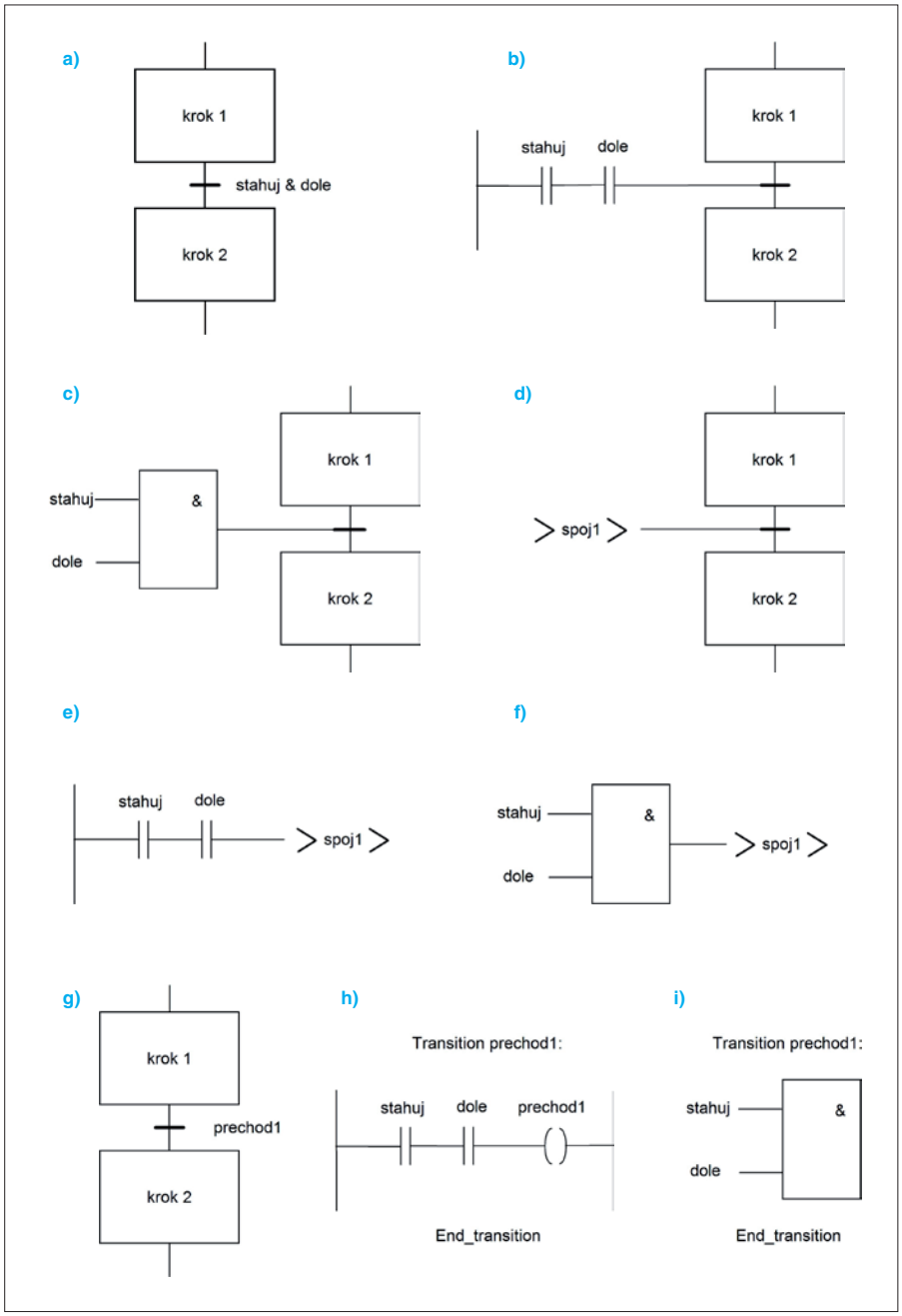
Esperanto programátorů PLC: programování podle normy IEC/EN 61131-3 (část 16)

Tato část seriálu se podrobněji věnuje popisu pravidel grafického nástroje SFC (*Sequential Function Chart*) k programování sekvenčních úloh. Nástroj SFC je sice součástí normy IEC EN 61131-3, ale v praxi není příliš používán – ne všichni výrobci jej implementují ve svých vývojových systémech, ne všichni uživatelé jej zakoupí, ne všichni programátoři jsou ochotni akceptovat odlišné přístupy k programování. Přesto je účelné tento nástroj používat, i kdyby byl jen metodickou pomůckou a sekvenční grafy byly vytvářeny jen jako „kreslené obrázky" pro potřeby návrhu a dokumentace programů. Nástroj SFC totiž vychází ze stavového popisu algoritmů, podobně jako konečné automaty a Petriho sítě. Představuje systematický přístup k programování. Zvyšuje produktivitu programátorské práce – zkracuje dobu tvorby programu, výrazně omezuje výskyt chyb programátora, zpřehledňuje program a usnadňuje provádění změn v programu.

