



## Vložené řídicí systémy Programování PLC I. – IEC 61131-3

Pavel Balda  
ZČU v Plzni, FAV, KKY



## Osnova přednášky

- n Společné rysy jazyků normy IEC 61131-3
- n Stručný přehled jazyků
- n Jednoduché příklady
- n Sekvenční funkční grafy (SFC)

2



## Doporučená literatura

- n Jack, F.: Automating Manufacturing Systems with PLCs.  
Dostupné na: [http://www.freebookspot.com/Comments.aspx?Element\\_ID=3024](http://www.freebookspot.com/Comments.aspx?Element_ID=3024)
- n Bonfatti, F.; Monari, D.P.; Sampieri, U.: IEC1131-3  
Programming Methodology. CJ International/Groupe ALterSys,  
France, 2001.
- n IEC-61161-3 2nd Ed. Final Draft, 2001-04-16
- n PLC open: <http://www.plcopen.org>

3



## Norma IEC 61131-3

- n Norma IEC 61131-3 se zabývá programováním PLC (Programmable Logic Controller, programovatelný automat)
- n **Společné prvky** (Common elements)
  - n Softwarový model
  - n Literály
  - n Identifikátory
  - n Datové typy
  - n Proměnné
- n **Programovací jazyky**
  - n Sekvenční funkční grafy **SFC** (Sequential Function Charts)
  - n Strukturovaný text **ST** (Structured Text)
  - n Seznam instrukcí **IL** (Instruction List)
  - n Funkční bloky **FBD** (Function Block Diagram)
  - n Liniové schéma **LD** (Ladder diagram)

4



## Identifikátory a komentáře

- n **Identifikátor**
  - n Řetězec složený písmeny, číslic a znaků '\_' (podtržítka) začínající písmenem nebo podtržítkem, např. `MAX_U_24`, `ahoj`, `a_hoj`
  - n Nerozlišují malá a velká písmena, tj. `ahoj`, `AHOJ` a `Ahoj` jsou stejné identifikátory
  - n Podtržítka nesmí být na konci a v ostatních případech nesmí jich být několik přímo za sebou, např. `__MAX_U`, `__MAX__U` nebo `MAX_U_` nejsou správné identifikátory!
- n **Klíčová slova**
  - n Vyhrazené identifikátory pro syntaktické prvky jazyka
  - n Neměla by být používána pro označování uživatelských proměnných
- n **Komentáře**
  - n Posloupnosti znaků mezi speciálními kombinacemi (`* a *`), např. `(* toto je komentář *)`
  - n Kromě IL a řetězcových literálů jsou dovoleny všude, kde může být mezera
  - n Nesmí se vnořovat, není povoleno např. `(* (* vnořený *) *)`

9

## Číselné literály

- n Číselné (numerické literály) slouží pro zápis číselných konstant
  - n Mohou uvnitř obsahovat znak '\_' (podtržítka) !
  - n **Celočíselné**
    - n Dekadické – mohou zapisovat běžným způsobem čísla v desítkové soustavě a mohou mít znaménko '+' nebo '-', např. `-12`, `0`, `123_456`, `+789`
    - n V soustavách o základech `2`, `8` nebo `16` – zapisují se s prefixem číselné soustavy následované znakem '#' a nesmějí obsahovat znaménko, např. `2#1110_0000` (240 dekadicky), `8#377` (255 dekadicky), `16#e5` (245 dekadicky)
    - n Dále mohou explicitně obsahovat typ uvedený jako prefix a oddělený znakem '#', např. `DINT#5`, `UINT#16#FE`
  - n **Reálné**
    - n obsahují desetinnou tečku a mohou mít exponent (i se znaménkem), např. `-12.0`, `3.14159_26`, `1.234e-6`
  - n **Booleovské** – `0` nebo `FALSE` a `1` nebo `TRUE`

10

## Řetězcové literály

- n **Řetězce jednobajtových znaků**
  - n Uzavřené do jednoduchých uvozek (apostrofově) `' '`, např. `'A'`, `' '`, `'...'`
  - n Náhradní posloupnosti dvouznačkové začínající '\$' – viz tabulka, např. `'$''`, `'$R$L'`, `'$$1.00'` (\$1.00)
  - n Tříznakové kombinace začínající znakem '\$' – druhé dva znaky se interpretují jako hexadecimální kód znaku kterým je nahrazena v textu, např. `'$0A'` (LF), `'$C4$CB'` ('Ě')
- n **Řetězce dvoubajtových znaků**
  - n Uzavřené do dvojítek uvozek `" "`
  - n Lze užívat 5-značkové kombinace začínající '\$' – další 4 znaky jsou kódem tištěného znaku
- n Mohou explicitně obsahovat prefix typu odděleného znakem '#', např. `STRING#"OK"`, `WSTRING#"OK"`

