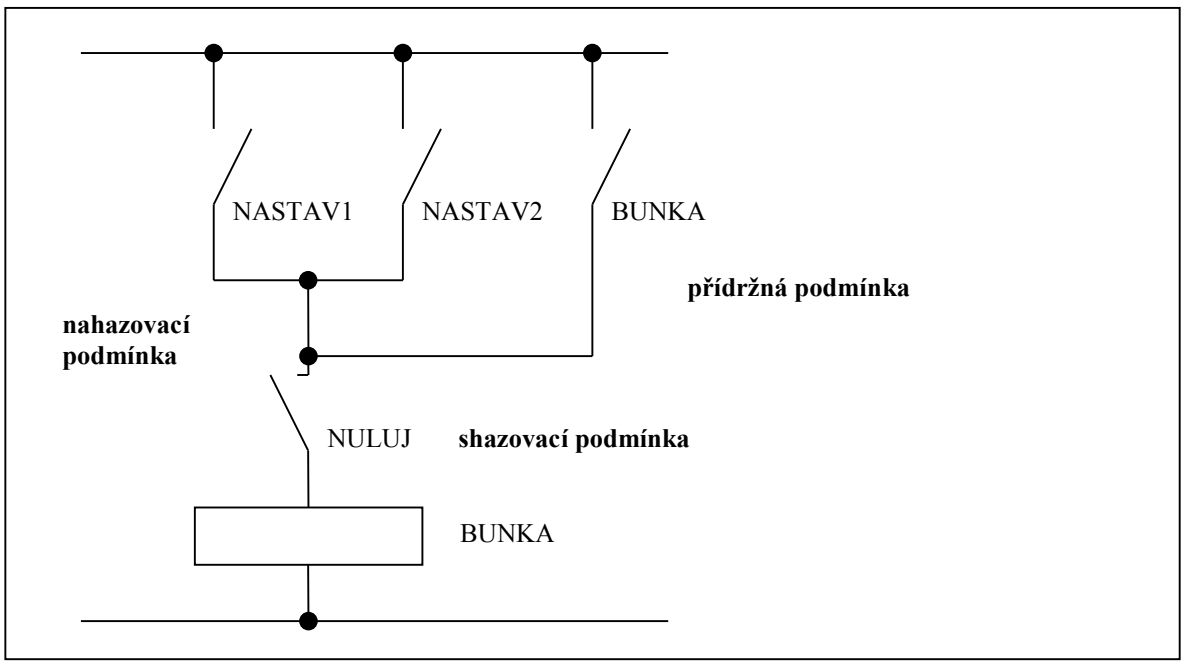
4

# 4. LOGICKÉ SEKVENČNÍ CELKY

## 4.1 Strukturalizace PLC programu

PLC program může být vytvořen různými způsoby. Klasický přístup při návrhu PLC programu je založen na navržení sekvenčně-kombinační logiky nebo přepsání reléové logiky do instrukcí jazyka PLC836. Každé relé má pak deklarovanou vlastní vnitřní bitovou buňku. Při řízení zápisu do této buňky je definována pomocí instrukcí ***nahazovací***, ***přídržná a shazovací*** podmínka. V programu je pak zápis do buňky na jediném místě.

Příklad:
Příklad klasického návrhu PLC programu.