Intuitivní přístup k programování

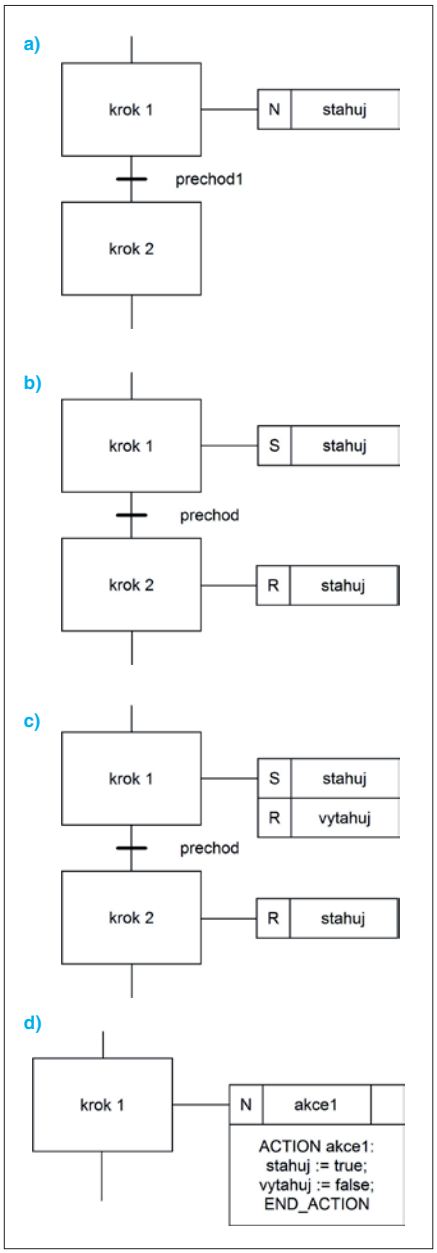
Dosavadní příklady sekvenčních funkcí v tomto seriálu (s výjimkou příkladu 33) byly řešeny intuitivně. Znamená to, že program je vytvářen bez jednotné systematické metodiky. Na základě intuice programátora jsou do programu umísťovány funkční bloky s funkcemi paměti („klopné obvody" typu SR, RS, T, detekce hran proměnných, čítačů a časovačů) a mezi nimi jsou doplňovány prvky, které vytvářejí kombinační logické funkce. Intuitivní návrh spočívá v postupu: „je zde třeba měřit čas, hodil by se nějaký časovač – tady je třeba hodnotu zapamatovat, dáme tam klopák (zápis funkcí S nebo R) – je třeba vyhodnotit počet událostí, dáme tam nějaký čítač – a někam je třeba doplnit podmínku, navrhneme pro ni kombinační logickou funkci atd.". Paměťové funkce, které jsou základem realizace sekvenčního chování (a mezi ně se počítají různé typy klopných obvodů, čítačů a časovačů, v nejhorším případě i zpětné vazby nebo podmíněné příkazy), jsou v programu umístěny zcela chaoticky.

Na předchozích příkladech bylo vidět, že funkce programu někdy závisejí i na pořadí, v jakém jsou jednotlivé funkční bloky a příkazy v programu umístěny – při změně pořadí se může změnit chování programu, obvykle směrem k chybě. Stejná vnitřní nebo výstupní proměnná bývá „oslovována" z různých míst programu a různým způsobem, např. prostým zápisem, nastavením (SET) nebo vynulováním (RESET). Stačí malá nepozornost (obzvláště při dodatečných změnách nebo opravách programu) k tomu, aby program vykazoval chybné chování. „Vděčnými zdroji chyb" bývají podmíněné příkazy v jazyce ST, popř. i zpětné vazby (těm je vhodné se v programu vyhnout, ale někdy nejsou snadno rozpoznatelné). Funkční vazby a souvislosti může zkušený programátor „podržet v hlavě" u jednoduchých programů, po omezenou dobu si lze pamato-



Obr. 55. Způsoby definování podmínky přechodu: (a) podmínkou v jazyce ST vpravo od značky, (b, c) podmínkou v LD a FBD vlevo od značky, (d) prostřednictvím konektoru směřujícího ke značce přechodu, (e, f) konektorem směřujícím od podmínky definované v LD nebo v FBD, (g) proměnnou, jejíž hodnota se odděleně vyčíslí v programu příkazy TRANSITION a END_TRANSITION: (h) v jazyce LD (i) v jazyce FBD

vat uživatelské funkce a funkční bloky. Pak lze úspěšně otestovat funkčnost laděné programové jednotky. Pro komplikovanější a rozsáhlejší programy je to ale téměř vyloučeno – obzvláště vrací-li se programátor k programu po delší době od jeho vytvoření nebo je nucen upravovat cizí program. Pak každá dodatečná změna nebo oprava v sobě nese riziko nových chyb.



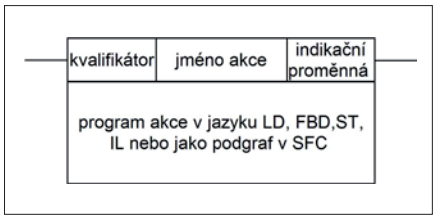
Obr. 56. Příklady jednoduchých akcí přidružených ke krokům: (a) s funkcí zápisu bez paměti s kvalifikátorem N, (b) s paměťovou funkcí S – R, (c) s dvojitou akcí a krok1, s akcí popsanou programem v ST

Zadání příkladů sekvenčních funkcí z předchozích kapitol je možné považovat spíše za nekomplikované a jejich řešení bylo několikrát kontrolováno. Přesto lze očekávat, že pozornému čtenáři se v některých podaří odhalit „chybky“, nekorektní či diskutabilní chování – obzvláště v mezních situacích, o kterých se v zadání nemluvilo.

