (kontrola chyb, adresování, řízení toku dat apod.).

Deterministický přístup na médium – centrální řízení

Metoda centrálního arbitra – sám rozhoduje o využití sdíleného prostředku – jednotlivým žadatelům přiděluje právo vysílat. Arbitr se rozhoduje na základě explicitních žádostí jednotlivý žadatelů. Je tedy nutné vyhradit část přenosové kapacity pro tyto žádosti nebo pro ně vytvořit vlastní spojení.

Metoda výzvy – pooling – v praxi se používá častěji. Centrální arbitr se postupně obrací na potenciální žadatele s dotazem, zda chtějí vysílat.

Centrálním arbitrem bývá jedna ze stanic, která má postavení řídicí stanice (master, supervizory), zatímco ostatní stanice vystupují v úloze podřízených stanic (slave). Řídicí stanice poskytuje jednotlivým stanicím právo vysílat na základě jejich kladné reakce na výzvu (pool). Stanice, která v daném okamžiku dostává právo vysílat se dostává do postavení hlavní stanice (primary) a sama si volí, komu chce data vyslat. Prostřednictvím jednoduché výzvy provede výběr (selection) jedné nebo několika dalších stanic, které budou přijímat její data. Ty pak vystupují jako vedlejší stanice (secondary).

Tento systém má jedno úskalí – v případě výpadku arbitra se celá síť stává nepoužitelnou.

Deterministický přístup na médium – distribuované řízení

Existují však i jiné metody, které nevyžadují existenci centrálního arbitra. Tyto metody jsou založené na myšlence, že žadatelé se mezi sebou dokážou dohodnout, kdo bude vysílat (decentralizovaný způsob řízení mnohobodového spojení). Pravidla pro přístup na síť definuje tzv. přístupová metoda (access method).

Tento způsob řízení je založen na různých přístupových metodách a je charakteristický hlavně pro lokální sítě.

Metoda logického kruhu – stanice sdílející přenosový kanál jsou označeny adresami a tyto adresy tvoří posloupnost. Každá ze stanic zná svou vlastní adresu a adresu stanice, která smí vysílat po ní. Jedna, ze stanic je vždy aktivní, v tomto stavu smí odvysílat datový paket, nebo předat řízení následující stanici speciálním paketem - pověřením (označovaným jako Token - pešek). Metoda je podle předávání pověření mezi stanicemi na sběrnici označována jako Token-Passing Bus nebo zkráceně Token Bus. Určitým problémem metody je její startování a změna posloupnosti stanic pro stanice, které během provozu sítě z logického kruhu odstupují nebo se do něj naopak chtějí zapojit. Metody pro modifikaci posloupnosti stanic v těchto případech jsou označovány jako metody rekonfigurace, příklady rekonfigurace si uvedeme pro síť ARCNet a pro sítě podle doporučení IEEE 802.4.

12.5.10 Náhodný přístup na médium

Náhodný přístup ke sdílenému přenosovému kanálu můžeme považovat za nejjednodušší techniku přístupu a za protipól deterministických metod, které si popíšeme později. Jednotlivé stanice podřizují přístup na kanál pouze svému odhadu nebo pozorování.

Příkladem jsou metody Aloha a zejména metody CSMA (metoda CSMA/CD je popsána v kapitole 14.2).

Shrnutí pojmů

Komunikační systémy používané v průmyslové komunikaci lze rozdělit na několik úrovní - **Actuator/Senzor Level**, **Field Level** a **Cell Level**.

Centralizované systémy jsou založeny na jedné výkonné řídicí jednotce, které má k dispozici všechny technologické signály a řídí kompletní technologii. Naopak **distribuované systémy** jsou sestaveny z několika menších řídicích jednotek, kde každá řídí jen část technologie a vzájemně jsou propojeny komunikační sběrnicí. Pro standardizaci propojování otevřených systémů vznikl **ISO-OSI model**. Má vrstvovou strukturu a každá vrstva zajišťuje určité funkce v procesu komunikace.

Fyzická vrstva zajišťuje přenos jednotlivých bitů mezi příjemcem a odesílatelem prostřednictvím fyzické přenosové cesty, kterou tato vrstva bezprostředně ovládá. Fyzická vrstva přenášené bity žádným způsobem neinterpretuje. Obsahuje **elektrické**, **mechanické** a **optické** rozhraní spolu s potřebnými softwarovými ovladači portů. Fyzická vrstva je ve skutečnosti jen reálným spojením mezi dvěma komunikujícími místy – uzly, nodes.

Linková vrstva se označuje také jako spojová vrstva. Tato vrstva má za úkol zajistit přenos celých bloků údajů, označovaných jako rámce, frames, jen mezi dvěma uzly. Tato vrstva zajišťuje **detekci/korekci chyb přenosu**, **potvrzování**, **řízení toku dat**, **přístup ke komunikačnímu médiu**.

Kontrolní otázky

1. Popište centrální a distribuované řídicí systémy.
2. Jaké jsou výhody a nevýhody distribuovaných řídicích systémů?
3. Co je to ISO-OSI model?

4. Popište vrstvy ISO-OSI modelu.
5. Jakým způsobem probíhá přenos dat mezi dvěma peer entitami umístěných na různých uzlech sítě?
6. Jaké jsou základní kvalitativní parametry přenosových médií?
7. Jaké typy fyzických médií se v současných komunikačních systémech používají?
8. Jaké jsou metody pro přenos bitů na fyzické lince?
9. Jaké jsou základní způsoby kódování bitů při přenosu v základním pásmu?
10. Jaké jsou způsoby modulace nosné?
11. Co je to komunikační protokol?
12. Jak je možné rozdělit komunikační kanál mezi několik uživatelů?
13. Jaké existují režimy přenosu dat mezi vysílačem a přijímačem?
14. Popište metody detekce a korekce chyb.
15. Jak probíhá asynchronní přenos bloku binárních dat?
16. Jaké existují metody přístupu na komunikační médium?

Další zdroje

12-1. Kováč F.: Distribuované riadiace systémy. Vydavatelstvo STU, Bratislava 1998.

12-2. Janeček J.: Distribuované systémy. Vydavatelství ČVUT, Praha 1997.

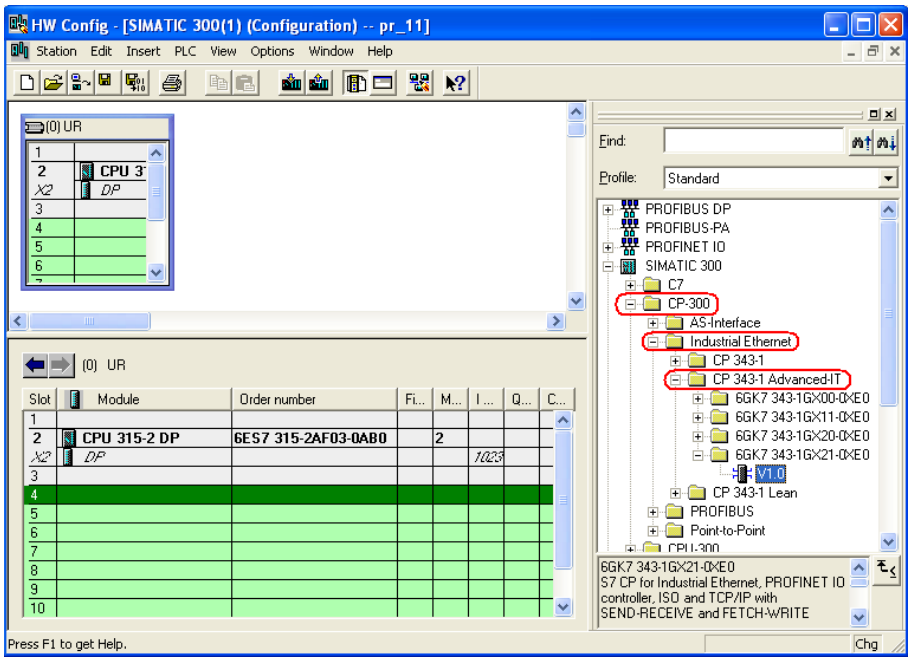
12-3. Janeček J., Bílý M.: Lokální sítě. Vydavatelství ČVUT, Praha 1998.

12-4. Časopis Automatizace 7/98.

Řešená úloha 12.1.

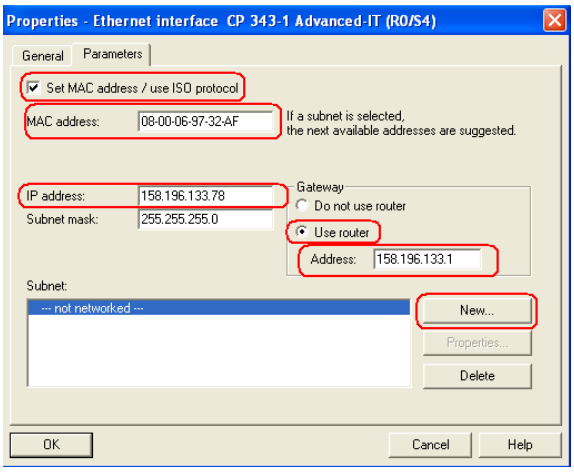
Vytvořte komunikaci mezi PC a PLC SIEMENS S7-300. Jako programovatelný automat je použit CPU 315-2AF03-0AB0 a jako ethernetový modul použijte komunikační procesor CP 343-1GX21-0XE0.

V hardwarové konfiguraci vložte nejprve lištu RACK a odpovídající CPU 315-2DP. Komunikační procesor se vkládá do slotu č.4. Komunikační procesor naleznete ve složce *CP300→Industrial Ethernet→ CP 343-1 Advanced-IT →6GK7 343-1GX21-0XE0* (obr. 12.10).

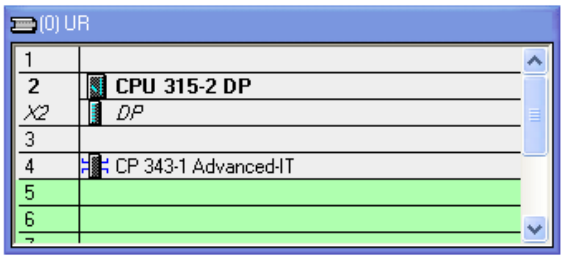


Obr. 12.10: Vložení komunikačního procesoru pro Ethernet.

Bezprostředně po vložení modulu do slotu 4, budete vyzvání pro vložení MAC adresy a IP adresy. Zkontrolujte, že je zatrhnuta volba *Set MAC address /use ISO protocol*. MAC adresa je specifikována již při výrobě může být změněna, ale nastavení MAC adresy neměňte. Do editovacího okénka vepište správnou MAC adresu např. *08-00-06-97-32-AF*. Dále nastavte IP adresu, která je tomuto komunikačnímu modulu přidělena např. *158.196.133.78* a musíte také aktivovat router. IP adresa routeru je *158.196.133.1*. Kliknutím na tlačítko New se vytvoří síť Ethernet v projektu (obr 12.11). Kliknutím na tlačítko OK se uzavře průvodce a komunikační procesor je nakonfigurován.

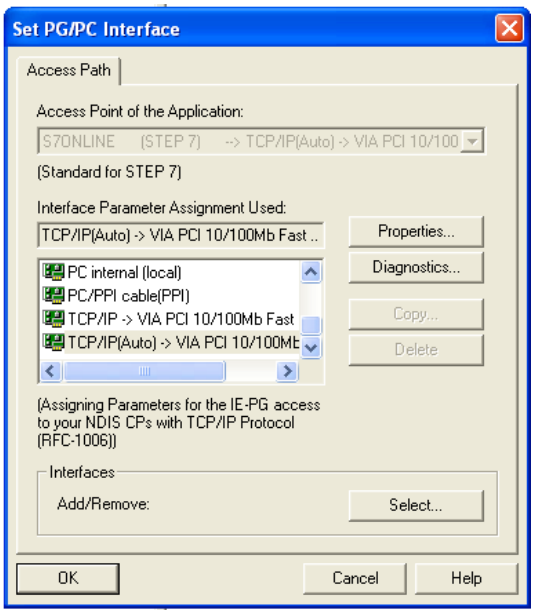


Obr 12.11: Nastavení CP 343-1.



Obr.12.12: Hardwarová konfigurace.

Pokud ještě PLC nebylo použito s tímto komunikačním modulem, nebude PLC ani odpovídající komunikační procesor znát IP adresu a když v menu zvolíte nastavení komunikace na Ethernet, jak ukazuje obr. 12.13, nebudete schopni zapsat nastavení do PLC. Musíte nastavit komunikaci přes MPI kabel a poté provést download hardwarové konfigurace do PLC. Po úspěšném downloadu již můžete si v prostředí STEP 7 nastavit komunikaci přes Ethernet.



Obr. 12.13: Nastavení komunikace přes Ethernet ve STEP7.

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 12\pr_12_1.zip

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 1\ anim2.avi.

DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 2\ anim3.avi.

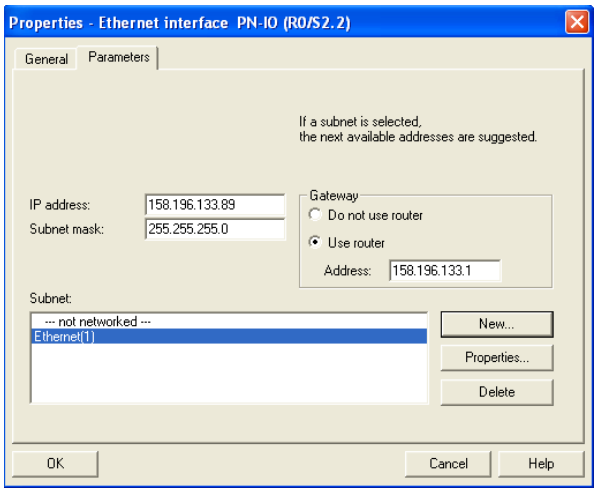
Řešená úloha 12.2.

Nastavte programovatelný automat CPU 315PN/DP pro komunikaci přes Ethernet.

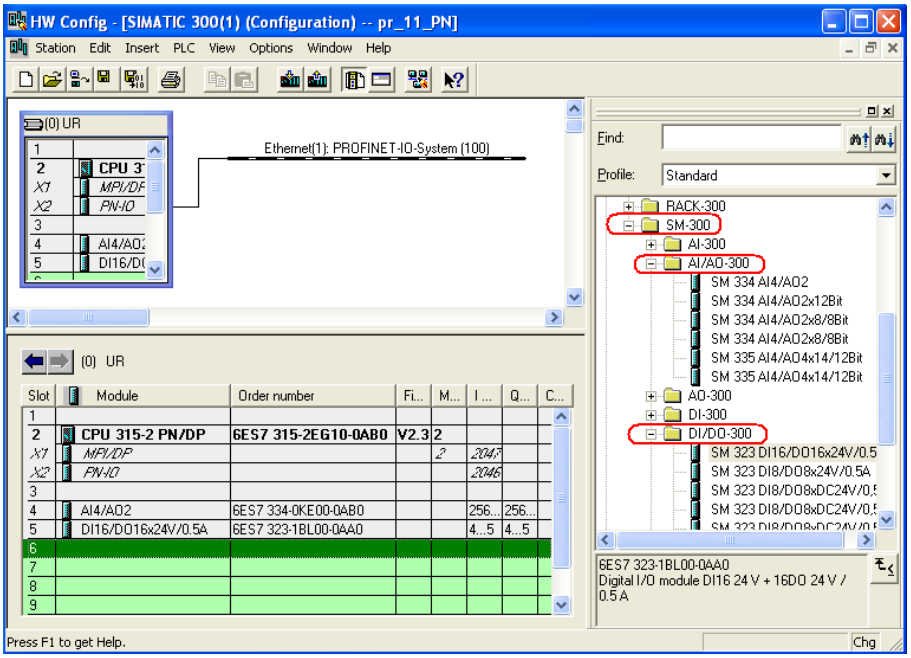
Jak bylo v přednášce číslo 2 uvedeno, vyrábí se také programovatelné automaty, které mají již např. integrováno toto komunikační rozhraní do základního modulu. Níže je uvedeno nastavení pro programovatelný automat CPU 315PN/DP.

Po vložení CPU do slotu 2 budete vyzváni k doplnění údajů o IP adrese. Jako IP adresa může být např. použita adresa *158.196.133.89*. Opět musí být také aktivován routek a vložena IP adresa

158.196.133.1 (obr. 12.14). Za hlavní jednotku do dalších slotů mohou být opět řazeny karty DI/DO, AI/AO apod. (obr. 12.15).



Obr.12.14: Nastavení rozhraní pro Ethernet.



Obr. 12.15. Hardwarová konfigurace.

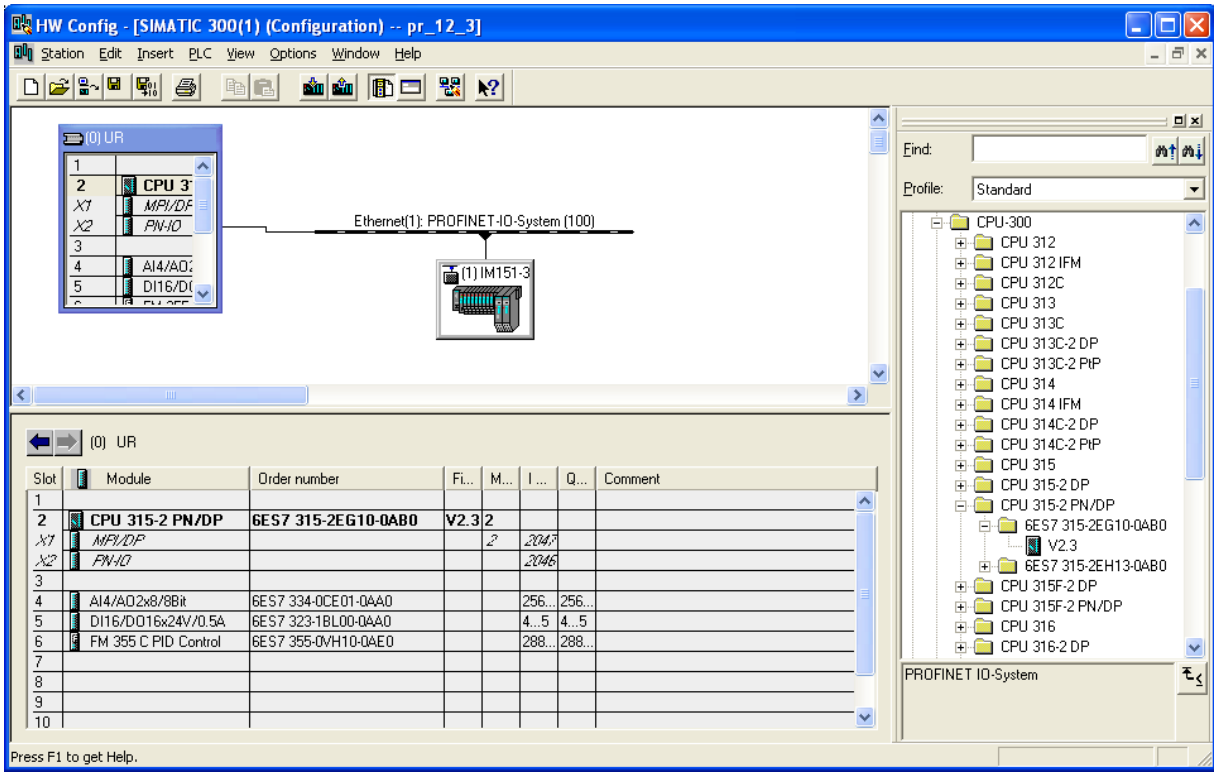
Opět pokud je tento PLC poprvé použit s touto IP adresou je nutné nejprve nastavit tuto konfiguraci přes MPI kabel programovatelnému automatu a až poté můžete využívat komunikace přes Ethernet.

DVD-ROM

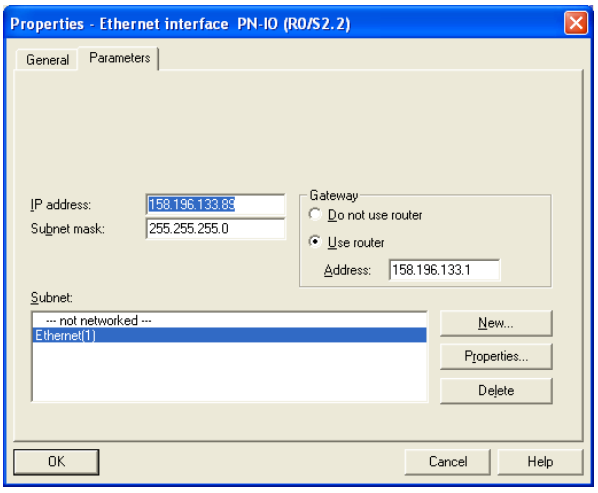
Řešený příklad naleznete na DVD: cvičení\cvičení 12\pr_12_2.zip

Řešená úloha 12.3.

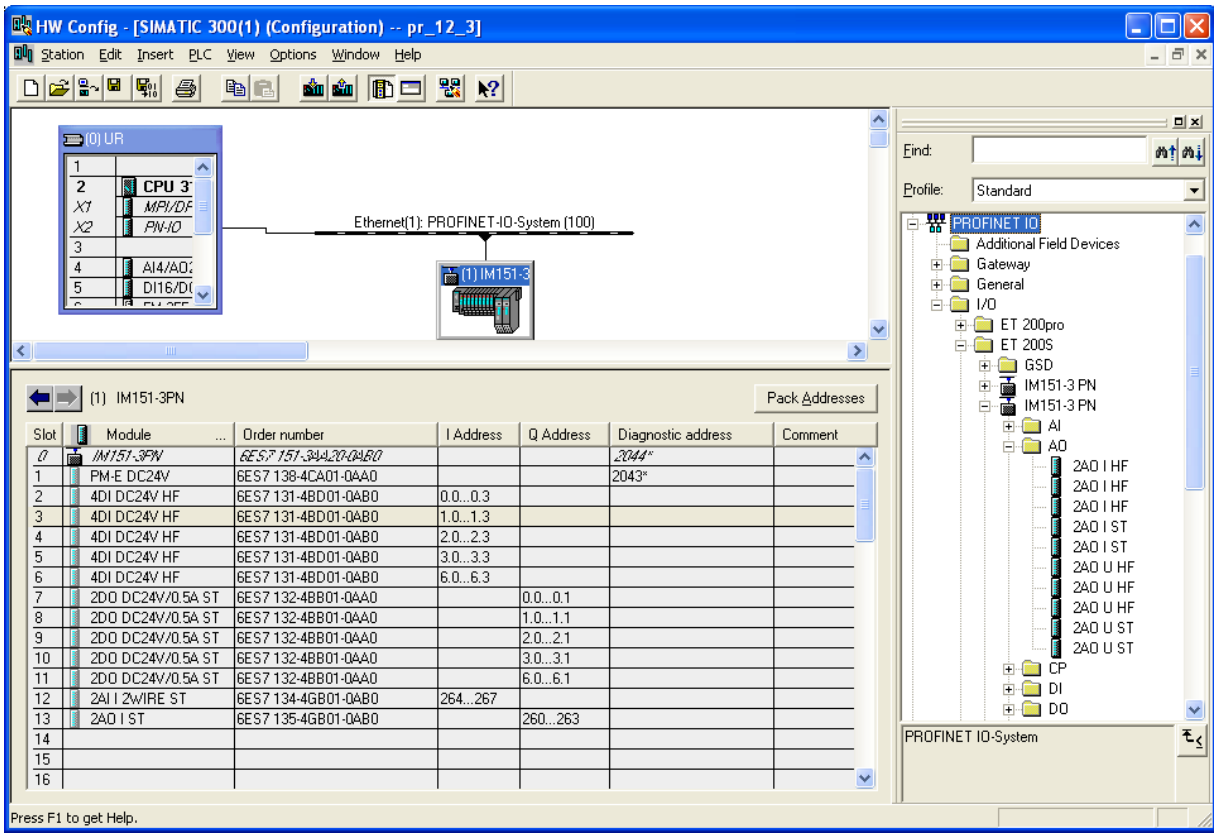
Nastavte komunikaci mezi PLC S7 315 PN/DP a modulem ET200S, které mezi sebou komunikují přes PROFINET.



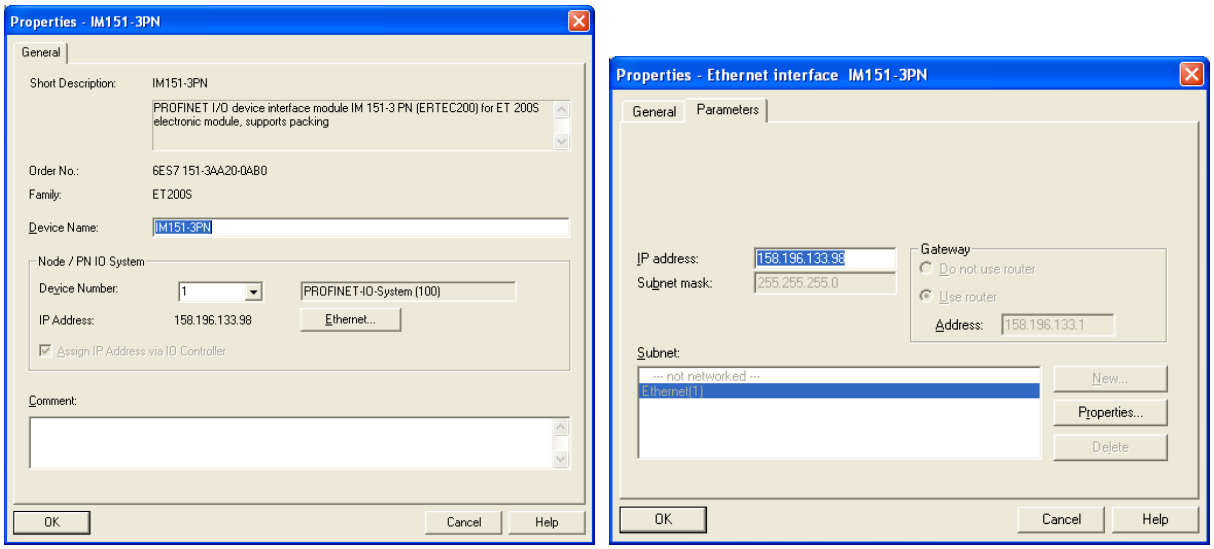
Obr. 12.16: HW konfigurace základního racku s CPU 315PN/DP.



Obr. 12.17: Nastavení IP adresy pro CPU 315PN/DP.



Obr. 12.18: Vložení jednotlivých modulů do ET200S.



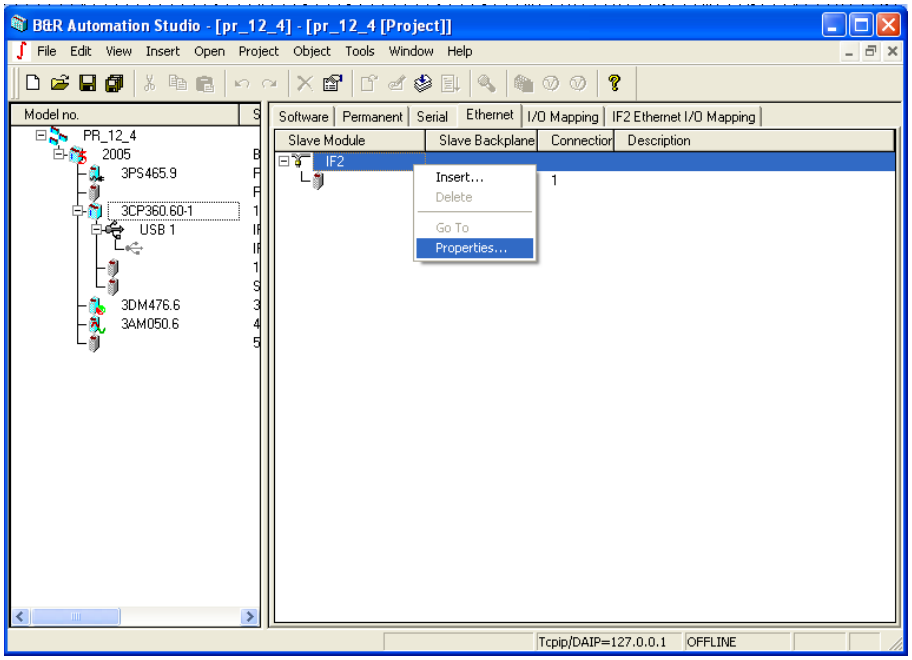
Obr. 12.19: Nastavení IP adresy pro ET200S.

DVD-ROM

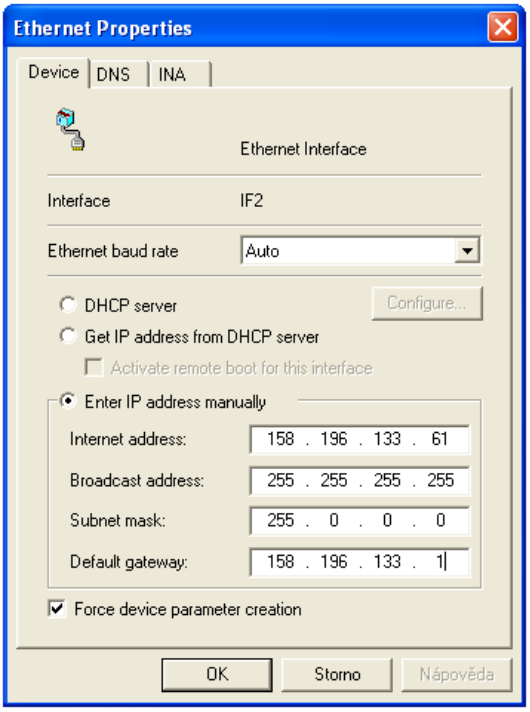
Řešený příklad naleznete na DVD: cvičení\cvičení 12\pr_12_3.zip

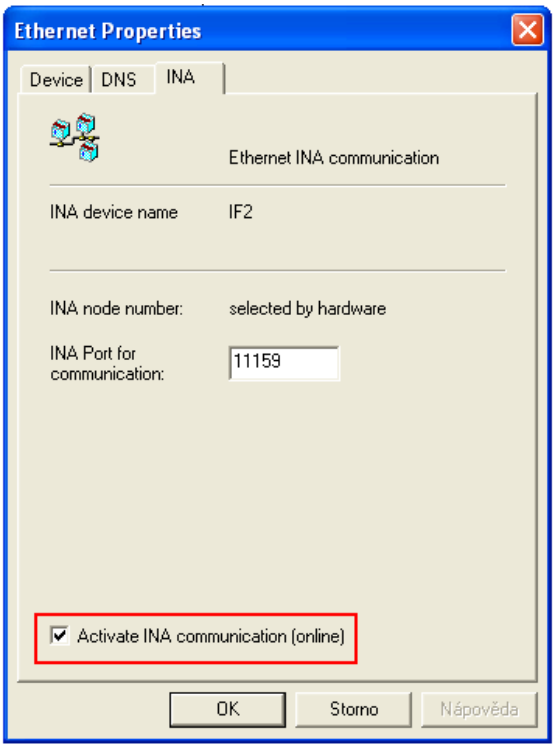
Řešená úloha 12.4.

Nastavte programovatelný automat Bernecker Rainer CP 360 pro komunikaci přes Ethernet.