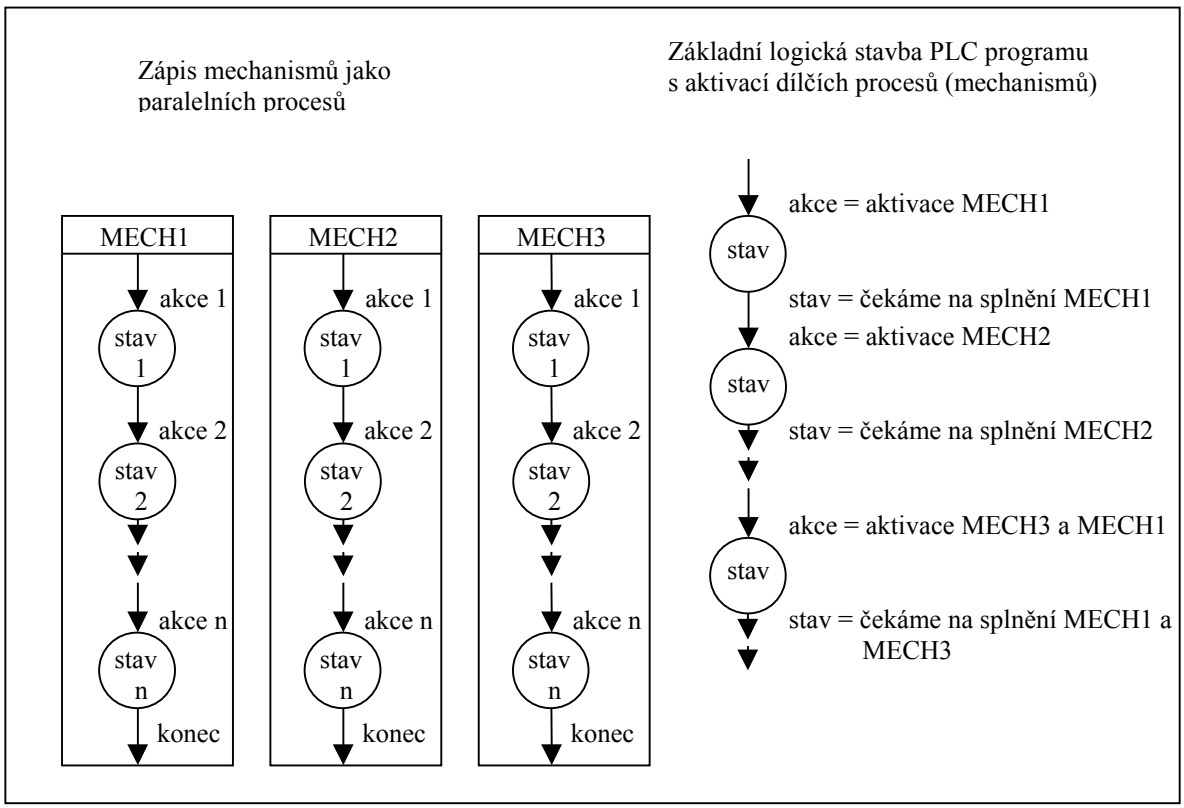

```
LDR     NASTAV1     ;nahazovací podmínka je logický součet
LO      NASTAV2     ;bitů NASTAV1 NASTAV2
LO      BUNKA       ;přídržná podmínka
LA      -NULUJ      ;bit NULUJ vynuluje klopný obvod
WR      BUNKA       ;paměťová bitová proměnná
```

Modernější přístup při návrhu PLC programu je založen na popisu jednotlivých procesů pomocí vývojových diagramů, které tvoří logické sekvenční celky (mechanismy).

Sekvenční logický celek (dále jen MECHANISMUS) je programový úsek, který se skládá z akcí a stavů. Akcí se rozumí nastavení příslušných bitů, které aktivují jednotlivé části mechanismů jako např. zapnutí hydrauliky, vypnutí stykačů ventilátorů atd. Stavem se rozumí kontrola splnění požadovaných vstupních podmínek mechanismů, např. "čekej na sepnutí kontaktu stykače od hydrauliky". Mechanismus lze znázornit vývojovým nebo stavovým diagramem.

Mechanismus se aktivuje aktivační proměnnou (NAZEV). Jedná se o bitovou proměnnou, deklarovanou automaticky zápisem mechanismu. Tuto proměnnou je možno různě nastavovat nebo testovat v programu interfejsu a tak řídit chod daného mechanismu. Mechanismus se aktivuje nastavením aktivační proměnné a po dokončení funkce mechanismu se tato proměnná automaticky vynuluje.

PLC program může obsahovat větší počet mechanismů pro řešení všech dílčích procesů stroje. (start vřetena, reverzace vřetena, stop vřetena, upínání a uvolňování os, orientovaný stop, přesun ramene, start a stop chlazení, jednotlivé kroky výměny nástroje, vyhledávání nástroje, řazení otáčkových řad atd...). V jeden okamžik může být aktivován libovolný počet mechanismů a tyto běží současně, nezávisle na sobě.