Kombinace	Při tisku
<code>\$\$</code>	Znak dolar
<code>\$'</code>	Jednoduchá uvozovka
<code>\$L</code> nebo <code>\$l</code>	Konec řádku (LF)
<code>\$N</code> nebo <code>\$n</code>	Nová řádka
<code>\$P</code> nebo <code>\$p</code>	Nová stránka (FF)
<code>\$R</code> nebo <code>\$r</code>	Návrat vozíku (CR)
<code>\$T</code> nebo <code>\$t</code>	Tabulátor
<code>\$"</code>	Dvojítká uvozovka

11

## Časové literály

- n **Literály pro dobu trvání**
  - n Pro měření uplynulého času od určitého okamžiku
  - n Udávají čas ve dnech, hodinách, minutách, vteřinách a milisekundách,
  - n Nejméně významná jednotka může být zapsána jako reálné číslo bez exponentu
  - n Prefixy `T#`, `TIME#`, `t#` nebo `time#`
  - n Příklady: `T#5d14h12m18s3.5ms`, `TIME#14.7h`, `time#25h15m`
- n **Literály pro absolutní čas a datum**
  - n Pro synchronizaci začátku a konce dané události s absolutním časem
  - n Datum se udává ve tvaru rok-měsíc-den, čas ve tvaru hodina:minuta:vteřina
  - n Prefixy `TIME_OF_DAY#`, `TOD#`, `DATE#`, `D#`, `DATE_AND_TIME#`, `DT#`
  - n Příklady: `DATE#2006-03-20`, `time_of_day#10:42:23.78`, `DT#2006-03-20-10:42:23.78`

12

## Jednoduché datové typy

Název	Datový typ	Bitů	Inic.	Název	Datový typ	Bitů	Inic.
<b>BOOL</b>	Boolean	1	0	<b>TIME</b>	Duration (doba trvání)	*	T#0s
<b>SINT</b>	Short integer	8	0	<b>DATE</b>	Date (only)	*	D#0001-01-01
<b>INT</b>	Integer	16	0	<b>TOD</b>	Time of day (only)	*	TOD#00:00:00
<b>DINT</b>	Double integer	32	0	<b>DT</b>	Date and time of day	*	jako D#+TOD#
<b>LINT</b>	Long integer	64	0	<b>STRING</b>	Variable-length char string	*	''
<b>USINT</b>	Unsigned short int	8	0	<b>WSTRING</b>	Var.-length 2-byte char str	*	""
<b>UINT</b>	Unsigned int	16	0	<b>BYTE</b>	Bit string of length 8	8	
<b>UDINT</b>	Unsign. double int	32	0	<b>WORD</b>	Bit string of length 16	16	
<b>ULINT</b>	Unsigned long int	64	0	<b>DWORD</b>	Bit string of length 32	32	
<b>REAL</b>	Real number	32	0.0	<b>LWORD</b>	Bit string of length 64	64	
<b>LREAL</b>	Long real number	64	0.0				

13

## Hierarchie jednoduchých typů

- n ANY
  - ANY\_DERIVED
  - ANY\_ELEMENTARY
  - ANY\_MAGNITUDE
  - ANY\_NUM
  - ANY\_REAL
    - LREAL, REAL
  - ANY\_INT
    - LINT, DINT, INT, SINT
    - ULINT, UDINT, UINT, USINT
  - TIME
  - ANY\_BIT
    - LWORD, DWORD, WORD, BYTE, BOOL
  - ANY\_STRING
    - STRING, WSTRING
  - ANY\_DATE
    - DATE\_AND\_TIME, DATE, TIME\_OF\_DAY
- n Generické datové typy – identifikované prefixem 'ANY'
  - n Používají se pro specifikaci přetížených vstupů a výstupů standardních funkcí a funkčních bloků

14

## Odvozené datové typy

(1/2)