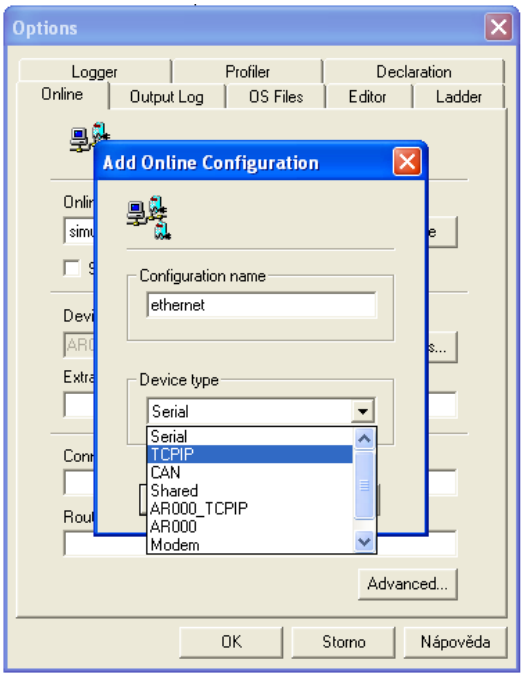
Obr. 12.20: Otevření okna Ethernet Properties pro zadání IP adresy PLC.





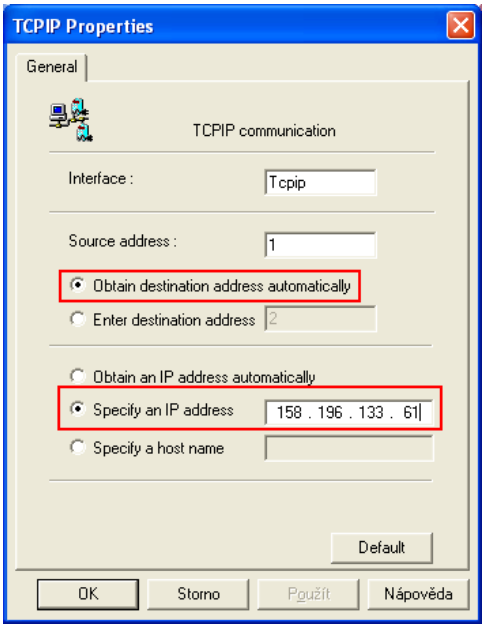
Obr. 12.21: Vložení IP adresy v záložce Device. Obr. 12.22: Aktivace INA komunikace.

Spojení s PLC Tools→Options. Pokud již není vytvořeno spojení přes ethernet, kliknutím na tlačítko Add se přidá nové spojení obr. 12.23.



Obr. 12.23: Přidání nového spojení přes Ethernet.

Kliknutím na tlačítko OK se přidá nové spojení. Kliknutím na tlačítko Properties v okně Properties se zapíše IP adresa automatu obr. 12.24.



Obr. 12.24: Nastavení IP adresy automatu pro spojení s PLC.

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 12\pr_12_4.zip

13. PRŮMYSLOVÉ SBĚRNICE ASI, PROFIBUS A JEJICH POUŽITÍ V ŘÍZENÍ

Čas ke studiu: 2 hodiny

Cíl:

Kapitola se věnuje dvěma velice často používaným komunikačním systémům – **AS Interface** a **Profibus**.

U sběrnice **AS Interface** jsou zde popsány základní parametry, fyzická a linková vrstva sběrnice a je popsáno použití tohoto komunikačního systému.

U sběrnice **Profibus** jsou opět popsány základní parametry, fyzická a linková vrstva sběrnice. Dále jsou uvedeny bližší informace k verzím **Profibus DP** a **Profibus PA**, včetně typických jejich architektur a použití.

Výklad

13.1 ASI (Actuator Senzor Interface)

Standard ASI (Actuator Sensor Interface) vznikl počátkem devadesátých let. V současnosti standard udržován konsorciem výrobců, je však přihlášen ke standardizaci organizací IEC. Paralelně existuje (od roku 1991) i organizace uživatelů. Pro další rozvoj standardu je důležitá jeho podpora firmou Siemens, která má také nejširší nabídku produktů s tímto rozhraním.

Charakteristické rysy lze shrnout do několika následujících bodů:

- Nestíněná dvoudrátová sběrnice s libovolnou topologií.
- Přenos dat a napájení po jediném vedení.
- Délku vedení maximálně 100 m, při použití opakovačů 300 m.
- Maximálně 31 účastnických stanic.
- Maximálně 124 senzorů a 124 akčních členů.
- Řízení komunikace na principu Master – Slave.
- Kódování dat kódem Manchester.
- Zabezpečení telegramu paritou.
- Vysoká rychlost komunikace (cyklus sběrnice kratší než 5 ms).
- Velmi jednoduchá instalace.

Cílem standardu je podpora binárních akčních členů a senzorů na nejnižších úrovních procesní automatizace. Jde o sběrnici s Master-Slave řízením, současně lze připojit až 31 účastnických jednotek. Důležitým rysem standardu je možnost jejich napájení po sběrnici, která tak slouží pro

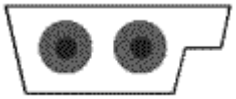
napájení i pro přenos dat. Velkou výhodou je také prakticky libovolná topologie a neexistence terminátorů na koncích vedení.

Fyzická vrstva sběrnice

Vzhledem k využití jediného vedení pro přenos napájení i datových signálů je definice fyzické vrstvy zcela specifická. Nominální hodnota napájecího napětí je 24 V. Součástí napájecího zdroje je i standardem definovaná indukčnost, která se podílí na přenosu dat. Při vysílání mění jednotlivé stanice odběr proudu, což vyvolá vznik napěťových impulsů (kladných a záporných) na vedení. Tyto napěťové impulsy jsou superponovány na napájecím napětí, jejich výskyt a polarita jsou sledovány a vyhodnocovány.

ASI kabel

Standard ASI používá vlastní kabel. Jde o plochý kabel se dvěma vodiči dle následujícího obrázku:

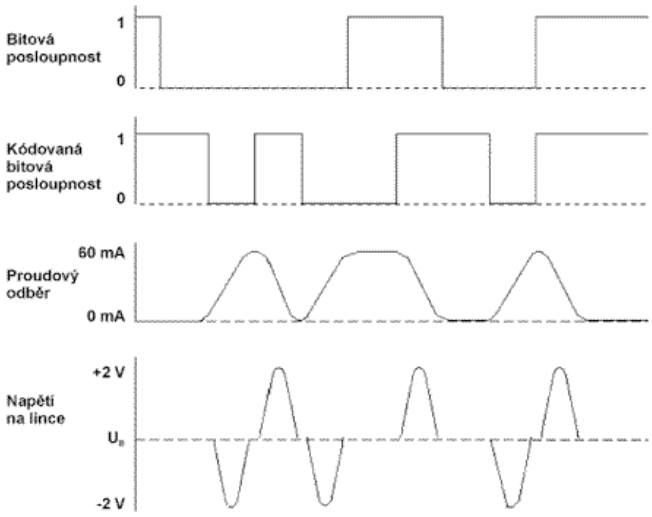


Obr. 13.1: Profil kabelu pro ASI.

Připojení modulů k tomuto kabelu je realizováno krempováním, tzn. nejsou třeba žádné konektory ani odizolování spojů. Tato metoda současně zrychluje a zjednodušuje instalaci, neboť tvar kabelu brání připojení s opačnou polaritou.

Kódování dat

Kódování dat definované standardem je zřejmé z následujícího obrázku.



Obr. 13.1: Kódování dat u ASI.

Binární posloupnost je nejprve zakódována kódem Manchester (ve středu bitového intervalu bitu s úrovní log. 0 je v zakódovaném signálu přechod log. 1 -> log. 0, ve středu bitového intervalu bitu s úrovní log. 1 je v zakódovaném signálu přechod log. 0 -> log. 1). Kódovaná bitová posloupnost je vysílačem převedena na změny odběru proudu, které se pak na lince projeví výše zobrazenými

napěťovými impulsy. Délka bitového intervalu je cca 6 ms, což odpovídá přenosové rychlosti 166 Kb/s. Skutečný přenosový výkon je vzhledem k prodlevám v komunikaci poněkud nižší.

Linková vrstva standardu

Linková vrstva používá řízení přístupu Master-Slave a specifický formát telegramů. Jednotlivé účastnické stanice jsou cyklicky oslovovány stanicí řídicí, celý cyklus trvá podle počtu účastnických stanic maximálně 5 ms pro 31 účastnických stanic. Při menším počtu dotazovaných účastnických stanic se doba trvání cyklu úměrně zkracuje.

Obr. 13.3: Struktura telegramu ASI.

Komunikace mezi řídicí a jednou účastnickou stanicí je zřejmá z následujícího obrázku. Tato posloupnost se cyklicky opakuje pro všechny účastnické stanice v síti. Význam jednotlivých symbolů je následující:

- MT - telegram řídicí stanice (Master telegram).
- MP - prodleva po telegramu řídicí stanice (Master prodleva).
- ST - telegram účastnické stanice (Slave telegram).
- SP - prodleva po telegramu účastnické stanice (Slave prodleva).

Telegram řídicí stanice představuje výzvu pro účastnickou stanici, telegram účastnické stanice je pak reakcí na tuto výzvu. Obě časové prodlevy pak pomáhají synchronizaci komunikace.

Zabezpečení přenosu

Datové přenosy jsou zabezpečeny následujícími prvky:

- Start bit - prvním impulsem, který se vyskytne na sběrnici při přenosu telegramu musí být negativní impuls.
- Alternace impulsů - vzhledem ke zvolenému typu modulace musí mít dva následující impulsy opačnou polaritu.
- Mezera mezi impulsy - mezi dvěma impulsy uvnitř telegramu smí být mezera o délce maximálně délky jednoho impulsu.
- Informační obsah - ve druhé polovině bitového intervalu musí být vždy impuls.
- Kontrola parity - v kódovém slově telegramu musí být sudý počet bitů hodnoty log. 1.
- Stop bit - Posledním impulsem, kterým končí přenos telegramu po sběrnici, musí být kladný impuls.
- Délka telegramu - po ukončení telegramu stop bitem již na sběrnici nesmí následovat žádné další impulsy.

13.2 Profibus

Průmyslová komunikační síť Profibus představuje v současné době jeden z velmi rozšířených komunikačních standardů v oblasti promyslové automatizace. Vychází z otevřeného komunikačního

modelu ISO/OSI a je určena pro všechny oblasti automatizace, tj. řízení technologií, řízení procesů a automatizaci budov. Historie sítě Profibus spadá do poloviny 80. let, kdy se firmy Bosch, Klockner & Moller a Siemens dohodly na společném projektu pro vývoj promyslové sběrnice označené jako Profibus (PROces Field BUS). Cílem bylo vytvořit architekturu komunikačního systému, který by na jedné straně respektoval potřebu připojit na sběrnici malá zařízení a současně vytvořil otevřené rozhraní pro komunikaci různých automatizačních zařízení (programovatelné automaty, operátorské panely, snímače, akční členy atd.). Norma Profibus je nezávislá na výrobci a její otevřenost je garantována normou EN 50170.

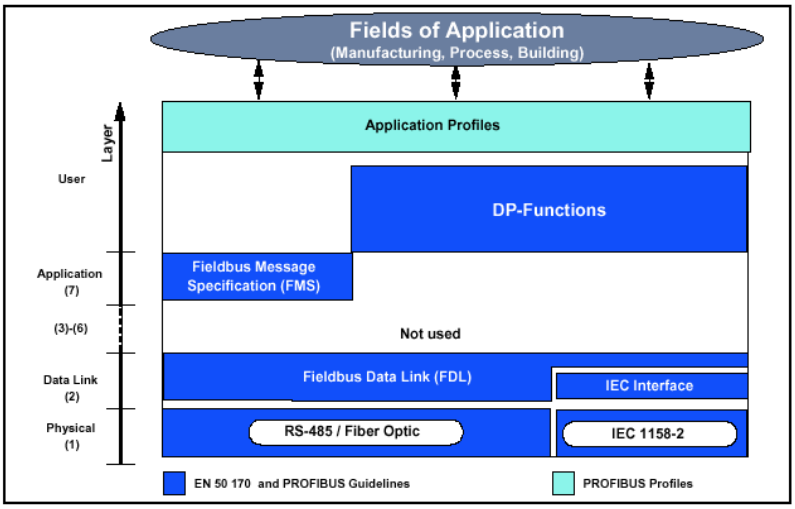
Architektura protokolu Profibus

Norma Profibus vyšla z modelu ISO/OSI. Z důvodu časové optimalizace definuje z tohoto modelu pouze vrstvy fyzickou, linkovou a aplikační:

- Fyzická vrstva definuje fyzické spojení mezi zařízeními včetně mechanických a elektrických vlastností tohoto spojení a současně je v této vrstvě definována topologie sítě; Profibus podporuje přenos po sběrnici RS-485, optickém vláknu a pro výbušné prostředí po proudové smyčce IEC 1158-2.
- Linková vrstva (Fieldbus Data Link) definuje mechanismus přístupu účastníka na přenosové médium (token passing, master-slave) a zabezpečuje tvorbu zprávy na úrovni bitového řetězce včetně generování kontrolních částí.
- Aplikační vrstva je nejvyšší vrstvou v referenčním modelu ISO/OSI, poskytuje jednotlivé služby nezbytné pro realizaci komunikace z hlediska uživatele.

Pro každou oblast automatizace existují určité požadavky na vlastnosti komunikační sítě. Tyto požadavky jsou shrnuty v tzv. profilech zařízení definovaných nad aplikační vrstvou pro každý typ protokolu. V současné době existují tři varianty komunikačního standardu Profibus:

- Profibus-DP -komunikační síť pro rychlou komunikaci typu master-slave. Uživatelská vrstva nabízí jednoduché funkce pro komunikaci, konfigurování a řízení provozu na síti. Komunikačním médiem je buď kroucená dvojlinka (standard RS-485), nebo optické vlákno.
- Profibus-FMS -nabízí komunikační standard pro komunikaci v heterogenním prostředí a s velkou množinou služeb pro práci s daty, programy a alarmy. Komunikačním médiem je podobně jako u varianty Profibus-DP buď kroucená dvojlinka (standard RS-485), nebo optické vlákno. Je určen pro komunikaci na vyšší úrovni řízení. Na této úrovni mezi sebou komunikují řídicí systémy (PLC, NC) a systémy vizualizace a sběru dat (operátorské panely, PC). Cílem této komunikační sítě není dosažení minimální doby reakce, ale především komunikace v heterogenním prostředí (komunikace v prostředí různých výrobců).
- Profibus-PA -používá rozšířenou normu Profibus-DP a je určen pro sběr dat ze snímačů v automatizaci procesů (tlak, teplota apod.). Aby bylo možné síť využít také v prostředí s nebezpečím výbuchu, je použita i speciální fyzická vrstva - proudová smyčka podle standardu IEC 1158-2. Dnes už se v nových aplikacích nenasazuje, je nahrazen Ethernetem.



Obr. 13.4: Architektura protokolu.

13.2.1 Fyzická vrstva sběrnice Profibus

Vlastnosti fyzické vrstvy komunikačního systému bývají nezřídka určující pro oblast použití té které komunikační sítě. Mimo obecných parametrů (bezpečnost přenosu, přenosová rychlost či maximální rozlehlost sítě) je zajímavá i jednoduchost instalace, cena, popř. realizace některých speciálních požadavků (napájení zařízení prostřednictvím komunikačního kabelu atd.). Protože je nemožné vyhovět současně všem požadavkům uživatelů, nabízí standard Profibus použití následujících tří variant fyzické vrstvy:

- Sběrnici RS-485 pro DP a FMS.
- Smyčku podle IEC 1158-2 pro PA.
- Optické vlákno (FO).

Přenosová cesta na bázi sběrnice RS-485

Nejčastěji používanou variantou přenosové vrstvy sítě Profibus je rozdílová napěťová sběrnice podle standardu RS-485 (v souvislosti se sítí Profibus se též můžeme setkat s označením H2). Aplikační oblast této varianty tvoří instalace Profibus-DP nebo FMS s požadavkem na jednoduchou, levnou síť s velkou přenosovou rychlostí. Pro RS-485 je použito stíněného kabelu s jedním měděným krouceným párem. U instalací, kde lze vyloučit vliv elektromagnetického rušení, je možné použít kabel nestíněný. Doporučené parametry kabelu pro síť Profibus se svými hodnotami včetně zakončení odlišují od standardu RS-485:

- Impedance: 135 Ω až 165 Ω.
- Kapacita: < 30 pF/m.
- Odpor smyčky: 110 Ω/km.
- Minimální průřez vodiče: > 0,34 mm^2.

Co do topologie, se jedná o sběrnicovou (liniovou) strukturu sítě. Povolená délka odbočky je 0,3 m. Na jeden segment sběrnice RS-485 je možné připojit až 32 účastníků sítě Profibus (aktivních, pasivních nebo opakovačů). Pro připojení účastníků na sběrnici je doporučeno použít devítikolíkový konektor D sub. Segment sběrnice musí být na obou svých koncích ukončen sběrnicovým terminátorem. Terminátor obsahuje mimo přizpůsobovacího odporu konce vedení R_t (220 Ω) ještě odpory R_d a R_u (390 Ω), pomocí nichž je definován stav vedení v době, kdy žádné zařízení nevysílá.

Pro zajištění správné funkce musí být oba terminátory napájeny. Většina výrobců dodává připojovací síťové konektory pro Profibus s integrovaným odpínatelným terminátorem nebo instalují odpínatelné terminátory přímo do jednotlivých zařízení. Zároveň je velmi výhodné použít konektory, ve kterých je propojen kabel přicházející k zařízení s kabelem vycházejícím ze zařízení. Při použití takovýchto konektoru je možné odpojit a připojit účastnická zařízení bez rozpojování segmentu. Maximální délka segmentu sítě závisí na použité přenosové rychlosti. Závislost je samozřejmě nepřímá, a proto nabízí Profibus v této variantě stupňovitě volitelnou přenosovou rychlost v hodnotách: 9,6 - 19,2 - 38,4 - 93,75 - 187,5 - 500 - 1500 - 3000 - 6000 - 9000 -12000kb/s. To umožňuje téměř vždy na1ézt vhodný kompromis mezi požadovanou rozlehlostí sítě a přenosovou rychlostí. Pokud je třeba připojit větší počet účastníků než 32 nebo je třeba zvětšit rozlehlost sítě při zachování přenosové rychlosti, je nutné pokračovat dalším segmentem. K propojování segmentů slouží opakovače (repeater). Opakovač galvanicky odděluje segmenty a regeneruje signál procházející opakovačem z jednoho segmentu do druhého. Pomocí opakovačů je možné realizovat jak sériové řazení segmentů, tak i odbočení. Jediným hlediskem pro spojování segmentů je čas potřebný pro přenos signálu mezi nejvzdálenějšími stanicemi. V praxi se sériově řadí maximálně 10 segmentů. Celkově však nesmí být překročen maximální počet účastníků připojených k jedné síti daný adresovacími možnostmi standardu Profibus, tj. 127 účastníků (aktivních nebo pasivních).

Přenosová cesta na bázi sběrnice lEC 1158-2 pro Profibus-PA

Sběrnice podle standardu IEC 1158-2 splňující požadavky aplikací v chemickém a petrochemickém průmyslu nese někdy označení H1 (neplést se starším označením sítě Industrial Ethernet firmy Siemens -SINEC H1). Tato varianta zaručuje tzv. jiskrovou bezpečnost a zároveň podporuje napájení zařízení po sběrnicovém kabelu. Proto se předpokládá použití převážně ve výbušných prostředích. Jako komunikační médium se pro tuto fyzickou vrstvu používá kabel s jedním měděným krouceným párem (stíněným nebo nestíněným) s následujícími doporučenými vlastnostmi:

- Impedance při 31 ,25 kHz: 100 Ω ±20 %.
- Kapacitní nesouměrnost: < 2 nF/km.
- Odpor smyčky: 44 Ω/km.
- Minimální průřez vodiče: > 0,88 mm2.
- Útlum při 39 kHz: 3 dB/km.

Přenos dat je zde založen na následujících principech:

- Každý segment. má pouze jeden zdroj proudu.
- Při vysílání nedodává zařízení do sítě žádný proud.
- V klidovém stavu odebírá každé zařízení konstantní proud.
- Každé účastnické zařízení pracuje jako pasivní proudový spotřebič.
- Segment je na obou koncích zakončen pasivním terminátorem.

Pro modulaci se předpokládá klidový proud minimálně 10mA pro napájení každého zařízení na síti. Logické úrovně komunikačního signálu jsou generovány vysílajícím zařízením modulací ±9mA ke klidovému proudu. U této varianty fyzické vrstvy je použito synchronního bitově orientovaného protokolu s přenosovou rychlosti pevně nastavenou na 31,25 kHz.

Přenos optickým vláknem - FO

Světlovody se v instalacích sítě Profibus používají pro aplikace v prostředí s vysokou úrovní elektromagnetického rušení a zároveň umožňují zvětšit maximální vzdálenosti pro velké přenosové rychlosti. Pro přenos optickým vláknem jsou používány jak levné plastové (délka až 50 m), tak i skleněné (délka až 2 km) světlovodné kabely. Existují však i varianty pro skleněné světlovody s délkou 15 km. Většina výrobců dodává speciální konektory s integrovaným převodníkem RS-485 na optické rozhraní a opačně. To umožňuje jednoduše kombinovat metalický a optický rozvod v jedné síti.

13.3 Linková vrstva - přístup na sběrnici

Druhá vrstva referenčního modelu ISOIOSI, linková, zabezpečuje mimo služeb pro blokový přenos dat především řízení přístupu na sběrnici. U standardu Profibus se tato vrstva nazývá Fieldbus Data Link (FDL) a je nejnižší vrstvou, ke které je v některých speciálních případech realizován přístup aplikačních programů.

Metoda řízení přístupu na sběrnici stanovuje, kdy je zařízení oprávněno vysílat na sběrnici. Musí být zajištěno, že v danou chvíli má právo na vysílání dat pouze jedno zařízení. Metoda řízení přístupu na sběrnici použitá u sítě Profibus byla navržena tak, aby uspokojovala následující dva požadavky:

- Pro komunikaci mezi složitějšími automatizačními zařízeními musí být zajištěno, že každému z těchto zařízení je přidělován dostatečný prostor na provádění jeho komunikačních úloh.
- Pro komunikaci mezi automatizačním zařízením a jemu přiřazenými jednoduchými vstupně/výstupními zařízeními je třeba zajistit co nejjednodušší a nejrychlejší cyklickou výměnu dat.

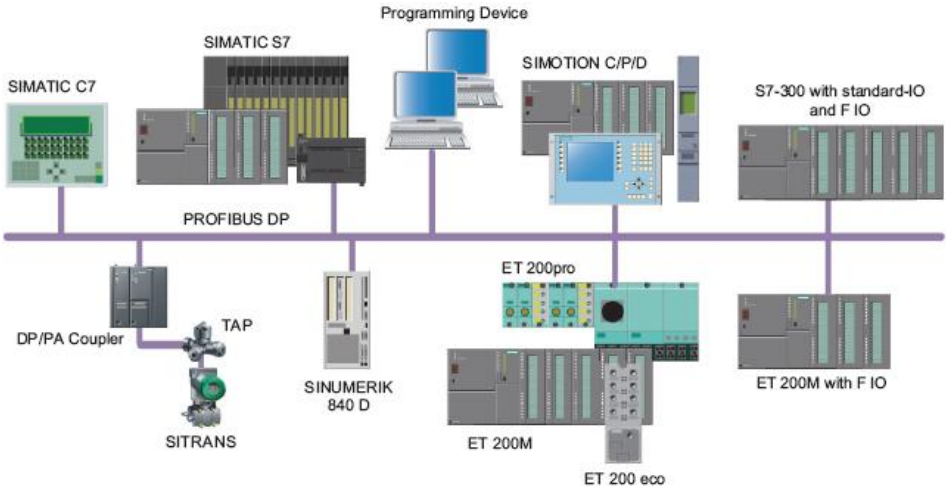
Proto v sobě Profibus kombinuje dvě základní metody řízení přístupu na sběrnici: metodu token passing (předávání pověření v logickém kruhu) pro komunikaci mezi aktivními zařízeními a metodu master-slave (centrálně řízené dotazování) pro komunikaci mezi aktivním a jemu přidělenými pasivními zařízeními. Pak je možné implementovat jak konfigurace používající pouze jednu z metod, tak i konfigurace s tzv. hybridním řízením přístupu na sběrnici (kombinace obou metod).

Protože metoda token passing patří k decentralizovaným metodám řízení přístupu na sběrnici, musí každé aktivní zařízení připojené na sběrnici realizovat mechanismus předávání pověření k přístupu na sběrnici. Jedním ze základních požadavků kladených na průmyslové komunikační sběrnice je možnost připojovat a odpojovat jednotlivé účastníky ke sběrnici, aniž by to mělo vliv na činnost ostatních zařízení. Proto nemůže uvedený mechanismus vycházet ze statických dat. Logický kruh, ve kterém dochází k předávání pověření, je tvořen aktivními stanicemi uspořádanými vzestupně podle jejich adresy. V ustáleném stavu zná tedy každá stanice v logickém kruhu svého předchůdce, od kterého dostane pověření (token) a svého následníka, kterému pověření zasílá. Typické pro deterministické metody přidělování přístupu na sběrnici, k nimž token passing patří, je existence parametru definujícího maximální dobu, za níž každá stanice získá právo vysílat. U standardu Profibus je to právě žádaná doba oběhu pověření označovaná T_{TR} (Target Rotation Time). Měření skutečné doby oběhu pověření T_{RR} (Real Rotation Time) provádí každá aktivní stanice zařazená do logického kruhu, která v době, kdy vlastní pověření, porovnává doby, T_{TR} a T_{RR}. Pokud je T_{RR} menší než T_{TR} realizuje stanice své komunikační požadavky v pořadí, v jakém vznikly a s ohledem na jejich prioritu. Pokud nemá žádné další požadavky nebo hodnota T_{RR} dosáhla hodnoty T_{TR}, odešle stanice pověření svému následníkovi v logickém kruhu a vynuluje časovač pro měření doby T_{RR}. V případě, že by došlo k situaci, kdy v době přijetí pověření stanicí bude hodnota T_{RR} větší než T_{TR}, má stanice právo odvysílat jednu zprávu s vysokou prioritou. Za účelem detekce nově připojených zařízení prohledává každé zařízení v rámci tzv. režijních činností na sběrnici adresní prostor. Adresní prostor zařízení v logickém kruhu, za který je dané zařízení "zodpovědné", začíná adresou o jedničku větší než vlastní adresa zařízení a končí adresou o jedničku menší, než je adresa jeho následníka v kruhu. Adresní prostor je

uzavřen tak, že se po adrese 126 pokračuje adresou 0. Parametrem zvaným GAP-Factor je možné specifikovat, jak často se má adresní prostor prohledávat. Protože prohledávání adresního prostoru patří k časově nejnáročnějším akcím (je prováděno pokusem o získání tzv. statusu stanice FDL), je možné adresní prostor shora omezit hodnotou parametru HSA (Highest Station Address). V případě odpojení zařízení nepřijde potvrzení o přijetí pověření následníkem a pověření je zasláno dalšímu nalezenému zařízení. Zároveň jsou připraveny mechanismy reagující na ztrátu pověření apod.

13.4 Profibus-DP

Protokol Profibus-DP je navržen pro rychlou cyklickou výměnu dat na nejnižší (technologické) úrovni komunikačního modelu CIM, tzn. pro komunikaci mezi programovatelnými řídicími automaty a jejich decentralizovanými vstupy a výstupy (v/v). Jedním ze speciálních požadavků kladených na komunikační systémy pro tuto úroveň řízení je zaručit kratší komunikační cyklus, než je cyklus zpracování řídicího algoritmu programovatelným automatem tak, aby nedocházelo ke zpoždění vlivem komunikace. Výměna dat probíhá mezi programovatelným automatem (řídicí, nadřízené zařízení - master) a jednotlivými v/v zařízeními (řízenými -slave) cyklicky. Avšak pro speciální účely (náběh komunikace, konfigurování zařízení, diagnostika atd.) jsou k dispozici acyklické služby.



Obr. 13.5: Typická architektura řídicího systému se sběrnicí Profibus DP.

Typy zařízení a konfigurace systému

Zařízení podporující komunikaci Profibus-DP je možné rozdělit do následujících tří skupin:

- DP Master Class 1 (DPM1) -řídicí zařízení realizující cyklickou komunikaci s podřízenými zařízeními: nejčastěji realizováno pomocí programovatelného automatu (Programmable Logic Controller -PLC), počítače PC atd.
- DP Master Class 2 (DPM2) -řídicí zařízení pro realizaci diagnostických a monitorovacích funkcí používaných při uvádění sítí Profibus-DP do provozu nebo při provozu pro monitorovací účely.
- DP Slave -periferní zařízení (v/v zařízení, pohony, ventily, operátorské panely atp.) získávající vstupní data pro řídicí zařízení a poskytující výstupní data od řídicího zařízení (množství vstupních a výstupních dat závisí na typu zařízení: maximálně může zařízení poskytovat 246 bajtů vstupních dat a požadovat 256 bajtů výstupních dat).

Z hlediska konfigurace sítě je možné provozovat konfiguraci buď monomaster, nebo multimaster. V konfiguraci monomaster je na sběrnici pouze jedno aktivní zařízení typu DPM1. Komunikace mezi

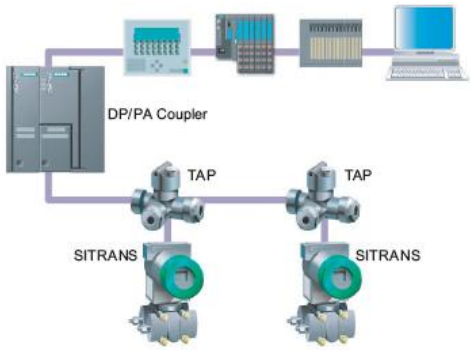
tímto zařízením a pasivními zařízeními typu DP Slave je realizována metodou master-slave. Toto řešení zaručuje nejkratší doby trvání komunikačního cyklu. Při konfiguraci multimaster se na sběrnici nachází několik aktivních řídicích zařízení. Potom můžeme na celý systém pohlížet jako na několik podsystémů vždy s jedním DPM1 zařízením, jemu přidělenými zařízeními DP Slave a případné diagnostické zařízení typu DPM2. Norma samozřejmě připouští i sdílení zařízení typu DP Slave několika řídicími zařízeními, ale pouze v případě vstupních zařízení typu DP Slave. Možnost zapisovat na výstupy má pouze zařízení DPM1, které konfigurovalo dané výstupní zařízení DP Slave. Mezi jednotlivými řídicími zařízeními dochází k předávání pověření k přístupu na sběrnici metodou token passing a v době, kdy řídicí zařízení vlastní právo přístupu na sběrnici, realizuje cyklickou výměnu dat se svými zařízeními DP Slave. Tato varianta vykazuje z hlediska jednotlivých podsystémů delší časy komunikačního cyklu.

13.5 Profibus PA

Profibus-PA je navržen pro přístrojovou instrumentaci v bezpečném i nebezpečném prostředí Zóna 0,1 a 2. PROFIBUS PA povoluje pružnou topologii sběrnice s délkou větve do 120m pro výbušná prostředí do 30m. Standardizované funkce PROFIBUS PA přes profily. Profily popisují funkční bloky, zařízení a volitelné parametry. Zařízení s PROFIBUS PA Profil III není jen univerzální, ale také vyměnitelný bez dalších potřebných technických úprav.

Až 31 procesních zařízení může být připojeno do každého DP/PA Link. Každý DP/PA Link může číst a psát 244 bytů informací po síti. Teoreticky je možné do jednoho PROFIBUS-DP lze zapojit více než 3800 procesních přístrojů.

Maximální délka PROFIBUS PA při dodržení konfiguračních pravidel je 1900m.



Obr. 13.6: Typická architektura řídicího systému se sběrnicí Profibus PA.