Návrh PLC programu se pomocí mechanismů velmi zjednoduší a zpřehlední. PLC program z hlediska návrhu získá ***strukturalizaci***. Strukturalizace je umožněna tím, že se pomocí mechanismů popíší všechny dílčí procesy stroje a tak hlavní logická stavba PLC programu už není zatížena jejich problematikou. Hlavní logická stavba PLC programu jenom aktivuje jednotlivé dílčí procesy (mechanismy) a kontroluje jejich splnění a případný výskyt chyb. Mechanismy se skládají z akcí a stavů. Pro každý stav jsou kontrolovány jen podmínky nevyhnutně potřebné pro přechod do následujícího stavu.

V každý okamžik možno sledovat průběh jednotlivých mechanismů: které z nich jsou aktivovány a v jakém jsou stavu. Tak přirozeně dochází k velkému ulehčení z hlediska ladění a diagnostikování provozu stroje, protože každý stav, ve kterém se mechanismus nachází je dán tzv. ***podmínkou pokračování*** (podmínková analýza). Podmínky pokračování (pro přechod do následujícího stavu) je užitečné sledovat při zastavení chodu mechanismu. Silnou stránkou mechanismů je i to, že splnění podmínek může být časově omezeno a tak nedojde k trvalému zastavení chodu mechanismu. Mechanismus se ukončí s chybou a v popisu chyby na obrazovce systému může být přesně v textové podobě vyjádřena podmínka pokračování daného mechanismu i s čísly svorek jednotlivých vstupů, které podmínka obsahuje. Tak jsou velmi ulehčeny servisní práce při poruchách.

Kromě podmínek, které možno sledovat a jsou užitečné v případě zastavení chodu mechanismu, možno sledovat i časový průběh chodu mechanismu, to je jak dlouho se průběh mechanismů zdržel v jednotlivých stavech a kterými stavy procházel (tzv. časová analýza).

## 4.2 Instrukce pro logické sekvenční celky

| | |
|---|---|
| **instrukce** | **MECH_BEGIN**<br>**MECH_END**<br>**MECH_INIT** |

| | | |
|---|---|---|
| **funkce** | **MECH_BEGIN** | **začátek logického celku** |
| | **MECH_END** | **konec logického celku** |
| | **MECH_INIT** | **inicializace logického celku** |
| **syntax** | **MECH_BEGIN** | **mech** |
| | **MECH_END** | **mech** |
| | **MECH_INIT** | **mech** |

Instrukce **MECH_BEGIN** musí být zapsána na začátku mechanismu. Proměnná "mech" je názvem mechanismu a současně názvem jeho aktivační bitové proměnné. Instrukce **MECH_END** musí být zapsána na konci příslušného mechanismu. Proměnná "mech" má stejný význam, t.j. je totožná s názvem v MECH_BEGIN.

Instrukce **MECH_INIT** uvede mechanismus do klidového stavu. Používá se v inicializaci interfejsu nebo v obsluze chybových stavů. Instrukci je nutno použít před prvním průchodem mechanismu, proto se doporučuje napsat instrukci do inicializačního modulu interfejsu (viz dále). Proměnná "mech" má opět stejný význam jako u MECH_BEGIN. Viz "Společné zásady" (dále).

| instrukce | EX |
|---|---|
| | EX0 |
| | EX1 |
| | BEX |

| funkce | EX | přerušení po dobu jednoho cyklu interfejsu |
|---|---|---|
| | EX0 | přerušení činnosti pokud RLO = 0 |
| | EX1 | přerušení činnosti pokud RLO = 1 |
| | BEX | začátek podmínky |

| syntax | EX |
|---|---|
| | EX0 |
| | EX1 |
| | BEX |

Instrukce pro definici stavu **EX, EX0, EX1** zastaví provádění sekvenčního logického celku do doby splnění kontrolované podmínky. Instrukce možno používat jenom uvnitř logických sekvenčních celků. Tato přerušení se nazývají stavy mechanismu.

