Zdrojové kódy používané v DP:

Globální proměnné (GVL):

GVL_Axis:

```
VAR_GLOBAL

// Referování os (axis) ustavovacích patek.

    axis1       : AXIS_REF;
    axis2       : AXIS_REF;
    axis3       : AXIS_REF;
    axis4       : AXIS_REF;
    axis5       : AXIS_REF;
    axis6       : AXIS_REF;

END_VAR
```

GVL_Konstanty:

```
VAR_GLOBAL CONSTANT

// Definice konstant používaných v celém programu.

    fgTihoveZrychleni       : REAL  := 9.81;        // Hodnota tíhového zrychlení [m/s^2].
    ngVzdalenostNodu        : INT   := 300;         // Vzdálenost mezi nody 1 až 3 a 2 až 4.
    ngSklon                 : INT   := 1000;        // Sklon je (v mm, nebo um) na 1000mm.
    ngPrevod                : INT   := 1000;        // Převod z um na mm (um přijdou vhodnější).
    fSklon_v_ose_X_pocatecni        : INT   := 0;   // Prvotní sklon v ose X, který zadává obsluha pří zapnutí stroje.
    fSklon_v_ose_Y_pocatecni        : INT   := 0;   // Prvotní sklon v ose Y, který zadává obsluha pří zapnutí stroje.
    fgVzdalenost_X1         : REAL  := 496;         // Přiřazení hodnoty vzdálenosti X1. Vzdálenost rohových patek 1 a 2 [mm].
    fgVzdalenost_X2         : REAL  := 592;         // Přiřazení hodnoty vzdálenosti X2. Vzdálenost rohových patek 5 a 6 [mm].
    fgVzdalenost_Y          : REAL  := 1268;        // Přiřazení hodnoty vzdálenosti Y. Vzdálenost dvojice patek 1,2 a 5,6 [mm].

END_VAR
```

GVL_Sklon_aktualni:

```
VAR_GLOBAL

    fgSklon_v_ose_X_aktualni        : REAL;     // Globální proměnná pro vypočítaný aktuální sklon v ose X.
    fgSklon_v_ose_Y_aktualni        : REAL;     // Globální proměnná pro vypočítaný aktuální sklon v ose Y.
    fgSklon_v_ose_X_aktualni_vp1    : REAL;     // Globální proměnná pro vypočítaný aktuální sklon vp1 v ose X.
    fgSklon_v_ose_Y_aktualni_vp1    : REAL;     // Globální proměnná pro vypočítaný aktuální sklon vp1 v ose Y.
    fgSklon_v_ose_X_aktualni_vp2    : REAL;     // Globální proměnná pro vypočítaný aktuální sklon vp2 v ose X.
    fgSklon_v_ose_Y_aktualni_vp2    : REAL;     // Globální proměnná pro vypočítaný aktuální sklon vp2 v ose Y.
    fgSklon_v_ose_X_aktualni_vp3    : REAL;     // Globální proměnná pro vypočítaný aktuální sklon vp3 v ose X.
    fgSklon_v_ose_Y_aktualni_vp3    : REAL;     // Globální proměnná pro vypočítaný aktuální sklon vp3 v ose Y.

END_VAR
```

