Informační a řídicí systémy I.
Programování PLC IV. – IEC 61131-3

Pavel Balda
ZČU v Plzni, FAV, KKY

Osnova přednášky

- Podrobně o SFC
  - Role SFC a ostatních jazyků
  - Vzájemně výlučné podmínky přechodu
  - Přesný význam kvalifikátorů akcí
- Příklady funkčních bloků
  - Implementace uvedená v normě IEC 61131-3, jen jako informativní (tj. nepovinná, nenormativní)
  - Analogické bloky z ŘS REX

Role jazyka SFC

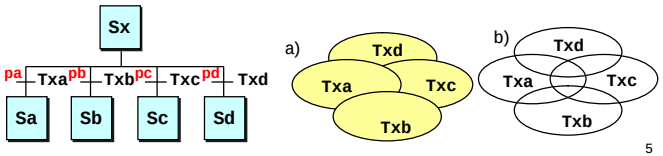
- SFC je považován za nejdůležitější z 5 jazyků normy IEC 61131-3 pro klíčové fáze vývoje softwaru pro PLC. Základními vlastnostmi jsou:
  - Vysoká vyjadřovací schopnost. SFC má stejné vyjadřovací schopnosti jako stavové diagramy a je příbuzný Petriho sítím.
  - Grafický formalismus. Umožňuje snadné naučení se práce se SFC, lze jej kombinovat s LD a FBD. Je vhodný pro reprezentaci procesu na různě detailních úrovních
  - Podpora počáteční fáze návrhu. Umožňuje přesnou analýzu i v případě, že mnoho požadavků není ještě definováno nebo známo návrháři. Tím výrazně snižuje pravděpodobnost nedorozumění mezi zákazníkem, návrhářem a programátorem
  - Podpora detailního návrhu. Počáteční návrh lze specifikovat a doplňovat tak, jak získává návrhář další informace. V akcích lze používat vnořená schémata v SFC
  - Přirozené spojení s dalšími jazyky. V podmínkách a akcích lze využít ostatní jazyky nejvhodnější k danému účelu
  - Podpora rozdělení kódu na menší úseky (fragmenation). Jednotlivé úseky kódu mohou běžet s různými periodami vzorkování a tím zmenšit celkové zatížení PLC.

Role ostatních jazyků

- Role ostatních jazyků je značně subjektivní dle zvyklostí jednotlivých uživatelů
  - Strukturovaný text (ST) – Vyšší programovací jazyk blízký Pascalu (a C), vhodný doplněk k SFC pro kódování podmínek a akcí
  - Seznam instrukcí (IL) – jazyk blízký assembleru, proto by měl mít větší výkonnost než ST, avšak dnes s rostoucím výkonem procesorů ztrácí tato výhoda na významu, neboť kód je mnohem méně přehledný
  - Schéma složené z funkčních bloků (FBD) – Analogické elektrickým schématům, nejlepší a nejpřehlednější prostředek pro konfiguraci regulačních algoritmů
  - Liniová (kontaktní) schémata (LD) – historický prostředek vývoje řídicích systémů logického typu. Dosud podporován mnoha výrobci, neumožňuje však všechny konstrukce, jako např. ST
- Doporučení:
  - SFC pro dekompozici složitých řídicích algoritmů a v kombinaci s ST pro úlohy logického typu
  - FBD pro analogovou regulaci, v kombinaci se SFC lze volit různé režimy provozu

Vzájemně výlučné podmínky přechodu

- Jednotlivé podmínky přechodu by se neměly překrývat (měly by být vzájemně výlučné, exklusivní), viz obr. a
- V případě, že se překrývají (obr. b), musí být určena priorita (**pa**, ..., **pd** v obr.) vyhodnocování – číslo uvedené u podmínky přechodu (menší hodnota odpovídá vyšší prioritě)
- Pokud se podmínky překrývají (obr. b) a priorita není uvedena, měla by být hlášena chyba
- Problém je v tom, že poslední případ je řešen na různých platformách různě – nekompatibilita !!! Proto je nejlepší podmínky konstruovat exklusivně, je to však mírně na úkor rychlosti!