Úloha 63: Analyzujte řešení příkladů z předchozích pokračování a pokuste se v nich odhalit nekorektní chování (snad i „chybky“) nebo chování, u něhož nelze jednoznačně určit, zda je v souladu se zadáním, či nikoliv. Vaše nálezy (i s autorstvím) zveřejníme v tomto seriálu a autora odměníme ročním předplatným časopisu Automa.

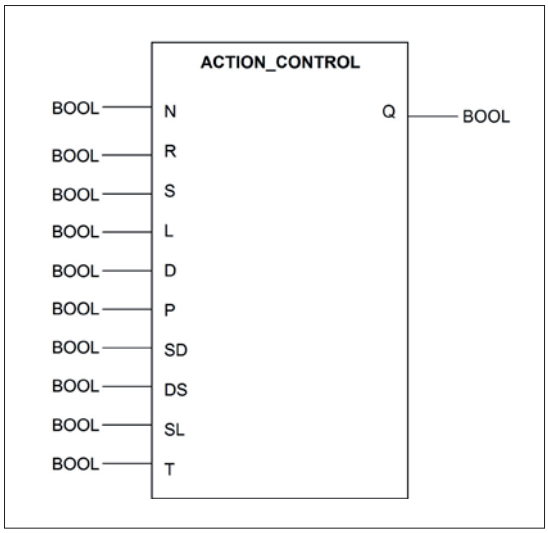
Stavový přístup k programování

Teoretické základy stavového přístupu k realizaci sekvenčních funkcí (a většina úloh z praxe řeší sekvenční problémy) tvoří teorie konečných automatů a Petriho sítí (P-sítí). Z nich vychází programový produkt Grafcet



Obr. 57. Obecná struktura bloku akce přidružené ke stavu

a nástroj SFC podle normy IEC EN 61131-3. Oba lze považovat za hybrid, vycházející z realizace konečného automatu Mooreova typu a podskupiny Petriho sítí. Podstatným znakem stavového přístupu je skutečnost, že celá sekvenční úloha se rozloží (dekompo-



Obr. 58. Vnější rozhraní neviditelného funkčního ACTION_CONTROL

nuje) na posloupnosti oddělených *událostí*. V nich lze chování programu popsat akcemi. *Akce* bývají reprezentovány jednoduchými příkazy, které obvykle ovlivňují stav souvisejících proměnných (vnitřních nebo výstupních) s případnými časovými závislostmi. Jako akce jsou ale uváděny i úseky programu v některém z jazyků PLC, popř. podřízený sekvenční graf (podgraf). *Dekompozice* sekvenčního algoritmu na události a přechody vytváří předpoklad k systematickému programování. Sekvenční graf lze vytvářet (a následně číst) jako příběh typu: „ve výchozím stavu *cekej* na odstartování (rozsvít signálku *pripraven*), pak přejdi do stavu *napoustej* (nastav proměnnou *cerpadlo*), čekej na spínač *plna*, který je podmínkou přechodu do stavu *predpirej*, …“. Většina strojů a technologických procesů je řízena podle obdobných algoritmů, se strukturou posloupnosti událostí (kroků). Spojením symbolů událostí a přechodů mezi nimi vznikne grafický útvar – síť, graf. Obvykle se nazývá *síť SFC*, *graf SFC*, *sekvenční graf*, *stavový graf*, *přechodový graf*, *přechodový diagram* (mnohoznačnost v názvech je způsobena odlišnostmi v terminologii výchozích teorií).

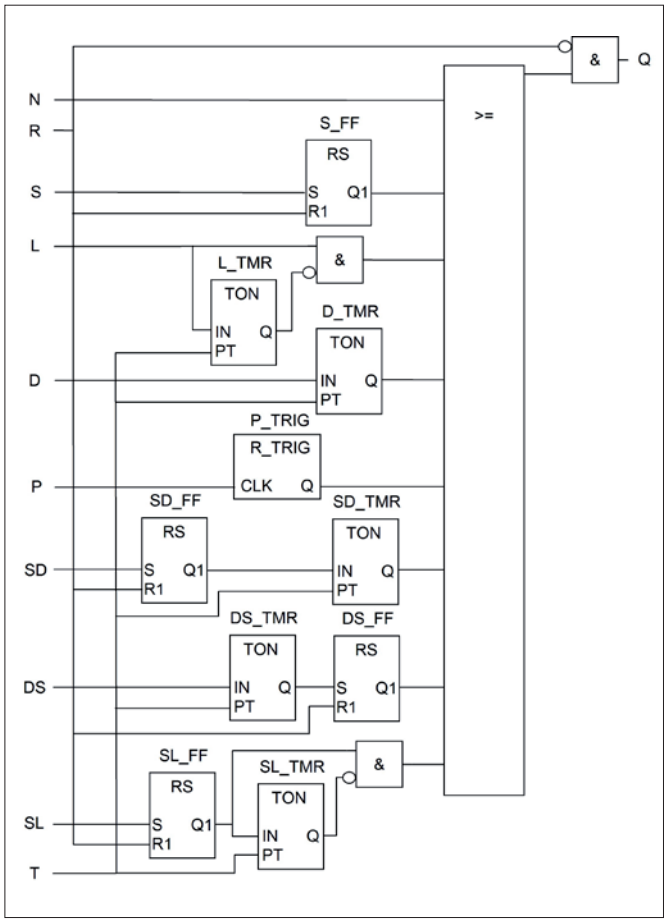
V teorii *konečných automatů* odpovídají událostem *stavy* – v sekvenčním grafu (přechodovém diagramu) jsou označovány kroužkem nebo oválem. V teorii Petriho sítí odpovídají událostem *místa*. V grafickém znázornění Petriho sítě jsou rovněž označovány kroužkem nebo oválem. V programových nástrojích typu Grafcet nebo SFC odpovídají událostem *kroky*, které jsou označovány obdélníkovou značkou s vepsaným jménem kroku. Počátečnímu kroku odpovídá obdélník s dvojitým orámováním. Značky se symboly událostí (stavy, místa, kroky) jsou podle potřeby propojeny spojnicemi – nazývají se *hrany*. Vedle každé hrany je uvedena *podmínka přechodu*, která určuje vznik nové události. Přechod mezi událostmi znamená opuštění (deaktivaci) dosavadní události a aktivaci následující události. U konečných automatů se podmínka přechodu vpisuje přímo ke hraně. U Petriho sítí se podmínka uvádí vedle *značky přechodu* (*transition*), která protíná hranu (obvykle vodorovně) – je to tučná krátká úsečka nebo úzký obdélník. V případě nástroje Grafcet nebo SFC je značkou přechodu krátká úsečka, která vodorovně přetíná svislou hranu. Je nutné, aby se v grafu střídaly symboly kroků se symboly přechodů. Není přípustná nepřerušovaná sekvence dvou kroků za sebou ani nepřerušená sekvence dvou přechodů.

