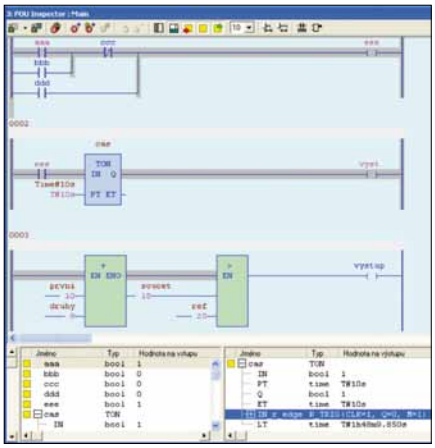
Esperanto programátorů PLC: programování podle normy IEC/EN 61131-3 (část 4)

Čtvrtý díl seriálu o programování PLC stručně seznamuje s programovacími jazyky, které definuje norma IEC EN 61131-3. Grafické jazyky LD a FBD jsou popsány velmi stručně. Podrobněji jsou přiblíženy textové jazyky IL a ST. Vzhled programů je ilustrován ukázkami obrazovek. Podrobný popis jazyků bude uveden v rámci komentáře k řešeným příkladům, které budou následovat. Jen přehledově je popsán nástroj sekvenčního programování (SFC). Zmíněn je rovněž grafický jazyk CFC, který sice není popsán normou, ale je velmi blízký jazyku FBD. Na závěr je uvedena diskuse o možnostech a účelnosti použití jednotlivých jazyků.



Obr. 9. Ukázka obrazovky s programem v jazyku LD v režimu monitorování běžícího programu se zviditelněním „vodivých cest“

Programovací jazyky

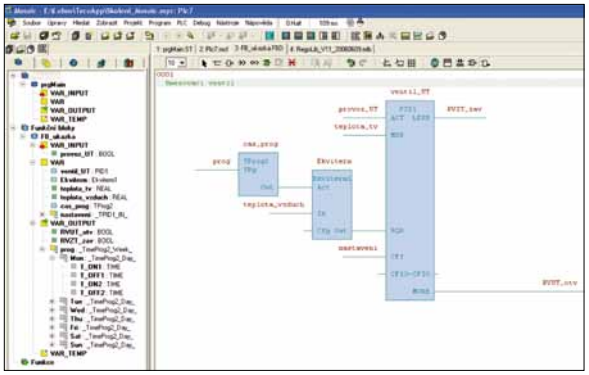
Norma kodifikuje čtyři typy programovacích jazyků pro PLC – dva grafické (LD, FBD) a dva textové (IL, ST). Jako pátý je někdy uváděn programovací nástroj SFC (*Sequential Function Chart*). Je to prostředek pro strukturovaný popis sekvenčních úloh, nadřazený ostatním jazykům. Někdy je mylně nazýván jako jazyk Grafcet, který je sice velmi podobný SFC (SFC je od něj odvozen), ale je obecnější a je popsán vlastní normou. Některé programovatelné automaty (a jejich vývojové systémy) umožňují programování ještě v dalších jazycích, např. v některé verzi jazyka C nebo v již zmíněném prostředku Grafcet.

Běžně je umožňováno používat programovací jazyky pro starší generace PLC stejného výrobce. Jejich dostupnost má význam především pro opravy a aktualizace programů starších zakázek s historickými PLC, které byly naprogramovány právě v těchto tradičních firemních jazycích. Je to i projev respektu k zákazníkům, kterým tyto jazyky vyhovují víc než jazyky podle normy – jsou na ně zvyklí a nemají chuť se učit nové zásady programování. Někdy je upřednostňují proto, že starší jazyky jsou více přizpůsobeny provedení hardwaru PLC nebo k řešení převažujících úloh.

Nabízeny jsou i zcela nové programovací jazyky. Někdy jde o experimenty lokálního významu, ale některé z nových jazyků se staly velmi populárními a rozšířenými. Patří sem i grafický jazyk CFC (*Continuous Function Chart*). V podstatě je to obdoba jazyka funkčních bloků (FBD) s volnějšími zásadami kreslení blokových schémat. Vytváření programu (kreslení blokového schématu) je jednodušší, program je přehlednější. Je pravděpodobné, že po čase bude jazyk CFC „adoptován“ i normou IEC EN 61131-3 nebo se stane její „neformální součástí“.