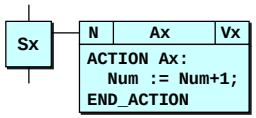


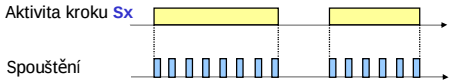
Kvalifikátory akcí v SFC (opakování)

Kvalif.	Definice	Chování akce
žádný	Non-stored (null quailier)	Jako kvalifikátor **N**
N	Non-stored	Provádí se, když je daný krok aktivní
R	overriding Reset	Ukončuje provádění akcí s kvalifikátorem **S**, **SD**, **SL**
S	Set (Stored)	Akce se provádí, dokud není dosažen stav v němž má kval. **R**
L	time Limited	Akce se provádí po dobu zadanou parametrem v kvalifikátoru
D	time Delayed	Akce se spustí za čas daný parametrem v kvalifikátoru
P	Pulse	Provede se, když je daný krok aktivován
SD	Stored and time Delayed	Po uplynutí zadaného zpoždění spouští akci jako při **S**
DS	Delayed and Stored	Akce se začne provádět, trvá-li daný stav alespoň zadaný čas
SL	Stored and time Limited	Akce se provádí jako při **S**, ale jen do uplynutí zadaného času
P1	Pulse (rising edge)	Akce se provede pouze jednou po náběžné hraně pulsu
P0	Pulse (falling edge)	Akce se provede pouze jednou po sestupné hraně pulsu

Akce s kvalifikátorem N – „Non-stored“

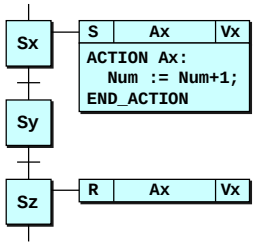
Akce, prováděná, když je daný krok aktivní

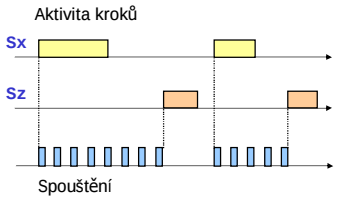




Akce s kvalifikátory S-R – „Set-Reset“

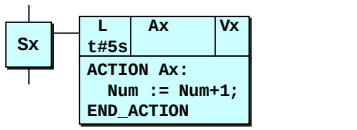
Akce se spouští vstupem do kroku, v němž je s kvalifikátorem S, ukončuje se v kroku s kvalifikátorem R

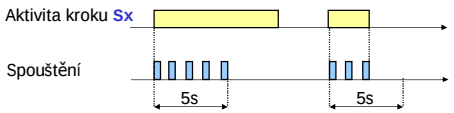




Akce s kvalifikátorem L – „Time Limited“

Akce se provádí od vstupu do kroku po dobu zadanou v kvalifikátoru L

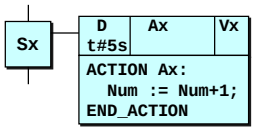


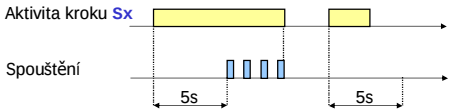


9

Akce s kvalifikátorem D – „Delayed“

Akce se začne provádět po vstupu do kroku se zpožděním zadaným v kvalifikátoru D

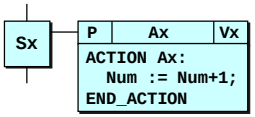


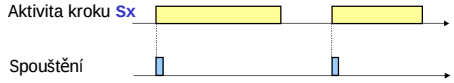


10

Akce s kvalifikátorem P – „Pulse“

Akce se provede jednorázově při vstupu do kroku

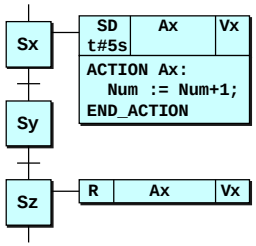


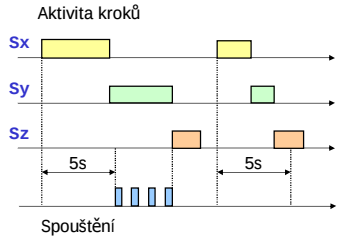


11

Akce s kvalifikátory SD-R

Akce se začne provádět se zadaným zpožděním po vstupu do kroku, v němž je s kvalifikátorem SD, ukončí se vstupem do kroku k kvalifikátorem R

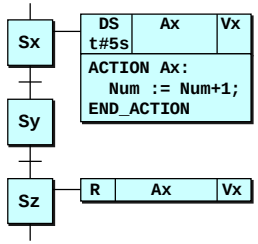


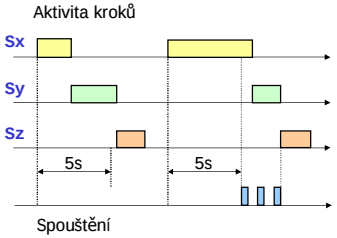


12

Akce s kvalifikátory DS-R

Akce se začne provádět v kroku, v němž je s kvalifikátorem DS se zadaným zpožděním, ukončí se vstupem do kroku k kvalifikátorem R

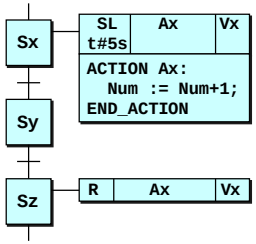


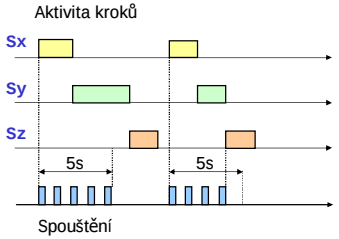


13

Akce s kvalifikátory SL-R

Akce se začne provádět v kroku, v němž je s kvalifikátorem SL po dobu danou parametrem nebo než se dosáhne kroku s kvalifikátorem R