GVL_Vstupy_a_vystupy:

```
VAR_GLOBAL

// Definice žadaných poloh:

    fP1zadana       AT      %Q*     : LREAL;// Žádaná poloha chytré patky 1.
    fP2zadana       AT      %Q*     : LREAL;// Žádaná poloha chytré patky 2.
    fP3zadana       AT      %Q*     : LREAL;// Žádaná poloha chytré patky 3.
    fP4zadana       AT      %Q*     : LREAL;// Žádaná poloha chytré patky 4.
    fP5zadana       AT      %Q*     : LREAL;// Žádaná poloha chytré patky 5.
    fP6zadana       AT      %Q*     : LREAL;// Žádaná poloha chytré patky 6.

// Definice aktuálních poloh:

    fP1aktualni     AT      %I*     : UINT;     // Aktuální poloha chytré patky 1.
    fP2aktualni     AT      %I*     : UINT;     // Aktuální poloha chytré patky 2.
    fP3aktualni     AT      %I*     : UINT;     // Aktuální poloha chytré patky 3.
    fP4aktualni     AT      %I*     : UINT;     // Aktuální poloha chytré patky 4.
    fP5aktualni     AT      %I*     : UINT;     // Aktuální poloha chytré patky 5.
    fP6aktualni     AT      %I*     : UINT;     // Aktuální poloha chytré patky 6.
```

```
// Definice sil na chytrých patkách:

    fF1     AT      %I*     : UINT;             // Síla na chytré patce 1.
    fF2     AT      %I*     : UINT;             // Síla na chytré patce 2.
    fF3     AT      %I*     : UINT;             // Síla na chytré patce 3.
    fF4     AT      %I*     : UINT;             // Síla na chytré patce 4.
    fF5     AT      %I*     : UINT;             // Síla na chytré patce 5.
    fF6     AT      %I*     : UINT;             // Síla na chytré patce 6.

// Definice odlehnutí na chytrých patkách:

    fOdlehnutiPatky1    AT      %I*     : UINT;         // Odlehnutí na chytré patce 1.
    fOdlehnutiPatky2    AT      %I*     : UINT;         // Odlehnutí na chytré patce 2.
    fOdlehnutiPatky3    AT      %I*     : UINT;         // Odlehnutí na chytré patce 3.
    fOdlehnutiPatky4    AT      %I*     : UINT;         // Odlehnutí na chytré patce 4.
    fOdlehnutiPatky5    AT      %I*     : UINT;         // Odlehnutí na chytré patce 5.
    fOdlehnutiPatky6    AT      %I*     : UINT;         // Odlehnutí na chytré patce 6.

// Hodnoty koncových uzlů libel, z kterých se vypočítá sklon OS v ose X a Y:

    fL1     AT      %I*     : UINT;             // Hodnota koncového uzlu 1.
    fL2     AT      %I*     : UINT;             // Hodnota koncového uzlu 2.
    fL3     AT      %I*     : UINT;             // Hodnota koncového uzlu 3.
    fL4     AT      %I*     : UINT;             // Hodnota koncového uzlu 4.

// Hodnoty vypočtených sklonů OS v ose X a Y:

    falfa1          : REAL;             // Hodnota vypočteného sklonu OS v ose X:
    fbeta1          : REAL;             // Hodnota vypočteného sklonu OS v ose Y:
    falfa2          : REAL;             // Hodnota vypočteného sklonu OS v ose X:
    fbeta2          : REAL;             // Hodnota vypočteného sklonu OS v ose Y:
    falfa3          : REAL;             // Hodnota vypočteného sklonu OS v ose X:
    fbeta3          : REAL;             // Hodnota vypočteného sklonu OS v ose Y:
```

```
// Za definování proměnné pro tlačítko TOTAL STOP:

        bgTotalStop     AT %I*          : BOOL;         // Proměnná pro tlačítko TOTAL STOP

END_VAR
```

Uživatelské datové typy (DUT):

DUT_PolohyZ:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o aktuálních polohách ustavovacích patek.

        DUT_PolohyZ    : ARRAY [1..6] OF LREAL;

END_TYPE
```

DUT_PolohyA:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o aktuálních polohách ustavovacích patek.

        DUT_PolohyA    : ARRAY [1..6] OF UINT;

END_TYPE
```

DUT_Libely:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o krajních hodnotách libel pro stanovení naklopení v osách X a Y.

        DUT_Libely      : ARRAY [1..4] OF UINT;

END_TYPE
```

DUT_Laktualní:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o vypočtených hodnotách Laktualni pro stanovení naklopení v osách X a Y.

        DUT_Laktualni           : ARRAY [1..4] OF REAL;

END_TYPE
```

DUT_Sily:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o silách neboli reakcích na ustavovacích patkách.

        DUT_Sily        : ARRAY [1..6] OF UINT;

END_TYPE
```

DUT_Sily_delta_opt:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o diference mezi aktuální silou a její optimální velikostí.

        DUT_Sily_delta_opt      : ARRAY [1..6] OF REAL;

END_TYPE
```

DUT_Sily_optimalni:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o optimálních silách neboli reakcích na ustavovacích patkách.

        DUT_Sily_optimalni      : ARRAY [1..6] OF REAL;

END_TYPE
```

DUT_OdlehnutiPatek:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o hodnotách zatížení na ustavovacích patkách. Jestli nedošlo k odlehnutí patky od lože.

        DUT_OdlehnutiPatek    : ARRAY [1..6] OF UINT;

END_TYPE
```

DUT_alfa:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o vypočtených sklonech OS v ose X.

        DUT_alfa        : ARRAY [1..3] OF REAL;

END_TYPE
```

DUT_beta:

```
TYPE

// Zde se definuje datový typ jednorozměrného pole.

// Pole obsahuje data o vypočtených sklonech OS v ose Y.