- n **Odvozené datové typy** (derived data types)
  - n Specifikovány uživatelem nebo výrobcem PLC
  - n Deklarují se pomocí konstrukce **TYPE...END\_TYPE**
  - n **TYPE Teplota : REAL := 10.0; END\_TYPE**
- n **Vyjmenované typy** (enumerated types)
  - n Hodnoty datových prvků mohou nabývat pouze hodnot z deklarovaného seznamu
  - n Počáteční hodnotou je hodnota prvního prvku nebo hodnota nastavená operátorem přiřazení
  - n **TYPE PracovníRežim : (Rucni, Auto, Vyp) := Vyp; END\_TYPE**
- n **Deklarace rozsahu** ( subrange)
  - n Určuje přípustný rozsah všech hodnot daného typu, zadávaný spodní a horní mezí. Není-li použit přiřazovací operátor, je počáteční hodnotou první (spodní) mez
  - n **TYPE Teplota : REAL (-1.0..+200.0) := 10.0; END\_TYPE**

15

## Odvozené datové typy

(2/2)

- n **Struktura** (structure)
  - n Deklarace struktury určuje typové datové typy vložených prvků (sub-elements), které jsou dostupné svými jmény
  - n Každý z vložených prvků může být jednoduchý typ nebo struktura
  - n **TYPE CidloTeploty**
    - Tepl : Teplota;
    - Kalibrace : DATE;
    - HorniMez : REAL := 180.0;
    - END\_TYPE
- n **Pole** (array)
  - n Deklarace specifikuje typ každého prvku a rozsah indexů
  - n Prvky daného pole mohou být inicializovány různými hodnotami
  - n **TYPE CelaCisla : ARRAY [1..10] OF INT; END\_TYPE**
  - n **TYPE TeplotyPece : ARRAY [1..20] OF Teplota := [10(-1.0), 10(1.0)]; END\_TYPE**
- n **Řetězec** (string)
  - n Maximální délka stringu je závislá na implementaci
  - n **TYPE STR10 : STRING[10] := 'ABCDEF'; END\_TYPE**

16

## Proměnné

(1/3)

- n **Proměnné** (variables) uchovávají data, jejichž obsah se může měnit
  - n Data přidružená ke vstupům (inputs), výstupům (outputs) a vnitřní paměti (memory) programovatelného automatu
  - n Proměnné se deklarují v **deklarační části** na začátku programu, funkce nebo funkčního bloku
- n **Deklarační část** (declaration part) specifikuje typ a případně fyzické nebo logické umístění proměnné v dané programové jednotce (Program Organization Unit, **POU**)
- n **Rozsah platnosti** (scope) proměnných je lokální v rámci dané POU, kde je proměnná deklarována. Výjimku tvoří jen globální proměnné (viz dále)
- n **VAR** – klíčové slovo pro deklaraci vnitřních (internal) proměnných
  - n Používají se interně v dané POU k ukládání výsledků výpočtů
- n **VAR\_INPUT** – klíčové slovo pro deklaraci vstupních proměnných
  - n Používají se pro proměnné, jejichž zdroj pochází z vnějšku dané POU
  - n Pro program odpovídají vstupům z čidel
  - n Pro funkci nebo FB jsou vstupními parametry

17

## Proměnné

(2/3)

- n **VAR\_OUTPUT** – klíčové slovo pro deklaraci výstupních proměnných
  - n Používají se pro proměnné nastavované v dané POU pro vnější entity
  - n Pro program odpovídají výstupům na akční členy (actuators) generovanými programem
  - n Pro funkci nebo FB jsou výstupními parametry; mohou být dále používány v dané POU
- n **VAR\_IN\_OUT** – klíčové slovo pro deklaraci vstupně-výstupních proměnných
  - n Získávají svou hodnotu mimo danou POU, mohou však v ní být modifikovány a tyto nové hodnoty lze používat vně dané POU.
  - n Lze jimi realizovat „volání odkazem“ známé z jazyků jako Pascal nebo C.
- n **VAR\_GLOBAL** – klíčové slovo pro deklaraci globálních proměnných
  - n Lze k nim přistupovat ze všech POU v daném programu
  - n V daném POU je třeba uvést klíčové slovo **VAR\_EXTERNAL**
  - n Typ uvedený ve **VAR\_GLOBAL** daného programu a **VAR\_EXTERNAL** dané POU musí souhlasit!
- n **VAR\_ACCESS** – klíčové slovo pro proměnné, které mohou být předávány prostřednictvím komunikace a přístupové cesty (access path) mezi programy