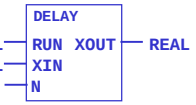
14

Příklady funkčních bloků (nenormativní)

- Ukazují možnosti implementace trochu složitějších FB
- Jména ani implementace ani nejsou povinné, jen informativní
- Příklady FB:
  - **DELAY** – zpoždění o **N** vzorků
  - **AVERAGE** – klouzavý průměr z **N** vzorků
  - **INTEGRAL** – integrátor signálu v čase
  - **DERIVATIVE** – diference vztažená k času
  - **PID** – PID regulátor

15

DELAY – zpoždění signálu o N vzorků

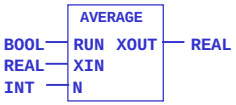



```
FUNCTION_BLOCK DELAY
  (* N-vzorkové zpoždění *)
  VAR_INPUT
    RUN : BOOL ; (* 1=run, 0=reset *)
    XIN : REAL ;
    N : INT ;    (* 0<=N<128 nebo výrobcem *)
  END_VAR        (* určená maximální hodnota *)

  VAR_OUTPUT XOUT : REAL; END_VAR (* Zpožděný výstup *)

  VAR
    X : ARRAY [0..127] OF REAL; (* Fronta FIFO s N vzorky *)
    I, IXIN, IXOUT : INT := 0;
  END_VAR
                 (* Chyba, když je RUN nastaven hned *)
  IF RUN THEN    (* při prvním průchodu *)
    IXIN := MOD(IXIN + 1, 128) ; X[IXIN] := XIN ;
    IXOUT := MOD(IXOUT + 1, 128) ; XOUT := X[IXOUT];
  ELSE XOUT := XIN ; IXIN := N ; IXOUT := 0;
    FOR I := 0 TO N DO X[I] := XIN; END_FOR;
  END_IF ;
END_FUNCTION_BLOCK
```

16

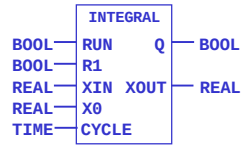
AVERAGE – klouzavý průměr z N vzorků

```
FUNCTION_BLOCK AVERAGE
  (* Klouzavý průměr z N vzorků *)
  VAR_INPUT
    RUN : BOOL ; (* 1=run, 0=reset *)
    XIN : REAL ;
    N : INT ;    (* 0<=N<128 nebo výrobcem *)
  END_VAR        (* určená maximální hodnota *)
  VAR_OUTPUT XOUT : REAL ; END_VAR (* Zprůměrovaný výstup *)
  VAR
    SUM : REAL := 0.0; (* Sčítací proměnná *)
    FIFO : DELAY ; (* Fronta FIFO s N vzorky *)
  END_VAR

  SUM := SUM - FIFO.XOUT ;
  FIFO (RUN := RUN , XIN := XIN, N := N) ;
  SUM := SUM + FIFO.XOUT ;
  IF RUN THEN
    XOUT := SUM/N ;
  ELSE
    SUM := N*XIN ; XOUT := XIN ;
  END_IF ;
END_FUNCTION_BLOCK
```



INTEGRAL – integrátor signálu v čase

```
FUNCTION_BLOCK INTEGRAL
  (* Integrace v čase *)
  VAR_INPUT
    RUN : BOOL ; (* 1=integruj, 0=drž *)
    R1 : BOOL ;  (* prioritní reset *)
    XIN : REAL ; (* vstupní proměnná *)
    X0 : REAL ;  (* počáteční hodnota *)
    CYCLE : TIME;(* perioda vzorkování *)
  END_VAR
  VAR_OUTPUT
    Q : BOOL ;   (* negace R1 *)
    XOUT : REAL; (* Integrovaný výstup *)
  END_VAR

  Q := NOT R1 ;
  IF R1 THEN
    XOUT := X0 ;
  ELSIF RUN THEN
    XOUT := XOUT + XIN * TIME_TO_REAL(CYCLE);
  END_IF ;
END_FUNCTION_BLOCK
```