Poznámka: V případě konečných automatů je událost charakterizována termínem stav, zatímco u Petriho sítí označuje termín stav souhrnnou aktivitu všech míst sítě. Podobně i pro Grafcet a SFC označuje termín stav aktivitu všech kroků sítě. Tato skutečnost je matoucí pro programátory, kteří jsou zvyklí na používání konečných automatů.

Implementace sekvenčních grafů

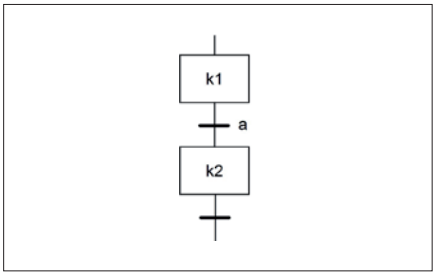
Existuje mnoho programových produktů určených pro popis sekvenčních úloh formou přechodového grafu konečného automatu nebo

některého typu Petriho sítě. Existují i programy, které realizují různé „variace na téma Grafcet nebo SFC“, některé jsou uživatelsky velmi přívětivé. Například může být na obrazovce zobrazena jen struktura přechodového grafu s kroky a přechody. Teprve po „kliknutí“ na zvolený krok se zobrazí program akce, který tomuto kroku odpovídá a při aktivaci kroku se vykonává. Zde bude důsledně používána symbolika SFC, zavedená v normě IEC EN 61131-3, i když jen ke zjednodušenému popisu. Podrobnější popis uvádí norma [1] a třetí díl skripta [7]. Je pravděpodobné, že implementace těchto nástrojů ve vývojových systémech různých výrobců PLC se v detailech mohou lišit, ale v důležitých zásadách by se měly shodovat.



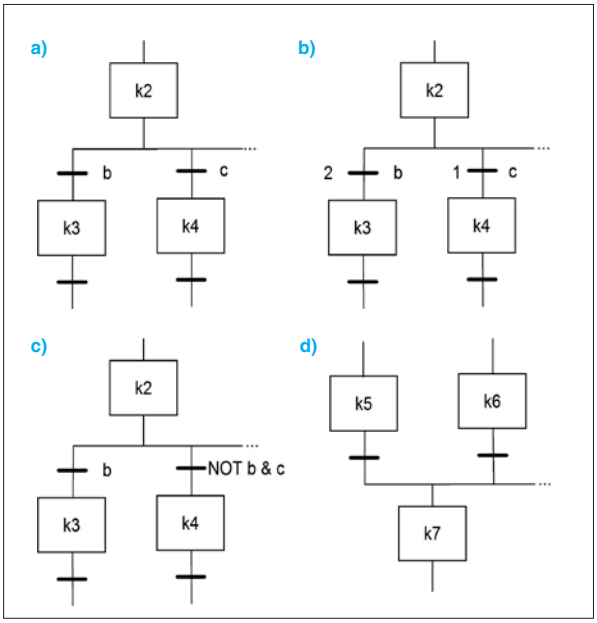
Obr. 59. Tělo neviditelného funkčního ACTION_CONTROL

I v případě, že čtenář nemá k dispozici programový produkt pro implementaci SFC nebo Grafcet, je účelné tuto symboliku používat, třeba jen pro ruční kreslení grafu – v zájmu systematického návrhu a přehledné dokumentace programu. Přechodový diagram (v jakékoliv formě, třeba i v symbolice konečných automatů) je dobrým komunikačním prostředkem mezi projektantem či konstruktérem a programátorem. Pro obě kategorie tvůrčích pracovníků je přechodový diagram srozumitelný. Lze tak předejít nedorozuměním, časovým ztrátám, sporům a reklamacím. Graf SFC je velmi kvalitním zadáním programu PLC a jeho „ruční“ převod do programu PLC je jen nenáročnou rutinou s minimálním rizikem vzniku chyb.