Jazyk kontaktních schémat – LD

V jazyku kontaktních schémat (LD, doslovně přeloženo „jazyku příčkových diagramů“) má program formu kontaktního schématu a je obdobou liniového schématu v elektrotechnice. Kontaktní schéma je ohraničeno dvěma svislými čarami (sběrnicemi) a jednotlivé funkce jsou realizovány vodorovnými kontaktními obvody („příčkami“ v pomyslném v žebříku) s kontakty zapojenými v sério-paralelních kombinacích. Kontakty reprezentují vstupní, výstupní nebo vnitřní proměnné. Mezi kontakty mohou být umístěny i obdélníkové značky funkčních bloků (jsou-li volány). Nejčastěji to jsou generátory impulzů od hran, klopné obvody typu set a reset, čítače a časovače nebo jakékoliv jiné. Výstupy jsou zde označovány jako „cívky“ – mohou mít charakter běžné cívky relé (s přímou nebo negovanou funkcí) nebo paměťového (klopného) obvodu s funkcí přednostního zápisu (obvod *set*) nebo přednostního nulování (obvod *reset*). Základní prvky kontaktního schématu se označují s využitím textových znaků uvedených v *tab. 5.*



Obr. 10. Ukázka obrazovky s programem regulační úlohy v jazyku FBD v edičním režimu při zadávání schématu; typ funkčního bloku se zvolí a umístí podobně jako při zadávání programu v LD

Tab. 5. Základní prvky jazyka kontaktních schémat (LD)

Funkce	Anglický název	Symbol
spínací kontakt	*open contact*	-| |-
rozpínací kontakt	*closed contact*	-|/|-
cívka s přímou funkcí	*coil*	-( )-
cívka s negovanou funkcí	*negated coil*	-(/)-
cívka s funkcí *set*	*set coil*	-(S)-
cívka s funkcí *reset*	*reset coil*	-(R)-

Jak pohodlně lze kontaktní schéma vytvořit, závisí na konkrétním vývojovém systému, který je pro programování PLC použit. Ukázka obrazovky v režimu zadávání kontaktního schématu (ediční režim vývojového systému Mosaic) je na *obr. 1* v Automě č. 10/2011, str. 60). Na liště nad polem programu jsou symboly kontaktů a cívek. Zvolený symbol lze myší „stáhnout“ a umístit na zvolené místo. Automaticky se pak nakreslí spoje a nabídne se dialogové okno, ve kterém lze odpovídající proměnnou pojmenovat a určit její datový typ a další specifikace. Při stisknutí symbolu funkčního bloku (zeleně) se otevře okno s nabídkou standardních funkcí a funkčních bloků (jsou vyjmenovány v kapitole o funkcích a funkčních blocích v Automě č. 11/2011, str. 44). S těmi se naloží podobně jako se symboly kontaktů nebo cívek. Na *obr. 9* je uveden stejný program v režimu monitorování běžícího programu, na kterém je zviditelněna vodivá cesta. Detailní

postup programování v jazyku LD zde nebude probírán, bude patrný ze sady příkladů v dalším textu.

Jazyk funkčních bloků – FBD

Jazyk funkčního blokového schématu FBD (*Function Block Diagram*) neboli jazyk funkčních bloků je další grafický jazyk, který popisuje norma. Program (blokové schéma) obsahuje obdélníkové značky funkcí a funkčních bloků, které jsou propojeny spojnicemi (spojovacími vodiči). Fragment programu ve FBD je uveden na *obr. 10* a *obr. 11*. Více podrobností bude opět uvedeno později na příkladech.

Jazyk seznamu instrukcí – IL

Program v textovém jazyku IL (*Instruction List*) neboli v jazyku seznamu instrukcí se sestavuje z textových zkratek několika základních instrukcí – mnemokódů. Je obdobou programovacích jazyků typu assembler pro programování počítačů nebo mikrořadičů (mikroprocesorů). Většina operací se v jazyku IL provádí podle schématu:

```
výsledek := výsledek OPERATOR operand
```

Požadovaná operace se tedy provádí s poslední hodnotou výsledku (aktuálním stavem střádače) a s programovaným operandem. Výsledek je opět uložen ve střádači. Pro realizaci logických úloh má jazyk IL k dispozici mnemokódy instrukcí pro různé typy operací (*tab. 6*). Některé z instrukcí lze doplnit modifikátorem, který stanoví, že operandem bude negovaná proměnná (N, *negation*), nebo zahajuje odloženou operaci (*defered operation* – realizuje otevírací závorku výrazu). Logické operátory jsou definovány pro operandy s rodovým typem dat ANY_BIT, tedy pro objekty typu BOOL, BYTE, WORD a DWORD (viz část 3 v Automě č. 11/2011, str. 46). Pro logické operace jsou tedy k dispozici standardní operátory s přípustnými modifikátory (příkazy) z *tab. 6*.