18

## Proměnné

(3/3)

- n **Proměnné ponechávající si hodnotu** (retentive variables)
  - n V některých implementacích může být počáteční hodnota proměnné nastavena z uchované hodnoty (retained value) uložené při ukončení předchozího běhu
  - n Tím je umožněno provádět tzv. teplý restart (warm restart)
- n **Jednoprvkové proměnné** (single-element variables)
  - n Proměnné tvořené jedním datovým prvkem jednoduchého, výčtového nebo rozsahového typu, nebo odvozeného jednoduchého typu od uvedených typů
  - n Mohou být reprezentovány symbolicky pomocí identifikátorů nebo přímo (directly) pomocí znaku '%', (procento) následovaného prefixem umístění (location) a prefixem velikosti (size), viz další stránku
- n **Víceprvkové proměnné** (multiple element variables)
  - n Proměnné typů pole nebo struktura
  - n K prvkům struktur se přistupuje pomocí znaku '.' (tečka)
  - n K prvkům pole pomocí indexů v hranatých závorkách '[' a '']
  - n Nelze je používat v jazyku seznam instrukcí (IL)
- n **AT** – klíčové slovo přiřazující fyzickou nebo logickou adresu symbolicky reprezentované proměnné

19

## Přimo reprezentované proměnné

- n **Přímá reprezentace proměnné** se skládá ze znaku '%', prefixu umístění (první část tabulky), prefixu velikosti (druhá část tabulky) a jednoho nebo několika celých čísel bez znaménka oddělených tečkami. Počet čísel oddělených tečkami je implementačně závislý parametr daný výrobcem

Prefix	Význam	Datový typ
<b>I</b>	Vstup (Input location)	
<b>Q</b>	Výstup (Output location)	
<b>M</b>	Paměť (Memory location)	
<b>X</b>	Jeden bit	<b>BOOL</b>
žádný	Jeden bit	<b>BOOL</b>
<b>B</b>	Byte (8 bitů)	<b>BYTE</b>
<b>W</b>	Word (16 bitů)	<b>WORD</b>
<b>D</b>	Double Word (32 bitů)	<b>DWORD</b>
<b>L</b>	Long (quad) Word (64 bitů)	<b>LWORD</b>

20

## Proměnné – příklady

(1/2)

- n Přímou reprezentované, ne-retentivní proměnné

```
VAR
  AT %IW6.2 : WORD;      (* 16 bitový řetězec *)
  AT %MW6 : INT;        (* 16 bit integer inicializovaný na 0 *)
  AT %QX5.1 : BOOL := 1; (* Boolean inicializovaný na 1 *)
  AT %MW7 : INT := 8;   (* 16 bit integer inicializovaný na 8 *)
END_VAR
```

- n Přímou reprezentované, retentivní proměnné

```
VAR RETAIN
  AT %QW4 : WORD;
  AT %QW5 : WORD := 16#FF00;
END_VAR
```

- n Po studeném startu je %QW4 inicializována na 0, %QW5 má nejvyšších 8 bitů inicializováno na 1, ostatní na 0

- n Umístění symbolických proměnných

```
VAR
  Lim5 AT %IX27 : BOOL; (* Lim5 je 27. vstupní bit *)
  Teplota AT %IW28 : INT; (* Teplota je 28. vstupní slovo *)
  PolohaVentilu AT %QW29 : INT := 100; (* 29. Výstupní WORD je
  přiřazen INT proměnné PolohaVentilu inicializované na 100 *)
END_VAR
```

21

## Proměnné – příklady

(2/2)

- n Umístění a inicializace polí

```
VAR
  Outs AT %QW6 : ARRAY [0..9] OF INT := [10(1)];
  Bits : ARRAY [0..7] OF BOOL := [1,1,0,0,0,1,0,0];
  Tbt : ARRAY [1..2][1..3] OF INT := [1,2,3(4),6];
END_VAR
```

- n Outs je pole 10 celých čísel inicializovaných na 1 souvisle alokovaných od pozice %QW6

- n Pole Tbt je pole 2\*3 celých čísel: Tbt[1,1]:=1; Tbt[1,2]:=2; Tbt[1,3]:=4; Tbt[2,1]:=4; Tbt[2,2]:=4; Tbt[2,3]:=6;