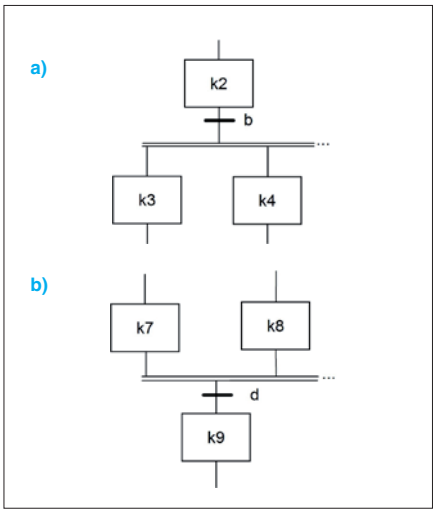


Obr. 60. Jednoduchá sekvence v grafu SFC

Nástroj SFC je všeobecně vnímám jako grafický prostředek pro tvorbu programu pro PLC. Méně je známo, že existují dvě odlišné formy programu SFC. První je již zmíněná forma grafu SFC. Stejnou strukturu grafu lze popsat i textovou formou, obdobně jako v jazyce ST, např. s využitím příkazů: STEP … END_STEP, TRANSITION … END_TRANSITION, ACTION … END_ACTION.



Obr. 61. Výlučné větvení grafu SFC: (a) začátek větvení v přirozeném pořadí priorit zleva doprava, (b) explicitní určení pořadí priorit, (c) výlučnost větvení je zajištěna vylučující se pravdivostí podmínek, (d) závěr výlučného větvení

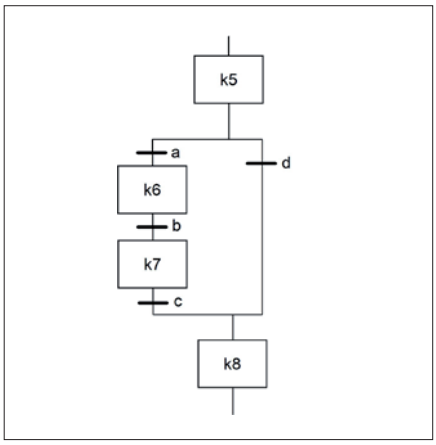


Obr. 62. Simultánní větvení grafu SFC (a) a synchronizace (b)

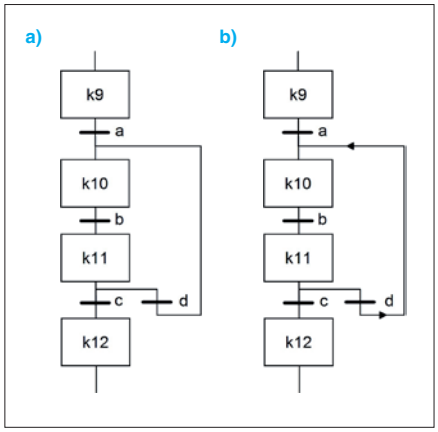
Krok a jeho příznaky

Symbolem události v SFC je krok. Je tvořen obdélníkovou značkou s vepsaným jménem kroku – může jím být libovolný identifikátor. Ke každému kroku je přiřazen *příznak kroku* (*Step Flag*). Je to proměnná typu BOOL, která je součástí struktury (pole) příznaků všech kroků. Je označována jako jméno kroku, za nímž následuje tečka a symbol X(x), např. *krok1.X*. Příznak kroku nabývá hodnotu 1 (TRUE), jestliže je odpovídající krok aktivní, jinak má hodnotu 0 (FALSE). Příznak libovolného kroku může být použit jako vstupní proměnná programu akce nebo podmínky přechodu (v libovolném z jazyků). V grafické formě je k dispozici na pravé straně značky kroku.

Ke kroku je ještě přiřazen *příznak doby setrvání v kroku* (doby aktivity kroku). Označován je jako jméno kroku, za nímž následuje tečka a symbol T(t), např. *krok1.T*. Je typu TIME a je součástí struktury (pole) časových příznaků pro všechny kroky. Při inicializaci grafu SFC nebo v okamžiku aktivace stavu je příznak kroku vynulován (nastaven na hodnotu T#0s). V průběhu aktivace kroku jeho hodnota narůstá a při deaktivaci kroku uchovává poslední hodnotu z okamžiku deaktivace. Příznak doby setrvání v libovolném kroku může být použit jako vstupní proměnná v programu akce nebo podmínky přechodu (v libovolném z jazyků).