Instrukce EX způsobí nepodmíněné zastavení provádění sekvencí na dobu jednoho cyklu interfejsu.

Instrukce BEX je podobná jako instrukce EX, ale neprovede se zastavení provádění sekvencí. Instrukce se používá pro definici začátku logické podmínky před instrukcemi EX0, EX1, TEX0 a TEX1. Instrukci je výhodné použít v časově náročných procesech, kdy ztráta jednoho cyklu interfejsu může vadit.

Instrukce EX0, resp. EX1 pozastaví provádění dalších sekvencí mechanismu po dobu, pokud RLO=0, resp. RLO=1. Před instrukcemi EX0 a EX1 může být napsána logická podmínka pro pokračování v daném stavu. Pro daný stav v každém cyklu dojde k vyhodnocování podmínek zapsaných v úseku mezi předposlední a poslední dosaženou instrukcí typu EX.

Ve skutečnosti instrukce EX0 a EX1 nečekají na daném místě pokud není splněna podmínka, ale provedou skok na konec mechanismu. V dalším průchodu mechanismem se skočí ze začátku mechanismu na předposlední instrukci typu EX a znovu se vyhodnotí podmínky po instrukci EX0 nebo EX1. Viz kapitolu "Společné zásady mechanismů" (dále).

Instrukce EX, EX0, EX1, TEX0, TEX1 a BEX jsou ***koncové instrukce*** pro logické rovnice. Instrukce nezachovávají RLO a DR registr.

| **instrukce** | **TEX0**<br>**TEX1** |
|---|---|

| | | |
|---|---|---|
| **funkce** | **TEX0** | **přerušení činnosti pokud RLO = 0 s časovou kontrolou** |
| | **TEX1** | **přerušení činnosti pokud RLO = 1 s časovou kontrolou** |
| **syntax** | **TEX0(TEX1)** | **citac, doba, error** |
| | **TEX0(TEX1)** | **citac, doba, error [, chyba ]** |
| | **TEX0(TEX1)** | **[TYPE.]citac, doba, error [, chyba ]** |
| | **TEX0(TEX1)** | **TYPE. (citac+n), (doba+m), error [, chyba]** |
| | **TEX0(TEX1)** | **- , doba, error [, chyba]** |
| | | **TYPE = BYTE. WORD.** |

Instrukce **TEX0**, resp. **TEX1** pozastaví provádění dalších sekvencí mechanismu, pokud RLO=0, resp. RLO=1, ale maximálně po předem stanovenou dobu "doba" (BYTE, WORD, konstanta). Pokud logická podmínka pokračování, která je napsána před instrukcemi TEX0 a TEX1, nebude splněna po zadanou dobu, bude program pokračovat od návěští, které je zadáno v parametru instrukce "error".

Parametr "citac" může být typu BYTE nebo WORD. Pro systémy řady CNC8x9 – DUAL od verze 6.028 je první parametr instrukce pro "citac" nepovinný. V tomto případě si instrukce deklaruje automatickou proměnnou v lokálních datech. Místo 1. parametru se v tomto případě napíše znak pomlčky nebo “NIL”.

Jedná se o silné instrukce, z kterých se mechanismy skládají, především protože umožňují ošetřit chybové stavy a tak nikdy nedojde k trvalému zastavení chodu mechanismu. V textovém popisu případné chyby může být pak přesně popsaná ***podmínka pokračování*** v daném stavu a tak je umožněno diagnostikování stroje.

Pro daný stav v každém cyklu dojde k vyhodnocování podmínek zapsaných v úseku mezi předposlední a posledně dosaženou instrukcí typu EX. Instrukce vyžaduje tři povinné parametry. Určení časového členu "citac", žádanou dobu zpoždění "doba" a návěští "error", od kterého bude program pokračovat, jestliže nebude splněna podmínka za stanovenou dobu. "Doba" může být přímá hodnota nebo adresa proměnné. Viz "Společné zásady" (dále).