        DUT_beta        : ARRAY [1..3] OF REAL;

END_TYPE
```

Funkční bloky (FB):

FB_Patka1: Ostatní patky jsou koncipovány stejně jako ukázka pro Patku 1.

```
FUNCTION_BLOCK FB_Patka1

VAR_INPUT

        bEnable1                : BOOL      := TRUE;        // Indikace zapnuté osy.
        bStartV1                : BOOL      := FALSE;       // Pokyn tlačítkem pro vysouvání ustavovací patky.
        bStartZ1                : BOOL      := FALSE;       // Pokyn tlačítkem pro zasouvání ustavovací patky.
        bAxisVysunuti1          : BOOL      := FALSE;       // Enablování osy pro vysouvání.
        bAxisZasunuti1          : BOOL      := FALSE;       // Enablování osy pro zasouvání.
```

```
        lrOverride1             : LREAL;        // Hodnoty override, které nastavuje obsluha.
        lrVelo                  : LREAL;        // Mění hodnoru rychlosti podle obsluhy.

END_VAR

VAR_OUTPUT

        lrHodnotaVysunuti1      : LREAL;        // Zobrauje hodnotu polohy vysunutí nebo zasunutí.
        lrNulovani1             : LREAL;        // Nulování hodnoty vysunutí patky1.

END_VAR

VAR

        lrOverride1p            : LREAL;        // Definice overrideru v mcSetOverride.

// Motion Control Function Block:

        mcPower                 : MC_Power;                 // Zapínání osy.
        mcJog                   : MC_Jog;                   // Ruční pohyb osy v kladném i záporném směru.
        mcBufferMode            : MC_BufferMode;
        mcSetOverride           : MC_SetOverride;           // Nastavování Override.
        mcReadActualPosition    : MC_ReadActualPosition;    // Informace o aktuální poloze vysunutí patky.

END_VAR
```

```
// Definice lineárních os (Axis) ustavovacích patek:

// Definování osy (Axis 1):

mcPower(
        Axis:= GVL_Axis.axis1,                  // Přiřazení, která osa se bude řídit.
        Enable:= bEnable1,                      // Zapínání MC bloku.
        Enable_Positive:= bAxisVysunuti1,       // Pohyb osy v kladném směru.
        Enable_Negative:= bAxisZasunuti1,       // Pohyb osy v záporném směru
        Override:= ,
        BufferMode:= mcBufferMode,
        Options:= ,
        Status=> ,
        Busy=> ,
        Active=> ,
        Error=> ,
        ErrorID=> );

// Zajišťuje v ručním režimu vysouvání a zasouvání ustavovací patky 1

mcJog(
        Axis:= GVL_Axis.axis1,                  // Přiřazení, která osa se bude řídit.
        JogForward:= bStartV1,                  // Vysouvání osy přes tlačítko vysouvání.
        JogBackwards:= bStartZ1,                // Zasouvání osy přes tlačítko zasouvání.
        Mode:= MC_JOGMODE_CONTINOUS,            // Pohyb zajištěn pokud bude tlačítko dávat signál TRUE.
        Position:= ,
        Velocity:= lrVelo,                      // Sem se načítá zvolená hodnota obsluhou.
        Acceleration:= ,
        Deceleration:= ,
        Jerk:= ,
        Done=> ,
        Busy=> ,
        Active=> ,
        CommandAborted=> ,
        Error=> ,
        ErrorID=> );
```

```
// Definuje override pro patku 1

        lrOverride1p := lrOverride1/100;        // Protože override pracuje jen s hodnotami
                                                // v intevalu <0;1> a obsluha v <0;100> je vytvořen přepočet.

mcSetOverride(
        Axis:= GVL_Axis.axis1,          // Přiřazení, která osa se bude řídit.
        Enable:= bEnable1,                      // Zapínání MC bloku.
        VelFactor:= lrOverride1,        // Přiřazuje okolik se sníží zadaná rychlost.
        AccFactor:= ,
        JerkFactor:= ,
        Enabled=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> );

// Čtení polohy patky 1 a zapsání do proměné lrHodnotaVysunuti1.

mcReadActualPosition(
        Axis:= GVL_Axis.axis1,
        Enable:= bEnable1,
        Valid=> ,
        Busy=> ,
        Error=> ,
        ErrorID=> ,
        Position=> lrHodnotaVysunuti1);

// Nulování aktuální hodnoty vysunutí patky1.

        lrNulovani1 := lrHodnotaVysunuti1 - lrHodnotaVysunuti1;
```

FB_Hmotnost_stroje:

```
FUNCTION_BLOCK FB_Hmotnost_stroje
VAR_INPUT

	fFa	: DUT_Sily;

END_VAR

VAR_OUTPUT

	fHmotnost	:REAL;		// Proměná pro vypočítanou hmotnost OS.