Obr. 63. Přeskok posloupnosti kroků



Obr. 64. Zpětná vazba překlenující posloupnost kroků: (a) standardní kreslení, (b) zdůraznění směru vývoje šipkami

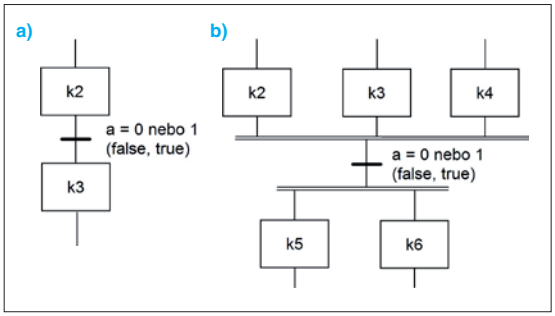
Přechody

Přechod představuje podmínku, která dovolí přechod z jednoho nebo několika výchozích stavů do jednoho nebo několika následných stavů. Může být dán podmínkou v jazycích ST, LD nebo FBD (*obr. 55 a, b, c*), konektorem (*obr. 55d, e, f*) nebo proměnnou (*obr. 55g*), jejíž hodnota se odděleně vyčíslí v programu příkazů TRANSITION a END_TRANSITION v jazyce LD (*obr. 55h*) nebo FBD (*obr. 55i*). Ekvivalentní je textová konstrukce:

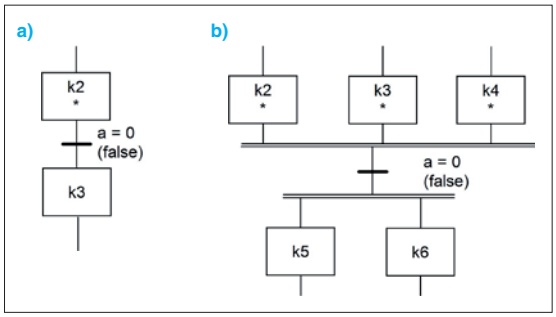
```
TRANSITION FROM krok1 TO krok2
        := stahuj & dole;
END_TRANSITION
```

Akce stavů

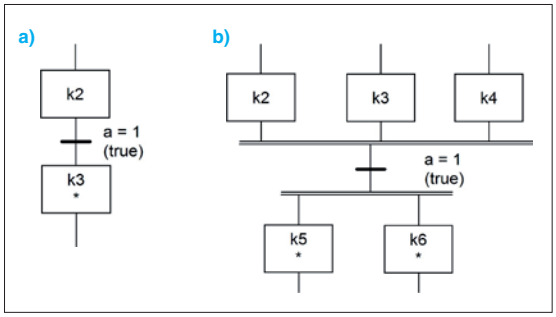
Mohou existovat kroky, ke kterým není přiřazena žádná akce. Program v nich nic nevykonává, jen čeká na splnění podmínky přechodu do následujícího kroku. Nejčastěji se vyskytují kroky, jejichž akce jednoduchým způsobem ovládají hodnoty zvolených proměnných. Na *obr. 56a* je v kroku *krok1* zapisována hodnota 1 (TRUE) do proměnné *stahuj* (kvalifikátor *N* znamená, že do „oslovené“ proměnné je pouze zapisována hodnota příznaku stavu – zápis bez paměti). Ke kroku *krok2* není přidružena žádná akce, krok jen čeká na konec své aktivace. Není „oslovena“ žádná proměnná, ani *stahuj*. Její hodnota se vrací do nuly, protože příznak kroku 1 (*krok1.x*) je již nulový, a nulová je tedy i jím ovládaná proměnná *stahuj*. Na *obr. 56b* je stejná situace řešena odlišně. V kroku 1 je proměnná *stahuj* nastavena na hodnotu 1, ovšem *S* označuje zápis hodnoty 1 s pamětí (s funkcí SET). Tato proměnná zachovává svou hodnotu, dokud není vynulována (zde akcí s kvalifikátorem *R*) u akce přidružené ke kroku *krok2*. Ke kroku může být přiřazeno několik akcí, např. na *obr. 56c* je ke kroku *krok1* přidružena další akce, která nuluje (resetuje) proměnnou *vytahuj*. Stejnou akci



Obr. 65. Mechanismus přechodu – přechod je nemožný: (a) jednoduchý přechod, (b) synchronizace a simultánní větvení



Obr. 66. Mechanismus přechodu – přechod čeká na splnění podmínky: (a) jednoduchý přechod, (b) synchronizace a simultánní větvení



Obr. 67. Mechanismus přechodu – přechod se uskutečnil: (a) jednoduchý přechod, (b) synchronizace a simultánní větvení

řeší konstrukce na *obr. 56d*, kdy je ve zvláštním poli uveden program (zde dva příkazy v jazyce ST), který provede požadovanou akci.

V obecném případě může mít blok akce strukturu podle *obr. 57*. V poli *kvalifikátor* je uveden symbol požadované akce ve významu:

- žádný (*None*): zápis 0 bez paměti,
- N (*Not Stored*): zápis 1 bez paměti,
- R (*Reset Overriding*): reset přednostní,
- S (*Set*): zápis 1 s pamětí,
- L (*Time Limited*): časově ohraničený,
- D (*Time Delayed*): časově zpožděný,
- P (*Pulse*): impulz náběžné hrany,
- SD (*Stored and Delayed*),
- SL(*Stored and Limited*),
- DS (*Delayed and Stored*).

Funkce v poli *kvalifikátor* jsou realizovány (pro uživatele neviditelným) funkčním blokem ACTION_CONTROL. Jeho vnější rozhraní je uvedeno na *obr. 58*, tělo bloku je na *obr. 59*. Je-li akce deklarována jako booleovská proměnná, je výstup *Q* tohoto bloku zapisován do této proměnné. Jestliže je akce deklarována jako úsek programu (posloupnost příkazů v ST, IL nebo obvodů v LD, FBD), tento úsek programu se bude průběžně vykonávat, dokud výstup *Q* bude mít hodnotu 1 (TRUE). Naposledy se vykoná při sestupné hraně výstupu *Q*. V poli *indikační proměnná* lze volitelně doplnit booleovské proměnné, které indikují např. dokončení kroku, vypršení předepsané doby kroku (*timeout*), podmínky chyby v kroku atd. Není-li toto pole uvedeno a pole *jméno akce* specifikuje, že akcí je booleovská proměnná, tato proměnná může být rovněž interpretována jako indikační proměnná.