Čtvrtým nepovinným parametrem "chyba" může být číselná hodnota (většinou číslo chyby), která zůstane v datovém DR registru při skoku na návěští "error". Tím je umožněno, že z různých instrukcí TEX0 a TEX1 možno v mechanismu skákat na stejné místo "error" se společnou obsluhou chybového stavu mechanismu.

Instrukce TEX0 a TEX1 jsou ***koncové instrukce*** pro logické rovnice. Instrukce nezachovávají RLO a DR registr, kromě odskoku na návěští "error", když je použitý čtvrtý parametr "chyba".

Možnost předefinování typu operandů "citac" a "doba" je popsáno v kapitole "Způsoby předefinování typu u datových proměnných. Předefinování typu se vztahuje na oba operandy "citac" a "doba", proto v tomto případě není umožněno použít operandu "doba" deklarovaného jako konstanta.

| instrukce | TIM | |
|---|---|---|
| **funkce** | **TIM** | **časové zpoždění** |
| **syntax** | **TIM** | **citac, doba** |
| | **TIM** | **[TYPE.]citac, doba** |
| | **TIM** | **TYPE.(citac+n), (doba+m)** |
| | **TIM** | **- , doba** |
| | | **TYPE = BYTE. WORD.** |

Instrukce časového zpoždění TIM zastaví provádění sekvenčního logického celku na stanovenou dobu. Instrukce vyžaduje dva parametry. Určení časového členu "citac" a žádanou dobu zpoždění "doba". "Doba" může být konstanta nebo adresa proměnné. Oba parametry mohou být typu BYTE nebo WORD. Instrukce TIM má podobné působení jako instrukce EX a může být použita jen uvnitř mechanismu.

Pro systémy řady CNC8x9 – DUAL od verze 6.028 je první parametr instrukce pro "citac" nepovinný. V tomto případě si instrukce deklaruje automatickou proměnnou v lokálních datech. Místo 1. parametru se v tomto případě napíše znak pomlčky nebo “NIL”.

Možnost předefinování typu operandů "citac" a "doba" je popsáno v kapitole "Způsoby předefinování typu u datových proměnných. Předefinování typu se vztahuje na oba operandy "citac" a "doba", proto v tomto případě není umožněno použít operandu "doba" deklarovaného jako konstanta.

## 4.3 Společné zásady mechanismů

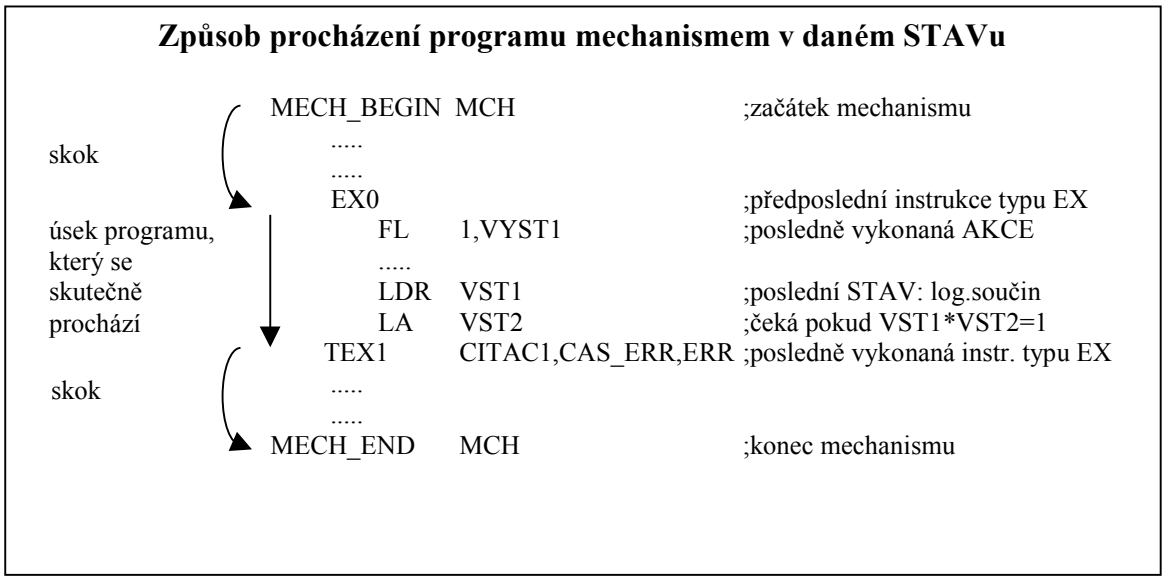
Instrukce EX, EX0, EX1, TEX0, TEX1 a TIM mohou být použity pouze uvnitř sekvenčních logických celků, vymezených instrukcemi MECH_BEGIN a MECH_END. Budeme je nazývat společně: ***instrukce typu EX***.