Například pro logický součin AND jsou k dispozici čtyři typy operátorů:

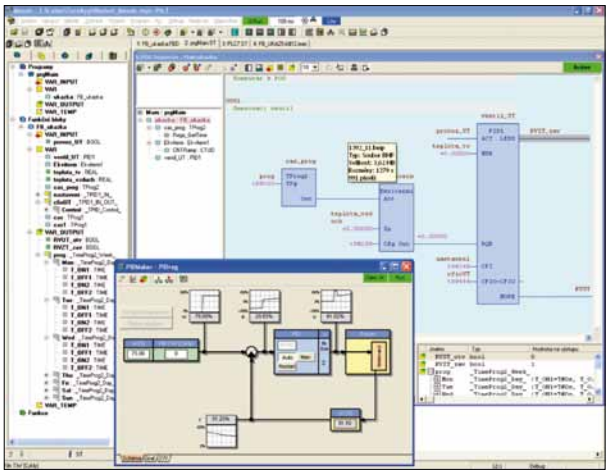
AND značí logický součin a v programu ho lze použít takto:

```
LD    operand1
AND   operand2
ST    soucinA
```

Program realizuje operaci:
soucinA := operand1 AND operand2

AND(značí logický součin a otevření závorky (zahájení odložené operace) a v programu ho lze použít takto:

```
LD    operand1
AND(  operand2
OR    operand3
)
ST soucinB
```



Obr. 11. Program regulační úlohy z obr. 10 v režimu běžícího programu se zviditelněním vodivých cest binárních systémů a s využitím nástroje PIDMaker pro zobrazení přechodového děje regulačního procesu

Tento program realizuje operaci:
soucinB := operand1 AND (operand2 OR operand3)

ANDN značí logický součin s negovaným operandem a v programu ho lze použít takto:

```
LD    operand1
ANDN  operand2
ST soucinC
```

Tento program realizuje operaci:
soucinC := operand1 AND NOT operand2

ANDN(značí logický součin s negovaným operandem a otevřením závorky a v programu ho lze použít takto:

```
LD    operand1
ANDN( operand2
OR    operand3
)
ST soucinD
```

Tento program realizuje operaci:
soucinD := operand1 AND (NOT operand2 OR operand3)

Pro výpočetní (numerické) operace jsou k dispozici operandy rodového typu ANY_NUM (tedy pro typy INT, SINT, DINT, UINT, USINT, UDINT, REAL, LREAL) s operátory a modifikátory podle *tab. 7*.

Tab. 6. Operátory a modifikátory pro logické operace (pro datový typ ANY_BIT) v jazyku IL

Operátor	Modifikátor	Popis funkce
LD	N	LD (*load*, sejmi) – nastaví výsledek podle aktuální hodnoty adresovaného operandu (sejme aktuální hodnotu adresovaného operandu do střádače)
AND	N, (	booleovský součin AND
OR	N, (	booleovský součet inkluzivní OR
XOR	N, (	booleovský součet exkluzivní XOR
ST	N	ST (*store*, ulož) – uložení aktuálního výsledku do adresované proměnné
S	*set* (podmíněné nastavení)	nastaví programovanou proměnou do log.1 (TRUE), jestliže je výsledek jedničkový, jinak nic
R	*reset* (podmíněné nulování)	nastaví programovanou proměnou do log. 0 (FALSE), jestliže je výsledek jedničkový, jinak nic
)	vyhodnocení odložené (*defered*) operace	uzavírací závorka výrazu, vyčíslení výrazu

Tab. 7. Operátory a modifikátory pro datový typ ANY_NUM v jazyku IL

Operátor	Modifikátor	Popis funkce
LD	N	*load* – přečtení adresovaného operandu do výsledku (akumulátoru)
ST	N	*store* – uložení výsledku do adresované proměnné
ADD	(	sčítání – přičte operand k výsledku
SUB	(	odčítání – odečte operand od výsledku
MUL	(	násobení – výsledek vynásobí operandem
DIV	(	dělení – výsledek dělí operandem
GT	(	porovnání > (výsledek je větší než operand)
GE	(	porovnání ≥ (výsledek je větší nebo roven operandu)
EQ	(	porovnání na rovnost = (výsledek je roven operandu)
NE	(	porovnání na neshodu < > (výsledek je odlišný od operandu)
LE	(	porovnání ≤ (výsledek je menší nebo roven operandu)
LT	(	porovnání < (výsledek je menší než operand)
)		uzavírací závorka výrazu – vyhodnocení poslední odložené operace, vyčíslení výrazu

Operátory a modifikátory pro přechody v běhu programu (skoky, volání a návraty) jsou uvedeny v *tab. 8*.