Shrnutí pojmů

V dnešní automatizaci se používá celá řada komunikačních sběrnic. Typickým představitelem nejnižší úrovně je **AS Interface**. Slouží pro připojení hlavně binárních čidel k řídicímu systému. Používá Master/Slave přístup ke komunikačnímu médiu. Fyzická vrstva je tvořena profilovaným dvouvodičovým kabelem, který přenáší jak komunikační signál, tak napájení pro jednotlivé Slavy.

O úroveň výše, na „field“ level se často používá komunikační sběrnice **Profibus**. Slouží pro přenášení větších objemů dat mezi stanicemi. Existuje několik modifikací, zejména **Profibus PA** a **DP**. Profibus DP slouží pro připojení vzdálených jednotek vstupů a výstupů k programovatelnému automatu, zatímco Profibus PA se používá v procesní automatizaci pro připojení procesní instrumentace.

Sběrnice Profibus používá několik typů přenosových médií, zpravidla sběrnicovou topologii a přístup ke komunikačnímu médiu **Master/Slave**.

Kontrolní otázky

1. Jaké jsou základní vlastnosti komunikační sběrnice AS Interface?
2. K čemu se používá sběrnice AS Interface?
3. Popište fyzickou vrstvu sběrnice AS Interface.
4. Popište linkovou vrstvu sběrnice AS Interface.
5. Jaké jsou základní vlastnosti komunikační sběrnice Profibus?
6. Jaké jsou typy sítě Profibus?
7. K čemu se používá sběrnice Profibus?
8. Popište fyzickou vrstvu sběrnice Profibus DP.
9. Popište linkovou vrstvu sběrnice Profibus DP.
10. Jakou používá Profibus-DP přístupovou metodu?
11. Popište fyzickou vrstvu sběrnice Profibus PA.

Další zdroje

13-1. Kováč F.: Distribuované riadiace systémy. Vydavatelstvo STU, Bratislava 1998.

13-2. Janeček J.: Distribuované systémy. Vydavatelství ČVUT, Praha 1997.

13-3. Janeček J., Bílý M.: Lokální sítě. Vydavatelství ČVUT, Praha 1998.

13-4. Časopis Automatizace 7/98.

13-5. Profibus System Description. Profibus Nutzeorgenization, Ocrober 2002.

13-6. Becker R. : AS-Interface, The Automation Solution. AS-International Association, 2002.

Řešená úloha 13.1.

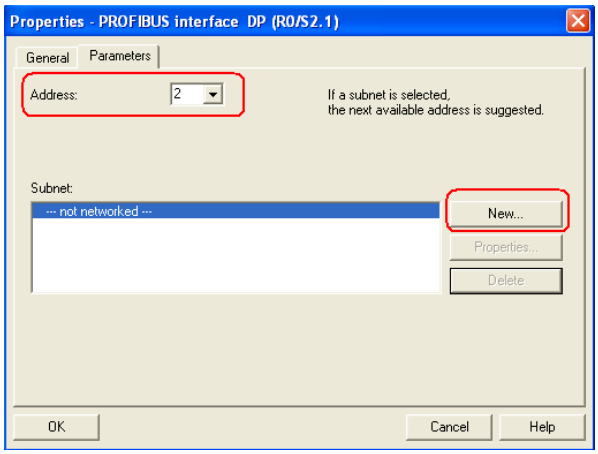
Vytvořte hardwarovou konfiguraci PLC s CPU 314C-2DP komunikující se vzdálenou periferií ET200M po sběrnici PROFIBUS-DP.

V přednášce č.2 byly popsány nyní dostupné programovatelné automaty PLC S7-300. Některé z nich měly také integrovaná rozhraní pro komunikaci např. přes PROFIBUS-DP.

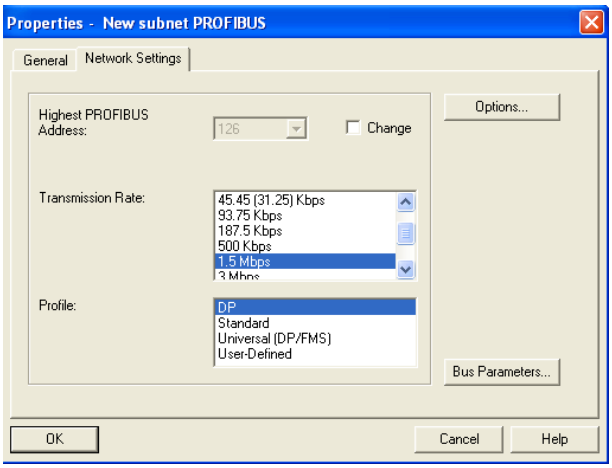
Základní popis hardwarové konfigurace byl ukázán ve cvičení číslo 2. Nyní bude předvedena hardwarová konfigurace s PLC S7-300, *CPU 314C-2DP (6ES7 314-6CF02-0AB0)* a periferií *ET 200M (6ES7 153-1AA03-0XB0).*

Tento typ CPU je umístěn ve složce CPU 300 a výběrem CPU 314C-2DP s číslem 6ES7 314-6CF02-0AB0 se vloží technikou (Drag&Drop) se vloží do slotu číslo 2. Bezprostředně po vložení se zobrazí okno s konfigurací sběrnice PROFIBUS-DP.

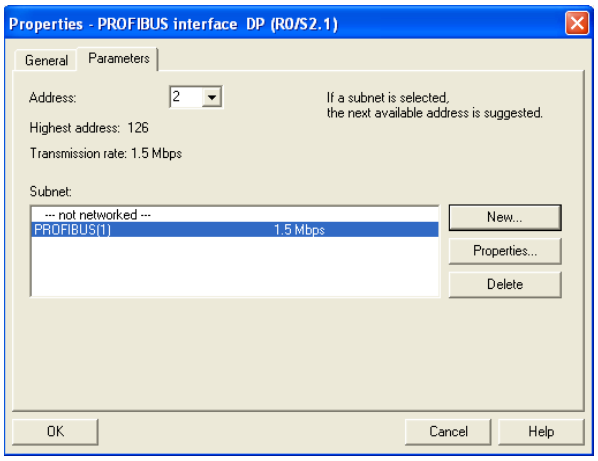
Adresa pro CPU je defaultně nastavena na 2. Kliknutím na tlačítko New (obr. 13.7) se spustí konfigurace sběrnice PROFIBUS-DP. V okně *New subnet PROFIBUS* (obr. 13.8) v záložce *General* lze zvolit přenosovou rychlost např. 1,5Mbps a profil tj. DP. Tato přenosová rychlost nemusí být vždy 1,5Mbps a závisí na také na vzdálenosti periferií od centrálního jednotky programovatelného automatu. Pokud se vyskytuje na trase nějaké rušení, je možné že tato rychlost musí být snížena, třeba i na 187,5Kbps. Protože ale v učebně se nepřekonávají větší vzdálenost než 20m, měla by ve většině případů vyhovovat přenosová rychlost 1,5Mbps. Kliknutím na tlačítko OK se zavře okno *New subnet PROFIBUS* a může být také zavřeno okno *PROFIBUS interface DP* (obr. 13.3).



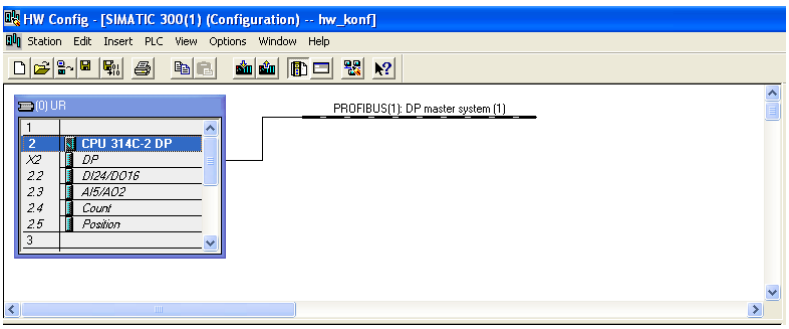
Obr. 13.7: Konfigurace PROFIBUS-DP.



Obr. 13.8: Volba přenosové rychlosti.

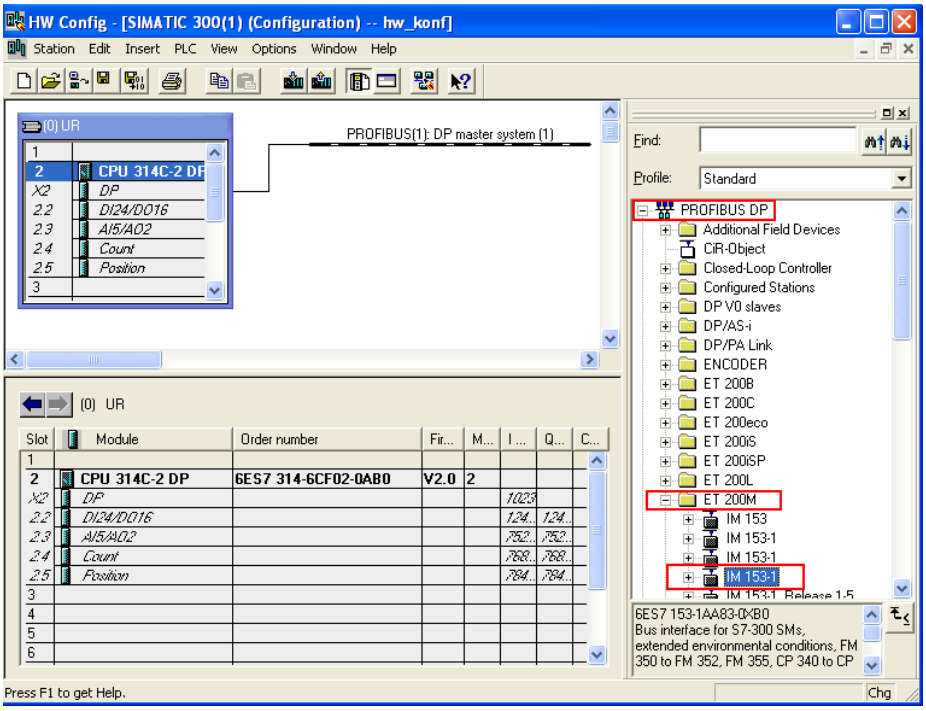


Obr. 13.9: Nastavení PROFIBUS-DP.

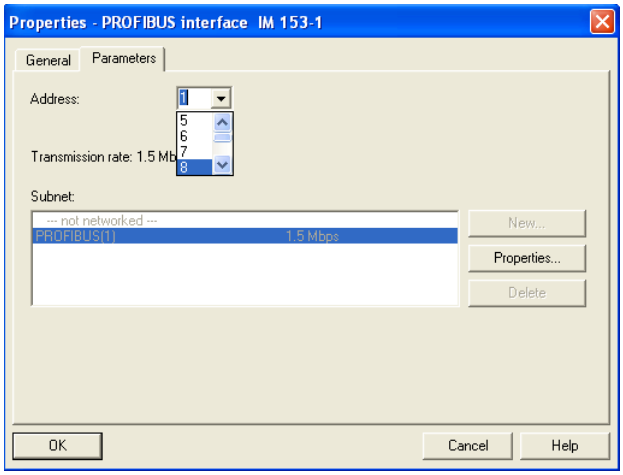


Obr. 13.10: Dokončení konfigurace pro CPU 314C-2DP.

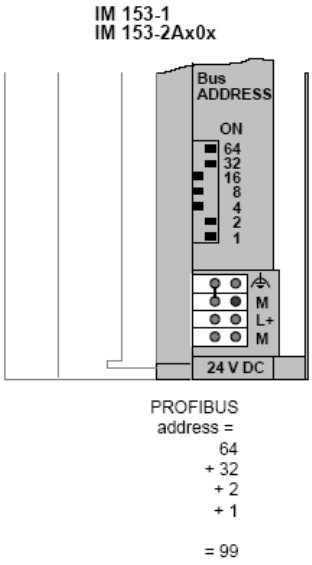
Modul ET200M najdete ve složce *PROFIBUS-DP → IM 153-1 (6ES7 153-1AA03-0XB0)*. Kliknutím na ikonu IM153-1 a opět přetažením technikou Drag&Drop umístěte myší, umístěte tento rozšiřovací modul na sběrnici PROFIBUS-DP. Po vložení budete vyzváni v okně *PROFIBUS interface IM153-1* (obr. 13.11), aby jste nastavili adresu modulu ET200M adresa je určena polohou přepínačů DIP, jak ukazuje obr. 13.12. Jestli je adresa vložena, kliknutím na tlačítko OK se nastaví tato adresa také v ikoně modulu ET200M



Obr. 13.11: Přidání vzdálené periferie ET200M na sběrnici PROFIBUS-DP.

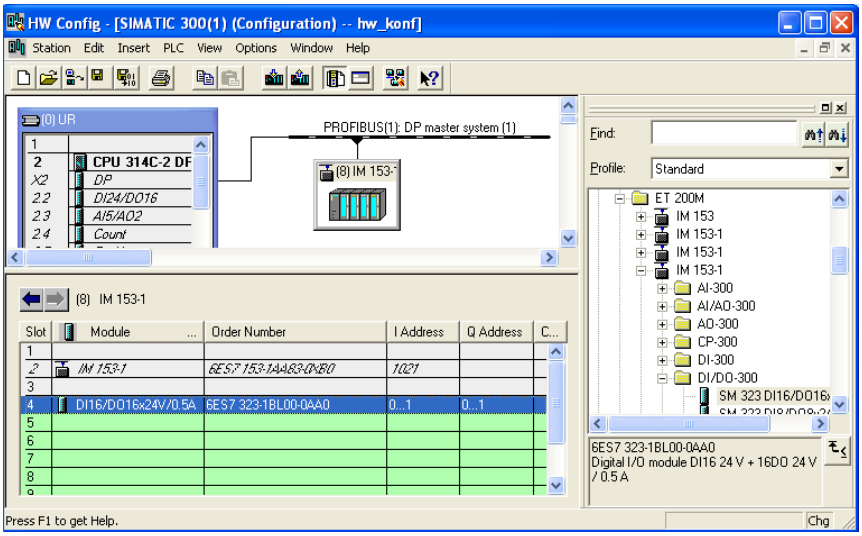


Obr. 13.12: Nastavení adresy ET200M.

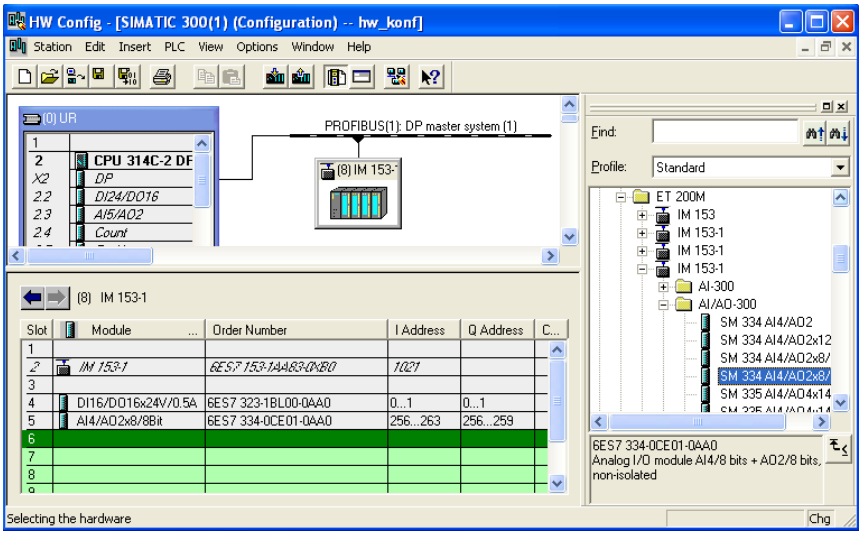


Obr. 13.13: Příklad výpočtu adresy na modulu ET200M.

Za modul ET200M je řazena např. dále karta DI/DO s označením 323-1BL00-0AA0 a AI/AO 334-0CE01-0AA0. Rozkliknutím ikony *IM 153-1 (6ES7 153-1AA03-0XB0)* naleznete další složky, které mohou obsahovat karty DI/DO apod. Kliknutím na složku DI/DO-300 lze poté vybrat odpovídající kartu DI/DO a vložit do slotu č.4 (obr. 13.14). Kartu analogových vstupů musí být vložena do slotu č.5

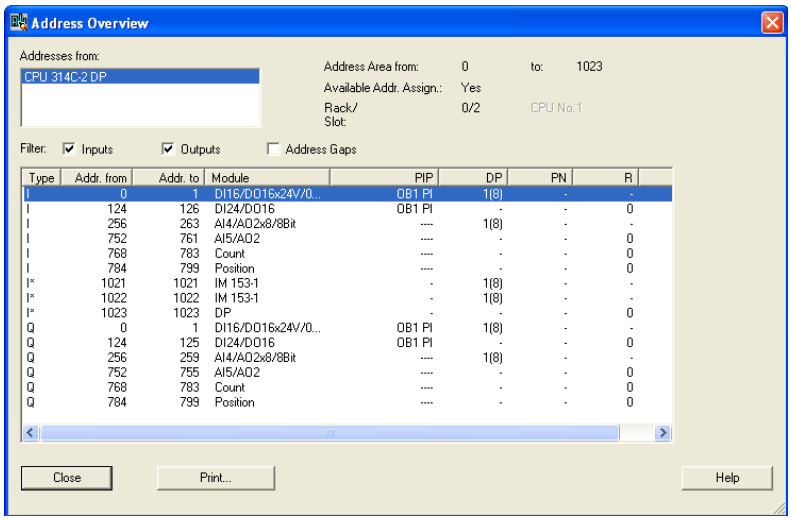


Obr.13.14 Vložení karty DI/DO.



Obr. 13.15: Vložení DI/DO.

Volbou Station → Save and Compile bude hardwarová konfigurace vytvořena. Pokud chcete získat přehled o jednotlivých adresách vstupů a výstupů můžete v menu View→ Address Overview získat přehled o všech adresách a modulech.



Obr. 13.16: Přehled adres.

DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 13\pr_13_1.zip

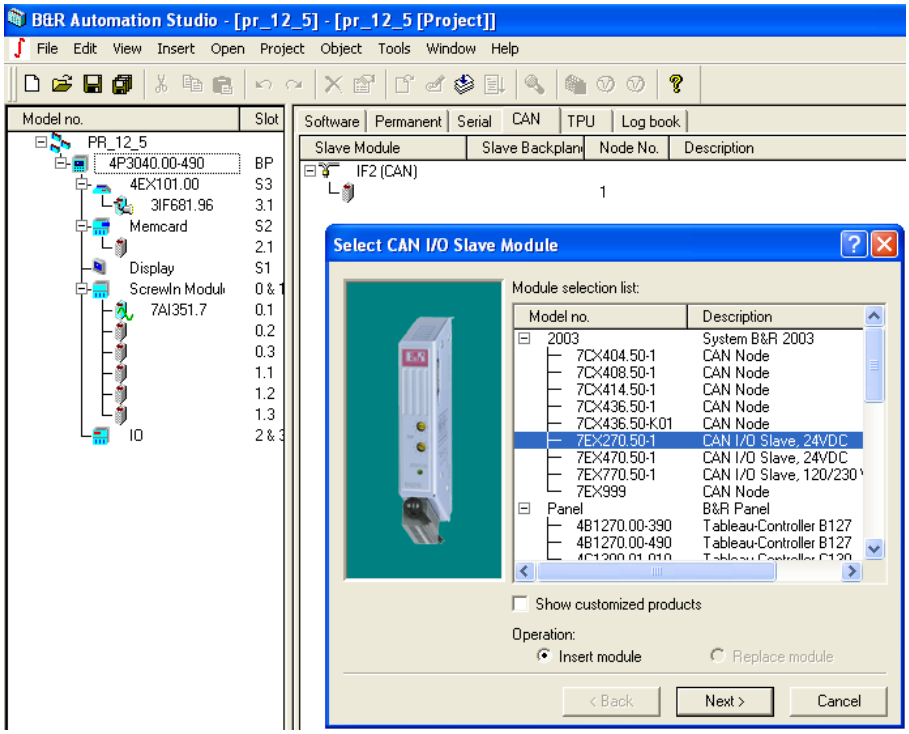
DVD-ROM

K této úloze je vytvořena animace, kterou naleznete na DVD: animace\animace 10\ anim1.avi.

Řešená úloha 13.2.

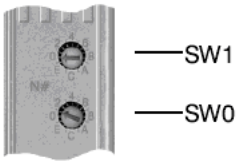
Vytvořte program, který bude realizovat komunikaci mezi PLC power panel PP41 a vzdálenou periferií CM 211 po sběrnici CAN. Pro ověření správné konfigurace realizujte funkci y = a*b, kde promenna a bude přiřazena digitálnímu vstupu 1 na PP41 a proměnná b bude přiřazena digitalnímu vstupu 1 na CM211. Proměnná y je přiřazena na digitální výstup 1 modulu CM 211.

Nejvýhodnější je opět Upload hardware. V levé části Automation Studia se zobrazí aktuální konfigurace power panelu PP41. Modul CM 211 je doplněn o CAN „ostrůvek“ EX270, který umožňuje realizaci komunikace přes CAN, musí být tento rozšiřovací modul dodatečně vložen, kliknutím na záložku *CAN* a kliknutím pravým tlačítkem myši na *IF2 (CAN)* a volbou *Insert* se vyvolá okno *Select CAN I/O Slave Module*, jak ukazuje obr. 13.17.



Obr. 13.17: Přidání modulu EX270.

Na každém modulu *EX270* jsou umístěny otočné přepínače, které specifikují adresu. V našem případě je to adresa *6 – Node Number*, protože na přepínači SW1 je hodnota 0 a na přepínači SW0 je nastavena hodnota 6 (obr. 13.18.).

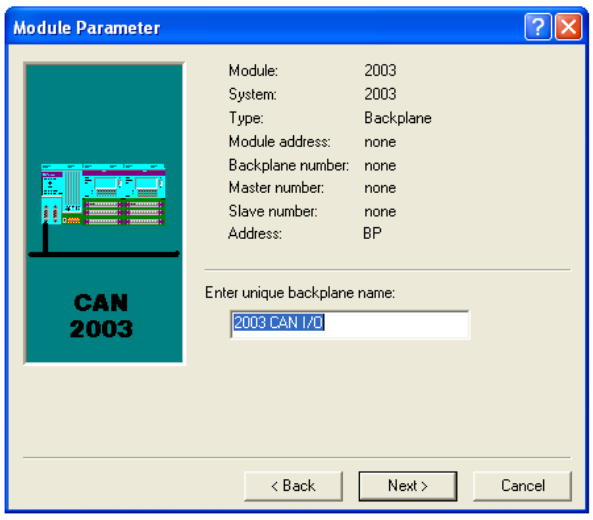


SW1	SW0	Node Number	Start Baudrate [kBit/s]
0	0	S-EEPROM	S-EEPROM
0	1 ... F	1 ... 15	250
1	0 ... F	16 ... 31	250
2	0 ... F	32 ... 47	250
3	0 ... F	48 ... 63	250

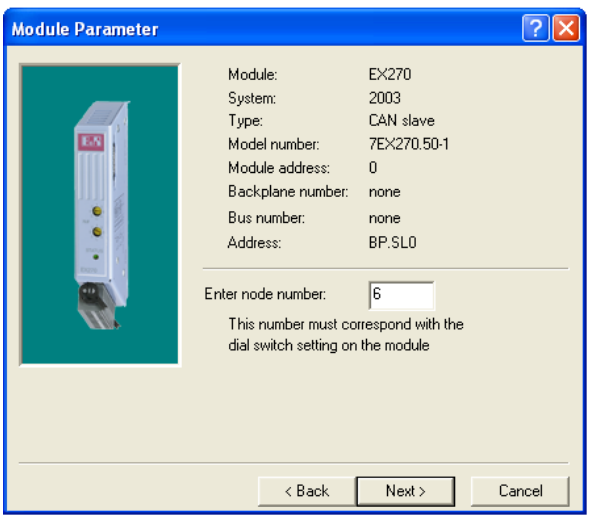
Obr. 13.18: Nastavení přenosové rychlosti a vyčtení adresy z přepínače.

Nastavením adresy, jak ukazuje obr. 13.21 se může ukončit průvodce konfigurací modulu EX270. Následuje konfigurace modulu CM 211. Kliknutím pravým tlačítkem myši na ikonu pod modul EX 270 (obr 13.22) a výběrem v pop-up menu Insert se spustí katalog pro přidání rozšiřujících modulů. Vyberte podle obr. 13.23. modul CM 211. Kliknutím na tlačítko OK se přidá modul do hardwarové konfigurace.

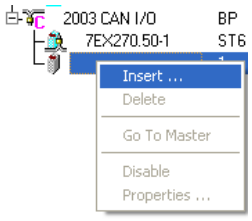
Bezprostředně poté je nutné nastavit také přenosovou rychlost pro CAN. Kliknutím na ikonu 4P3040.00-90, dále kliknutím na záložku CAN a kliknutím pravým tlačítkem myši na ikonu IF2(CAN) a výběrem Properties se vyvolá okno CAN I/O Properties, kde je nutné nastavit přenosovou rychlost, jak ukazuje obr. 13.23. Přenosová rychlost se opět určí na základě přepínačů SW0 a SW1, jak ukazuje obr. 13.18.



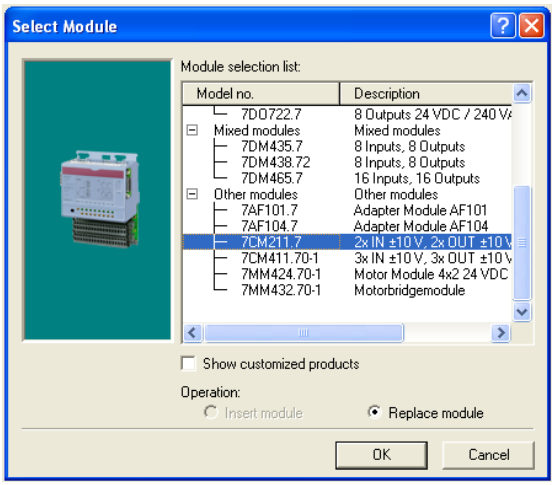
Obr. 13.19: Jméno CAN.



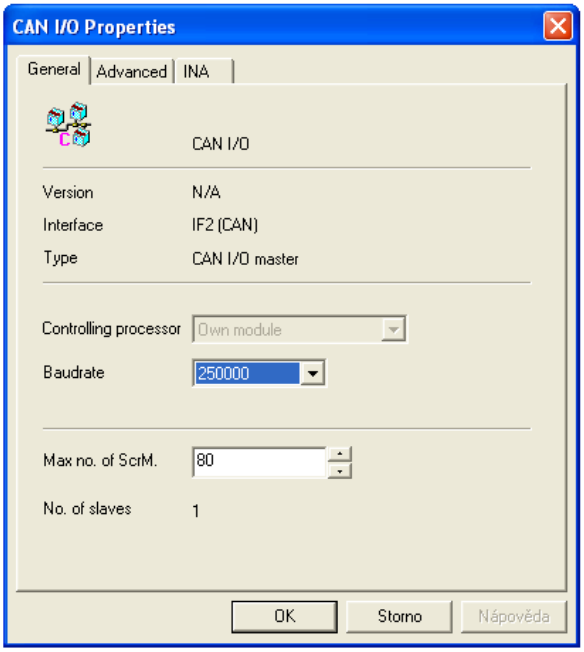
Obr. 13.20: Vložení adresy Node Number.



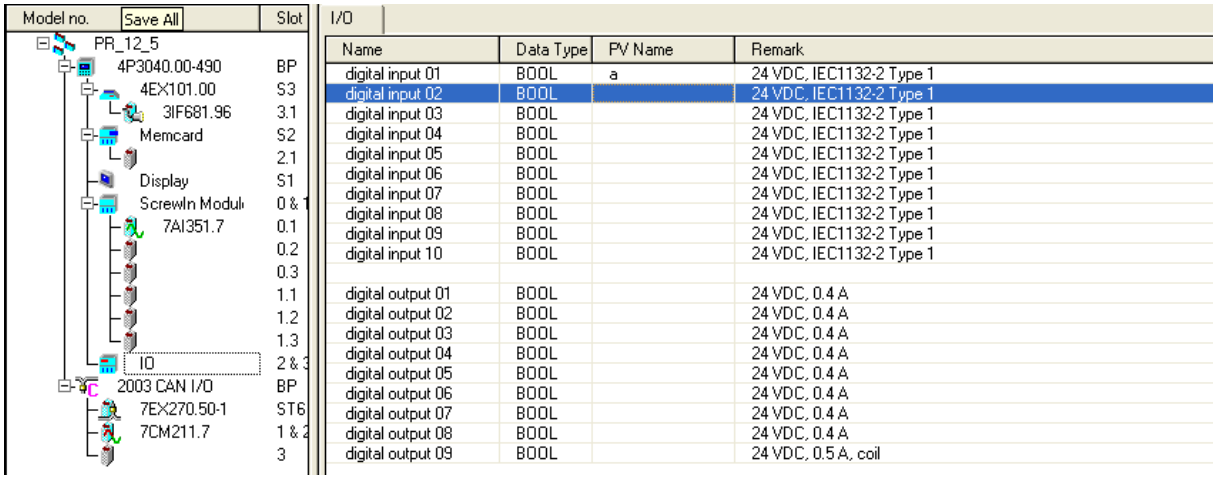
Obr. 13.21: Vložení CM211.



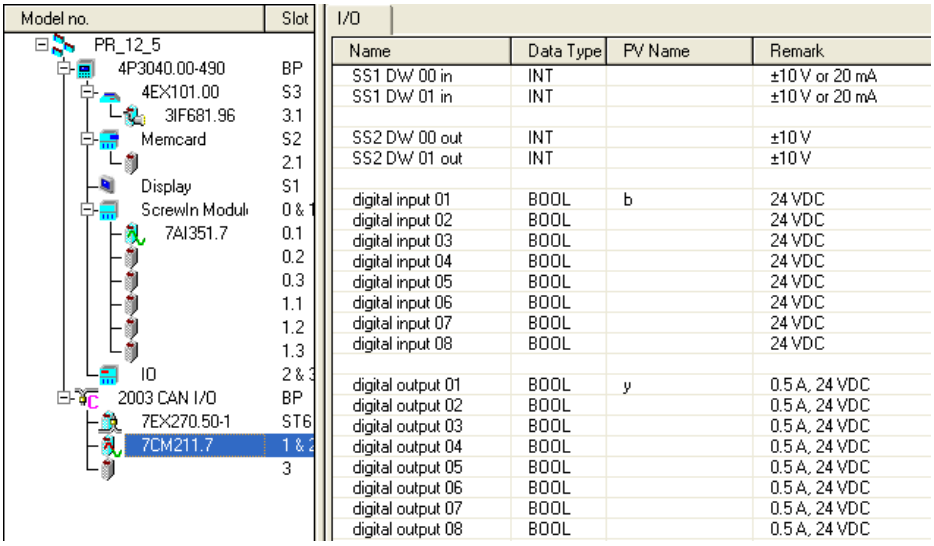
Obr. 13.22: Výběr CM 211 z katalogu.



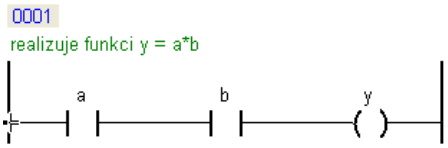
Obr. 13.23: Nastavení přenosové rychlosti.



Obr.13.24: Přiřazení proměnné a digitálnímu vstupu na PP41.



Obr. 13.25: Přiřazení proměnné b (y) digitálnímu vstupu a výstupu CM211.



DVD-ROM

Řešený příklad naleznete na DVD: cvičení\cvičení 13\pr_13_2.pgd.zip

14. PRŮMYSLOVÉ SBĚRNICE CAN, INDUSTRIAL ETHERNET A JEJICH POUŽITÍ V ŘÍZENÍ

Čas ke studiu: 3 hodiny

Cíl:

Kapitola se zabývá směrnicemi **CAN**, **Industrial Ethernet** a **Profinet**.