- n Deklarace a inicializace retentivního pole

```
VAR RETAIN
  RTbt : ARRAY [1..2][1..3] OF INT := [1,2,3(4)];
END_VAR
```

- n Po studeném startu je RTbt inicializováno na stejné hodnoty jako Tbt v předchozím příkladu, kromě RTbt[2,3]=0;

- n Automatická alokace symbolických proměnných

```
VAR
  Den : WORD; (* Den je 16 bit. string *)
  Awd, Bwd, Cwd : INT; (* 3 celočíselné proměnné *)
  Okay : STRING[10] := 'OK'; (* zainicializovaný řetězec *)
END_VAR
```

22

## Sekvenční funkční grafy (SFC)

- n Sekvenční funkční grafy (Sequential Function Charts, SFC) jsou určeny pro návrh sekvenčního řízení

- n Výrazově bohatý grafický formalismus podobný stavovým diagramům

- n Umožňují rozčlenit složitý program na množinu kroků (stavů) a přechodů mezi nimi

- n Ke každému kroku je přidružena množina akcí

- n Ke každému přechodu je přidružena množina podmínek

23

## SFC – Jednoduchý příklad

- n Příklad „blikač“

- n S1 – výchozí stav

- n T1 – při nastavení signálu **Start** na **TRUE** se přechází do stavu S2

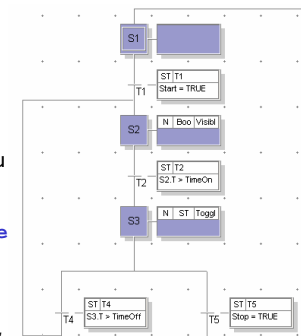
- n S2 – Nastaví se proměnná **Visible** na **TRUE**

- n T2 – po setrvání ve stavu S2 po dobu větší než **TimeOn** se přechází do stavu S3

- n S3 – invertuje se proměnná **Visible**

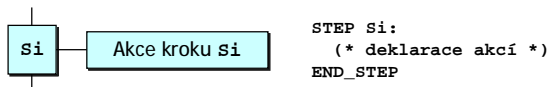
- n T4 – po uplynutí času většího než **TimeOff** se ze stavu S3 přechází do S2 a postup se opakuje

- n T5 – je-li nastavena proměnná **Stop**, přechází se ze stavu S3 do výchozího stavu S1



24

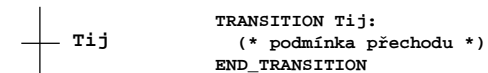
## Krok



- n **Krok (Step)** určuje situaci, kdy chování vstupů a výstupů dané POU je definováno přidruženými akcemi
  - n Krok lze reprezentovat graficky nebo textově, viz obrázek
- n **Si.X** – příznak kroku (step flag)
  - n Udává, zda je krok aktivní nebo neaktivní a je reprezentován booleovskou proměnnou **si.x**, kde **si** je název kroku
- n **Si.T** – uplynulý čas (elapsed time)
  - n Udává čas po který setrvává SFC v daném kroku
  - n Tento čas v proměnné typu **TIME** je po opuštění kroku ponechán na hodnotě, jakou měl v okamžiku opuštění, při aktivaci kroku je nastaven na hodnotu **t#0s**