Tab. 8. Operátory a modifikátory pro skoky, volání a návraty v jazyku IL

Operátor	Modifikátor	Popis
JMP	C, N	skok na návěští
CAL	C, N	volání funkčního bloku
Func_name		volání funkce
RET	C, N	návrat z funkce nebo z podprogramu

Modifikátor C označuje, že naznačená operace se má provést pouze tehdy, je-li aktuální výsledek roven log. 1 (TRUE), jestliže nebyl použit modifikátor N. Byl-li použit modifikátor N, naznačená operace se provede pouze tehdy, jestliže výsledek má hodnotu log. 0 (FALSE). Ukázka programu v jazyku IL je na *obr. 12*. Podrobněji budou čtenáři s programem seznámeni na příkladech.

Jazyk ST – strukturovaný text

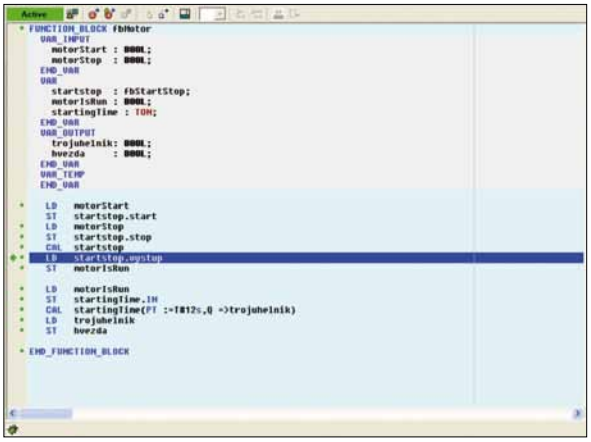
Jazyk strukturovaného textu (ST) je velmi výkonný textový jazyk. Kořeny má ve známých jazycích typu Pascal, Ada, C. Je objektově orientován. Obsahuje všechny důležité prvky moderního programovacího jazyka. Je velmi účinným nástrojem pro zápis náročných algoritmů a pro vytváření komplexních funkčních bloků. Významnou možností jazyka ST je aparát pro zápis výrazů. Výrazy provádějí vyčíslení hodnot dílčích výsledků pro uložení do adresovaných proměnných, vyčíslují i hodnoty podmínek pro další příkazy. Operandy použitelné pro tvorbu výrazů jsou uvedeny v *tab. 9*.

Vyhodnocení výrazu spočívá v aplikování operátorů na operandy s respektováním priorit operátorů. Nejdříve jsou aplikovány operátory s nejvyšší prioritou, pak se postupně používají operátory s klesající prioritou, dokud není vyhodnocení výrazu dokončeno. Operátory se stejnou prioritou se používají v pořadí, v kterém byly zapsány ve výrazu – tedy zleva doprava. Vlevo od výrazu je umístěn symbol přiřazení (dosazení, „pascalské rovnítko“) :=. Před ním je obvykle umístěn identifikátor proměnné, kam má být výsledek uložen. V *tab. 10* je uveden soubor příkazů jazyka ST.

Tab. 9. Operátory v jazyku ST

Operátor	Operace	Priorita
(,)	závorky	nejvyšší
**	umocňování	
- NOT	změna znaménka negace, logický doplněk	
* / MOD	násobení dělení modulo, zbytek po dělení	
+ –	sčítání odčítání	
<, >, ≤, ≥	porovnání, nerovnost	
= <, >	rovnost, shoda neshoda	
&, AND	logický (booleovský) součin AND	
XOR	výlučný (exkluzivní) součet	
OR	logický (booleovský, inkluzivní) součet	nejnižší

Ukázka programu v jazyku FBD byla uvedena na *obr. 8* v Automě č. 11/2011, str. 45. Podrobnější seznámení se základními možnostmi jazyka ST bude uvedeno na příkladech.



Obr. 12. Detail obrazovky s programem v jazyku IL

Tab. 10. Příkazy jazyka ST

Příkaz	Popis	Poznámka
:=	přiřazení	přiřazení hodnoty vyčíslené z výrazu na pravé straně do proměnné s identifikátorem na levé straně
	volání	volání (instance) funkčního bloku s předáváním parametrů
IF IF...ELSE... END_IF	příkaz výběru	výběr alternativy podmíněné vyčísleným logickým výrazem
CASE	příkaz výběru	výběr bloku příkazů podmíněný číselnou hodnotou vyčísleného výrazu
FOR	iterační příkaz	smyčka FOR s počáteční a koncovou hodnotou inkrementu
WHILE	iterační příkaz	smyčka WHILE s podmínkou ukončení smyčky na jejím začátku
REPEAR	iterační příkaz	smyčka REPEAT s podmínkou ukončení na jejím konci
EXIT	ukončení smyčky	předčasné ukončení iteračního příkazu
RETURN	návrat	opuštění právě vykonávané POU a návrat do volající POU
;	prázdný příkaz	