V úvodu kapitoly jsou uvedeny základní vlastnosti sběrnice CAN a je popsána fyzická vrstva. Následuje popis linkové vrstvy CANu, kde je hlavní pozornost věnována **metodě přístupu ke komunikačnímu médiu**, která je u této sběrnice specifická. Rovněž jsou uvedeny **typy rámců** a **metody detekce chyb**.

Další část kapitoly je věnována sběrnici Ethernet a její průmyslové modifikaci Industrial Ethernet. Jsou popsány fyzická vrstva, linková vrstva a formát datového rámce.

Poslední část kapitoly je věnována sběrnici Profinet, u níž jsou popsány základní parametry, formát rámce, komunikační kontroléry a způsoby nasazení. Text se věnuje i využití Profinetu pro připojení vzdálených vstupů/výstupů – tedy **Profinet IO**. Je zde demonstrováno rovněž použití tzv. komponentně orientované automatizace – **Profinet CBA**.

V závěru kapitoly je uveden poměrně podrobný popis aktivních síťových prvků od firmy Siemens, zejména switchů **Scalance** pro klasické i bezdrátové sběrnicové systémy.

Výklad

14.1 CAN

CAN – Controller Area Network, je výkonná sériová sběrnice pro tvorbu distribuovaných řídicích systémů. Byla vyvinuta v roce 1980 firmou Robert Bosch GmbH. CAN je standardizován International Standardization Organization (ISO) a Society of Automotive Engineers (SAE).

CAN je nejčastěji používanou sběrnicí v automobilových aplikacích. Vzhledem k tomu, že přední výrobci integrovaných obvodů implementovali podporu protokolu CAN do svých produktů, je stále častěji využíván i v průmyslu, ve vlacích, budovách, medicínské technice apod. Důvody jsou především nízká cena, dostupná potřebná součástková základna, snadné použití, spolehlivost, relativně velká přenosová rychlost a snadná rozšiřitelnost. V současné době má protokol CAN své pevné místo mezi ostatními průmyslovými komunikačními protokoly.

CAN využívá fyzickou a linkovou vrstvu ISO-OSI modelu, existuje však také řada protokolů vyšších vrstev pro CAN. Většinou se jedná o implementaci do aplikační vrstvy ISO/OSI modelu. Například:

CAL (CAN Application Layer) – v aplikační vrstvě jsou definovány služby týkající se Network Managementu, protokol CMS pro výměnu aplikačních dat, služby pro nastavování parametru fyzické vrstvy apod.

CANOpen – využívá služeb CAL a rozšiřuje jej o další služby pro práci s uživatelskou vrstvou.

Device Net – je průmyslová sběrnice založená na CANu. Definuje svou fyzickou vrstvu, linkovou (tedy základ komunikačního protokolu) přebírá od CANu, ale navíc definuje vyšší vrstvy.

CAN Aerospace – definuje aplikační vrstvu CANu pro aplikace v letectví apod.

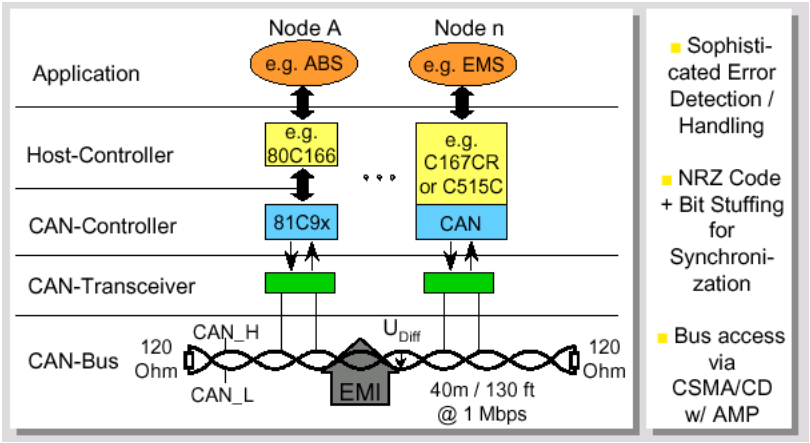
14.1.1 Základní koncepty sběrnice CAN

Nejdůležitějšími vlastnostmi sběrnice CAN jsou (viz. Obr. 14.1):

Sofistikovaný systém detekce a obsluhu chyb.

Kódování NRZ a vkládání bitů (bit stuffing) pro zajištění synchronizace přenosu.

Metoda náhodného přístupu ke komunikačnímu médiu, kdy vzniku kolizí je zabráněno pomocí systému priorit zpráv.



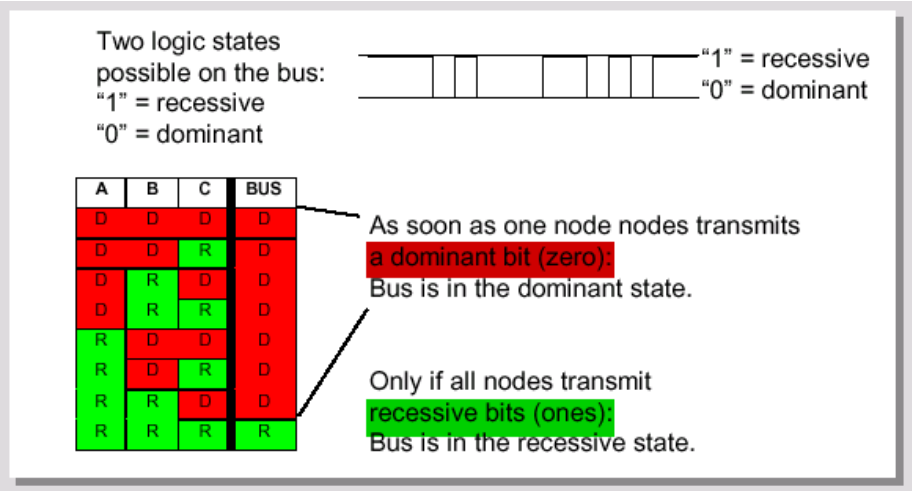
Obr. 14.1: Základní koncepty sběrnice CAN.

14.1.2 Fyzická vrstva

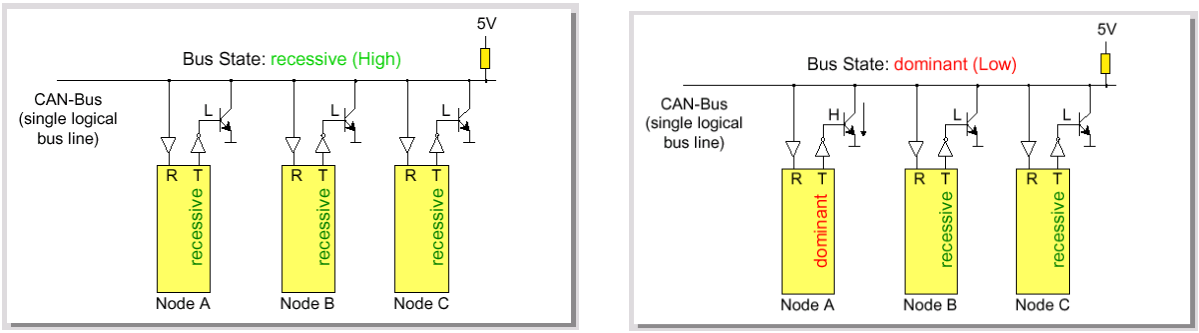
Základním požadavkem na fyzické přenosové médium protokolu CAN je, aby realizovalo funkci logického součinu (viz. Obr. 14.2). Standard protokolu CAN definuje dvě vzájemně komplementární hodnoty bitů na sběrnici - dominant a recessive. Jedná se v podstatě o jakýsi zobecnělý ekvivalent logických úrovní, jejichž hodnoty nejsou určeny a skutečná reprezentace záleží na konkrétní realizaci fyzické vrstvy.

Pravidla pro stav na sběrnici jsou jednoduchá a jednoznačná. Vysílají-li všechna zařízení na sběrnici recessive bit, pak na sběrnici je úroveň recessive. Vysílá-li alespoň jedno zařízení dominant bit, je na sběrnici úroveň dominant. Příkladem může být optické vlákno, kde stavu dominant bude odpovídat stav svítí a stavu recessive stav nesvítí. Jiným příkladem může být sběrnice buzená hradly s otevřeným kolektorem, kde stavu dominant bude odpovídat logická nula na sběrnici a stavu recessive logická jednička (viz. Obr. 14.3). Pak, je-li jeden tranzistor sepnut, je na sběrnici úroveň logické nuly (dominant) a nezáleží již na stavech ostatních zařízení. Pokud není sepnut žádný tranzistor, je na sběrnici úroveň logické jedničky (recessive).

Je použito kódování NRZ – Non-Return-To-Zero- kódování bitů. To znamená, že signál vysílaný na sběrnici je během bitového intervalu konstantní.

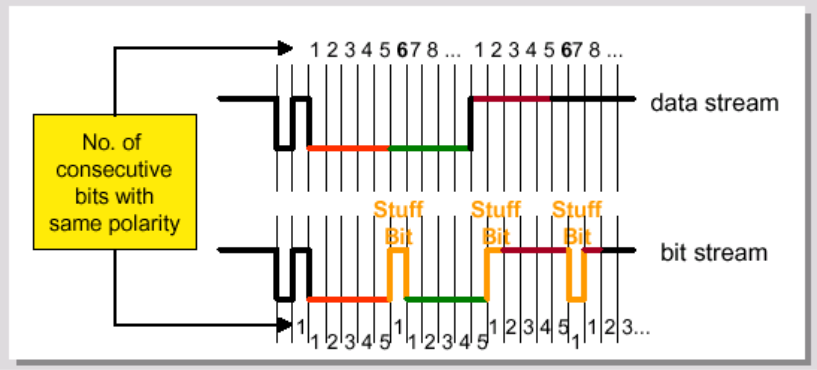


Obr. 14.2: Stavy na sběrnici.



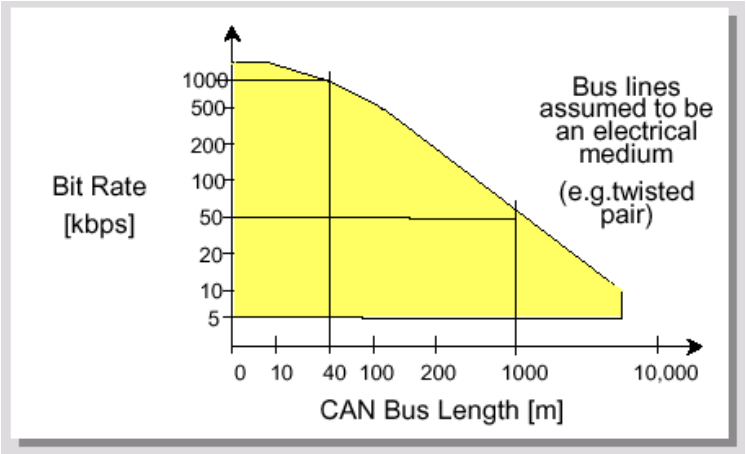
Obr. 14.3: Příklad realizace logického součinu na elektrickém médiu.

Fyzická vrstva provádí rovněž vkládání bitů - Bit Stuffing, čímž zajišťuje dostatečný počet R→D hran pro synchronizaci přijímače a vysílače. Stuff bit je vložen za 5 po sobě jdoucích bitů stejné úrovně a je inverzí předchozího bitu (viz. Obr. 14.4). Na přijímací straně jsou tyto vložené bity zase odebírány.



Obr. 14.4: Princip vkládání bitů.

Přenosová rychlost u sběrnice CAN je maximálně 1Mb/s při délce max. 40m. Při délce 1000m však dosahuje už jen 50kb/s. Pro větší délku musí být použito speciálních budičů sběrnice (viz. Obr. 14.5).



Obr. 14.5: Vztah mezi přenosovou rychlostí a délkou sběrnice.

Pro realizaci fyzického přenosového média se nejčastěji používá diferenciální sběrnice definovaná podle normy ISO 11898. Tato norma definuje jednak elektrické vlastnosti vysílacího budiče i přijímače, jednak principy časování, synchronizace a kódování jednotlivých bitů. Sběrnici tvoří dva vodiče (označované CAN_H a CAN_L), kde úroveň dominant nebo recessive na sběrnici je definována rozdílem napětí těchto dvou vodičů:

Úroveň Recessive – CAN_H i CAN_L jsou zhruba 2,5V, takže rozdílové napětí je 0V.

Úroveň Dominant – rozdílové napětí je kolem 2V.

Pro eliminaci odrazů na vedení je sběrnice na obou koncích přizpůsobena terminátory o hodnotě 120 Ω.

14.1.3 Linková vrstva

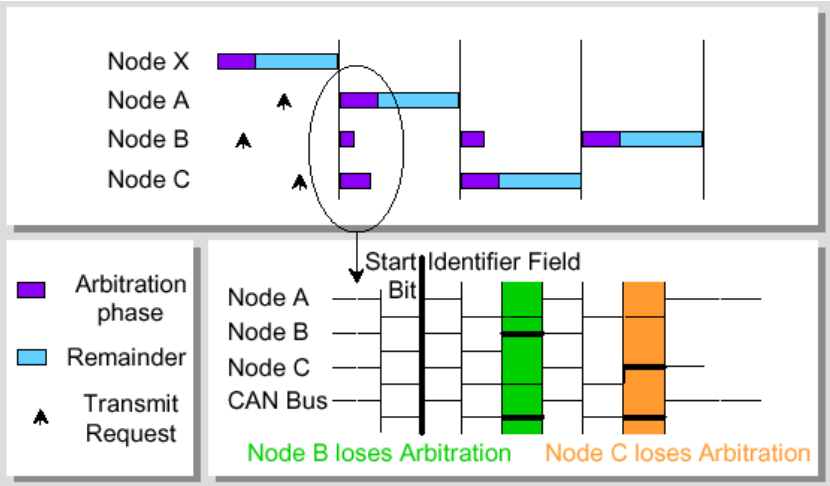
Linková vrstva protokolu CAN je tvořena dvěma podvrstvami označovanými jako MAC a LLC:

- Podvrstva MAC (Medium Access Control) zabezpečuje přístup k médiu s úkolem kódovat data, vkládat doplňkové bity do komunikace (stuffing/destuffing), řídit přístup všech zařízení k médiu s rozlišením priorit zpráv, detekce chyb a jejich hlášení a potvrzování správně přijatých zpráv
- Podvrstva LLC (Logical Link Control) má za úkol filtrovat přijaté zpráv (acceptance filtering) a hlášení o přetížení (overload notification).

Přístup ke komunikačnímu médiu

Je použita metoda CSMA/CD w/ AMP - Carrier Sense Multiple/Collision Detection Access with Arbitration on Message Priority. Jedná se o metodu náhodného přístupu ke komunikačnímu médiu, kde všechna zařízení mohou kdykoliv začít vysílat svá data, ale vzniku případných kolizí je zde zabráněno pomocí systému priorit zpráv.

Princip přístupu ke komunikačnímu médiu popisuje obrázek 14.6.



Obr. 14.6: Přístup ke komunikačnímu médiu.

Je-li sběrnice volná (stav bus free), může zahájit vysílání kterékoliv z připojených zařízení. Zahájí-li některé zařízení vysílání dříve než ostatní, získává sběrnici pro sebe a ostatní zařízení mohou začít vysílat až poté, co odešle kompletní zprávu. Jedinou výjimku zde tvoří chybové rámce, které může vyslat libovolné zařízení, detekuje-li chybu v právě přenášené zprávě. Začne-li vysílat několik zařízení současně, pak přístup na sběrnici získá zařízení přenášející zprávu s vyšší prioritou (nižším identifikátorem uváděným na začátku zprávy). Každý vysílač porovnává hodnotu právě vysílaného bitu s hodnotou na sběrnici a zjistí-li, že na sběrnici je jiná hodnota, než vysílá (jedinou možností je, že vysílač vysílá recessive bit a na sběrnici je úroveň dominant), okamžitě další vysílání přeruší. Tím je zajištěno, že zpráva s vyšší prioritou bude odeslána přednostně a nedojde k jejímu poškození, což by mělo za následek opakování zprávy a zbytečné prodloužení doby potřebné k přenosu zprávy. Zařízení, které při kolizi nezískalo přístup na sběrnici, může vyslat opět zprávu, jestliže se sběrnice uvolní (bus free).

Typy a formáty rámců

CAN používá 5 typů rámců:

Datový rámec.

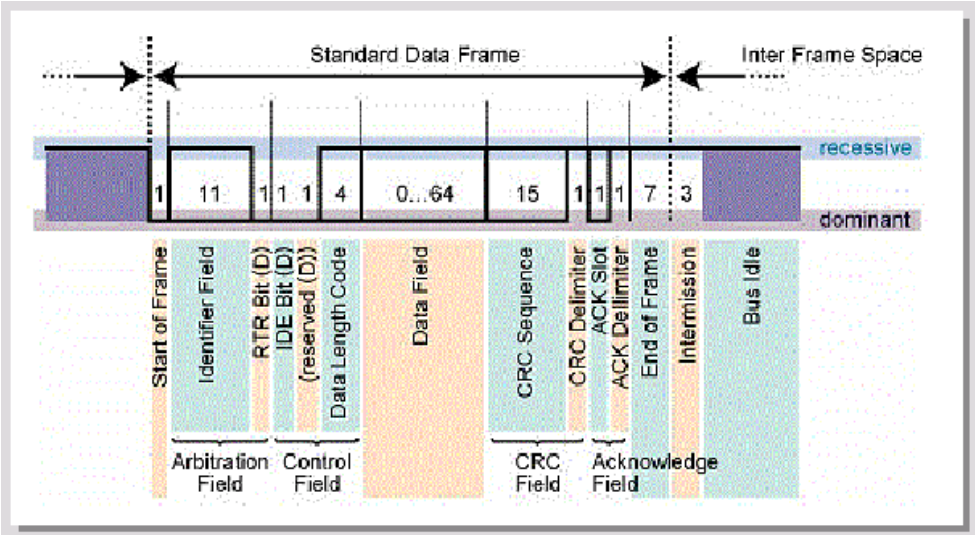
Vzdálený rámec.

Chybový rámec.

Rámec přetížení.

Mezera mezi rámci.

Základním rámcem je datový rámec. Je zobrazen na obrázku 14.7.



Obr. 14.7: Datový rámec.

Datový rámec je generován CAN nodem, pokud node chce vyslat data. Začíná Start of Frame bitem, který je dominant.

Následuje Arbitration Field, které obsahuje 12b – 11b Identifikátor a 1b Remote Transmission Request (RTR). RTR se používá k rozlišení datového (RTR=dominant) a vzdáleného rámce (RTR=recessive).

Následuje Control Field – 6b. První bit – IDE (Identifier Extension) – standard frame = dominant. Další bit je chráněný a je dominant. Následují 4b udávající délku zprávy (DLC) – počet bytů ve zprávě (0-8B).

Datové pole – 0 – 8B.

Cyclic Redundancy Code Field (CRC) – 16b = 15b kontrolního součtu a 1b CRC delimiteru.

Acknowledge field se skládá z ACK slot a ACK dellimiter. Během ACK slotu node posílá recessive a nody, které detekují chybu vyšlou dominant. ACK Dellimiter – recessive, nesmí být přepsán hodnotou dominant.

7 recessive bitů = End of Frame.

Metody detekce chyb

CAN používá 5 detekce chyb:

Cyclic Redundancy Check Error – chyba kontrolního součtu.

Form Error – chyba formátu.

Stuff Error – chyba vkládání bitů.

ACK Error – chyba potvrzení.

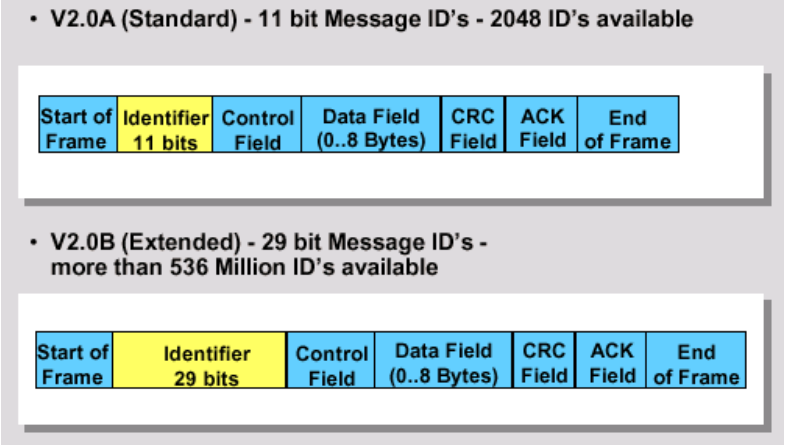
Bit Error – chyba stavu bitu.

Pro představu o spolehlivosti detekce chyb CAN protokolu si lze představit následující příklad:

Vozidlo vybavené sběrnici CAN je v provozu 2000h za rok. CAN je provozován s rychlostí 500kb/s a zatížení sběrnice je 25%. Při těchto podmínkách nastane 1 nedetekovaná chyba každých 1000 let.

Verze sběrnice CAN

Základní verzí sběrnice CAN je verze V2.0A. Existuje však i rozšířená verze, u které je modifikován formát zprávy tím, že je rozšířeno pole identifikátoru – z 11b. na 29b. To umožňuje použít podstatně vyšší počet zpráv (viz. obr. 14.8).



Obr. 14.8: Verze sběrnice CAN.

14.2 Ethernet

Historie Ethernetu začíná na Hawaii. Tehdy na tamní univerzitě pracovali na rádiové síti ALOHA pro propojení ostrovů, která je prapředkem všech sítí se sdíleným médiem.

Ethernet jako takový byl vyvíjen již začátkem 70-tých let pracovníky vývojových laboratoří firmy Xerox. Ve výzkumném středisku PARC (Palo Alto Research Center) vědci pracovali na propojení pracovních stanic Alto, které byly ve středisku také vyvíjeny.

První verze Ethernetu, koncepce pana Metcalfa a jeho spolupracovníků, pracovala s přenosovou rychlostí 2,94 Mb/s, používala koaxiální kabel o impedanci 70 ohmů dlouhý až 1 km a od pozdějších verzí Ethernetu se v několika aspektech lišila. Důležité však bylo, že již v polovině sedmdesátých let, kdy tato ranná verze vznikla, se ukázala být velmi dobře funkční. Dokonce natolik, že přilákala pozornost dalších dvou firem, které se kolem roku 1979 zapojily do vývojových prací. Byly to firmy DEC a Intel. Nová vylepšená verze, která vznikla díky jejich úsilí v roce 1980 byla označována jako DIX (**D**EC, **I**ntel, **X**erox) Ethernet a pracovala na rychlosti 10Mb/s.

Další vývoj Ethernetu byl podpořen tím, že firmy DEC, Intel a Xerox se rozhodly neponechat si Ethernet pouze jako své vlastní řešení, ale naopak předaly jeho specifikace a nechaly jej standardizovat. Jako standardizační orgán, který se staral o další vývoj Ethernetu, byl vcelku jednoznačně zvolen institut IEEE (Institute of Electrical and Electronic Engineers). Firmy DEC, Intel a Xerox předložily návrh specifikací Ethernetu pracovní skupině IEEE 802 společnosti IEEE (konkrétně podskupině 802.3, které byl standart Ethernet přidělen). Reakce IEEE na předložený návrh byla pozitivní a předložené specifikace se posléze staly standardem IEEE - bohužel s jistými drobnými věcnými změnami, které nebyly tak úplně zanedbatelné a které odrážely poněkud odlišné představy a postoje lidí podílejících se na standardizaci Ethernetu v rámci IEEE. Původní autoři DIX Ethernetu tyto odlišnosti do značné míry zapracovali do další verze DIX Ethernetu označované jako Ethernet II. Touto úpravou ovšem původní vývojová větev Ethernetu skončila a DIX Ethernet se již dále nevyvíjel.

První standard Ethernetu vypracovaný organizací IEEE byl publikován v roce 1985 pod označením „IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification“. Samotné jméno „Ethernet“ si jako svou značku zaregistrovala firma Xerox. Z těchto příčin verze Ethernetu pocházející od IEEE formálně vůbec nenesou jméno Ethernet.

Další vývoj Ethernetu se již odehrával výhradně v institutu IEEE. Ovšem ne tak, že by vlastní technická řešení byla vyvíjena právě na půdě IEEE - nová technická řešení byla vyvíjena nejrůznějšími institucemi a firmami, které to považovaly za účelné. Jakmile ale jejich řešení dospělo do stupně schopného standardizace, přicházeli s ním na půdu IEEE a zde se rozhodovalo o akceptování a následné standardizaci. Tímto způsobem postupně vznikla celá řada standardů Ethernetu majících různé parametry, různá přenosová média a různé způsoby provozu.

Kromě organizace IEEE se normalizací datové komunikace zabývá i mezinárodní normalizační úřad ISO (International Standards Organization), který vypracoval takzvaný referenční model OSI (Reference Model for Open System Interconection). Jeho podstata je založena na rozdělení všech činností při výměně dat do sedmi částí - vrstev realizovaných hardwarovými nebo softwarovými prostředky. Každá z vrstev zajišťuje funkce pro vyšší vrstvu a využívá služeb nižší vrstvy. Mezi jednotlivými vrstvami jsou (formou standardů a doporučení) definována rozhraní (protokoly) a mezi prvky stejné vrstvy jsou definována pravidla komunikace (vrstvové protokoly). Dnes je Ethernet standardizován i normou ISO 8802/3.

14.2.1 Popis sítě Ethernet a Industrial Ethernet

V dnešní době je Ethernet rozšířen do tisíců aplikací, s použitím široké škály přenosových protokolů. Rozšiřování technologie Ethernet do řídicích systémů má několik příčin. Především je úzce spjata s nejdůležitějším informačním prostředkem současnosti – sítí Internet. Tím enormně narostl počet ethernetových rozhraní a následně výrazně poklesla jejich cena.

Ethernet lze z hlediska determinističnosti rozdělit na komerční a průmyslový. Klasický komerční Ethernet používá nedeterministické metody přístupu na médium a proto ho nelze použít v průmyslových aplikacích. Z tohoto důvodu vznikly modifikované protokoly a jistá opatření vedoucí ke zvýšení determinističnosti této sítě a tím i k možnosti použití v distribuovaných systémech. Síť s těmito modifikacemi se pak nazývá průmyslový (Industrial) Ethernet.

Zde je výčet několika opatření, které vedly ke zlepšení odezvy:

- Strukturovanost topologie, kde namísto sběrnice s rozsáhlou doménou dochází v současných i kancelářských aplikacích k rozbití rozsáhlé domény použitím rozdělovačů (hub) a přepínačů (switch) na Krátké domény s mnohem menší pravděpodobností kolizí, plynoucích z přístupové metody CSMA/CD.
- Vysokou a stále rostoucí rychlostí ethernetových budičů a tím zkrácením doby odezvy na komunikačním kanálu.
- Směrováním zpráv jen do segmentů, pro které jsou tyto zprávy určeny, čímž opět dochází ke zmenšení pravděpodobnosti kolize.
- Segmentací zpráv, pomocí níž jsou odděleny zprávy RT od zpráv non-RT a topologií je zaručeno oddělení segmentů RT a non-RT.
- Aplikací prioritních slotů do metody CSMA/CD (Carrier Sense Multiple Access/Collision Detection), které zaručí definovaný okamžik přístupu k médiu.
- Použití UDP (User Datagram Protocol) místo protokolu TCP (Transmission Control Protocol) pro časově kritické úlohy (tato nespojovaná služba umožňuje poslání zprávy již v dalším taktu po odhalení kolize).
- Použití PTP (Precission Time Protocol) podle standardu IEEE 1588.

Precission time protocol je zatím nejúčinnější a levný mechanismus pro zaručení synchronismu zpráv v komunikaci Ethernet staví na distribuovaných hodinách reálného času v každé komunikující entitě. Dosahuje se tak vyššího stupně synchronizace zpráv (menší jitter) než u stávajících průmyslových sběrnic, které jsou takto navrhovány ze své podstaty.

14.2.2 Rozdělení Ethernetu podle modelu ISO/OSI

Pro toto rozdělení je vhodné si ve stručnosti připomenout základní strukturu modelu ISO/OSI. Model ISO/OSI je referenční komunikační model označený zkratkou slovního spojení "International Standards Organization / Open System Interconnection" (Mezinárodní organizace pro normalizaci / propojení otevřených systémů). Jedná se o doporučený model definovaný organizací ISO v roce 1983, který rozděluje vzájemnou komunikaci mezi počítači do sedmi souvisejících vrstev. Ethernet je definován na dvou vrstvách tohoto modelu, fyzické a linkové.

Fyzická vrstva (Physical Layer)

Úkol této vrstvy se zdá být velmi jednoduchý - zajistit přenos jednotlivých bitů mezi příjemcem a odesilatelem prostřednictvím fyzického média (metalický nebo optický kabel), který tato vrstva bezprostředně ovládá. Navíc je ovšem třeba vyřešit otázky převážně technického charakteru - např. jakou úrovní napětí bude reprezentována logická jednička a jakou logická nula, jaká může být maximální frekvence bitů, kolik kontaktů a jaký tvar mají mít konektory kabelů, jaké signály jsou přenášeny, jaký je jejich význam, časový průběh apod. Problematika fyzické vrstvy proto spadá spíše do působnosti elektroinženýrů a techniků.

Ethernet se podle této vrstvy dá rozdělit do několika skupin podle rychlosti a použitého propojovacího média.

Ethernet s přenosovou kapacitou 10 Mbit/s:

10Base-2

- Používá jako přenosové médium dvakrát stíněný koaxiální kabel označovaný jako Thin Ethernet (v jednodušší verzi Cheapernet) s impedancí 50 ohm.
- Délka segmentu kabelu může být maximálně 185 m (i když existují i varianty karet umožňující délku až 300 m).
- Na jednom segmentu může být maximálně 25 stanic.
- Segment musí být na obou koncích ukončen pomocí tzv. terminátorů .

10Base-5

- Používá jako přenosové médium pětkrát stíněný koaxiální kabel ozn. jako Thick Ethernet neboli Yellow Cable s impedancí 50 ohm.
- Délka segmentu může být maximálně 500 m; na kabel jsou připevňovány transceivery, stanice může být max. 50 m od transceiveru.
- Transceivery musí být připevňovány ve vzdálenostech násobku 2,5 m (na kabelech bývá označení)
- Segment musí být na obou koncích ukončen pomocí tzv. terminátorů.

10Base-T

- Používá jako přenosové médium kroucený dvoudrát (stíněný nebo nestíněný) s impedancí 100 ohm (min. Cat 3).
- Délka kabelu mezi uzlem a aktivním prvkem může být max. 100 m.

10Base-FL

- Používá jako přenosové médium multimodový optický kabel.

- Délka kabelu mezi uzly může být max. 2 km.
- Existuje i modifikace používající singlemodový optický kabel.

Fast Ethernet s přenosovou kapacitou 100 Mbit/s:

100Base-TX

- Používá jako přenosové médium kroucený dvoudrát (stíněný nebo nestíněný) s impedancí 100 ohm (min. Cat 5).
- Délka kabelu mezi uzlem a aktivním prvkem může být max. 100 m.

100Base-T4

- Používá jako přenosové médium kroucený dvoudrát (stíněný nebo nestíněný) s impedancí 100 ohm (min. Cat 3).
- Délka kabelu mezi uzlem a aktivním prvkem může být max. 100 m.
- Používá všechny 4 páry kabelu.
- Technologie není příliš rozšířena.