25

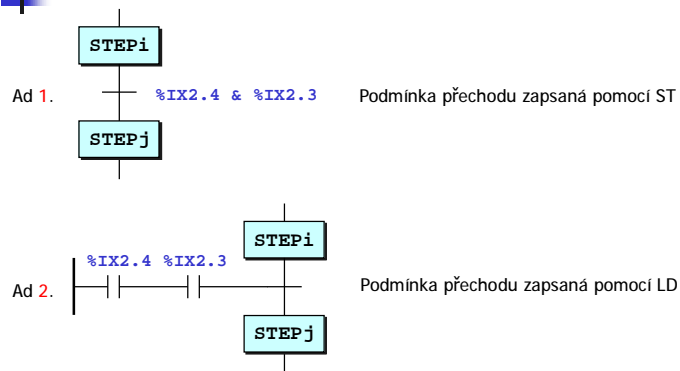
## Přechod



- n **Přechod (Transition)** představuje podmínku, za které se řízení předává z jednoho nebo více předchozích kroků do jednoho nebo více následujících kroků
  - n Přechod lze reprezentovat graficky nebo textově, viz obrázek.
  - n Při splnění podmínky vyhodnocované jako booleovský výraz dojde k přechodu ve směru shora dolů
- n Podmínka přechodu může být k daném přechodu přidružena následovně (viz příklady na následujících stránkách):
  1. Jako booleovský výraz v ST
  2. Jako liové schéma (LD) jehož výstup přiléhá (adjacent) k vertikální lince
  3. Jako FBD jehož výstup přiléhá k vertikální lince
  4. Jako LD nebo FBD připojený k vertikální lince pomocí konektorů
  5. Jako konstrukce **TRANSITION...END\_TRANSITION** užívající ST
  6. Jako konstrukce **TRANSITION...END\_TRANSITION** užívající IL
  7. Jménem přechodu přidruženého ke konstrukci **TRANSITION...END\_TRANSITION**, v jejímž těle je kód v LD, FBD, IL nebo ST vracející hodnotu booleovské proměnné

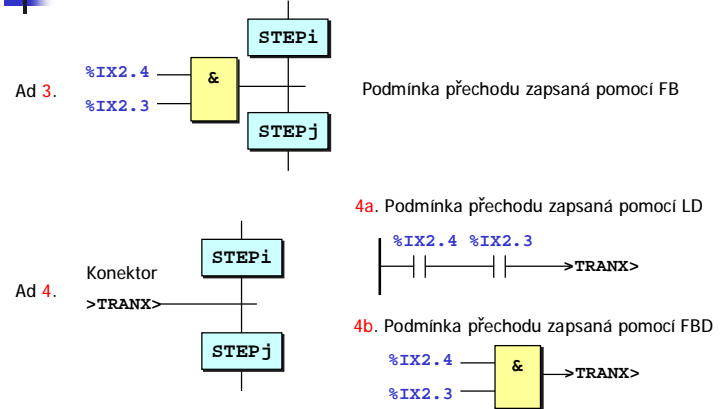
26

## Příklady přechodů (1/3)



27

## Příklady přechodů (2/3)



28

## Příklady přechodů

(3/3)

Ad 5. Textový ekvivalent případu 1. využívající jazyk ST

```
STEP STEPi: END_STEP
TRANSITION FROM STEPi TO STEPj
:= %IX2.4 & %IX2.3;
END_TRANSITION
STEP STEPj: END_STEP
```

Ad 6. Textový ekvivalent případu 1. využívající jazyk IL

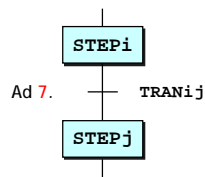
```
STEP STEPi: END_STEP
TRANSITION FROM STEPi TO STEPj:
LD %IX2.4
AND %IX2.3;
END_TRANSITION
STEP STEPj: END_STEP
```

7c. Podmínka přechodu zapsaná pomocí IL

```
TRANSITION TRAnij FROM STEPi TO STEPj:
LD %IX2.4
AND %IX2.3;
END_TRANSITION
```

7d. Podmínka přechodu zapsaná pomocí ST

```
TRANSITION TRAnij FROM STEPi TO STEPj
:= %IX2.4 & %IX2.3;
END_TRANSITION
```



29

## Akce

(1/2)

- n Akce jsou v SFC přidruženy ke krokům
  - n Ke každému kroku může být přidružena žádná, jedna nebo několik akcí
  - n Krok, který nemá žádnou akci se považuje za „čekací“ do té doby, než nastane podmínka jeho opuštění
- n Deklarace akce se skládá ze jména akce a z těla akce
  - n Rozsah platnosti dané akce (scope) je v rámci dané POU obsahující deklaraci
- n Tělem akce může být:
  - n Booleovská proměnná
  - n Posloupnost instrukcí v IL
  - n Posloupnost příkazů v ST
  - n Posloupnost příček v LD
  - n Propojení bloků ve FBD
  - n Další SFC
- n Akce se přidružují ke krokům buď v grafických blocích nebo textově

30

## Akce

(2/2)

- n Chování akce je určeno tzv. kvalifikátorem akce, viz tabulku
  - n Podrobněji v samostatné přednášce o SFC