Nástroj sekvenčního programování – SFC

Detailní popis nástroje sekvenčního programování SFC (*Sequentional Functiom Chart*) se vymyká možnostem základního kurzu programování PLC. Jen pro hrubou představu je na *obr. 13* (poskytnutém Ing. Marií Martináskovou, Ph.D.) uvedena struktura jednoduché úlohy. Program má formu přechodového grafu objektu, který modeluje sekvenční chování. Je zobecněním konečného automatu Mooreovat typu, kombinového s Petriho sítí. I bez znalosti teorie lze graf snadno interpretovat. Obdélníky představují kroky (Step). Krok je časově oddělená situace, pro kterou je charakteristické určité chování, popsané v rámci přiřazené akce (zde nejsou akce uvedeny). Kroky jsou spojeny spojnicemi (hranami). Na nich jsou krátkými vodorovnými čárkami znázorněny přechody (*transitions*). Přechod z aktuálního kroku do následného cílového kroku se uskuteční při splnění podmínky (pravdivost proměnné typu BOOL, která odpovídá podmínce přechodu, má hodnotu log. 1 (TRUE)). Při přechodu se aktivuje následný krok a předchozí krok se deaktivuje. Stav grafu je určen jako soubor aktivních kroků a je výslednicí akcí jednotlivých kroků.

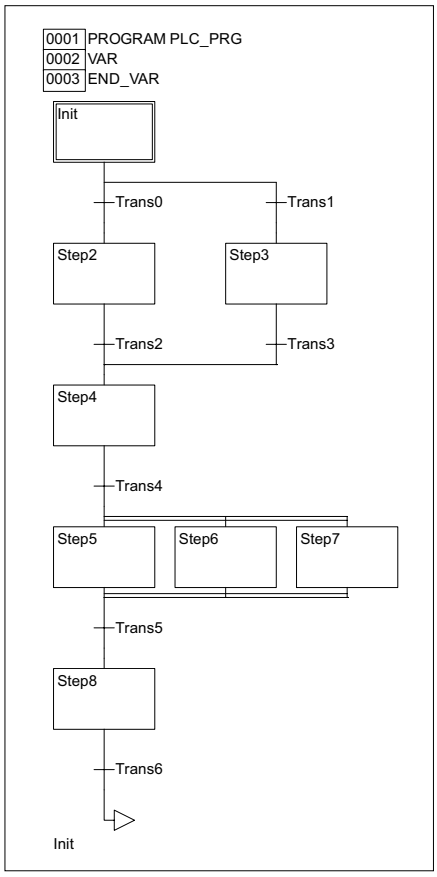
Jazyk CFC

CFC (*Continuous Function Chart*) je grafický jazyk, který je obdobou jazyka blokových schémat. Je poměrně populární, ač není kodifikován normou. Při jeho implementaci je ale obvykle využíván aparát normy (používání POU, definice datových typů a proměnných apod.). Pro uživatele jsou příjemné volnější zásady pro kreslení blokových schémat. Z toho vyplývá snadnější programování (kreslení) a přehlednější program. Ukázka je uvedena na *obr. 14*.

Používání a oblíbenost jazyků

Každý z programovacích jazyků má své příznivce a odpůrce a lze parafrázovat rčení, že „není jazyk ten, který by se zalíbil programátorům všem“. Někdy o oblíbenosti jazyka rozhodují důvody čistě subjektivní, především způsob myšlení a vizualizace představ ve vědomí programátora. Důležitá je i dosavadní profese a praxe programátora i to, kde a jaký jazyk se poprvé naučil.

Například programátor PLC, který byl původně provozním elektrikářem nebo elektro-



Obr. 13. Ukázka struktury programu v programu SFC (zdroj: Ing. Marie Martinásková, Ph.D.)

instalatérem, bude nepochybně preferovat jazyk kontaktních schémat (LD). Tento typ jazyka je také nejstarší z jazyků pro PLC a přispěl k úspěšnému rozšíření PLC. Stejný jazyk bude vyhovovat i údržbářům strojů a automatizovaných technologií pro jeho názornost, ale i pro možnost „zviditelnit vodivé cesty v kontaktním schématu“. Tuto možnost nabízejí některé vývojové systémy jako účinný prostředek diagnostiky, který dovoluje snadno určit příčinu závady v zařízení bez hlubšího pochopení funkce programu. Jazyk LD vyhovuje i mnoha konstruktérům elektrické výzbroje strojů (např. pro řešení algoritmu přizpůsobovacích obvodů stroje v PLC části systémů CNC pro obráběcí či tvářecí stroje nebo roboty) a pro řízení jiných mechanismů a mechatronických systémů. Lze odhadnout, že jazyk LD je patrně nejrozšířenější z jazyků pro PLC.