100Base-FX

- Používá jako přenosové médium multimodový optický kabel
- Délka kabelu mezi uzly může být v případě plně duplexního provozu max. 2 km; v příp. poloviční duplexu je vzdálenost ovlivněna zapojením sítě
- Existuje i modifikace používající singlemodový optický kabel

Gigabit Ethernet s přenosovou kapacitou 1000 Mbit/s:

1000Base-SX

- Používá jako přenosové médium multimodový optický kabel.
- Délka kabelu mezi uzlem a aktivním prvkem je ovlivněna parametry kabelu.

1000Base-LX

- Používá jako přenosové médium multimodový nebo singlemodový optický kabel.
- Délka kabelu mezi uzlem a aktivním prvkem je ovlivněna typem a parametry kabelu .

14.2.3 Linková vrstva (Data Link Layer)

Fyzická vrstva poskytuje jako své služby prostředky pro přenos jednotlivých bitů. Bezprostředně vyšší linková vrstva (někdy nazývaná též spojová vrstva či vrstva datového spoje) pak má za úkol zajistit pomocí těchto služeb bezchybný přenos celých bloků dat (velikosti řádově stovek bytů), označovaných jako rámce (frames). Jelikož fyzická vrstva nijak neinterpretuje jednotlivé přenášené bity, je na linkové vrstvě, aby správně rozpoznala začátek a konec rámce, i jeho jednotlivé části.

Na přenosové cestě může docházet k nejrůznějším poruchám a rušení, v jejichž důsledku jsou přijaty jiné hodnoty bitů, než jaké byly původně vyslány. Jelikož fyzická vrstva se nezabývá významem jednotlivých bitů, rozpozná tento druh chyb až linková vrstva. Ta kontroluje celé rámce, zda byly přeneseny správně. Odesilateli potvrzuje přijetí bezchybně přenesených rámců, zatímco v případě poškozených rámců si vyžádá jejich opětovné vyslání.

Popis linkové vrstvy Ethernetu - metoda přístupu k médiu CSMA/CD:

Protokol IEEE 802.3 používá metodu přístupu CSMA/CD (Carrier Sense Multiple Access/ Collision Detection), což znamená metoda mnohonásobného přístupu k médiu s nasloucháním nosné a detekce kolizí. Tato metoda sleduje před vysíláním provoz na médiu, čímž je zajištěno, že vysílač nebude vysílat dřív, než je médium volné. Ovšem i při tomto opatření může díky zpoždění na lince dojít ke kolizi dvou vysílačů a jejich vyslané rámce se vzájemně znehodnotí. V takovém případě stanice, která kolizi detekuje, vyšle krátký signál jam (32 bitů), který informuje ostatní stanice o tom, že došlo ke kolizi. Po té se stanice odmlčí a pokusí se o nové vysílání.

Mezi pokusy o vysílání čeká stanice vždy náhodnou dobu. Čekací interval, který si stanice zvolila při prvním pokusu, se po každém dalším pokusu zdvojnásobuje. Stanice tak omezuje množství pokusů v čase a zároveň zvyšuje pravděpodobnost, že se o médium s ostatními stanicemi podělí. Pokud se vyslání rámce nezdaří ani po šestnácti pokusech, pak linková vrstva pošle vyšší vrstvě zprávu o neúspěchu.

Kolize může nastat jen v době od počátku vysílání do doby, než obsadí celou délku komunikačního média (pak už ostatní stanice zjistí, že médium je obsazené a své pokusy o vysílání odloží). Tomuto intervalu se říká kolizní okénko a musí být kratší než nejkratší vyslaný rámec, jinak by docházelo k nedetekovaným kolizím (dvě vzdálené stanice by vyslaly krátké rámce, které by se protnuly a zkomolily, ale obě stanice by ukončily vysílání dřív než by byla zjištěna kolize).

Tato metoda je efektivní při nižším počtu účastníků na médiu, je to asi 30% šířky pásma. Její efektivita klesá při zvyšování počtu požadavku o vysílání, kdy pravděpodobnost kolizí začne exponenciálně růst. Efektivita CSMA/CD přístupu je účinnější pro delší rámce, protože při jejich přenosu je lepší poměr trvání kolizního okénka vůči vysílanému rámci.

14.2.4 Rámec protokolu 802.3

Tab. 14.1: Rámec protokolu Ethernet.

Počet oktetů:	1	6	6	2	46-1500	4
Význam:	Preambule (Synchronizace)	Cílová adresa	Zdrojová adresa	Délka datového pole	Datové pole	Kontrolní součet

Popis prvků rámce Ethernet:

- **Preambule** se skládá ze střídajících se jedniček a nul, přičemž poslední z oktetů má tvar 10101011 a označuje začátek vlastního rámce. U 802.3 se uvádí jako preambule jen prvních sedm oktetů rámce a osmý je tzv. omezovač počátku rámce (starting frame delimiter - SFD).
- **Cílová i zdrojová adresa** jsou fyzické (MAC) adresy o délce 16 nebo 48 bitů (záleží na konkrétní aplikaci, avšak typ MAC adresy musí být stejného typu pro celý segment LAN), kde zdrojová je vždy unikátní a cílová adresa muže být individuální (unicast), skupinová (multicast) nebo všeobecná (broadcast).
- **Délka datového pole** určuje počet oktetů datového pole.
- **Datové pole** se skládá minimálně ze 46 oktetů a maximální velikost je omezena na 1500 oktetů.
- **Kontrolní součet** (frame check sequence) je třicetidvou bitový cyklický kontrolní kód, který se počítá přes všechna pole kromě preambule.

14.3 PROFINET

Jedním z aktuálních požadavků současné automatizace je zajištění homogenní komunikace na všech úrovních podnikové automatizace, tzn. z úrovně akčních členů a přístrojové techniky v technologickém provozu do úrovně řízení (PLC) a výše do nadřazených stupňů zpracování a vyhodnocování technologických dat. Standardizované propojení, jednotná správa sítě, osvědčené techniky ze světa IT, komplexní diagnostika, to vše může znamenat úsporu jak ve fázích projektování, tak i ve fázi uvádění do provozu či při běžném provozu. Výhody poskytující průmyslové sběrnicové systémy (tzv. fieldbuses) a na druhé straně standardizované funkce ze světa IT založené na Ethernetu by měly být využívány současně pomocí jednotné komunikační architektury. Proto byl mezinárodní organizací Profibus International definován rozsáhlý standard Profinet vycházející ze standardu Ethernet, který provozním jednotkám umístěným přímo v technologii otevírá zcela nové možnosti, kterými jsou:

- Plná integrace s IT na vyšších úrovních.
- Distribuovaná automatizace..
- Využití bezdrátových sítí v průmyslu.
- Zajištění odezvy v reálném čase.

Profinet je tedy otevřený, na výrobci nezávislý komunikační standart pro všechny úrovně průmyslové automatizace, založený na průmyslovém ethernetu. Nabízí integraci existujících průmyslových sítí, jako například Profibus bez nutnosti modifikace stávajícího zapojení. Tím je zajištěna ochrana všech stávajících investic. S Profinetem se používá pro všechny úrovně jen jedna komunikační sběrnice. To přispívá ke zjednodušení instalace, údržby i školení obsluhujících pracovníků.

Opětovné používání strojů či jejich částí jako inteligentních technologických modulů v koncepci automatizace založené na komponentech CBA (Component Based Automation) znamená úsporu při vývoji a oživování technologie. CBA je přímo založena na Profinet a proto je distribuovaná automatizace v takovémto pojetí velice zajímavá.

Pomocí webové integrace je možno dosáhnout nových možností co se týče parametrizace, diagnostiky a vizualizace.

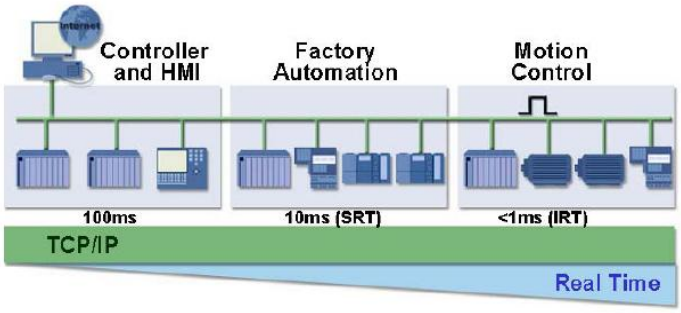
Pro vývoj aplikací se používá standardně software Step 7. Pro distribuovaná automatizační řešení dle koncepce CBA je k dispozici nový softwarový nástroj SIMATIC iMap – na výrobci nezávislý editor pro spojování komponent (technických modulů).

14.3.1 Popis protokolu

Profinet je nový protokol, který je součástí normy IEC 61158. Slouží ke komunikaci průmyslových automatizačních systémů koncepčně vycházející ze standartu Profibus, ale komunikující na základě standartu Ethernet. Výhodou řešení s tímto standardem je možnost použití jak metalického vedení, optických kabelů, tak především bezdrátových sítí WIFI, což může usnadnit řešení mnoha složitých situací při projektování rozvodů sítě. Profinet podporuje integraci jednoduchých distribuovaných periférií v časově kritických aplikacích na bázi ethernetové komunikace a zároveň umožňuje rozložit distribuovaný systém na logické subsystémy, které spolu komunikují přes definovaná rozhraní podle zásad component-based (komponentově založeného) distribuovaného systému.

Profinet používá odstupňovanou komunikační architekturu. Rozlišujeme v ní tři stupně komunikace: standardní komunikace (TCP/IP), komunikace pro reálný čas (RT) a komunikaci izochronního reálného času (IRT). Tyto tři stupně pokrývají široké spektrum požadavků dnešní automatizace. Klíčovým rysem komunikace Profinet je koexistence realtimové i klasické TCP komunikace.

Velkou výhodou použití Ethernetu jako základu pro komunikaci Profinet je možnost využití bezdrátových spojů WIFI podle standardu 802.11. Ty umožňují rádiové připojení automatů nebo periferií tam, kde je obtížné zavádět klasickou kroucenou dvoulinku nebo optický kabel.



Obr. 14.9: Rozdělení komunikačních kanálů pro různé časové odezvy.

14.3.2 Standardní TCP/IP komunikace

Jak bylo řečeno Profinet používá také standardní TCP/IP komunikaci pro výměnu časově nekritických dat. Z hlediska možnosti komunikace různorodých aplikací nestačí TCP/UDP kanál (4. úroveň modelu ISO/OSI) pro distribuované periferie v průmyslu. Faktem je, že TCP/IP uskutečňuje pouze základní komunikaci zařízení mezi lokální síti a distribuovanou síti. Pro komunikaci na tomto kanále jsou požadovány protokoly na vyšších úrovních než je TCP/UDP. Globální komunikace je zajištěna teprve pokud je u všech zařízení nastaven stejný protokol (SMTP, FTP, HTTP...).

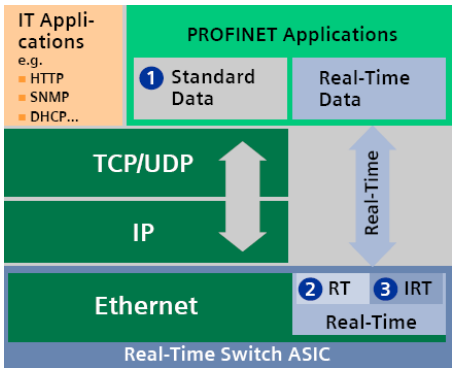
14.3.3 RT komunikace

V průmyslových aplikacích se požadují obnovovací časy dat v rozsahu 5 – 10 milisekund. Implementace RT komunikace může vést k výsledku jen tehdy, pokud procesor distribuovaného zařízení není zatížen samotnou komunikací, ale zpracovává aplikační software v potřebném čase – data by byla znehodnocena. Zkušenost ukazuje, že přenosový čas přes linku Fast Ethernet (100Mb/s) nebo vyšší je zanedbatelný v porovnání s časem zpracování v distribuované periferii. Čas nutný k vystavení dat v periferii není navyšován časem komunikace. Každé významnější zlepšení času vystavení by bylo možno dosáhnout optimalizací komunikační fronty modelu producent-konzument.

V zájmu uspokojení RT aplikací v automatizaci má Profinet vyhrazený časový kanál. Tento kanál je založen na standardu IEEE 802.1q, ve kterém jsou rámce opatřeny prioritou. Tuto prioritu jsou schopny zachytávat aktivní prvky sítě a potlačit rámce s nižší prioritou. Proces je srovnatelný s provozem na volné cestě, z níž levý pruh je vyhrazen pro časově kritický provoz a hlídá časově nekritický provoz před připojením do tohoto pruhu. Případné zablokování pravého pruhu (časově nekritický provoz) neovlivní provoz časově kritický. Doba odezvy zařízení na realtimové rámce může

kolísat, což je způsobeno dobou nutnou k dokončení přenosu předchozího rámce, který může mít různou délku i prioritu.

Toto řešení minimalizuje znatelně časy běhu v komunikační frontě a zvyšuje kvalitu obnovování dat. Eliminace některých vrstev modelu ISO/OSI snížily délku posílané zprávy a také dobu, za kterou je možno zprávu poslat. Díky tomu byly sníženy výpočetní nároky na procesor, co se týká komunikace.



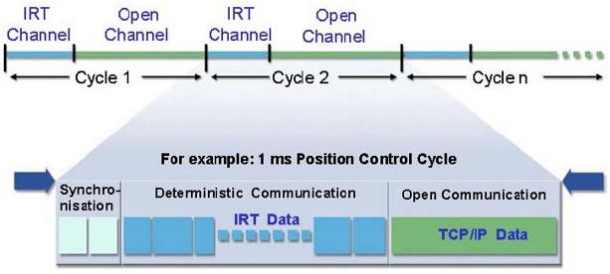
Obr. 14.10: Přemostění vrstev modelu ISO/OSI v RT komunikaci pro její urychlení.

Standardní priorita pro realtimová data je šest. To dovoluje zachytávání priorit jiných aplikací, jako např. internetová telefonie s prioritou pět.

14.3.4 Isochronní RealTime (IRT)

Výše uvedené řešení není dostačující pro kontrolu pohonů. Tyto aplikace vyžadují obnovit data do 1 milisekundy s přesností kolem 1 mikrosekundy až pro 100 pohonů. Pro dosažení těchto požadavků Profinet definoval komunikaci s časovými sloty na druhé úrovni modelu ISO/OSI. Aby tato striktně deterministická komunikace mohla probíhat, je nutno již při hardwarové konfiguraci těch zařízení, která budou v IRT módu komunikovat, tato zařízení zařadit do synchronizované skupiny, kde jedno zařízení bude synchronizaci řídit a ostatní se budou podle ní synchronizovat. Synchronizace probíhá zároveň na všech aktivních prvcích sítě (switche), protože je nutné zcela zastavit veškerý ostatní provoz na síti a umožnit přenos IRT rámců.

Implementace isochronního módu je na hardwarové úrovni. V komunikačním rozhraní je implementován obvod typu ASIC, který zajišťuje synchronizaci a rezervaci časového slotu pro časově kritická data. Hardwarová implementace umožňuje realizovat požadovanou kadenci dat podle jejich významnosti a také tím, že je oddělena od procesoru, který nezatěžuje. Ušetřený čas pak může být použit pro aplikaci běžící na dané periferii.[14-2, 14-3]



Obr. 14.11: Rozdělení komunikace Profinet na časové sloty.

Na obr. 14.11 je vidět jeden cyklus komunikace Profinet. V první fázi se synchronizují všechna zařízení, která se účastní IRT komunikace. V druhé fázi jsou těmto zařízením zaslány izochronní data. Když jsou data odeslána, je médium otevřeno pro otevřenou komunikaci RT a TCP/IP. Jak již bylo

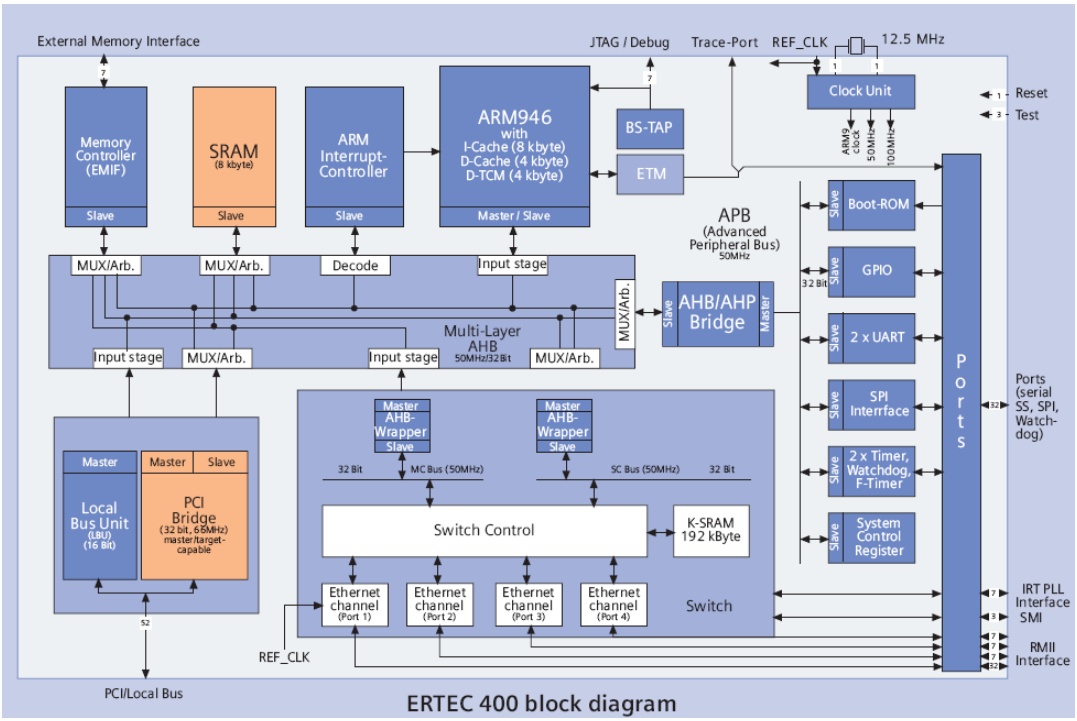
uvedeno, realtimové rámce se od běžných rámců TCP/IP komunikace odlišují prioritou podle standardu IEEE 802.1q.

14.3.5 Komunikační kontroléry ERTEC 200 a ERTEC 400

Rodina obvodů ERTEC zahrnuje všechny funkce nezbytné pro vysoce výkonná řešení v průmyslové automatizaci. Byly vyvinuty za účelem sloučení klasické komunikace používané v lokálních sítích, internetu a realtimové komunikace požadované na nižších vrstvách průmyslové automatizace.

ERTEC (Enhanced Real-Time Ethernet Controller) 200 je Ethernetový kontrolér s integrovaným dvouportovým switchem a 32-bitovým mikroprocesorem, který dovoluje zavést nové možnosti komunikace. Doba cyklu mikrokontroléru je 0,25ms a odchylka je menší než 1µs. Implementace obvodu ERTEC 200 eliminuje potřebu externích switchů za účelem propojování s dalšími zařízeními a tím snižuje cenu výsledného řešení komunikace. Jsou v něm sloučeny všechny funkce vyplývající z požadavků na realtimovou komunikaci, liniovou topologii a integraci do stávajících komunikačních systémů, pracujících na základě standartu Ethernet.

Zapojení ethernetového kontroléru obvykle vyžaduje propojení přes oddělovací moduly pro spojení interního a externího rozhraní, což u kontrolérů ERTEC není potřeba, protože ho mají integrovány v sobě. Díky integrovanému procesoru ARM 946 může být čip použit pro jednoduché aplikace se „systémem na čipu“. Cyklická výměna dat Profinet IO s realtimovým a izochronním módem, která je obvykle obsluhována procesorem, je v tomto případě vyřešena zcela hardwarově. Toto řešení umožňuje využití procesoru pro řízení jednoduchých periférií bez použití externího procesoru.



Obr. 14.12: Blokové schéma obvodu ERTEC 400.

ERTEC 400 – cílem tohoto čipu jsou automatizační zařízení jako např. kontroléry, řízení pohonů, PC stanic a síťových komponent. ERTEC 400 je ethernetový kontrolér se čtyř portovým switchem, 32-bitovým mikroprocesorem a PCI rozhraním (verze 2.2, takt sběrnice 66MHz). Čtyř portový switch umožňuje realizaci několika topologií – hvězda, strom a linie bez externích síťových komponent. Velký integrovaný komunikační buffer umožňuje zpracování velkého množství rámců tak, jak to požadují programovatelné automaty a kontroléry pohonů.

Integrované rozhraní PCI umožňuje použití pro kontrolní stanice na bázi PC, aniž by bylo nutno vytvářet na kartě rozhraní PCI most.

Integrovaný procesor ARM 946 obsluhuje veškeré komunikační rutiny. To zmenšuje zatížení procesoru pro řízení daného procesu. Veškerá cyklická komunikace je ošetřena hardwarově, stejně jako u čipu ERTEC 200, což zkracuje doby čtení a zápisu na porty periferie. [14-13, 14-17]

14.3.6 Rámec protokolu Profinet

Rámec protokolu se v základu skládá z částí popsaných v komunikaci standartu IEEE802.3, ale navíc se v něm vyskytují prvky pro zlepšení vlastností komunikace vzhledem k realtimovým aplikacím. Rámec byl rozšířen především o příznakové byty VLAN sítí standartu IEEE 802.1Q, které určují, zda se jedná o běžnou komunikaci nebo realtimovou komunikaci. Dále je každý rámec odlišen svým identifikačním číslem, což zaručuje, že komunikace bude spojitá.

Tab. 14.2: Rámec protokolu PROFINET.

Počet bytů:	7	1	6	6	2	2
Význam:	Preambule	Synchronizace	Cílová adresa	Zdrojová adresa	Ether-type I	VLAN

2	2	40 - 1440	2	1	1	4
Ether-type II	Frame ID	RT data	Čítač cyklů	Datový status byte	Transfer status byte	FCS

Tab. 14.2 Popis rámce komunikace Profinet IO - RT

Popis prvků rámce Profinet RT:

Preambule je sekvence synchronizačním bytů, která je převzata ze standartu IEE802.3, skládá se z posloupnosti bitů 10101010.

Synchronizace je byte pro ukončení synchronizace s posloupností bitů 10101011.

Cílová adresa obsahuje šest bytů, které slouží pro informaci, kam rámec směřuje. Tato adresa je specifická pro každé síťové zařízení a je přidělována výrobcem.

Zdrojová adresa obsahuje šest bytů, které slouží pro informaci o odesílateli daného rámce.

Ether-type I představuje část, která je specifikována ve standartu IEEE 802.1Q. Proto switche, které mají přenášet takto modifikované rámce musí podporovat standart IEEE 802.1Q. V opačném případě jsou všechny modifikované rámce zahozeny. Modifikace představuje dva byty, které slouží k rozlišení, zda se jedná o běžný rámec non-realtimové komunikace nebo rámec s VLAN doplňkem. RT rámec protokolu Profinet má hodnotu této části 8100_{hex}. Běžný rámec pro přenos non-RT dat (TCP/IP) má hodnotu této části 8000_{hex}.

VLAN část je složena také z dvou bytů, tedy z 16 bitů. První tři bity udávají prioritu rámce v osni úrovních 0-7, přičemž 7 je priorita nejvyšší. Čtvrtý bit je vždy nulový a zbývajících 12 bitů udává číslo virtuální sítě LAN, do které zařízení náleží.

Ether-type II část ukazuje, že se jedná o RT rámec Profinetu. Jeho hodnota je vždy 8892_{hex}.

Frame-ID je identifikační číslo rámce. Slouží k odlišení typů rámců.

RT-data představují datový obsah rámce s informacemi o nastavení DI/DO a AI/AO. V případě alarmového rámce obsahují data informace o chybě v systému.

Čítač cyklů se v průběhu komunikace inkrementuje o hodnotu, která představuje čas mezi dvěma rámci. Jedna jednotka čítače představuje hodnotu 31,25µs.

Datový status byte podává informace o posílaných datech:

- Bit - obvykle je v log. 1 a udává, kterou větví byly data transportovány v případě, že je komunikace zapojena redundantně.
- Bit – vždy log. 0.
- Bit – log. 1 udává, že data jsou v pořádku. Log. 0 se vyskytuje ve startovací fázi.
- Bit – vždy log. 0 .
- Bit – log. 1 udává, že proces, který data vygeneroval, stále běží (spíše informativní charakter).
- Bit – je indikátor případných závad. Log. 1 indikuje, že vše je v pořádku a log. 0, informuje o tom, že nastal problém a rámec obsahuje diagnostiku tohoto problému.
- Bit – vždy log. 0.
- Bit – vždy log. 0 .

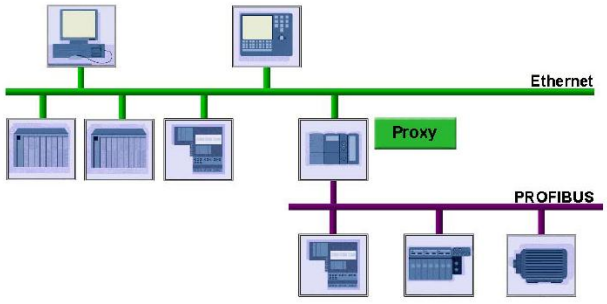
Transfer status byte je vždy nulový.

FCS je kontrolní součet rámce.

14.3.7 PROFINET IO

Profinet IO se používá pro přímé připojení distribuovaných periférií. Je zachován obvyklý model sítě, známý z Profibus DP, avšak zde mají všichni účastníci stejná práva, proto je uspořádání master-slave známé z Profibus DP přeneseno do modelu producent-konzument. Producent je vysílající stanice, která vysílá data bez čekání na výzvu od komunikačního partnera. Spotřebitel data zpracovává. Přiřazení mezi poskytovateli a spotřebiteli je definováno při hardwarové konfiguraci.

Instalace sítí Profinet jsou založeny na specifických požadavcích pro sítě Ethernet v průmyslovém prostředí. Klíčovým aspektem Profinetu je hladký přechod od stávající komunikační technologie jako je např. Profibus DP na ethernetově založenou komunikaci Profinet. Do sítě Profinet mohou být integrovány segmenty s jinými komunikačními protokoly, než je Profinet. Segment se pak považuje za nižší vrstvu a připojuje se pomocí tzv. proxy rozhraní.



Obr. 14.13: Připojení dalšího segmentu s jiným protokolem přes Proxy.

Typy zařízení Profinet:

- Vstupně/výstupní kontrolér (PLC), řídicí jednotka ve, které je prováděn program.
- Vstupně/výstupní zařízení (periferie), distribuovaná jednotka v technologickém provozu, přidělená řídicí jednotce I/O .

- Vstupně/výstupní Supervisor je programovací jednotka nebo PC s funkcemi pro uvádění do provozu a s diagnostickými funkcemi nebo jednotka pro operátorské rozhraní.

Supervisoři se používají k nastavování komunikace systému, diagnostiku chyb a spouštění komunikace. Mohou číst a zapisovat diagnostická data, která jsou asociována s komunikací Profinet IO nebo diagnostická data, poskytnutá aplikačními programy a zařízeními. Supervisoři mohou také číst a zapisovat konfigurační data užitím speciálních acyklických záznamů ve formě servisních objektů. Tento typ zařízení je možno používat pouze pro oživování komunikace Profinet IO, popřípadě jako zařízení HMI ke zobrazování diagnostických dat uživateli.

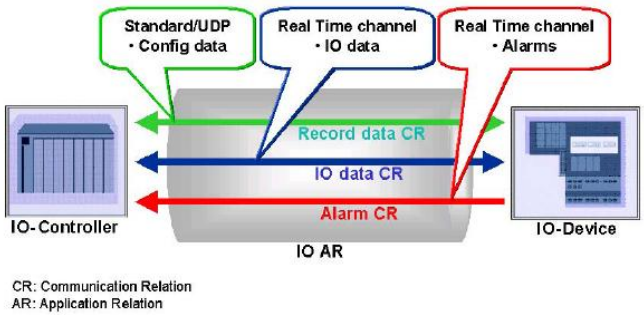
Profinet IO systém vyžaduje minimálně jeden vstupně/výstupní kontrolér a jedno vstupně výstupní zařízení. Systému může být konfigurován několika způsoby, více kontrolérů ovládá jedno periferní zařízení, jeden kontrolér ovládá více periferních zařízení nebo více kontrolérů ovládá více periferních zařízení.

Data mohou být přenášena těmito metodami:

- Cyklickým přenosem dat skrz RT kanál.
- Událostmi řízenými alarmy skrz RT kanál.
- Skrz nadefinované proměnné, které se čtou přes UDP kanál (Communication Relationships).

Pro nastartování relačních vazeb mezi PLC a periférií je nutno přenést konfiguraci. To se provádí standardně pomocí UDP kanálu. Konfigurace se přenáší ze vstupně/výstupního „supervisora“ užitím služeb objektového přenosu datového záznamu. Tyto služby dovolují externím zařízením číst a zapisovat data do diagnostické haldy Profinet IO nebo aplikace. Datový záznam je vždy přenášen acyklicky ve spojově orientovaném, potvrzovaném módu. Datový záznam obsahuje diagnostická data, záznamy alarmů, identifikační data zařízení a vstupně/výstupní datové objekty.

IP adresa zařízení Profinet IO je uložena v trvalé (flash) paměti zařízení. Může být modifikována „supervisorem“. Profinet IO zařízení může být nastaveno také na automatické přidělování IP adresy z DHCP serveru.



Obr. 14.14: Model komunikace Profinet mezi PLC a periferiemi.

PLC přenáší parametrizaci a konfigurační data přiřazené periférii přes "Record Data CR". Cyklická výměna dat se provádí přes "IO CR". Asynchronní události a alarmy jsou přenášeny přes "alarm CR" do kontroléru, kde jsou potvrzeny.

Alarmy Profinetu:

- Odpojení (uplug).
- Připojení (plugin).
- Diagnostika.
- Status.
- Update alarm.
- Alarmy specifikované výrobcem.

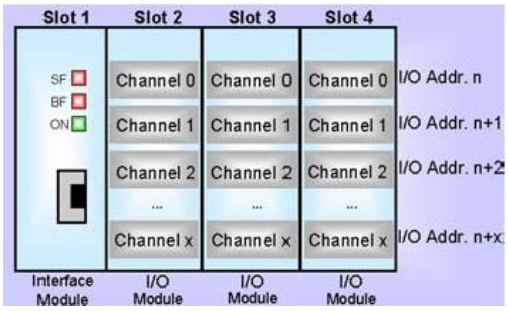
U všech těchto alarmů může být nastavena vysoká nebo nízká priorita.

Model distribuovaných periférií:

Pro Profinet IO je specifikován unifikovaný model Profinet IO-Device, který umožňuje konfiguraci modulárních i kompaktních jednotek. To je orientováno na charakteristiky Profibus DP, kde modulární jednotky obsahující sloty pro zasunutí různého množství IO modulů. Tyto moduly jsou zpraženy s IO kanály, které jsou obsluhovány procesními signály.

Toto schéma dovoluje existující distribuovanou síť Profibus DP zahrnout do sítě Profinet bez jakékoliv modifikace. To zajišťuje jistou ochranu investic do zařízení různých výrobců (zařízení budou moci být dále využívány ve stejném protokolu).

Každé vstupně/výstupní zařízení má přiřazeno globálně unikátní 32bitové identifikační číslo, které se skládá z 16bitů identifikace výrobce a 16 bitů identifikace zařízení.



Obr. 14.15: Modulové rozdělení periferie převzaté z Profibusu.

Diagnostika:

Profinet IO podporuje víceúrovňovou diagnostiku, která umožňuje bližší specifikaci poruchy. Pokud nastane chyba, IO zařízení generuje diagnostický alarm v komunikačním rozhraní. Tento alarm volá odpovídající programovou rutinu v PLC, která vyvolá reakci na vzniklou chybu. Pokud periferie hlásí, že je nutno ji celou vyměnit, je možno provést okamžitou výměnu.