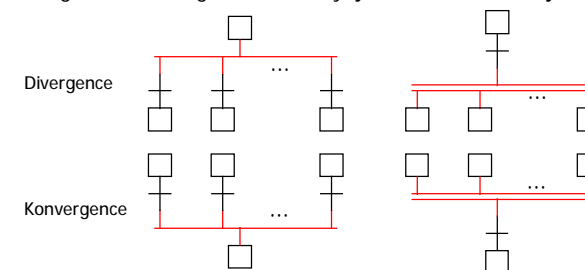
Kvalif.	Definice	Chování akce
žádný	Non-stored (null qualifier)	Jako kvalifikátor <b>N</b>
<b>N</b>	Non-stored	Provádí se, když je daný krok aktivní
<b>R</b>	overriding Reset	Ukončuje provádění akcí s kvalifikátorem <b>S, SD, SL</b>
<b>S</b>	Set (Stored)	Akce se provádí, dokud není dosažen stav v němž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

31

## Divergence a konvergence

(1/3)

- n Prostředky větvení programu v SFC
- n **Divergence** – vícenásobné spojení z jednoho symbolu (kroku nebo přechodu) na několik symbolů opačného typu
- n **Konvergence** – vícenásobné spojení z několika symbolů (kroků nebo přechodů) na jeden symbol opačného typu
- n Divergence a konvergence mohou být jednoduché nebo dvojité, viz obr.



32



## Divergence a konvergence (2/3)

- n **Jednoduchá divergence** – spojení jednoho kroku s několika přechody
  - n Z přidružených podmínek přechodu může být splněna nejvýše jedna (detaily viz samostatnou přednášku k SFC)
  - n Při splnění jedné podmínky se začne vykonávat po ní následující krok
  - n Může tedy běžet vždy jen jedna z větví vzniklých touto divergencí, proto se jí někdy říká **OR-divergence**
- n **Jednoduchá konvergence** – spojení několika přechodů s jedním krokem
  - n Používá se pro spojení větví vzniklých jednoduchou divergencí
- n **Speciální případy:**
  - n Tzv. **přeskočení sekvence** (sequence skip), kdy některá větev neobsahuje ani kroky ani přechody
  - n Sekvenční smyčka (sequence loop), kdy některá nebo několik větví se vrací do některého z předchozích stavů

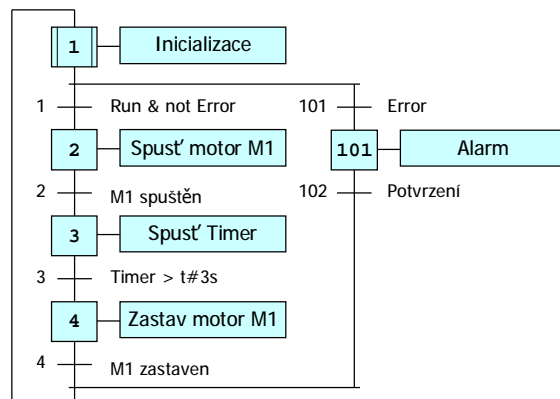
33

## Divergence a konvergence (3/3)

- n **Dvojitá divergence** – spojení jednoho přechodu s několika kroky
  - n Odpovídá paralelnímu běhu procesů reprezentovaných jednotlivými větvemi, nazývanými **současně běžící sekvence** (simultaneous sequences)
  - n Proto se jí někdy říká **AND-divergence**
  - n Spustí se, je-li aktivní předchozí krok a splní-li se podmínka vycházející z něj do této divergence. Pak se inicializují všechny počáteční kroky všech současně běžících sekvencí
- n **Dvojitá konvergence** – spojení několika kroků s jediným přechodem
  - n Používá se pro spojení větví vzniklých jednoduchou divergencí
  - n Spouští se, když jsou aktivní všechny kroky k ní připojené a je splněna z ní vycházející podmínka
  - n Za konvergencí jsou všechny předchozí aktivní kroky deaktivovány a je aktivován jediný krok následující po podmínce vycházející z divergence

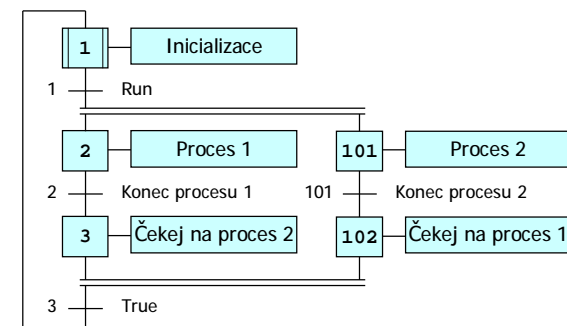
34

## Příklad jednoduché divergence a konvergence



35

## Příklad dvojitě divergence a konvergence



36