Programátoři z řad návrhářů pevné logiky s integrovanými obvody budou patrně preferovat jazyk FBD nebo CFC pro možnost zapsat program jako logické schéma. Stejnou možnost uvítají i programátoři z oboru regulační techniky, kteří jsou zvyklí své představy ztvárňovat v podobě blokového diagramu regulačního obvodu nebo jiného algoritmu. Jazyk strukturovaného textu (ST) bude nepochybně vyhovovat programátorům zvyklým na programování počítačů v některém z vyšších textových jazyků a absolventům škol, kde se vyučuje programování ve vyšších textových jazycích. Jazyk seznamu instrukcí (IL) je pravděpodobně nejméně oblíbený a vyhovuje snad jen „zarytým asembleristům“ a těm, kteří v tomto jazyku začínali programovat a zvykli si na něj.

Překvapivě je málo používán nástroj SFC, popř. jazyk Grafcet, přestože jde o velmi výkonný prostředek, který zproduktivňuje programování, nutí programátory k systematickému postupu a minimalizuje programátorské chyby. Důvodem je především skutečnost, že SFC nabízí jen nemnoho výrobců PLC. Důvodem může být i vysoká cena produktu a někdy nepříliš kvalitní implementace (např. pomalost nebo nadbytečný objem přeloženého programu pro PLC). Nedostatečná je metodická podpora programování s využitím teorie konečných automatů a Petriho sítí ve výuce na školách i pro odborníky z praxe.

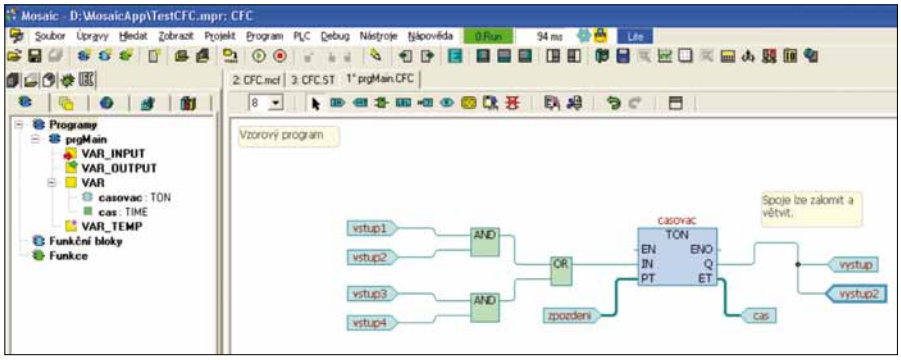
Jazyk LD – pro a proti

Volba různých jazyků pro PLC má ale své racionální důvody. Důvodem pro použití jazyka kontaktních schémat (LD) bývá především jeho jednoduchost, přehlednost a názornost u nepříliš složitých programů pro úlohy převážně logického typu. Nabízejí jej snad všichni výrobci PLC. Proto je vhodný pro výuku základů programování PLC. Ze stejných důvodů bude i tento kurz začínat blokem příkladů v jazyku LD. Při psaní obsáhlých nebo logicky komplikovaných programů ale už program v LD svou přehlednost ztrácí – obzvláště je-li program psán nesystematicky nebo má více autorů, často se mění a opravuje. Jeho nevýhodou je, že logické vazby jsou zprostředkovány jen prostřednictvím jmen proměnných, nikoliv viditelnými spoji (čarami). V obsáhlém programu lze „snadno zabloudit“, obzvláště je-li program (kontaktní schéma) neuspořádaný, stejné proměnné jsou nastavovány („oslovovány“) z různých míst programu a jsou vedeny vazby z konce na začátek (což je obvyklé zejména při nesystematických opravách a doplňcích programu). Jazyk LD je tedy vhodný jen pro nepříliš komplikované úlohy s algoritmy logického typu. Pro úlohy výpočetního typu (numerické úlohy) je LD zcela nevhodný.