Vyhodnocení grafu SFC

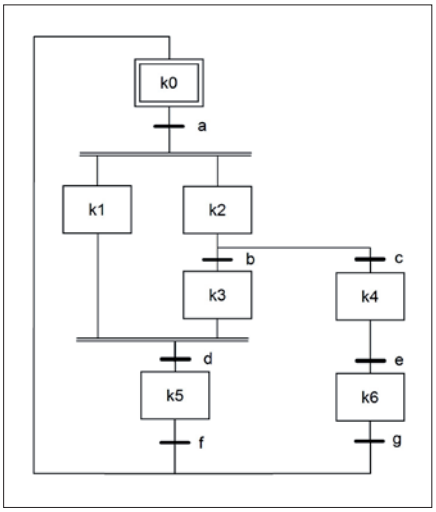
Často má graf SFC jednoduchou strukturu ve formě řetězce kroků a přechodů, který končí v koncovém kroku nebo se z koncového kroku vrací zpět do počátečního. Je tvořen jednoduchými sekvencemi podle *obr. 60*. Někdy je třeba provést větvení grafu SFC. Podobně jako v případě konečných automatů je třeba zajistit, aby v celém grafu byl aktivní jen jeden krok. Proto je zapotřebí při větvení (výběru sekvence) respektovat pořadí priorit pro jednotlivé větve. Přirozené je pořadí zleva doprava (na *obr. 61a* je nejprve vyhodnocována podmínka *b* a pak *c*). Jestliže tento způsob nevyhovuje, lze pořadí priorit jednotlivých větví explicitně určit čísly před značkami přechodů. Podmínky přechodů se pak stanovují v určeném pořadí. V situaci na *obr. 61b* se nejprve bude vyhodnocovat podmínka přechodu *c* a pak teprve podmínka *b*. Optimální situace nastává, jestli-

že se pravdivosti podmínek na začátku větvení navzájem vylučují. Na *obr. 61c* pravdivost první podmínky (*b*) vylučuje pravdivost druhé (NOT *b* & *c*) a naopak. Nevyřčenou podmínkou korektnosti grafu SFC je, že logický součet pravdivostí podmínek všech přechodů na začátku větvení musí být roven jedné. Z toho vyplývá, že vždy je aktivována některá z větví. V opačném případě hrozí riziko, že žádná z větví nebude aktivována a vykonávání grafu se zablokuje. Na *obr. 61d* je zobrazen závěr výlučně aktivovaných větví.

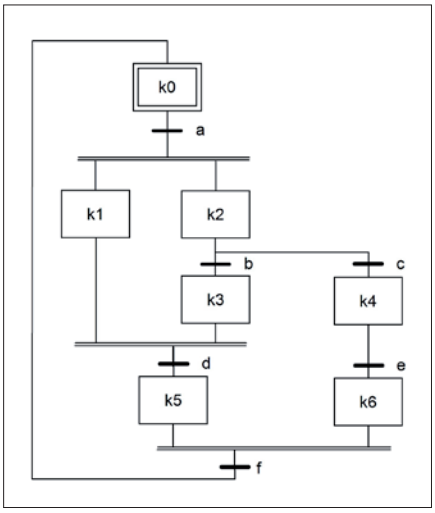
Existují stroje, linky a technologické procesy, které vyžadují, aby byl hlavní proces rozdělen na dva nebo více dílčích procesů, které probíhají souběžně a nezávisle na sobě až do fáze, kdy je třeba je synchronizovat a opět pokračovat ve společném procesu. V grafu SFC to představuje nutnost rozdělit hlavní větev na několik souběžně aktivovaných větví. Mluví se o simultánním větvení grafu. Příklad začátku simultánního větvení je uveden na *obr. 62a*. Je-li aktivní krok *k2* a je splněna podmínka *b*, budou aktivovány kroky *k3* a *k4*, popř. další kroky připojené pod dvojitou vodorovnou čáru (a krok *k2* bude deaktivován). Začátek simultánního větvení je znázorněn dvojitou vodorovnou čarou. Z každého ze simultánně aktivovaných kroků mohou vést libovolně strukturované větve, které se vyvíjejí nezávisle na sobě. Protějškem simultánního větvení je synchronizace, znázorněná na *obr. 62b*. K aktivaci kroku *k9* je nutné, aby byly současně aktivovány kroky *k7* a *k8*, popř. další kroky nad dvojitou čarou a splněna podmínka přechodu *d*. Před uskutečněním přechodu se počká, až bude aktivní poslední z kroků, připojený shora ke dvojité čáře, teprve pak je vyhodnocena podmínka přechodu.

Zvláštním případem výlučného větvení je situace podle *obr. 63*, kdy jedna z větví (zde za podmínkou *d*) neobsahuje žádný krok a překlenuje větev s posloupností stavů. Opačným případem je zpětnovazební smyčka znázorněná na *obr. 64a* a *obr. 64b*. Je opět tvořena větví, která neobsahuje žádné kroky, ale směřuje opačným směrem – od podmínky *d* vede zpět ke kroku *k10*. Z *obr. 64a* je sice zřejmý opačný směr větve, ale jistější je potvrdit jej šipkou, podle *obr. 64b*.