***Každý mechanismus musí být inicializován pomocí instrukce MECH_INIT v modulu PIS_INIT a případně v PIS_CLEAR (viz kapitolu "Struktura PLC programu").***

Modul přípravných a závěrečných funkcí je také sekvenčním logickým modulem a proto v nich lze využít instrukce všechny instrukce typu EX i samostatně. Použití těchto instrukcí v uvedených modulech způsobí zastavení provádění přípravných nebo závěrečných funkcí do doby splnění kontrolované podmínky (viz příklad).

Instrukce EX0, EX1, TEX0, TEX1 a TIM pozastaví provádění dalších sekvencí mechanismu po dobu danou podmínkou ( pokud RLO=0, resp. RLO=1) nebo dobu danou nastaveným časem ( u instrukci TIM). Před instrukcemi EX0, EX1, TEX0 a TEX1 může být napsána logická podmínka pro pokračování v daném stavu. Pro daný stav v každém cyklu dojde k vyhodnocování podmínek zapsaných v úseku mezi předposlední a poslední dosaženou instrukcí typu EX.

Ve skutečnosti instrukce typu EX nečekají na daném místě pokud není splněna podmínka, ale provedou skok na konec mechanismu. V dalším průchodu mechanismem se skočí ze začátku mechanismu na předposlední instrukci typu EX a znovu se vyhodnotí podmínky po poslední instrukci typu EX.