FBD, CFC a používání funkčních bloků

Rovněž jazyky FBD a CFC jsou vhodné jen pro úlohy do určité složitosti. Zde to ovšem mohou být algoritmy nejenom logického, ale i numerického a smíšeného typu. Výhodou je, že program (blokové schéma) je obvykle sestavován z ověřených „prefabrikátů“ – funkčních bloků. Ty mohou být použity z knihoven standardních funkčních bloků podle normy, z nakoupených knihoven od výrobce PLC pro řešení určitých tříd úloh nebo vlastních uživatelských funkčních bloků. Používání funkčních bloků zproduktivňuje programování v každém z jazyků. Platí to pouze za předpokladu, že funkční bloky jsou správně vytvořeny, důkladně ověřeny, správně a úplně popsány a programátor je správně používá. Při částečném pochopení funkce a nesprávném volání nebo při použití nedostatečně ověřených bloků mohou vzniknout velmi nepříjemné a obtížně lokalizovatelné chyby.

To je důvod, proč si mnozí programátoři raději vytvářejí své vlastní knihovny funkčních bloků. Někdy bývá jednodušší a spolehlivější vytvořit si funkční blok (knihovnu nebo úsek programu) podle svých představ, než se snažit z nepřesného popisu pochopit funkci, která uživateli vyhovuje jen zčásti a nemá důvěru v její správnost. Programátoři všeobecně neradi přejímají cizí programy (a tedy i funkční bloky s komplikovanými a komplexními funkcemi) a raději programují „na zelené louce“. Důvody bývají zčásti subjektivní, zčásti věcné.



Obr. 14. Ukázka programu v grafickém jazyku CFC

Plocha pokrytá programem

Rovněž je třeba si uvědomit, že k zobrazení programu (blokového schématu) je obvykle zapotřebí plocha větší, než vyžaduje kontaktní schéma. Z tohoto pohledu je málo úsporný i textový jazyk seznamu instrukcí (IL), podobně jako jazyky typu assembler. Jazyk IL je nevýhodný i z jiných důvodů – každou, i jednoduchou funkci je zde nutné rozepsat až na úroveň základních instrukcí, což je pracné, zdlouhavé a nese to s sebou riziko chyb. Vzniklý program je dlouhý a nepřehledný, obtížně se v něm hledají a opravují chyby. Lze předpokládat, že norma tento jazyk při-

jala jen z tradice. Jazyk IL patří k nejstarším jazykům pro PLC, je téměř současníkem jazyka LD a většina historických programovacích přístrojů jej nabízela. Nyní již není mnoho důvodů pro jeho používání. Z tohoto pohledu je (může být) nejúspornější jazyk strukturovaného textu ST.

Výhody jazyka ST

Naproti tomu pro komplikované úlohy logického, numerického a smíšeného typu (pro hybridní algoritmy) je velmi vhodný jazyk strukturovaného textu (ST). Je zcela obecný, má bohatý aparát příkazů pro úlohy různých typů. Při systematickém používání a při použití jeho logických a aritmetických výrazů a struktur dat je jazyk ST velmi efektivní. I pro složité úlohy lze na malé ploše vytvořit program, který je přehledný, názorný, snadno odladitelný, opravitelný a měnitelný. Je ale třeba připomenout, že v každém jazyce lze při chaotickém programování vytvořit nepřehledný a neodladitelný „paskvil“, že i ve zdánlivě jednoduchém řetězci podmíněných příkazů lze snadno zabloudit a vytvořit program plný chyb.

Nástroj SFC nebo jazyk Grafcet je programovací prostředek nejvyšší úrovně, který dovoluje systematicky přistupovat k programování, zajišťuje dobrou produktivitu a spolehlivost programátorské práce. Obvykle je spojován s řešením logických úloh sekvenčního charakteru. Je ale vhodný i pro úlohy, kde převažují numerické algoritmy – přesněji řečeno: téměř všechny úlohy, které řeší program PLC, jsou smíšeného typu, téměř vždy jsou s PLC realizovány hybridní algoritmy. Ne vždy to musí být úlohy sekvenčního řízení a regulace, ale třeba i úlohy rozpoznání sekvenčních událostí, chybových stavů v technické diagnostice apod.

Volba jazyka z organizačních důvodů

Nezávisle na uvedených argumentech mohou o volbě programovacího jazyka rozhodovat zcela odlišné důvody. Pro dodavatelskou firmu může být rozhodující používání jednotného programovacího jazyka všemi programátory. Obdobně může být striktně doporučeno používání určitých knihoven funkcí a funkčních bloků (ať již nakoupených od určitého dodavatele PLC nebo vytvořených ve firmě). Podobně nelze diskutovat o volbě programovacího jazyka, jestliže si konečný uživatel a zákazník přeje program v konkrétním jazyku.