Diagnostické zprávy jsou hierarchicky uspořádány:

- Číslo slotu.
- Číslo kanálu.
- Typ kanálu - vstup, výstup.
- Kód chyby.
- Další části specifikované výrobcem.

14.3.8 PROFINET CBA

Vývoj na poli automatizace se zaměřil na tvorbu modulárních linek a strojů. Tato struktura dala impuls dalšímu vývoji na úrovni distribuované automatizace. Profinet poskytuje dané řešení také z toho důvodu, aby se zabránilo nižšímu dělení na úrovni technologických modulů (distribuovaných periférií).

Distribuovaná automatizace:

Profinet komponentový model popisuje autonomní moduly strojů a zařízení jako technologické moduly. Distribuovaný automatizační systém vytvořený na bázi technologických modulů zjednodušuje modulární návrh systémů a značně usnadňuje opakované použití některých již vytvořených součástí. To výrazně snižuje pracnost návrhu. Využívá přitom objektově orientovaného přístupu k prvkům sítě.

Profinet na bázi komponentového modelu je popsán pomocí PLD (Profinet Component Description). Tyto soubory jsou na bázi XML a může být vytvořen buďto konfiguračním SW výrobce nebo Profinet komponent editorem. Navrhování distribuovaných sítí se dělí na programování kontrolní logiky jednotlivých PLC v síti (pomocí SW výrobce) a na konfiguraci všech zařízení jako celku, která definuje komunikační pravidla mezi technologickými modely.

Profinet CBA komponenty:

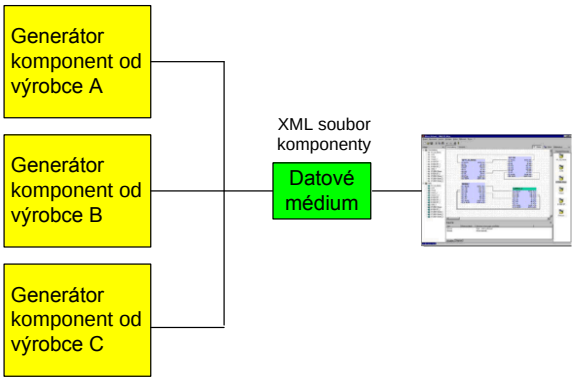
Reprezentantem technologického modulu z klasického navrhování distribuovaných systémů je v Profinetu tzv. Profinet komponent. Každý Profinet komponent má svoje rozhraní, které obsahuje technologické proměnné a ty jsou pak sdíleny mezi jednotlivými komponenty.

Profinet komponenty jsou modelovány se standardizovanou technologii COM. COM je další prostředí na bázi objektově orientovaného programování a dovoluje navrhování aplikací z vyhotovených komponentů, které jsou připraveny na svou modifikaci. Výhoda komponent spočívá v možnosti jejich propojování mezi sebou grafickou formou. Komponenty mohou být flexibilně kombinovány jako stavebnice z bloků. Tyto bloky mohou být několikrát použity bez ohledu na jejich vnitřní implementaci. Mechanismus pro přístup k rozhraní komponent je jednotně definován v Profinetu.

Při pohledu na obecnost modulů je důležité uvážit jejich možnost znovupoužití v různorodých systémech. Podstatou koncepce je co nejflexibilnější používání modulů v zájmu dosažení konečného systému v co nejkratší době. Na jednu stranu jemnější struktura tvoří model, který je komplexnější, ale to vyúsťuje ve vyšší nároky na znalosti programátora. Na druhou stranu hrubší struktura zužuje možnosti pro znovupoužití modulu a to vyúsťuje ve vyšších nárocích na implementaci. Softwarové komponenty vytvářejí výrobci linek nebo strojů. Komponentová struktura má hlavní dopad na snižování nároků na hardware a na návrh aplikace.

Tvorba komponent:

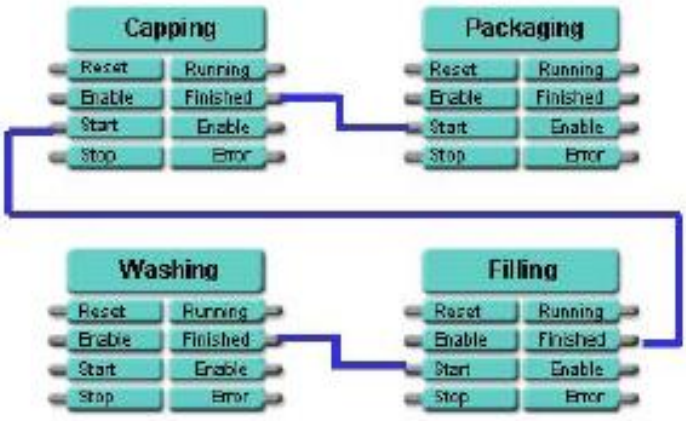
Komponenty se vytvářejí jako obrazy technologických modulů, které programátor zapojuje fyzicky na síť Profinet. Programování a konfigurace periférií je implementována jako dříve v nástrojích výrobce. To umožňuje využití stávajících programů a programátorských zvyklostí. Ve finále se aplikační software uloží ve formě Profinet komponentu (soubory formátu PCD - Profinet Component Description). Tento komponent se pak importuje do connection editoru.



Obr. 14.16: Nezávislost obsahu komponenty na výrobci.

Spojování komponent:

Provádí se pomocí Profinet Connection Editor, což je v našem případě nástroj firmy Siemens, SIMATIC iMap. Vytvořené komponenty se natáhnou z knihovny a jednoduchými tahy myší se provedou potřebná spojení. Tato spojení na grafické úrovni nahrazují předchozí nepřehledné programování komunikačních vazeb. Programování vyžaduje dobrou znalost komunikačních vazeb a programových sekvencí daného zařízení. Při programování musí být jasno, které zařízení se kterým bude komunikovat a skrz jaké médium bude komunikovat. Není nutno znát funkce pro konfiguraci komunikace, protože ty se konfigurují automaticky. Connection editor pak podle nákresu distribuuje definované proměnné skrz síť.



Obr. 14.17: Ukázka propojených komponent

Download:

Po spojení všech komponent jsou informace o spojení, kód a konfigurační data komponent natažena do všech zařízení na lince Profinet. Tak zná každé zařízení své partnery na síti a data, která má s kým sdílet.

PCD - component description:

Profinet component description je soubor, který je napsán v jazyce XML. Je tvořen nástroji od výrobce, které v sobě zahrnují generátor komponent. Alternativně se komponenty mohou vytvářet pomocí univerzálního generátoru.

Soubor PCD obsahuje informace o funkci a o objektech náležících Profinet komponentům, což zahrnuje:

- Popis komponent jako knihovních prvků, ID komponentu a jméno.
- Popis hardware: IP adresu, přístup k diagnostickým datům, informace o propojení.
- Popis funkčnosti softwaru: propojení software s hardwarem, rozhraní komponent, vlastnosti proměnných, jejich jména, datový typ a směr komunikace vstup/výstup.
- Buffer pro objekt komponenty.

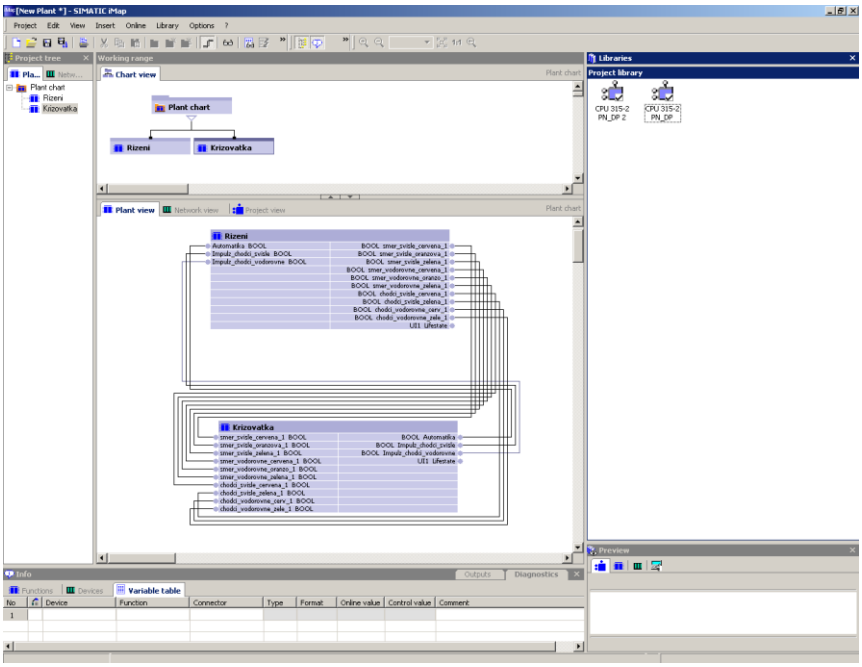
Connection Editor:

Je to nástroj pro spojování komponent natažených z knihoven. Jak bylo již výše uvedeno, jedná se o software iMap.

Tento editor nabízí dva pohledy:

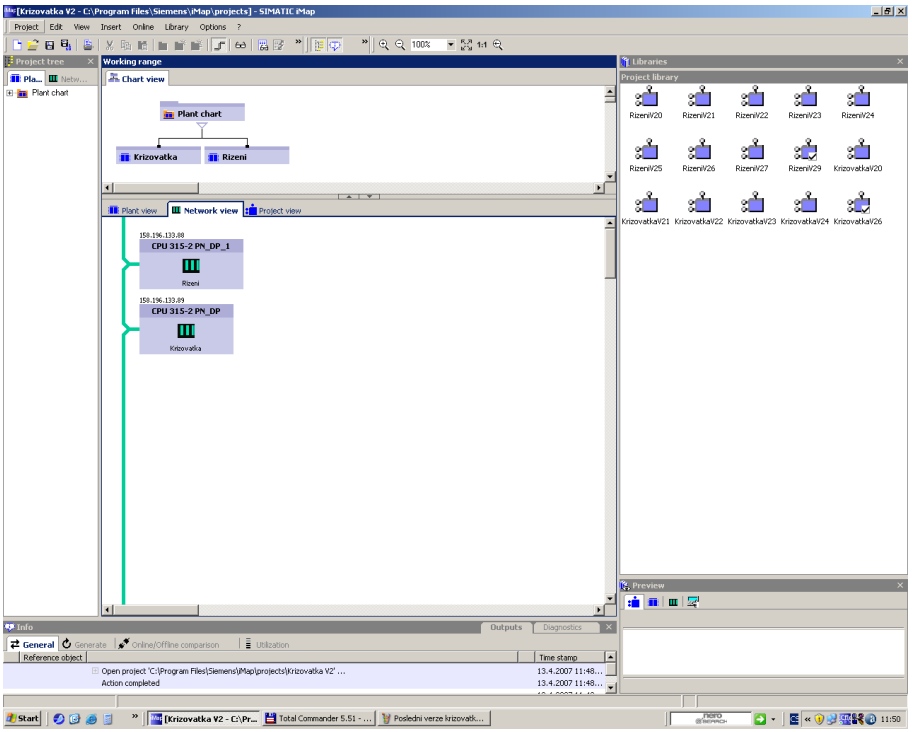
- Objektový.
- Síťový.

Objektový pohled - zde jsou nezbytné komponenty importovány z knihovny, umístěny na obrazovku a jsou vyobrazeny jejich spojení. Výsledný obrázek vytváří přehled o technologické struktuře a jejich logických spojeních.



Obr. 14.18:Ukázka systémového zobrazení iMapu.

Síťový pohled - vyobrazuje topologické schéma celého distribuovaného systému. Jsou vidět jednotlivé distribuované periferie, rozhraní mezi komunikačními protokoly a PLC.



Obr. 14.19: Ukázka síťového zobrazení iMapu.

Komunikace Profinet IO:

Při startu používá Profinet IO UDP/IP protokol s funkcemi vzdáleného volání procedur (RPC), pro inicializaci vazeb pro výměnu dat, nastavení parametrů distribuované periferie a spouštěcí diagnostiku.

Díky otevření standardního protokolu vzdáleného volání procedur, mohou do sítě přistupovat zařízení HMI a inženýrské systémy (IO-Supervisors). Realtimový kanál je pak použit pro přenos vstupně/výstupních dat a alarmů. Při typické vstupně/výstupní konfiguraci si PLC cyklicky vyměňuje data s distribuovanými periferiemi. V každém obnovovacím cyklu jsou do PLC zaslána data ze všech spřažených periférií, pak jsou zpracována a poslána zpět do periférií. Komunikační vazby jsou kontrolovány přijatými zprávami cyklické komunikace. Pokud selže vstupní rámec třikrát, je periferie identifikována jako vadná.

Profinet používá packetu s délkou datové oblasti 66 bytů až 1500bytů. Režie celkového protokolu pro RT data je 28 bytů.

Komunikace mezi komponenty:

U Profinet komponent se používá protokol v DCOM (Distributed COM), který pracuje na bázi TCP/IP. DCOM je rozšířené COM pro distribuci objektů a jejich spolupráci skrze síť.

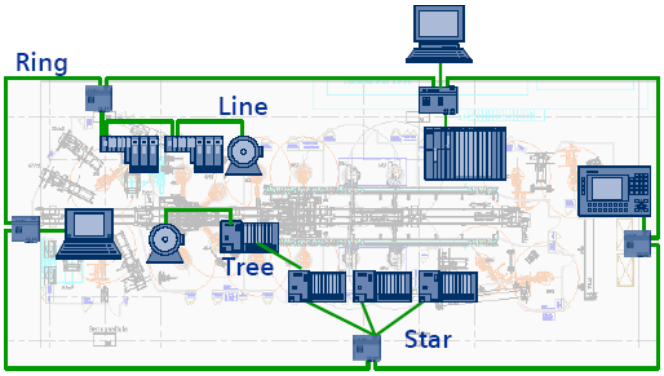
Používání DCOM pro výměnu dat mezi komponenty Profinet není nezbytné. Uživatel konfiguruje, zda data budou vyměněna skrz DCOM nebo real-timový kanál distribuovaného systému. Když periferie startují komunikaci, posoudí také, zda je třeba real-timového protokolu, protože komunikace mezi PLC a periferií nemůže být prováděna běžným TCP/IP protokolem.

TCP/IP a DCOM protokol může být použit pouze pro start komunikace mezi periferiemi a PLC. Poté se použije Profinet RT kanál. Uživatel může při konfiguraci nastavit kvalitu obsluhy zařízení nastavením obnovovací frekvence, kde data mohou být přenášena cyklicky nebo pouze při požadavku některého z komponent modelu. Cyklický přenos je lepší v rámci vysokých přenosových rychlostí,

protože dotazy na obnovu zaberou více výpočetního času procesoru než cyklická výměna bez dotazování.

14.3.9 Možnosti zapojení

S PROFINETem můžete vytvářet různé druhy sítí. Rozlišují se 4 druhy zapojení z nichž každé zapojení má své výhody a nevýhody. Podle následujícího obrázku lze ale vidět, že lze do jedné sítě zapojit různé druhy sítí, protože PROFINET využívá přepínaný Ethernet. Kruhová struktura garantuje vysokou bezporuchovost a také rychlost, sběrnicová struktura zajišťuje zase minimalizaci nákladů na kabely.



Obr.14.20: Zapojení SCALANCE v síti PROFINET.

Do sítě PROFINET mohou být také zahrnuty aktivní prvky jako SWITCH a ROUTER. Switch může být již také integrován do procesorů CPU 414-3 PN/DP, CPU 416-3 PN/DP, CPU 416F-3 PN/DP. Tyto CPU obsahují integrovaný 2 portový switch, webserver. Každý switch obsahuje kartu C-PLUG (Configuration Plug), do které se ukládá aktuální nastavení switche a v případě, že tento switch bude zničen, vyjmutím této karty a vložením této karty do nového switche získáte opětovné nastavení switche. Odpadá tímto nutnost opětovně nastavovat tento nový switch.

Obr. 14.21: Karta C-Plug.

14.3.10 Switch SCALANCE X

Switche SCALANCE X jsou aktivní síťové komponenty, používané v sítích průmyslový Ethernet. Předností modulů SCALANCE X je robustní provedení, vysoká odolnost proti průmyslovým vlivům, spolehlivost, realizace redundantních síťových struktur, široké diagnostické možnosti, speciální funkce plynoucí z koncepce TIA (např. zahrnutí do projektu ve STEP 7), jednoduchá parametrizace, podpora všech potřebných protokolů atd. Skupina produktů SCALANCE X zahrnuje 4 řady výrobků, které se navzájem doplňují a jsou sestaveny tak, aby uživatel našel vždy modul, který bude plně vyhovovat jeho potřebám. Všechny switche těchto řad jsou pečlivě testovány společně s ostatními produkty divize A&D, což je pro uživatele zárukou správné funkce celého systému. [14-14, 14-15]

SCALANCE X005 Vstupní úroveň

Nemanažovatelný switch v průmyslovém provedení s pěti metalickými RJ45-Porty pro použití v méně rozsáhlých komunikačních sítích. Cenově optimalizovaný modul.

SCALANCE X100 Základní

Průmyslové switche s až 24 porty (metalické i optické) a integrovanou diagnostikou ve formě signalizačního kontaktu a LED, nemanažovatelné.

SCALANCE X200 Manažovatelné

Univerzálně použitelné switche v průmyslovém provedení (některé i IP65), až 24 portů (metalické i optické). Integrované www stránky pro parametrizaci a diagnostiku. Možnosti diagnostiky: www, signalizační kontakt, SNMP, email, LED, integrace do projektu v softwaru SIMATIC Step 7. Možnost vytváření redundantních struktur (kruh, ...) s krátkou dobou rekonfigurace. Paměťové médium C-plug pro zálohování konfigurace.

SCALANCE X400 Modulární

Pro implementaci do vysoce výkonných průmyslových provozních sítí a propojení s kancelářskými sítěmi. Částečně modulární konstrukční koncepce nabízí flexibilitu – výběr počtu portů (metalické i optické, i Gigabit Ethernet). Integrované www stránky pro parametrizaci a diagnostiku. Možnosti diagnostiky: www, signalizační kontakt, SNMP, RMON, email, LED, integrace do projektu v softwaru SIMATIC Step 7. Možnost vytváření redundantních struktur s krátkou dobou rekonfigurace. Podpora směrování (routing) a IT funkcí (VLAN, IGMP, RSTP). Paměťové médium C-plug pro zálohování konfigurace.

SCALANCE X100 Konvertory

Průmyslové převodníky metalických sítí na optické a sítí s komunikační rychlostí 10 Mb/s na 100Mb/s.

Switche SCALANCE se také liší počtem vstupů a integrovanou funkcí a použitím. Každý switch je také vybaven spínacím kontaktem, který je v případě chyby na switchi sepnut. Switche SCALANCE X005 jsou označovány jako low-end a slouží jako klasické switche pro PC s tím, že jsou přizpůsobeny průmyslovému prostředí. Switche bez správy X100 jsou switche, které neobsahují další rozšiřovací funkce jako SNMP, Web Server apod. Můžete do nich připojit jak metalické vedení, tak vedení pomocí světlovodů. Switche se správou X-200 jsou switche, které obsahují integrované funkce SNMP (Simple Network Management Protocol) , Redundancy Manager, apod. U této řady je důležité si povšimnout zejména typu X204 IRT, který může být nakonfigurován jako tzv. správce kruhové sítě, pokud je síť složena pouze se switch řady se správou, bez správy, low-end, tak tento switch zajišťuje to, aby při výskytu poruchy na vedení - přerušení kabelového vedení, mohl přesměrovat komunikaci např. na opačný směr. Pokud vytvoříte kruhovou topologii z těchto switchů tj. X005,X100,X200 dosáhnete max. přenosové rychlosti 100Mbit/s. Pokud vytvoříte kruhovou síť se switchi X414 můžete dosáhnou přenosové rychlosti až 1Gbit/s.

Pozn.

Simple Network Management Protocol (**SNMP**) je součástí sady internetových protokolů. Slouží k potřebám správy sítí. Umožňuje průběžný sběr nejrůznějších dat pro potřeby správy sítě, a jejich následné vyhodnocování. Na tomto protokolu je dnes založena většina prostředků a nástrojů pro správu sítě.

Má tři verze: druhá obsahuje navíc autentizaci a třetí šifrování. Nejvíce zařízení podporuje druhou verzi.

Rozlišuje se mezi stranou monitorovanou (dohledovaný systém) a monitorovací (sběrna dat). Tyto strany mohou běžet buď odděleně na různých fyzických strojích, nebo v rámci jednoho stroje. Na

monitorované straně je spuštěn agent a na straně monitorovací manager. Na straně monitorované jsou operativně shromažďovány informace o stavu systému (zařízení). Manager vznáší požadavky agentovi, zpravidla na zaslání požadovaných informací (zpráv). Agent zajišťuje realizaci reakcí na požadavky managera. Získaný obsah zpráv se na straně monitorovací může dále různým způsobem zpracovávat (tabulky, grafy…). Komunikace mezi agentem a managerem se označuje jako SNMP operace.

Na straně monitorované může existovat možnost takové konfigurace, kdy agent zašle managerovi informace automaticky bez jeho požadavku. K tomu dojde zpravidla potom, kdy byla splněna předem definovaná podmínka (výpadek, kolize, dosažení hraniční hodnoty…), agent nečeká na odpověď. V tomto případě se jedná o SNMP operaci TRAP. [14-18, 14-19]

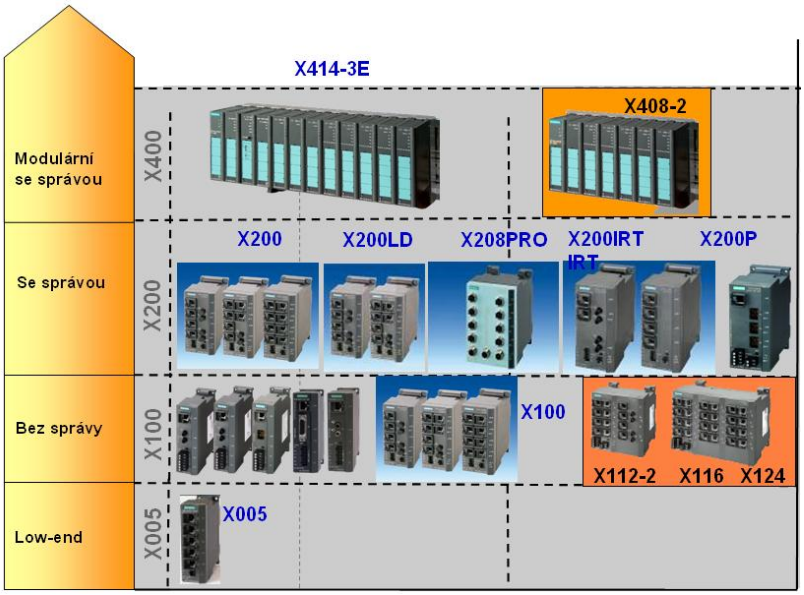
Standard RMON (Remote Monitoring), RMON (Remote Monitoring) bývá integrální součástí software přepínače a jeho hlavním určením je trvale získávat statistiky o činnosti přepínače. Tyto statistiky jsou poté správci sítě dostupné prostřednictvím jednotlivých výše uvedených rozhraní pro správu. Existují dvě verze RMON, a to RMON-1 a RMON-2. RMON-1 je primárně orientován na činnost přepínače na druhé vrstvě a má definováno 10 skupin, mezi které patří například statistiky, historie, alarmy, události, informace o klientech sítě a jejich komunikaci a informace o spojeních mezi jednotlivými klienty. RMON-2 rozšiřuje schopnosti RMON-1 systému o informace z vyšších vrstev (informace o protokolech a adresách). [14-18, 14-19]

Protokol IGMP (Internet Group Management Protocol)

Použití vícesměrového vysílání IP v sítích TCP/IP je definováno jako standard TCP/IP ve specifikaci RFC 1112, Internet Group Management Protocol (IGMP). Tento dokument RFC definuje kromě rozšíření adres a hostitelů pro podporu vícesměrového vysílání hostiteli IP také verzi 1 protokolu IGMP (Internet Group Management Protocol). V dokumentu RFC 2236, Internet Group Management Protocol (IGMP) version 2, je definována verze 2 protokolu IGMP. Obě verze protokolu IGMP slouží k výměně a aktualizaci informací o členství hostitelů ve specifických skupinách vícesměrového vysílání. Pomocí verze 3 protokolu IGMP hostitelé mohou zadat, že chtějí přijímat vícesměrové vysílání ze zadaných zdrojů nebo ze všech zdrojů s výjimkou určité sady zdrojů.

Princip vícesměrového vysílání IP

Data vícesměrového vysílání IP jsou odesílána na jedinou adresu, ale zpracovává je více hostitelů. Princip vícesměrového vysílání IP je podobný principu novinového předplatného. Podobně jako právě vydané noviny obdrží pouze jejich předplatitelé, data protokolu IP odeslaná na adresu IP rezervovanou pro skupinu vícesměrového vysílání přijmou a zpracují pouze hostitelské počítače, které patří do této skupiny. Skupina hostitelů, kteří přijímají zprávy odeslané na určitou adresu IP pro vícesměrové vysílání, se nazývá skupina vícesměrového vysílání. [14-18, 14-19]

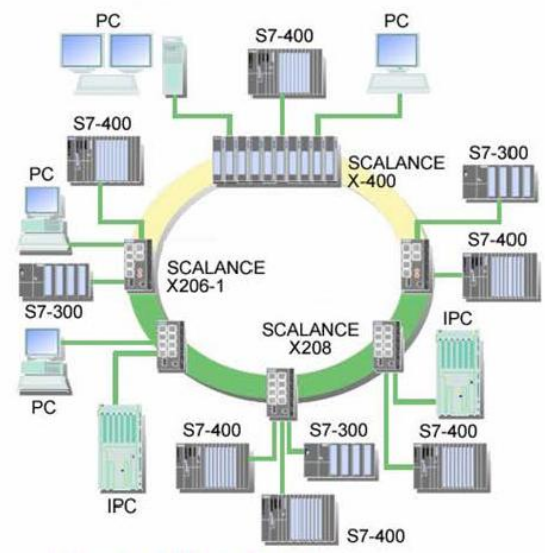


Obr. 14.22:Přehled produktů SCALANCE.

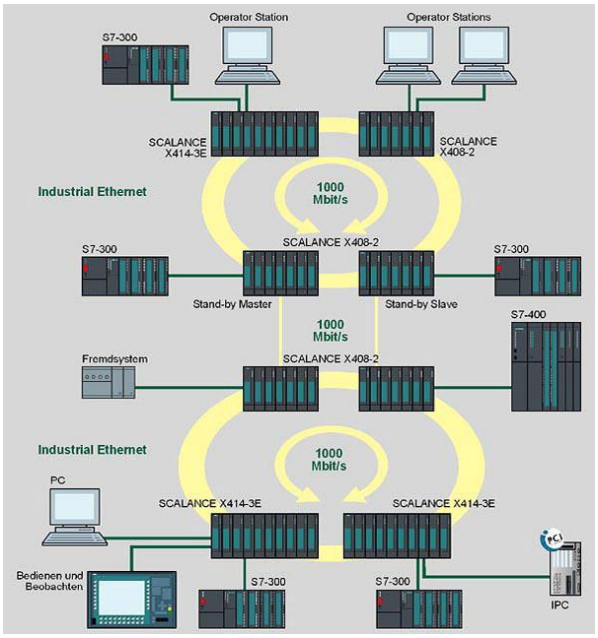
Nejprve se pojďme podívat na kruhovou topologii. Výše bylo uvedeno, že tato topologie zaručuje spolehlivost komunikace. Níže uvedený obrázek ukazuje možnost základního kruhového spojení.

Kruhové spojení může být realizováno jak z metalických kabelů tak optických vláken. Existuje zde několik pravidel. Když zapojujete do kruhu jednotlivé switche mohou být v kruhu zapojeny různé typy, ale jeden z nich musí být konfigurován jako Redundancy Manager. Pokud využíváte jenom jedno kruhové spojení musí být switch SCALANCE X204 IRT nakonfigurován jako Redundancy Manager. Pokud máte spojeny vzájemně dvě kruhové sítě nemůže být switch SCALANCE X204IRT konfigurován jako Redundancy Manager, ale musí být zvolen režim Standy Manager. Na níže uvedeném obrázku by byl Redundancy managerem switch řady X-400, protože žádný ze switchů řady X200, zde nemá funkci Redundancy Manager tj. typ X20xIRT. Redundancy manager dohlíží na správnost a úplnost zapojení v kruhové topologii a v případě přerušení jedné části kabelu mezi switchi, dokáže tuto komunikaci přesměrovat a výpadek nijak neovlivní přenos dat.

Se switchi řady X400, X200 mohou být také vytvářeny redundantní spojení s tím, že dva switche musí být nastaveny v režimu Stand by, který zajišťuje detekci redundantního spojení a v případě chyby na jedné části sítě, převezme komunikaci obr. 14.24.

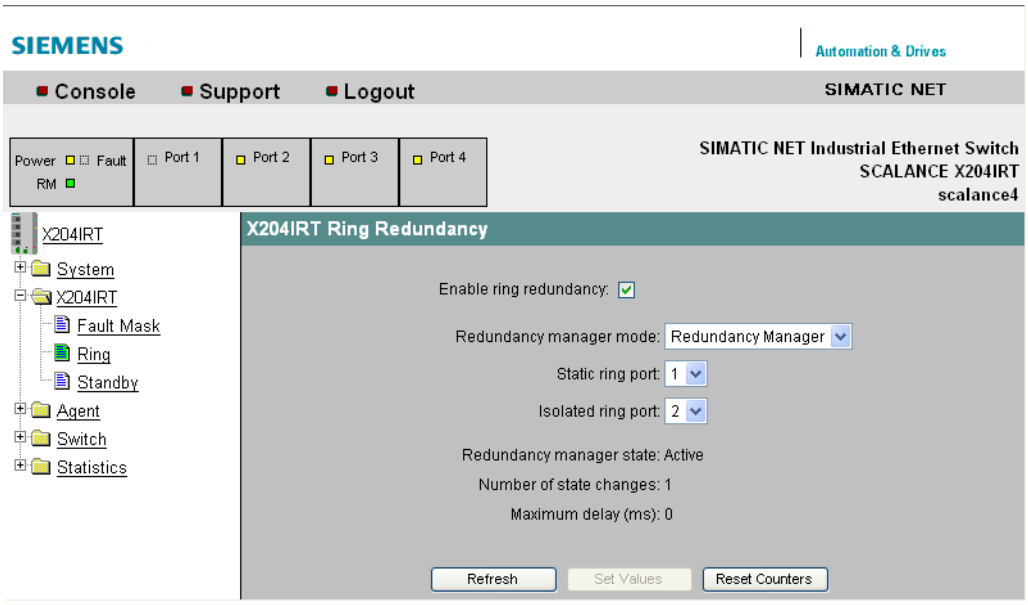


Obr. 14.23: Kruhová topologie.



Obr. 14.24: Dvě kruhové redundantní sítě.

Každý switch je možné nakonfigurovat. Aby jste mohli tyto switche nastavovat, musíte do internetového prohlížeče zapsat jeho IP adresu. Budete dále vyzváni pro zadání hesla a poté se otevře Web Based Management, kde se provádí doplňující nastavení. Příklad konfigurace ukazuje následující obrázek



Obr. 14.25: Konfigurace SCALANCE X 204 IRT jako Redundancy Manager.

14.3.11 SCALANCE W

Typickými průmyslovými aplikacemi bezdrátové komunikace se Scalance W, jsou obecně automatizované systémy, kde je potřeba připojit pohybující se části technologických zařízení do komunikační sítě – často u automaticky řízených vozíků, jeřábů, rotačních částí, apod., s využitím Profinet IO. Možné je i připojení vzdálených částí technologie či těžko dostupných částí technologie bezdrátovým spojením. Další oblastí je bezdrátový sběr dat, užívání mobilních terminálů, mobilní provádění servisu a diagnostiky automatizovaných systémů.