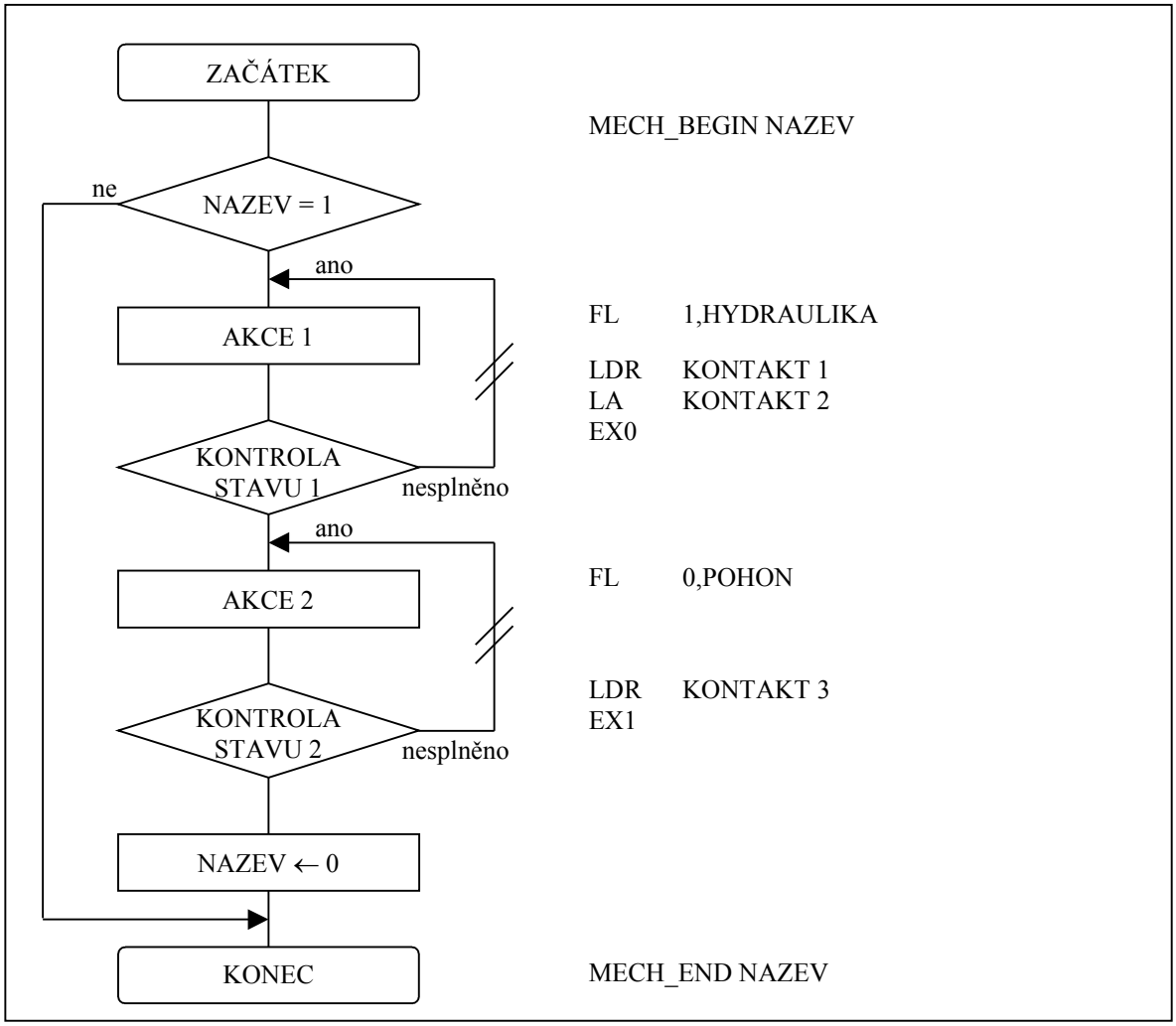

Mechanismy možno aktivovat a dezaktivovat i z jiných mechanismů. Případ ***dezaktivace mechanismů*** se provede instrukci MECH_INIT, která vynuluje bitovou řídící proměnnou mechanismu a nastaví pokračující adresu na začátek. Dezaktivaci mechanismů je výhodné použít v případě ***protichůdných mechanismů,*** kdy mechanismus s větší prioritou dezaktivuje z bezpečnostních důvodů svůj protichůdný mechanismus. Například mechanismus stopu vřetene dezaktivuje případně rozpracovaný mechanismus pro start vřetene.

Mechanismy je také možno zacyklit a tak se mohou s výhodou používat i ***trvale zacyklené mechanismy.*** Tyto pak mají vlastnost sekvenčních driverů. Trvale zacyklený mechanismus musí mít v těle cyklu aspoň jednu instrukci typu EX !

Definicí mechanismu *"mech"* se provede automatická deklarace bitové proměnné s názvem "*mech*", deklarace paměťové buňky typu WORD pro pokračující adresu mechanismu a paměťové buňky typu WORD s názvem "mech_LINE" pro aktuální linku programu, ve které se mechanismus nachází. Při kompilaci PLC programu programem TECHNOL (viz. dále) se vytvoří také kontrolní listing programu s rozšířením ".LS1" který je doplněn o čísla řádků (linky) programu. Při každé definici stavu pomocí příkazů EX, EX0, EX1, TEX0, TEX1 a TIM se pro kontrolu zapisuje číslo linky do buňky *"mech*_LINE". Pomocí ní je umožněno jednoduché sledování, ve kterém stavu se mechanismus nachází. Pokud mechanismus není aktivován, je buňka "*mech*_LINE" vynulována.