Ladislav Šmejkal

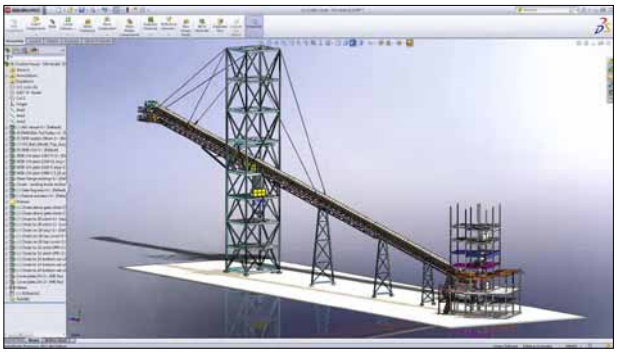
Software SolidWorks 2012 představen v Praze

Představitelé vedení společnosti SolidWorks v Evropě přijeli v říjnu do Prahy představit novou verzi softwaru SolidWorks 2012. Společnost SolidWorks založil v roce 1993 Jon Hirschtick, který přišel s tehdy převratnou myšlenkou, spojit do jednoho systému software pro modelování a konstrukční software CAD. Firma se sídlem v USA patří v současnosti do koncernu Dassault Systemes. Obrat v oblasti EMEA (Evropa, Střední východ a Afrika) činí 42 % z celkového obratu společnosti SolidWorks, který v roce 2010 dosáhl 417,8 milionu amerických dolarů a v první polovině roku 2011 se vyšplhal na 263,3 milionu dolarů.

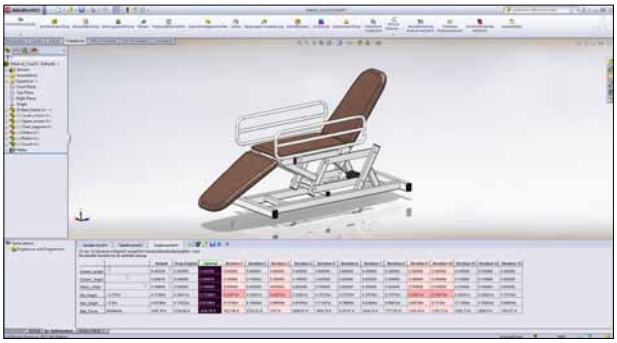
V prodeji převažuje software CAD, ale postupně roste zájem o ostatní typy softwaru: pro simulace, správu dat, dokumentaci, analýzu chování součástí v různém prostředí apod. Podíl tohoto „nonCAD“ softwaru na obratu firmy stále vzrůstá – z 10 % v roce 2007 se zvýšil na 20 % v prvním pololetí roku 2011.

Na říjnové tiskové konferenci v Praze byl představen systém SolidWorks Enterprise PDM 2012, který poskytuje podporu při navrhování konstrukčních dílů a sestav, pro simulaci jejich funkce, vytváření dokumentace a pro správu dat v celém životním cyklu výrobků.

Nová verze softwaru má zajímavé nástroje usnadňující konstruktérům jejich práci. Například SolidWorks Costing vypočítá náklady na výrobu plechových a obráběných dílů. Práci s velkými konstrukčními sestavami zjednoduší nástroj Large Design Review (*obr. 1*). Funkci Feature Freeze použije konstruktér, jestliže chce určitý díl v sestavě „zmrazit“, tedy ponechat jej při úpravách sestavy bez jakýchkoliv změn. Novinkou je funkce ProStep EDMP, která je součástí nástroje CircuitWorks 2012 a umožňuje, aby při projektování mechatronických přístrojů efektivně spolupracovali návrháři elektroniky s konstruktéry mechanických součástí.



Obr. 1. Příklad pracovní plochy nástroje Large Design Review pro práci s velkými konstrukčními sestavami



Obr. 2. Návrh pohonů a mechanismů pro pohyb polohovatelného lůžka s využitím funkce SolidWorks Motion

Zajímavé možnosti přináší konstruktérům nová verze simulačního nástroje, SolidWorks Simulations 2012. Obsahuje funkci SolidWorks Motion, která pomáhá při navrhování pohybových mechanismů. Například při konstrukci polohovatelného lůžka (*obr. 2*) konstruktér využívá simulace a postupně navrhuje mechanismy a správnou velikost a výkon jednotlivých pohonů, přičemž cílem je dosáhnout požadovaných funkcí lůžka při nízkých nákladech. Pomocí nástroje SolidWorksFlow Simulation 2012 lze zkoumat, jak se bude daná součást chovat ve skutečných podmínkách, např. při různém namáhání, zahřívání apod. Nová verze softwaru SolidWorks 2012 má také zdokonalené uživatelské rozhraní s lepším vyhledávačem s přímou vazbou na Windows Explorer a dialogové okno pro vyhledávání souborů ve Windows.

(ev)