Bezdrátová komunikace s prvky Scalance W je založena na IWLAN (Industrial Wireless LAN), což je rozšířená verze standardu IEEE 802.11 určená pro použití v průmyslu. Komponenty využívající WLAN např. notebooky s rozhraním Wifi mohou fungovat v jedné síti společně s komponenty založenými na IWLAN.

Přístupové body a klientské moduly firmy Siemens s označením Scalance W jsou ideálními prvky pro realizaci průmyslových bezdrátových sítí na bázi IWLAN. Pracují na frekvencích 2,4 GHz nebo 5 GHz s rychlostí přenosu dat až 54 Mb/s (IEEE 802.11 b/g, 802.11 a/h). Zabezpečení komunikace řeší mechanizmy WPA a AES, WEP, Radius. Konfiguraci modulů provádíme přes webové rozhraní nebo příkazový řádek, usnadní ji připravení průvodci (Wizards). Všechny komponenty Scalance W jsou určeny pro nasazení v průmyslových podmínkách, tomu je přizpůsobeno jejich provedení. Vysokou odolnost proti vnějším vlivům předurčuje jejich robustní kovový kryt. Moduly Scalance W mají stupeň krytí IP 65/67, široký rozsah pracovních teplot a jsou odolné proti vibracím a mechanickým rázům.

Siemens dodává několik typů přístupových bodů s jedním nebo dvěma rádiovými rozhraními i body s funkčností Rapid roaming zmíněnou výše. Tři typy klientských modulů slouží k připojení zařízení do bezdrátové sítě. Nechybí ani potřebné příslušenství – různé antény pro komunikaci až na kilometrové vzdálenosti, Rcoax kabel vyzařující definované radiové pole použitelný místo antény (typicky např.v tunelech, výtahových šachtách, na dopravnících apod.), napájecí zdroje, apod. Zařízením zasluhujícím zvláštní pozornost je IWLAN/PB Link PN IO sloužící pro realizaci přímého přechodu mezi IWLAN a sítí Profibus. Dostupný je i software SINEMA E pro návrh a zprovozňování bezdrátových sítí. [14-15]

Bezdrátová technologie je kompatibilní s následujícími standardy:

IEEE 802.11 b (11 Mbit/s, 2400.0-2483.5 MHz).

IEEE 802.11 g (54 Mbit/s, 2400.0-2483.5 MHz).

IEEE 802.11 a/h (54 Mbit/s, 5150-5350 MHz; 5725-5825 MHz).

Pro bezdrátovou technologii můžete využívat:

Přístupové body.

SCALANCE W788-1PRO/W788-2PRO.

SCALANCE W788-1RR/W788-2RR.

Komunikační procesory.

CP1515 např. pro Field PG.

CP 1515 např. pro zařízení MOBIC.

Klientské moduly.

SCALANCE W744-1PRO.

SCALANCE W746-1PRO.

SCALANCE W747-1RR.

IWLAN/PB Link PN IO network transition.

IWLAN RCoax Cable.

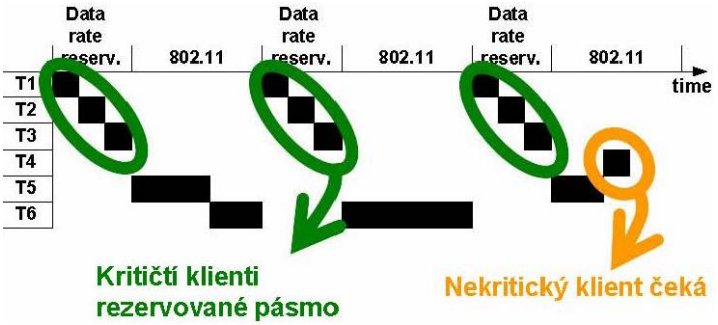
Různé antény a jiné.

Tato bezdrátová technologie nachází uplatnění zejména při automaticky řízených dopravních systémech, propojení segmentů sítě, integrace sítí PROFIBUS, mobilních připojeních – servis,

diagnostika, monitorování. SWITCH SCALANCE W obsahuje opět funkce pro diagnostiku, které mohou být SNMP, Web Based Management. SCALANCE W přináší techniku IWLAN tj. Industrial LAN.

Technika IWLAN (Industrial LAN)

Celá komunikace IWLAN je řešena pomocí komunikace mezi klienty, kde industrial znamená, že komunikace má vyhrazené pásmo iQoS (industrial Duality of Service). Tato metoda zajišťuje, že se může komunikovat s jakýmkoliv klientem a také zajišťuje cyklickou výměnu dat. Metoda je kompatibilní se standardem IEEE 802.11. Pomocí Web Based Managementu se musí také nastavit šířka přenášeného pásma např. 200Kbit/s.



Obr.14.26: Princip rezervace přenosového pásma iQoS.

Obohacení IWLAN oproti WLAN

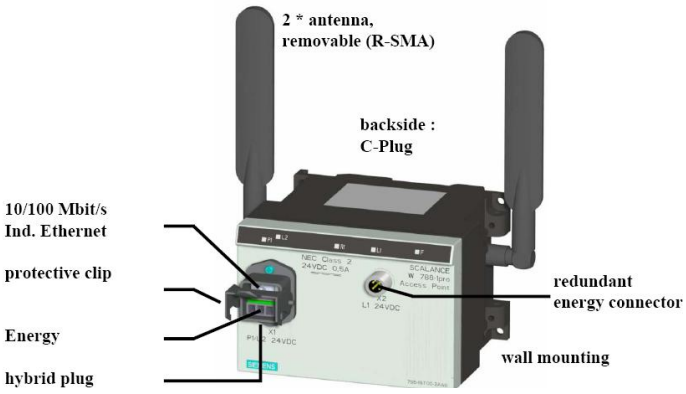
- Podpora komunikace v reálném čase.
- Rychlý roaming (přesměrování komunikace klienta z jednoho přístupového bodu na druhý).
- Diverzita antén (ze dvou antén je užívána ta, která má lepší podmínky).
- Sledování bezdrátového spojení.
- Wireless distribution system (bezdrátové propojování jednotlivých přístupových bodů).
- Redundantní bezdrátová spojení.
- Filtrování zpráv typu multicast/broadcast.
- Vynucený roaming.
- Spanning tree protokol.
- atd.

Pozn.

QoS (**Quality of Service**) je řízení datových toků v síti. Tento protokol zajišťuje spravedlivé dělení rychlostí a nedochází tak k zahlcování sítě. Problém může nastat v okamžiku, kdy jeden subjekt sítě (budeme brát v tomto případě uživatele (více PC) připojeného ke sdílenému internetu v rámci třeba malé firmy nebo domácnosti s více PC). Jestliže se rozhodne jeden uživatel stahovat přes den např. film (700MB) a druhý chce např. volat přes VoIP a používat Web, tak se stane to, že druhý uživatel bude mít velké problémy v komunikaci, bude docházet k vysoké odezvě (ping) a web bude pomalý. Toto nastane pokud je sdílena celá konektivita a není nijak uplatňováno QoS, nebo alespoň nejjednodušší omezování rychlosti bandwidth. [14-18, 14-19]

SCALANCE W700

SCALANCE W700 nabízí spolehlivý rádiový přenos, všestranné redundantní spojení a vysokorychlostní přenos z jednoho přístupového bodu na další přístupový bod (roaming). SCALANCE W700 umožňují snadné začlenění do stávající sítě. Industrial Wireless LAN (IWLAN) mohou být použity v časově kritických aplikacích. SCALANCE W700 mají krytí IP65 a relativně velký teplotní rozsah -20°C až 60°C. Použitím RCoax kabelu se může také tato technologie uplatnit v dopravních systémech. Přístupové body pracují podle normy IEE 802.11 a/b/g/h, kde přenosová rychlost může být až 54Mbit/s. Přístupové body mohou pracovat na frekvencích 2,4GHz a také 5GHz.

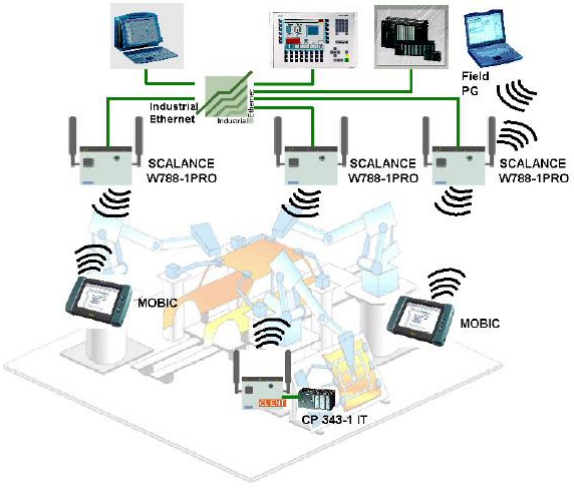


Obr. 14.27: SCALANCE W 7xx.

14.3.12 Přístupové body

SCALANCE W788-1PRO

SCALANCE W788-1PRO jsou přístupové body, ke kterým se lze připojit pomocí např. mobilních zařízení různých typů PG vybavené komunikačními kartami CP7515 apod. Tím je zajištěn přístup do sítě Industrial Ethernet. Přístupový bod nemusí být jenom jeden pokud je nedostačující a může být jich několik. Může se přecházet od jednoho přístupového bodu ke druhému bez časové mezery (roaming). Přístupový bod musí být schopen vyměňovat data přes Ethernet pokud není použit Wireless Distribution System (WDS). Při komunikaci využívá technologii 802.11 a/b/g

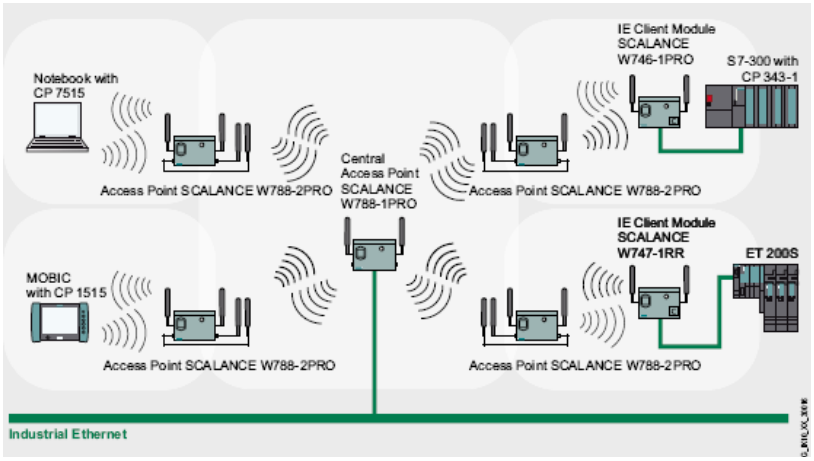


Obr.14.28: Možnosti přístupu do sítě Industrial Ethernet.

Pokud nemůže být přístupový bod již připojen kabelem k síti Ethernet, musí být pro tento přístupový bod zvolen operační mód Wireless Distribution System tzn. že informaci, kterou dostal musí dál předat. SCALANCE W788-1PRO může komunikovat až s 8 dalšími přístupovými body, které nejsou

spojeny se síti Ethernet přímo pomocí kabelu. Pro tuto možnost se používají přístupové body SCALANCE W788-2PRO.

SCALANCE W78x-2PRO jsou rozšířenou verzí přístupových bodů SCALANCE W788-1 PRO s možností využití technologie 802.11 a/b/g a 10/100Mbps Ethernet uplink. Každý z těchto modulů pracuje jako dvojitý přístupový bod s jedním sdíleným Ethernet uplinkem, který může pracovat jako most mezi 802.11a, 802.11g a Ethernetem. Mohou se také použít pro komunikaci s 802.11g klienty.

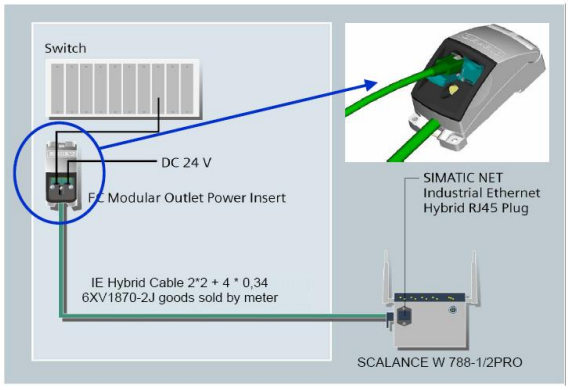


Obr.14.29: WDS systém se čtyřmi vzdálenými přístupovými body bez přímého spojení s Industrial Ethernetem.

V této konfiguraci podle obrázku výše je rádiový přenos každému ze 4 W788-2PRO zajištěn přístupovým bodem W788-1PRO pomocí WDS. Přístupové body mohou vzájemně komunikovat.

Speciální verzí WDS může být pomocí point-to-point spojení. V tomto operačním módu jsou dva Ethernetové segmenty spojeny pomocí rádiového přenosu např. když je obtížné propojit dva segmenty sítě kabelem. Řešením je použít dvě směrové antény, kde vzdálenost mezi nimi může být ve venkovním prostoru až 100m.

Protože přístupový bod např. SCALANCE W788-1PRO je většinou umístěn někde na stropě místnosti apod. musely by se s k němu instalovat dva kabely jeden napájecí a jeden datový, tak se tento problém řeší tím, že se používá kabel typu IE Hybrid Cable 2x2 + 4x0,34 spolu s module FC Modular Outlet Power Insert, jak ukazuje následující obrázek 14.30.



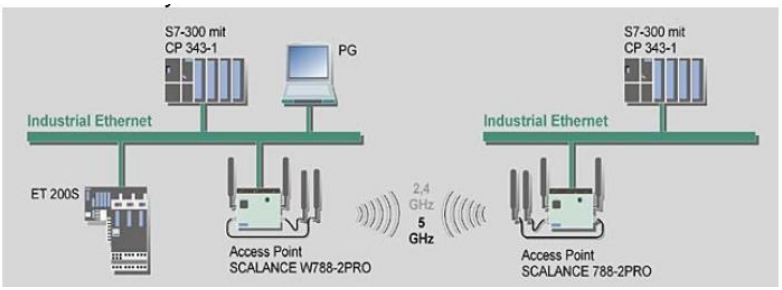
Obr. 14.30: Připojení přístupového bodu SCALANCE W788-1/2 PRO pomocí kabelu IE Hybrid Cable 4x0.34.

SCALANCE W788-2PRO – redundantní zapojení

SCALANCE W788-2PRO je obzvláště vhodný pro vytváření redundantních spojení v aplikacích, které požadují vysokou spolehlivost. Tato možnost je rozšířena o Spanning Tree (ST) mechanismus, který v případě chyby na jednom spojení okamžitě vyhledává redundantní cestu.

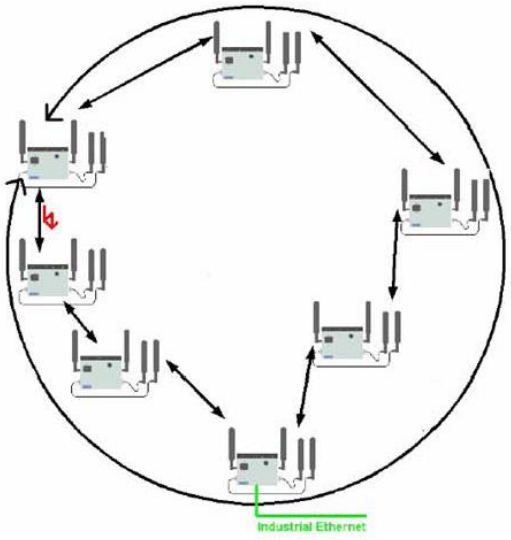
Redundancy mode

V tomto operačním módu jsou dvě Industrial Ethernetové sítě spojeny a tok dat je přenášen paralelním bezdrátovým systémem. Např. na frekvenčním pásmu 2,4GHz nebo 5GHz. S tím je dosažena maximální spolehlivost.



Obr.14.31: Redundantní spojení.

S duálními přístupovými body můžete dosáhnout konfigurace záložní cesty na jiném kanálu a tím vytvořit redundantní spojení pomocí Spanning tree protokolu.



Obr.14.32: Spanning Tree protokol.

Pozn.

Redundantní topologie a větvící se strom

Redundantní síťové topologie jsou navrhovány za účelem funkčnosti sítě i v případě, že selže jeden bod sítě. Každý výpadek by měl být tak krátký, jak je to jen možné.

V hlavičce druhé vrstvy není hodnota Time To Live (TTL). Pokud je rámec druhé vrstvy poslán do přepínané sítě se smyčkami, bude tam přeposílán do nekonečna. To vede k plýtvání šířky pásma a vede k nepoužitelnosti sítě.

Z výše popsaného vyplývá, že přepínaná síť pro správnou funkci nemůže mít fyzické smyčky. Ale pro zvýšení dostupnosti sítě jsou fyzické smyčky klíčové. Řešením je mít sice fyzické smyčky, ale vytvořit logickou topologii bez smyček.

Logická topologie bez smyček se nazývá strom (anglicky tree). Logická topologie je hvězda nebo rozšířená (extended) hvězda. Tato topologie je větvící se strom (anglicky spanning-tree). Jmenuje se tak protože všechny zařízení jsou dostupná přímo nebo přes větve stromu.

Algoritmus, který vytvoří logickou topologii bez smyček, se jmenuje spanning-tree algoritmus. Tento algoritmus je časově poměrně náročný. Proto byl vyvinut rapid spanning-tree algoritmus, který spočítá logickou topologii bez smyček rychleji. [14-18, 14-19]

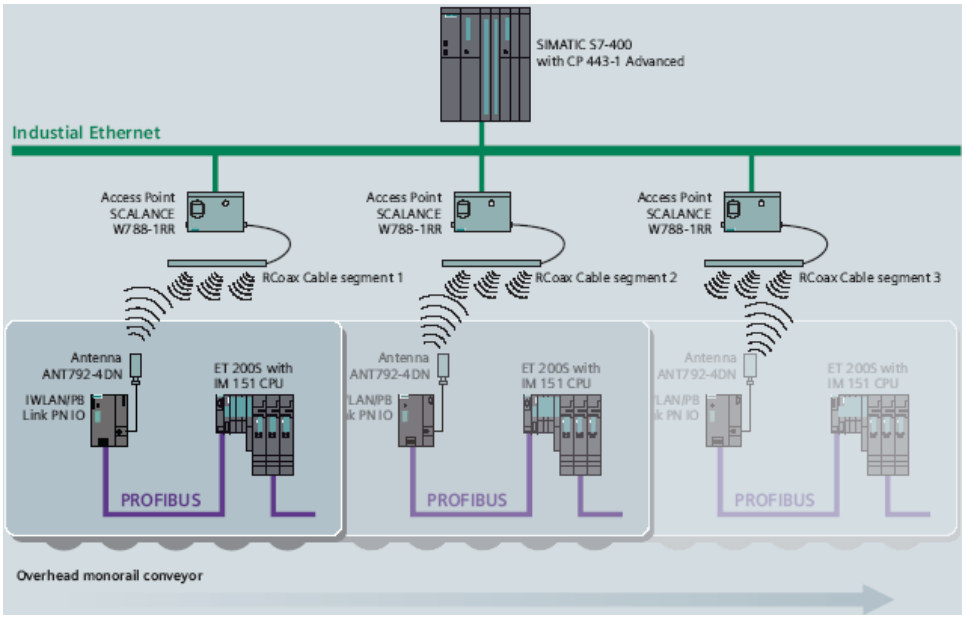
Spanning Tree protokol (STP – 802.1D) je systém komunikace přepínačů mezi sebou, který dovolí do vaší sítě jednoduše začlenit redundanci linek a základní ochranu proti vzniku kruhu (network loop), který následně způsobuje zahlcení sítě všesměrovými pakety. STP umožňuje na přepínačích vytvářet alternativní cesty spojení a umožní přepínačům pro přenášená data nalézt nejvhodnější cestu povolením rychlejších datových cest mezi přepínači a zablokováním těch pomalejších (záložních) spojů.

Vznikl i nový standard nazvaný **Rapid Reconfiguration of Spannig Tree Protocol** (RSTP - 802.1w), který jak už z jeho názvu vyplývá, zajišťuje podstatně rychlejší konvergenci v sítích řízených tímto protokolem. Například pokud při výpadku spojení v síti řízené STP trvá možná konvergence v řádu desítek vteřin, RSTP realizuje změny datových toku v několika vteřinách. [14-18, 14-19]

SCALANCE W788-2RR

SCALANCE W788-2RR (duální přístupové body) jsou vytvořeny spojením SCALANCE W788-1RR a SCALANCE W788-1PRO. K tomuto přístupovému bodu může být připojen RCoax kabel, který ve spojení s IWLAN/PB Linkem PN IO a rychlým roamingem umožňuje přenášet data z pohyblivých zařízení obr. 14.34.

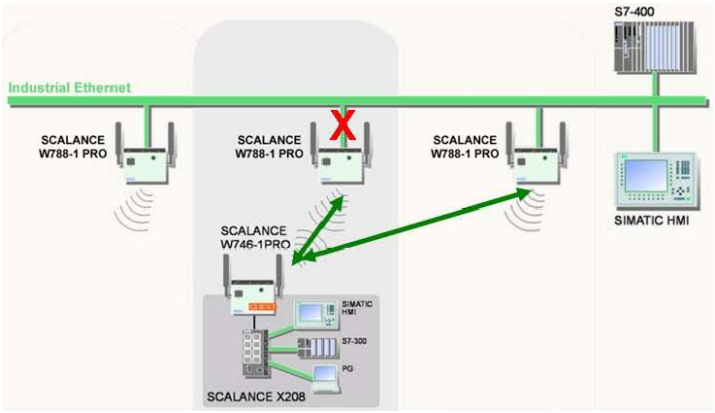
RCoax kabel je speciální typ kabelu, který pracuje jako anténa pro SCALANCE W přístupové body. Kolem kabelu je definované rádiové pole. Proto je obzvláště vhodný pro SCALANCE W740, IWLAN/PB Link. Kabel musí být na začátku a na konci zakončen terminátorem s impedancí 50Ω.



Obr.14.34: Využití rychlého roamingu při řízení vozíku.

SCALANCE W a vynucený roaming

Princip vynuceného roamingu spočívá v tom, že např. předmět osazen klientským modulem SCALANCE W746-1PRO vyšle nějaká data na nejbližší přístupový bod, který musí tyto data dále přeposlat do nadřízené stanice. V případě, že se tento přístupový bod např. W788-1PRO nedokáže spojit s nadřazenou sítí, klientský modul W746-1PRO začne svá data posílat na nejbližší další přístupový bod W788-1PRO obr. 14.35.

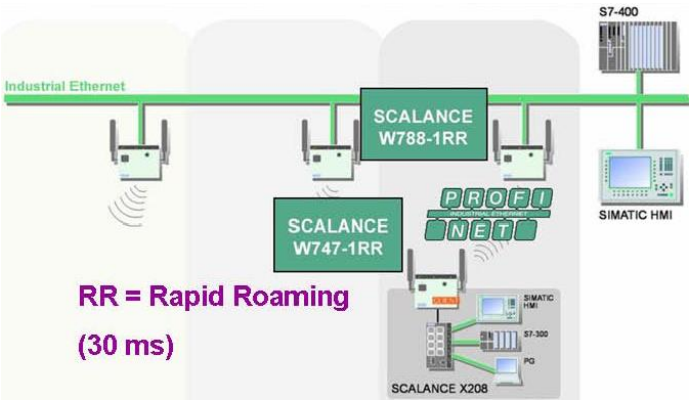


Obr. 14.35: Vynucený roaming.

SCALANCE W a rychlý roaming

Pro rychlý roaming se používají SCALANCE W788-1RR(SCALANCE W788-2RR). Tyto moduly mají možnost extrémně rychlého roamingu z jedné buňky na další. Přenos je tak rychlý, že je zde možnost využití PROFINET IO bez přerušení během roamingu (update time max 20ms).

- Rychlý roaming
- Cyklická komunikace s Profinetem IO



Obr. 14.36: Vynucený roaming.

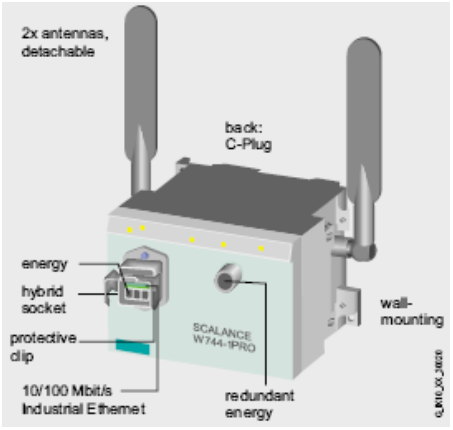
14.3.13 Klientské moduly

SCALANCE W-740 moduly opět využívají frekvence 2,4 až GHz. Vyznačují se spolehlivým přenosem až do rychlosti 54Mbit/s. tyto moduly opět podporují standard IEEE802.11h. Zaručují definovanou odezvu a dodržování komunikace v reálném čase. [14-14, 14-15]

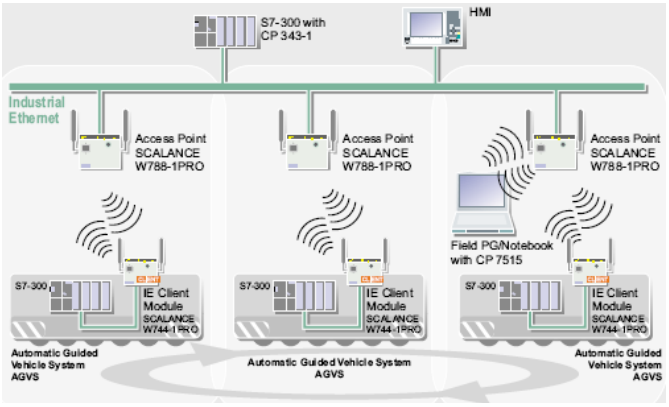
SCALANCE W744-1PRO

Tento ethernetovský klient SCALANCE W744-1PRO zajišťuje přenos mezi mobilním zařízením a sítí Industrial Ethernet. Clientovský model implementuje bezdrátovou technologii, ve které se tento klientský modul může pohybovat volně. Ethernetovský klient modul může opět využívat mezi jednotlivými přístupovými body možnost roamingu.

SCALANCE W744-1PRO může spravovat jednu jednoduchou IP adresu a to připojeného zařízení. Jestliže je tato adresa změněna klientský modul rozpozná tuto změnu a spravuje tuto novou adresu. Pomocí Infractruture módu SCALANCE W744-1PRO může pracovat v ad hoc módu. Tímto může klientský modul SCALANCE W744-1PRO komunikovat s dalšími klientskými moduly bez možnosti využití přístupového bodu.



Obr. 14.37: Klientský modul.



Obr.14.38: Použití klientského modulu při řízení mobilního vozíku.

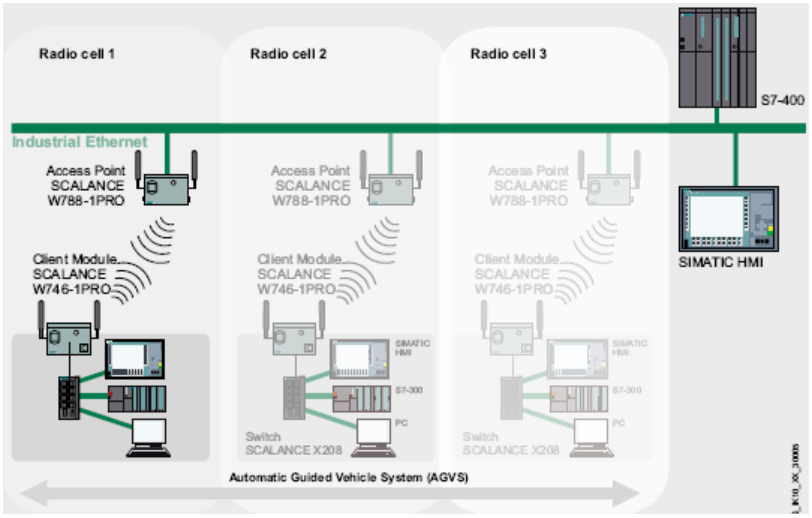
Pozn.

Ad-hoc označuje dočasné síťové spojení mezi dvěma rovnocennými prvky např. dva laptopy spojené pomocí Wi-Fi bez Přístupového Bodu (AP). V principu pak celá síť funguje tak, že první spuštěný klient tvoří jakýsi imaginární access point, který pak řídí další komunikaci ostatních klientů. Ti však komunikují bez tohoto „hlavního“ počítače. Pokud hlavní klient vypadne, síť se na malý okamžik rozpadne, než se funkce access pointu ujme jiný klient (většinou zcela náhodně).

SCALANCE W746-1PRO

Tento ethernetovský klient SCALANCE W746-1PRO zajišťuje přenos mezi mobilním zařízením jako administrátor až pro 8 zařízení s připojením na Ethernet. Tím je možné vytvářet malé mobilní jednotky

(max. s 8 zařízeními), které se pohybují v poli IWLAN. Jestliže je jedno zařízení změněno ethernetovský klient rozpozná tuto změnu a spravuje novou adresu. Využití tohoto modulu je u mobilních vozíků apod. obr.14.39.

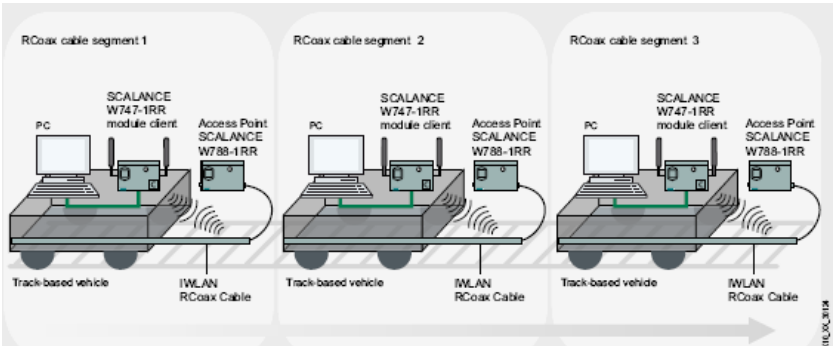


Obr. 14.39: Využití klientského modulu SCALANCE W746-1PRO při řízení mobilního vozíku.

SCALANCE W747-1RR

Mobilní zařízení SCALANCE W788-1RR zajišťuje v síti vynucený cyklický roaming tam, kde se používá PROFINET IO a kde se požadují velmi rychlé aktualizace časů. Vynucený roaming je rozšířením IWLAN a nabízí real-time mobilní komunikaci dokonce i při pohybu od jedné mobilní buňky k další (roaming). Update time může být až 16ms. Jak víme IWLAN komunikace pracuje s vynuceným roamingem pro časově kritické aplikace i v několika stanicích. RCoax kabel (radiating cable) se s výhodou používá při řešení těchto aplikací. Pokud se musí data přenášet s pevnou dobou odezvy je nutné využít tento rychlý vynucený roaming.

SCALANCE W747-1RR ethernetovský klient administrátorský modul se používá při mobilní komunikaci s vynuceným roamingem až s 8 zařízeními. Tím je možné vytvářet malé mobilní jednotky (max. s 8 zařízeními), které se pohybují v poli IWLAN. Jestliže jedno je jedno zařízení změněno ethernetovský klient rozpozná tuto změnu a spravuje novou adresu. Využití tohoto modulu je u mobilních vozíků apod. – viz. obr. 14.40.



Obr. 14.40: Využití IWLAN RCoax Kabelu při řízení vozíku.