Postup vyhodnocení přechodů je názorně ukázán na *obr. 65* až *obr. 67*. Na *obr. 65* je znázorněna situace, kdy nemůže dojít k přechodu (na *obr. 65a* pro jednoduchý přechod, na *obr. 65b* pro synchronizaci simultánně aktivovaných větví a následné simultánní větvení). Pokud výchozí krok není aktivován, nemůže dojít k přechodu, ať je hodnota podmínky jakákoliv. Na *obr. 66a* je znázorněna



Obr. 68. Příklad nebezpečné sítě



Obr. 69. Příklad nedosažitelné sítě, bezvýchodná situace

situace, kdy výchozí krok je sice aktivní, ale přechod se neuskuteční, dokud není podmínka splněna (zde má nulovou hodnotu). Podobná situace je znázorněna na *obr. 66b*, kde jsou aktivovány všechny tři výchozí kroky, ale není splněna podmínka přechodu. Teprve *obr. 67* ukazuje situaci, kdy byla splněna podmínka a přechod se uskutečnil. Hvězdičkami jsou označeny aktivní kroky.

Použití grafu SFC k návrhu programu sice omezuje riziko chyb programátora, ale zcela je nemůže vyloučit – i v SFC lze vytvořit chybné programy. Na *obr. 68* je uveden příklad sítě SFC, která je označována jako *nebezpečná síť* (*unsafe*). Předpokládejme, že po simultánním větvení po podmínce *a* jsou aktivovány kroky *k1* a *k2*. Po splnění podmínky *c* se postupně aktivují kroky *k4*, *k6*, *k1* a opět jsou splněny předpoklady aktivace *k1* s *k2*. Krok *k1* ale stále zůstává aktivní a byl by aktivován podruhé, v dalším oběhu smyčky potřetí atd. Může tak docházet k nekontrolovanému množení značek aktivace kroků.

Na *obr. 69* je uveden příklad sítě SFC, která je označována jako *nedosažitelná* (*unreachable*). Po simultánním rozvětvení jsou aktivovány kroky *k1* a *k2*. Je-li splněna podmínka *b*, bude aktivován krok *k3* a po splnění podmínky *d* bude aktivován krok *k5*. K dalšímu přechodu ale nemůže dojít, protože krok *k6* nemůže být aktivován (stav je nedosažitelný). Při předpokladu, že po aktivaci *k2* bude splněna podmínka *c* a pak *e*, bude aktivován *k6*. Opět tak nastává bezvýchodná situace, protože nelze aktivovat *k5* a ani přechod do kroku *k0* s podmínkou *f*. Nastává tak situace, která je označována jako *bezvýchodná*, *uzamčená* (*locked up*).

Poznámka: V seriálu se střetávají dva styly textu. V textu jsou podle zásad matematické sazby uváděna jména proměnných kurzívou (např. *vstup1*, *vstup2*, *vystup*). Program v textové formě jazyka ST je ale psán ve vývojovém systému, který nerozlišuje styl písma. Proto jsou tytéž proměnné v programu ST psány stojatým písmem:

```
vystup := vstup1 AND vstup2;
```

Podobná situace je i u obrázků v této i v předchozích částech seriálu. Obrázky představují fragmenty grafu SFC, který je v praxi rovněž vytvářen programem vývojového systému, a texty v nich jsou tedy zobrazovány stojatým písmem, proto je v textu uvedena např. proměnná *krok1* a stejná proměnná je v obrázku vyznačena jako krok 1.

Ladislav Šmejkal, Josef Černý

krátké zprávy

▶ Software zenon 7.10 a SAP: integrace výrobních a obchodních dat

Společnosti KCT Data a COPA-DATA uzavřely strategické partnerství, které poskytne podnikům využívajícím systémy SAP a zenon 7.10 možnost bezpečně propojit úroveň řízení podniku (ERP) s řízením na úrovni provozů (SCADA). Toto propojení umožňuje rozhraní SAP zenon, které vytváří uzavřenou smyčku. Obousměrné propojení mezi úrovněmi SCADA a ERP v praxi znamená, že informace z výroby mohou být rychle zpracovány a požadavky specifikované v systému ERP bezprostředně uplatňovány na provozní úrovni. Výrobní data a události jsou navíc dostupné v reálném čase, což dovoluje optimálně využít pracovní sílu i materiál. KCT Data, s. r. o., se zaměřuje na poradenství a vývoj zákaznických aplikací systému SAP. Rakouská společnost Copa Data od roku 1987 vyvíjí a dodává software kategorie SCADA/HMI pod názvem zenon. Nejnovější verze softwaru, zenon 7.10, je nyní kompatibilní s operačním systémem Windows 8. Po úspěšných zkouškách kompatibility potvrdila společnost Microsoft, že zenon 7.10 může hladce běžet pod Windows 8 a plně odpovídá požadavkům na kvalitu a bezpečnost tohoto systému. Nastavení systému zenon 7.10 bylo upraveno tak, aby uživatelům dovolovalo spouštět zenon přímo prostřednictvím ikony. *(ev)*