Příklad:
Nastartování mechanismu s názvem "CW" v přípravních funkcích a čekání na provedení mechanismu:

```
FL      1,CW    ;Nastavení aktivační proměnné.
EX
LDR     CW      ;Kontrola vykonání mechanismu.
EX1             ;Čeká, pokud CW = 1
```




Příklad:
Nastartování mechanismu s názvem "CW" v přípravných funkcích a čekání na provedení mechanismu po dobu 10 sec. Jestli se mechanismus za 10 sec neprovede, pokračuj na obsluhu chyby ERR1:

```
        EQUI    D500,500
        EQUI    CHYBA1,12H          ;ERROR 4.12

        FL      1,CW                ;Nastavení aktivační proměnné.
        EX
        LDR     CW                  ;Kontrola vykonání mechanismu.
        TEX1    CITAC,D500,ERR1     ;Čeká, pokud CW = 1, ale maximálně
        ....                        ;10 sec. Jestli CW = 1 déle než 10 sec.,
        ....                        ;pokračuje na ERR1.

ERR1:   LOD     CHYBA1              ;Obsluha chyby. Překročena časová
        STO     BZH11               ;kontrola správnosti chodu mechanismu.
        ....
        ....
        ....
```

Příklad:
Zapište mechanismus pro reverzaci otáček vřetena z CW do CCW. Mechanismus nazvěte CWCCW.

```
        MECH_BEGIN      CWCCW       ;Začátek mechanismu.
        FL              0,SMP       ;Vynuluj stykač motoru-pozitiv.
        LDR             KSMP
        EX1                         ;Čekej, pokud kontakt stykače
                                    ;motoru-pozitiv je v jedničce.
        TIM             CAS1,D02    ;Zpoždění 0,2 sec.
        FL              1,SMN       ;Zapni stykač motoru-negativ.
        LDR             KSMN
        EX0                         ;Čekej, pokud kontakt stykače
                                    ;motoru-negativ je v nule.
        MECH_END        CWCCW       ;Konec mechanismu.
```

*Poznámka:*
*V reálném případě by bylo vhodnější v mechanismu CWCCW místo instrukcí EX1 a EX0 použít instrukce TEX1 a TEX0, tak jak je to v následujícím příkladě.*

Příklad:
Zapište mechanismus pro reverzaci otáček vřetena z CW do CCW. Mechanismus nazvěte CWCCW. Jestli nespadne kontakt od stykače motoru-pozitiv KSMP do doby 1 sec., zahlas chybu 4.04. Jestli nepřijde kontakt od stykače motoru-negativ KSMN do doby 1 sec., vypni stykač motoru a zahlas chybu 4.05.

```
                EQUI        D1,50                   ;Doba 1 sec.
                EQUI        D02,10                  ;Doba 0,2 sec.

MECH_BEGIN                  CWCCW                   ;Začátek mechanismu.
                FL          0,SMP                   ;Vypni stykač motoru-pozitiv.
                LDR         KSMP
                TEX1        CITAC1,D1,CW_ERR,04     ;Čekej, pokud kontakt stykače
                                                    ;motoru-pozitiv je v jedničce.
                TIM         CITAC,D02               ;Zpoždění 0,2 sec.
                FL          1,SMN                   ;Zapni stykač motoru negativ.
                LDR         KSMN
                TEX0        CITAC,D1,CW_ERR,05      ;Čekej, pokud kontakt stykače
                                                    ;motoru-negativ je v nule, ale
                                                    ;maximálně 1 sec.
                JUM         CW_END                  ;Skok na konec mechanismu.
                                                    ;Obsluha chyby mechanismu.
CW_ERR:         FL          0,SMN                   ;Vypni stykač motoru-negativ.
                STO         BZH11                   ;Nastav chybu 4.05, 4.04
CW_END:
MECH_END                    CWCCW                   ;Konec mechanismu.
```

Příklad:
Způsoby zápisu instrukcí TEX0,TEX1 a TIM s předeklarováním typu:

```
CITAC:      DS      100         ;pole čítačů
REKONFIG:   DS      100         ;pole rekonfigurovatelných proměnných

            TEX0    WORD.(CITAC+30),(REKONFIG+14),ERROR
            TEX1    BYTE.(CITAC+13),(REKONFIG+21),ERROR,23h
            TIM     WORD.(CITAC+20),(REKONFIG+10)
```