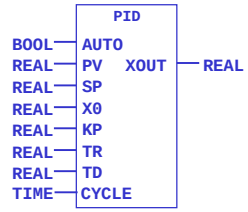


DERIVATIVE – aproximace derivace

```
FUNCTION_BLOCK DERIVATIVE
  (* Diferenciace v čase *)
  VAR_INPUT
    RUN : BOOL ; (* 0 = reset *)
    XIN : REAL ; (* diferencovaný vstup *)
    CYCLE : TIME;(* perioda vzorkování *)
  END_VAR

  VAR_OUTPUT
    XOUT : REAL ; (* Diferencovaný výstup *)
  END_VAR

  VAR X1, X2, X3 : REAL ; END_VAR

  IF RUN THEN
    XOUT := (3.0 * (XIN - X3) + X1 - X2)
      / (10.0 * TIME_TO_REAL(CYCLE)) ;
    X3 := X2 ; X2 := X1 ; X1 := XIN ;
  ELSE
    XOUT := 0.0; X1 := XIN ; X2 := XIN ; X3 := XIN ;
  END_IF ;
END_FUNCTION_BLOCK
```

PID – PID regulátor

```
FUNCTION_BLOCK PID
  VAR_INPUT
    AUTO : BOOL ; (* 0-man, 1-auto *)
    PV : REAL ; (* řízená veličina *)
    SP : REAL ; (* požadovaná hodnota *)
    X0 : REAL ; (* výstup v ručním režimu *)
    KP : REAL ; (* Proporcionální zesílení *)
    TR : REAL ; (* integrační čas. konstanta *)
    TD : REAL ; (* derivační čas. konstanta *)
    CYCLE : TIME ; (* perioda vzorkování *)
  END_VAR
  VAR_OUTPUT XOUT : REAL; END_VAR
  VAR
    ERROR : REAL ; (* PV - SP *)
    ITERM : INTEGRAL ; (* FB pro integrační část *)
    DTERM : DERIVATIVE ; (* FB pro derivační část *)
  END_VAR
  ERROR := PV - SP ;
  (*** Přizpůsobení ITERM, aby XOUT := X0 když AUTO = 0 ***)
  ITERM (RUN := AUTO, R1 := NOT AUTO, XIN := ERROR,
         X0 := TR * (X0 - ERROR), CYCLE := CYCLE) ;
  DTERM (RUN := AUTO, XIN := ERROR, CYCLE := CYCLE) ;
  XOUT := KP * (ERROR + ITERM.XOUT/TR + DTERM.XOUT*TD) ;
END_FUNCTION_BLOCK
```



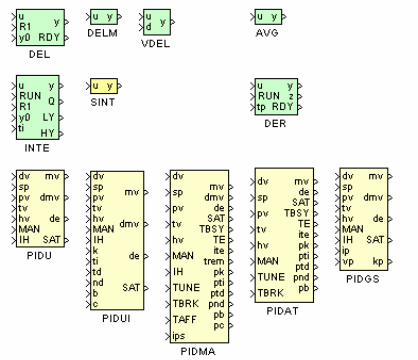
Příklady FB – zhodnocení

- FB v běžných PLC
  - + Jednoduché algoritmy
  - + Možnost vytváření vlastních FB a jejich knihoven s využitím existujících
  - – Často nejsou ošetřeny mezní stavy, např. v PID regulátoru unášení integrační složky
  - – Ruční ošetření mezních stavů je pracné
- Nedostatky uvedených příkladů
  - Krátké buffery (**DELAY**, **AVERAGE**), lze snadno zvětšit, je-li dost paměti
  - **DELAY**: chyba při prvním průchodu, je-li nastaven příznak RUN
  - **AVERAGE**: není vyřešena kumulace chyb
  - **INTEGRAL**: příliš hrubá aproximace obdélníkovým pravidlem
  - **DERIVATIVE**: odhad derivace je příliš závislý na šumu v signálu (použití 4 bodové formule)
  - **PID**: neřeší standardní požadavky, např. bezrázové přepínání, unášení integrační složky, apod.

Obdobné příklady v ŘS REX 1/2

- Bloky z knihovny FB RexLib (dostupná z www.rexcontrols.cz)
- Autorem většiny bloků je prof. Schlegel
- **DEL**, **DELM**, **VDEL** - bloky pro zpoždění
  - Podstatně delší buffer, v současné implementaci až 10000 vzorků
- **AVG** – klouzavý průměr z až 10000 vzorků
  - Odstranění kumulace chyb
- **INTE** – řízený integrátor, **SINT** – jednoduchý integrátor
  - Integrace se provádí lichoběžníkovou metodou
- **DER** – derivace, filtrace, predikce z n+1 vzorků metodou nejmenších čtverců
  - V případě velkého šumu lze zvětšovat hodnotu n
- **PIDU**, **PIDUI**, **PIDMA**, **PIDAT**, **PIDGS** – různé varianty PID regulátorů
  - Regulátory PIDMA a PIDAT jsou vybaveny vestavěnými algoritmy pro automatický návrh parametrů (autotunery)
- Detaily viz: Funkční bloky systému REX – Referenční příručka. REX Controls, Plzeň, leden 2007. Dostupná na internetu

Obdobné příklady v ŘS REX 2/2



Bloky z knihovny AnalLib

Bloky z knihovny RegLib