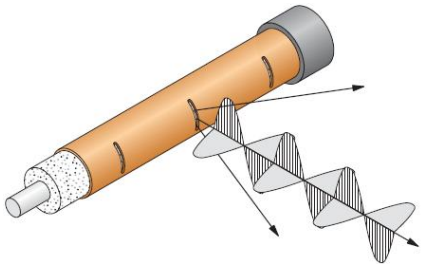
14.3.14 IWLAN RCoax Kabel

Tento typ kabelu využívá pásmo 2,4GHz a 5 GHz pro aplikace v IWLAN podle standardu IEEE 802.11 b/g nebo a/h. RCoax Kabel obr. 14.42 je stíněným kabelem, který dovoluje vysílat elektromagnetické vlny z kabelu do prostoru a také tyto vlny zpětně přijímat z okolí. [14-7]

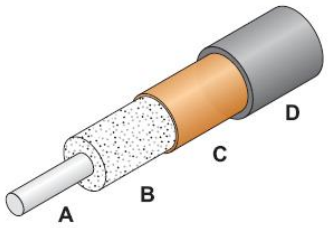
Koaxiální kabel se skládá obr. 14.42:

- Z vnitřní žíly A.
- Vnitřní dielektrikum B.
- Venkovní stínění C.
- Pláště kabelu chránící kabel před mechanickým poškozením D.

Plášť kabelu je v pravidelných délkách přerušován spolu s dielektrikem, aby příchozí vlny mohly vycházet z kabelu případně do kabelu vstupovat obr. 14.41. Venkovní stínění na povrchu dielektrika má za úkol odstranit rušení. Vycházející vlny se z kabelu šíří kolmo na kabel. Tento RCoax Kabel může být max. dlouhý 200 metrů na jeden segment. Maximální propustnost je přibližně 24Mbps což odpovídá standardu 802.11g.

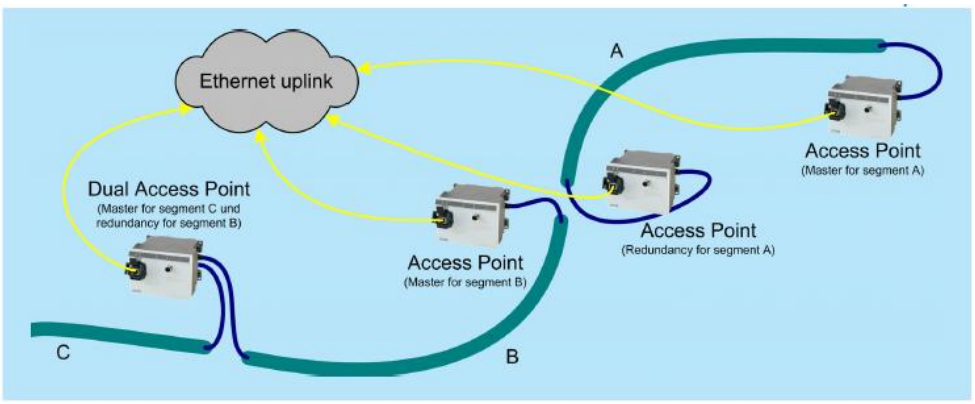


Obr. 14.41: RCoax Kabel.



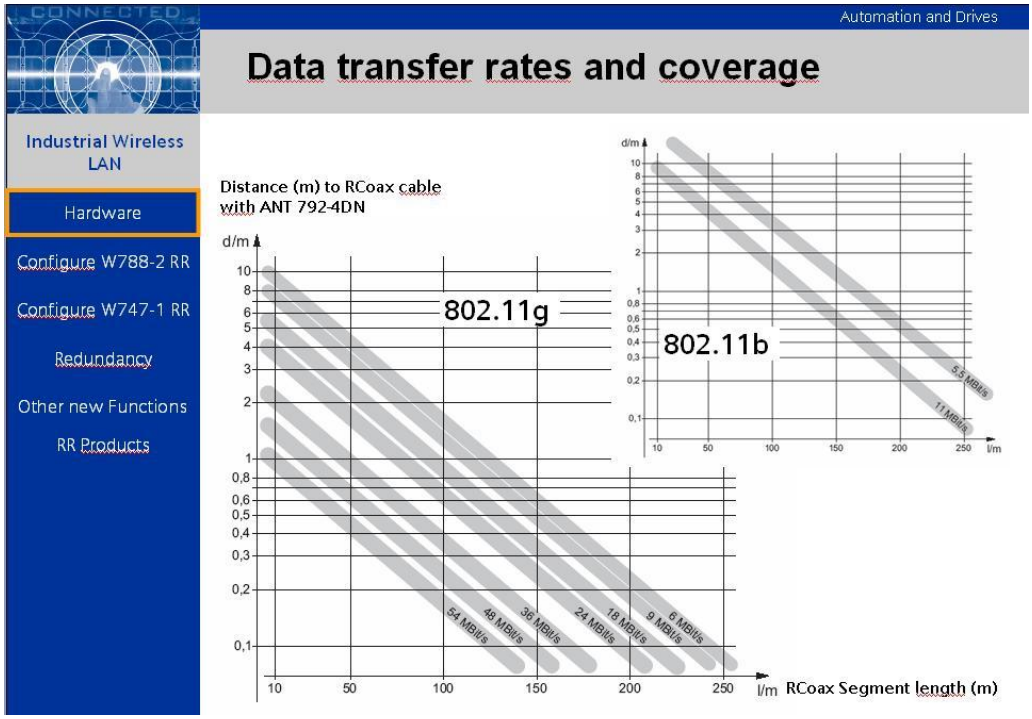
Obr.14.42: RCoax Kabel.

Opět i s tímto kabelem je možné vytvářet redundantní síť obr. 14.42.

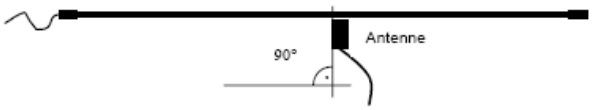


Obr. 14.43: Redundantní spojení.

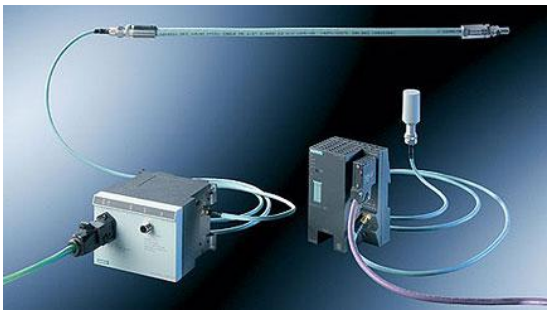
Protože tento kabel se ve spojení např. s nějakými dopravníky, které se pohybují v kolejišti, tak musí být mezi tímto kabelem a jednotkou IWLAN PB Linkem, která informace přenáší např. na jednotku ET200S minimální vzdálenost, protože s rostoucí vzdáleností, jak ukazuje následující obr. 14.44, také klesá přenosová rychlost. Samozřejmě, že také anténa, která je umístěna na IWLAN PB Link musí být kolmo umístěná na tento RCoax Kabel obr. 14.45. Kompletní spojení RCoax kabelu, který je připojen na přístupový bod SCALANCE W 7xx a také IWLAN PB Link s anténou je na obr. 14.46.



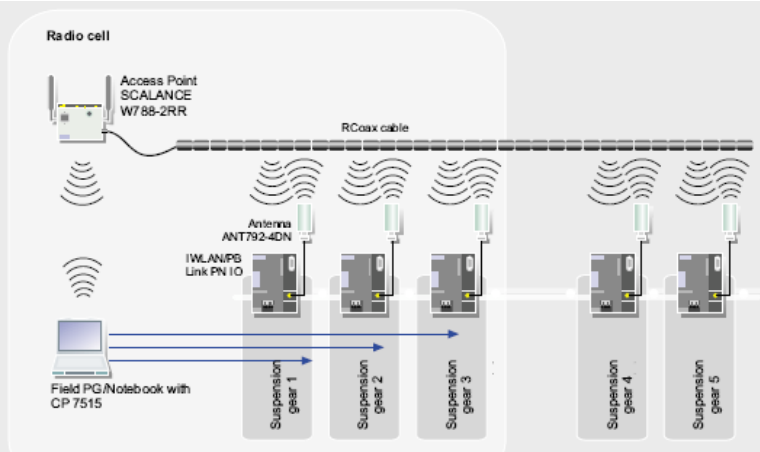
Obr. 14.44: Vliv vzdálenosti na přenosovou rychlost.



Obr. 14.45: Poloha antény k RCoax kabelu.



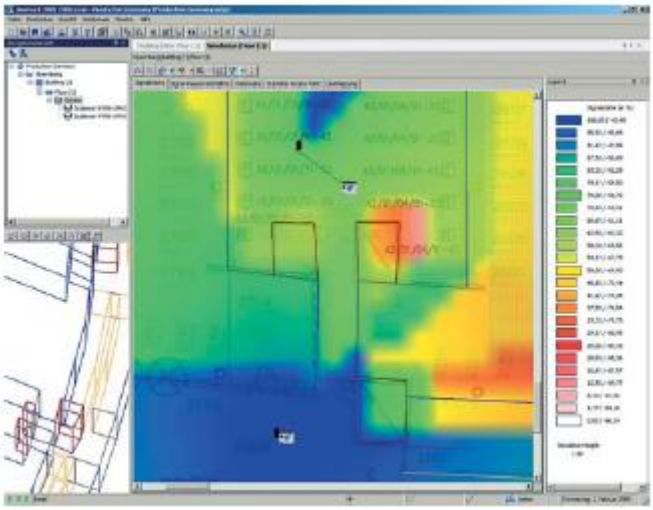
Obr. 14.46: Typická sestava pro komunikaci s RCoax Kabelem.



Obr. 14.47: Využití R-Coax Kabelu pro řízení pohybu.

Návrh bezdrátové technologie

Návrh bezdrátové technologie je možný již při výstavbě nějakého nového komplexu, kde bude tato bezdrátová technologie využita, protože je možné převést si výkresy půdorysu z Autocadu do software SINEMA E Lean (SIMATIC Network Manager Engineering), kterým je možné si ověřit, kde by bylo nejvhodnější umístění přístupových bodů a také jaká zde bude maximální možná přenosová rychlost.



Obr.14.48: SINEMA E Lean.

14.3.15 SCALANCE S

Moduly SCALANCE S jsou určeny pro nasazení v sítích průmyslový Ethernet a Profinet. Zabezpečení komunikačních sítí funguje na principu VPN (virtuální privátní síť – zabezpečený „tunel“) nebo (a) firewall (filtrace datové dopravy). [14-14, 14-15, 14-16, 14-17]

Moduly SCALANCE S zajistí:

- ochranu proti chybám operátora
- prevenci proti neautorizovanému přístupu

Pro vzdálenou správu systémů (Simatic S7, atd.) přes internet využíváme VPN tunel vytvořený buď mezi dvěma moduly SCALANCE S nebo mezi jedním modulem a softwarovým VPN klientem.

Ochranu před přetížením komunikační sítě (omezení všesměrových zpráv, apod.) např. po propojení s podnikovou sítí vytvoříme vhodným nastavením pravidel pro filtraci datové dopravy – firewall.

Přednostmi modulů SCALANCE S jsou především robustní průmyslové provedení, jednoduchá obsluha a bohatá funkční výbava. Před nasazením SCALANCE S zpravidla není potřeba žádná modifikace stávající sítě průmyslový Ethernet.

SCALANCE S602 firewall

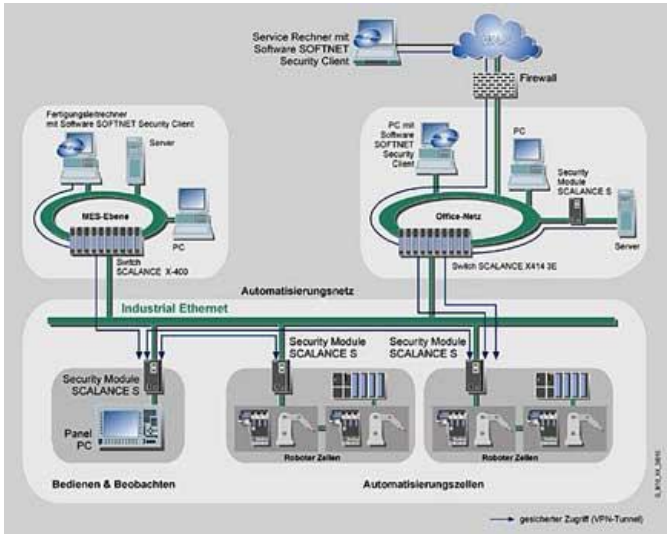
Podporuje routing, DHCP server, NAT (překlad síťových adres), NAPT (překlad portů), Syslog server pro archivaci událostí spojených s provozem modulu. Zálohování konfigurace na paměťové médium C-plug.

SCALANCE S612 a SCALANCE S613 VPN, firewall

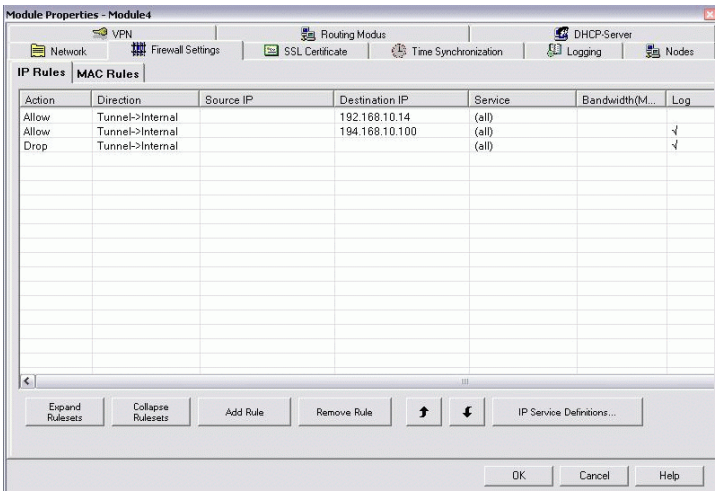
Podporují routing, DHCP server, NAT (překlad síťových adres), NAPT (překlad portů), Syslog server pro archivaci událostí spojených s provozem modulu, VPN. Zálohování konfigurace na paměťové médium C-plug.

SOFTNET Security client VPN klient

Software pro PC stanici umožňující vytvoření VPN „tunelu“ mezi touto stanicí a modulem SCALANCE S612 nebo SCALANCE S613.



Obr. 14.49: Příklad využití SCALANCE S.



Obr. 14.50: Nastavení IP adresy SCALANCE X-200.

Tab. 14.3:Přehled možných nastavitelných zabezpečení

Funkce	**S602**	**S612 V1**	**S612 V2**	**S613 V1**	**S613 V2**
Firewall	ANO	ANO	ANO	ANO	ANO
NAT/NAPT router	ANO	NE	ANO	NE	ANO
DHCP server	ANO	NE	ANO	NE	ANO
Network Syslog	ANO	NE	ANO	NE	ANO
IPSec tunel (VPN, Virtual Private Network)	NE	ANO	ANO	ANO	ANO
Softnet Security Klient (in flat network only)	NE	ANO	ANO	ANO	ANO

Shrnutí pojmů

Komunikační sběrnice **CAN** se velice často používá v automobilových aplikacích, ale lze se s ní setkat i v průmyslových systémech. U sběrnice CAN jsou definovány dvě nejnižší vrstvy podle modelu ISO-OSI, tedy fyzická a linková. Existuje řada rozšíření sběrnice CAN, kde je navíc definována nejvyšší, aplikační vrstva.

Pro správnou činnost sběrnice je nutné, aby fyzická vrstva realizovala **funkci logického součinu**. U sběrnice je použita **metoda náhodného přístupu ke komunikačnímu médiu - CSMA/CD w/ AMP**. Sběrnice CAN používá pět typů přenosových rámců a pět metod detekce chyby.

Komunikační sběrnice **Ethernet (resp. Industrial Ethernet)** se používá ve vyšších úrovních počítačově řízené výroby, protože její nedeterminističnost zabraňuje použití v nižších vrstvách. Výhodou této sběrnice jsou její velice dobré výkonové vlastnosti a všeobecná rozšířenost.

V průmyslových aplikacích se v současné době začínají stále více používat sběrnice na bázi Ethernetu, které jsou deterministické a je možné je využívat i v nižších vrstvách řízení. Příkladem takové sběrnice je **Profinet**.

Profinet je tedy otevřený, na výrobci nezávislý komunikační standard pro všechny úrovně průmyslové automatizace, založený na průmyslovém ethernetu.

Pro realizaci průmyslových komunikačních systému na bázi Ethernetu nebo Profinetu je vhodné (a někdy i nezbytné) používat pasivní a aktivní komponenty sběrnice navržené pro průmyslové aplikace. Příkladem mohou být switche **Scalance** a další síťové prvky firmy Siemens.

Kontrolní otázky

1. Jaké programovatelné automaty jsou v základní řadě firmy B&R?
2. Popište základní vlastnosti automaty B&R System 2003.
3. Popište základní vlastnosti automaty B&R X20.
4. Jaké jsou možnosti rozšíření programovatelných automatů B&R System 2003.
5. Jaké jsou možnosti rozšíření programovatelných automatů B&R X20.
6. Jaké existují generace procesorových jednotek programovatelných automatů B&R a čím se liší?
7. Jaké programovací jazyky lze využít v software Automation Studio pro psaní řídicích aplikací?
8. Popište deterministický multitasking.
9. Co je to tasková třída a jaké jsou jejich typy?
10. Co je to task?
11. Jak funguje čtení vstupů a zápis výstupů u programovatelných automatů B&R.
12. K čemu slouží nástroj Datové reference v prostředí Step7?
13. Jakým způsobem je možné monitorovat stav programu ve Step7?
14. Jakým způsobem je možné monitorovat a měnit data ve Step7?
15. K čemu slouží nástroj PLCSIM?
16. Jaké nástroje jsou ve Step7 k dispozici pro diagnostiku chyb program a hardware?
17. K čemu slouží Logbook u programovatelných automatů B&R?
18. Jak lze v B&R Automation Studiu monitorovat program?
19. Jak lze v B&R Automation Studiu monitorovat a měnit data?
20. K čemu slouží v B&R Automation Studiu nástroj Profiler?
21. K čemu slouží v B&R Automation Studiu nástroj Trace?
22. Popište možnosti realizace PID zpětnovazebního řízení u programovatelných automatů Simatic.
23. Popište realizaci PID řízení pomocí integrovaných funkcí z knihovny PID Control.
24. Popište realizaci PID řízení pomocí integrovaných funkcí z knihovny PID Temperature Control.
25. Popište možnosti automatického nastavení konstant u funkcí z knihovny PID Temperature Control.
26. Popište nástroje Standard PID Control a Modular PID Control.
27. Popište základní vlastnosti nástroje PID Self Tuner.
28. Co jsou to funkční moduly, jak je lze využít pro zpětnovazební řízení?
29. Jaké jsou možnosti realizace PID zpětnovazebního řízení u programovatelných automatů B&R?
30. Popište funkce LCPID(), LCPIDpara() a LCPIDAutoTune().
31. Popište postup při návrhu řídicích aplikací.
32. Co je to systémová analýza ?

33. Jaké jsou prostředky systémové analýzy?
34. Jaké modely je možné využít při návrhu řídicí aplikace?
35. Co jsou to Petriho sítě?
36. Z jakých prvku se skládají P/T Petriho sítě.
37. Jaké jsou základní vlastnosti komunikační sběrnice AS Interface?
38. K čemu se používá sběrnice AS Interface?
39. Popište fyzickou vrstvu sběrnice AS Interface.
40. Popište linkovou vrstvu sběrnice AS Interface.
41. Jaké jsou základní vlastnosti komunikační sběrnice Profibus?
42. Jaké jsou typy sítě Profibus?
43. K čemu se používá sběrnice Profibus?
44. Popište fyzickou vrstvu sběrnice Profibus DP.
45. Popište linkovou vrstvu sběrnice Profibus DP.
46. Jakou používá Profibus-DP přístupovou metodu?
47. Popište fyzickou vrstvu sběrnice Profibus PA.
48. Popište základní vlastnosti sběrnice CAN.
49. Popište fyzickou vrstvu sběrnice CAN.
50. Popište linkovou vrstvu sběrnice CAN.
51. Jak funguje metoda přístupu ke komunikačnímu médiu použitá u sběrnice CAN?
52. Popište základní vlastnosti sběrnice Ethernet.
53. Jak funguje metoda přístupu ke komunikačnímu médiu použitá u sběrnice Ethernet?
54. Jaký je formát rámce u sběrnice Ethernet.
55. Popište základní vlastnosti sběrnice Profinet.
56. Jaké typy komunikačních kanálů existují u sběrnice Profinet?
57. Co je to Profinet IO a k čemu slouží?
58. Co je to Profinet CBA a k čemu slouží?
59. Jaké sběrnicové topologie umožňuje Profinet?
60. K čemu slouží zařízení označené jako Siemens Scalance?

Další zdroje

14-1. Siemens: SIMATIC Communication with SIMATIC System Manual, 12/2005, EWA 4NEB 710 6075-02 03

14-2. Siemens: SIMATIC PROFINET System Description System Manual, 10/2006 A5E00298288-03

14-3. Siemens: SIMATIC PROFINET IO From PROFIBUS DP to PROFINET IO Programming Manual, Edition 07/2004

14-4. Siemens: SIMATIC SCALANCE Industrial Ethernet SCALALNCE X-100 and SCALANCE X-200 Product Line, Comminssioning Manual, 10/2005, A5E00349864 Release 6

14-5. Siemens: SIMATIC SCALANCE Industrial Ethernet SCALALNCE X-100 and SCALANCE X-200 Product Line, Operating instructions, 09/2006, A5E00349864 Release 7

14-6. Siemens: SIMATIC Products for Totally Integrated Automation and Micro Automation, Katalog ST 70.2005

14-7. Siemens: SIMATIC NET RCoax System Manual Rcoax Cable 2,4GHz, RCoaxCable 5GHz, release 01/2006, C79000-G8976-C189-04

14-8. Siemens: Produkt Analysis: Siemens SCALANCE W Rapid Roaming for Real-Time Applications in Wireless LAN, ComConsultResearch, Cornelius Hochel-Winter

14-9. http://cs.wikipedia.org/wiki/Quality_of_Service

14-10. http://cs.wikipedia.org/wiki/Ad-hoc

14-11. http://cs.wikipedia.org/wiki/Simple_Network_Management_Protocol

14-12. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, FAKULTA INFORMAČNÍCH TECHNOLOGIÍ, Seminární práce do kurzu CC3 – CCNA3-modul 7 Spanning Tree Protocol.

14-13. http://www.profinet.com/

14-14. www.siemens.com/scalance

14-15. http://www1.siemens.cz/ad/current/prezentace/as/index.php?mi=1

14-16. http://support.automation.siemens.com

14-17. http://www.automation.siemens.com/profinet/index_76.htm

14-18. http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=23&clanekID=34

14-19. http://www.cs.vsb.cz/grygarek/SPS/

Řešená úloha 14.1.

ŘEŠENÍ SEMESTRÁLNÍHO PROJEKTU

15. SEZNAM ZKRATEK

10 Base-T/F	Standard pro Ethernet který dovoluje přenos až do 10 Mbps
100 Base-T/F	Standard pro Ethernet který dovoluje přenos až do 100 Mbps
1000 Base-T/F	Standard pro Ethernet který dovoluje přenos až do 1000 Mbps
A/D	(Analog to Digital) A/D převodník.
AB	(Automation Basic).
AC	Střídavé napětí
AI	(Analog Input) Analogový vstup
AO	(Analog Output) Analogový výstup
AR	(Automation Runtime).
AS	(Automation Studio).
ASCII	(American Standard Code for Information Interchange).
ASI	(Actuator Sensor Interface) .
B&R	Bernecker Rainer
BCD	(Binare Coded Decimal).
CAN	(Control Area Network)
CBA	(Component Based Automation).
COM	(Component Object Model) Programovací standart, slouží k distribuci objektů mezi procesy na jednom PC
CP	(Communications Procesor) Komunikační procesor
CPN	(Colored Petri Nets) Barevné Petriho sítě.
CPU	(Central Procesor Unit).
CRC	(Cyclic Redundancy Code).
CSMA/CD	(Carrier Sense Multiple Access/ Collision Detect) Metoda náhodného přístupu na médium standartu Ethernet.
D/A	(Digital to Analog) D/A převodník.
DB	Datový blok.
DC	Stejnosměrné napětí
DCOM	(Distributed Component Object Model) Rozšíření modelu objektové komunikace COM o komunikaci po síti.
DCP	(Discovery and Basic Configuration) Nástroj výrobce, který definuje přiřazení parametrů a IP adresy periferii.
DCS	(Distributed Control System) Distribuovaný řídicí systém.
DHCP	(Dynamic Host Configuration Protocol) Metoda pro dynamické přidělování IP adres v určitém segmentu sítě.
DI	(Digital Input) Digitální vstup.
DLE	(Data Link Escape).
DO	(Digital Output) Digitální výstup.

DP	(Distributed peripheral) Distribuované periferie.
DRAM	(Dynamic RAM).
DSR	Distribuovaný systém řízení.
ERTEC	ERTEC - Enhanced Real Time Ethernet Controller
	Nové mikroprocesory řady ERTEC 200 a ERTEC 400, které se využívají v oblasti automatizace – PROFINET, kde se vyžaduje IRT operace.
Ethernet	Komunikační standart s metodou přístupu CSMA/CD definovaný na první a druhé vrstvě modelu ISO/OSI.
ETX	(End of Text).
EXC	(Exception Task Class).
FB	(Function Block) Funkční blok.
FBD	Instruction List, zápis programu pomocí hradlového zápisu.
FC	(Function) Funkce.
FDM	(Frequency Division Multiplexing).
FM	(Function Module) Funkční modul.
FPROM	(Flash PROM).
FTP	(File Transfer Protocol) Protokol pro přenášení souborů po síti.
GSD	(General Station Description) Soubor psaný v jazyce XML, který obsahuje ID zařízení, parametry komunikace, typ zařízení, konfigurační data, údaje o diagnostice atd.
HMI	(Human Machine Interface) Rozhraní mezi člověkem a strojem, obecný výraz pro operátorská zařízení na procesní úrovni průmyslové automatizace.
HTML	(Hypertext Markup Language) Jazyk pro zápis internetových stránek.
HTTP	Protokol pro sdílení internetových stránek po sítích TCP/IP.
HW	Hardware.
CHR	(Chiena, Hronese and Reswick) Metoda automatického nastavení PID regulátoru
I/O	Vstupně/výstupní, případně vstupy/výstupy.
I/O – Controller	Kontrolér s rozhraním Profinet IO, na kterém probíhá řídicí program.
I/O – Device	Zařízení přidružené ke kontroléru přes rozhraní Profinet IO.
I/O – Supervisor	Programovatelné zařízení s rozhraním Profinet IO pro diagnostiku a zprovoznění automatizovaných celků.
IC	(Industrial PC).
IEC	(International Electrotechnical Commission).
IEEE	(Institute of Electrical and Electronics Engineers).
IL	(Instruction List) Zápis programu pomocí instrukcí.
IP Address	(Internet Protocol Address).

IPC	(Industrial PC).
IRT	(Isochronous Real Time) Komunikační kanál určený pro úlohy s nejvyššími požadavky na přesnost odezvy, jako jsou aplikace pro synchronizaci pohonů. Kanál umožňuje komunikaci s periodou volání 1ms a přesností synchronizace 1µs.
ISO-OSI	(International Standards Organization - Open System Interconnection).
KPA	Kompaktní programovetný automat.
LD	(Ladder Diagram) Zápis programu pomocí liniového schématu.
LLC	(Link Layer Control).
MAC	(Media Access Control).
MAC address	(Medium Access Control Address) Každé zařízení pro PROFINET má přiřazenu jednoznačnou MAC adresu.
MES	(Manufacturing Execution Systems) Skupina softwarových aplikací pro přenos a vizualizaci dat nadřazených podnikových informačních systémů, na nejvyšší úrovni průmyslové automatizace.
MMC	(Micro Memory Card).
MPA	Modularní programotelný automat
MPI	(Multi Point Interface).
MRP/ERP	(Manufacturing/Enteerprice Resource Planning).
NRZ	(Non Return to Zero).
OB	(Organization Block) Organizační blok.
OLE	(Object Linking and Embedding) Mechanizmus pro vytváření a editaci dokumentů s obsahem objektů, které byly vytvořeny různými aplikacemi
OPC	(OLE for Process Control) Rozhraní specifikované v roce 1996 pro výměnu dat mezi vizualizačními aplikacemi v automatizaci.
PCC	(Programmable Computer Controller) Označení programotelných automatů používané firmou B&R.
PCI	(Peripheral Component Interconnect).
PG	Programátorská stanice.
PID Algorithm	(Proportional-Derivative-Integration Algorithm).
PLC	(Programmable Logic Controller) Programovatelný automat.
PN	(Petri Nets) Petriho sítě.
POU	(Program Organization Unit).
PROM	(Programmable ROM).
Proxy	Zástupce objektu v objektovém modelu, vytváří rozhraní mezi protokolem Profinet a jinými protokoly.
PTN	(Place-Transition Petri Nets).
PtP	(Point to Point).

PVI	(Process Visualization Interface) Obecné rozhraní pro komunikaci s programotelnými automaty B&R.
QoS	(Quality of Service) Zajišťuje rovnoměrné vyvažování zátěže sítě s ohledem na druh přenášených dat , spravedlivě rozděluje konektivitu mezi jednotlivé zákazníky dle nastavených priorit a zabraňuje přetížení sítě.
RAM	(Random Access Memory).
RPC	(Remote Procedure Call) Rozhraní pro volání funkcí na vzdálených stanicích.
RS	Klopný obvod.
RS232	Sériový komunikační standard.
RT	Real-Time.
RZ	(Return to Zero).
SCADA	(Supervisory Control And Data Acquisition).
SFB	(System Function Block) Systémový funkční blok.
SFC	(System Function) Systémová funkce.
SFC	(Sequential Function Chart) Grafický jazyk pro programování sekvenčních úloh.
SG3, SG4	(System Generation).
SM	(Signal Module) Slouží jako rozhraní mezi řízeným procesem a PLC.
SMTP	(Simple Mail Transfer Protocol).
SNMP	(Simple Network Management Protocol) Je součástí sady internetových protokolů. Slouží k potřebám správy sítí. Umožňuje průběžný sběr nejrůznějších dat pro potřeby správy sítě, a jejich následné vyhodnocování.
SoftPLC	(Software Programmable Logic Controller).
SRAM	(Static RAM).
SRT	(Soft Real Time) realtimový kanál pro časově kritická data procesů, užívaný v úlohách průmyslové automatizace. Je implementován softwarově na automatech s rozhraním Profinet.
ST	(Structured Text) Zápis programu pomocí strukturovaného textu.
STEP7	Nástroj pro vytváření programů pro PLC v různých jazycích pro SIMATIC S7.
STL	(Statement List) Zápis programu pomocí instrukcí.
STX	(Start of Text).
SW	Software.
TC	(Task Class) Tasková třída
TCP	Transportní protokol pro komunikaci po síti.
TDM	(Time Division Multiplexing).
UDP	(User Datagram Protocol).

UDT	(User Data Type) Uživatelsky definované datové typy.
USB	(Universal Serial Bus).
VLAN	(Virtual Local Area Network) Nádstavba ethernetového rámce ve standartu 802.1q pro definici priority rámce a čísla virtuální podsítě, do které náleží
VNC	(Virtual Network Computing).
WIFI	(Wireless Fidelity) Zkratka pro standart IEEE 802.11, který slouží k bezdrátovému spojení výpočetních systémů do lokálních sítí.
XML	(Extensible Markup Language).