END_VAR

VAR

	fg	: REAL	:= GVL_Konstanty.fgTihoveZrychleni;		// Dekladace tíhového zrychlení

END_VAR

// Funkčním blokem je vypočítávána hmotnost OS z udajů
// získaných ze silových senzorů (čidel).

fHmotnost := (fFa[1]+fFa[2]+fFa[3]+fFa[4]+fFa[5]+fFa[6])/fg;
```

FB_Poloha_COG:

```
FUNCTION_BLOCK FB_Poloha_COG

VAR_INPUT

        fFa             : DUT_Sily;     // Pole zatěžujících sil po odlehčení vnitřních patek [N].
        fX1             : REAL;         // Vzdálenost rohových patek 1 a 2 [mm].
        fX2             : REAL;         // Vzdálenost rohových patek 5 a 6 [mm].
        fY              : REAL;         // Vzdálenost dvojice patek 1,2 a 5,6 [mm].

END_VAR

VAR_OUTPUT

        fA              : REAL;         // Hodnota polohy v ose X [mm].
        fB              : REAL;         // Hodnota polohy v ose Y [mm].

END_VAR
// Výpočet COG v ose Y.

fB := (fFa[5]+fFa[6])*fY/(fFa[1]+fFa[2]+fFa[5]+fFa[6]);

// Výpočet COG v ose X.

fA := (fFa[2]*fX1+fFa[6]*(fX1+((fX2-fX1)/2))-fFa[5]*((fX2-fX1)/2))/(fFa[1]+fFa[2]+fFa[5]+fFa[6]);
```

FB_SklonXY:

```
FUNCTION_BLOCK FB_SklonXY

VAR_INPUT

    fSklon_v_ose_X_pocatecni    : REAL;         // Prvotní sklon v ose X, který zadává obsluha pří zapnutí stroje.
    fSklon_v_ose_Y_pocatecni    : REAL;         // Prvotní sklon v ose Y, který zadává obsluha pří zapnutí stroje.
    fLdelta                     : DUT_Libely;   // Pole definující Hodnoty z Nodu 1. až 4.

    nVzdalenostNodu             : INT;          // Vzdálenost mezi nodi 1 až 3 a 2 až 4. Přiřazovat hodnotu vzdálenosti až v MAIN.
    nSklon                      : INT;          // Sklon je (v mm, nebo um) na 1000mm.
    nPrevod                     : INT;          // Převod z um na mm (um přijdou vhodnější).

END_VAR

VAR_OUTPUT

    fLaktualni                  : DUT_Laktualni;    // Vypočtené aktuální hodnoty nodu 1 až 4.
    fSklon_v_ose_X_aktualni     : REAL;             // Vypočítaný sklon v ose X.
    fSklon_v_ose_Y_aktualni     : REAL;             // Vypočítaný sklon v ose Y.

END_VAR

// Funkční blok, kde se vypočítávají:
// 1) Aktuální hodnoty nodu 1 až 4 (fLaktualni[...]).

    fLaktualni[1] := (fSklon_v_ose_Y_pocatecni*nVzdalenostNodu/nSklon/nPrevod)+fLdelta[1];
    fLaktualni[2] := (fSklon_v_ose_X_pocatecni*nVzdalenostNodu/nSklon/nPrevod)+fLdelta[2];
    fLaktualni[3] := 0+fLdelta[3];
    fLaktualni[4] := 0+fLdelta[4];

// 2) Vypočte se sklon OS v ose X a Y => Stůl je umístěn ve výchozí poloze.

    fSklon_v_ose_X_aktualni := nSklon/nVzdalenostNodu*(fLaktualni[2]-fLaktualni[4])*nPrevod;
    fSklon_v_ose_Y_aktualni := nSklon/nVzdalenostNodu*(fLaktualni[1]-fLaktualni[3])*nPrevod;
```

FB_Optimalni_sily:

```
FUNCTION_BLOCK FB_Optimalni_sily

VAR_INPUT

        fbHmotnostStroje        : FB_Hmotnost_stroje;
        fbVypocetPolohyCOG      : FB_Poloha_COG;
        fX1                     : REAL;         // Vzdálenost rohových patek 1 a 2 [mm].
        fX2                     : REAL;         // Vzdálenost rohových patek 5 a 6 [mm].
        fY                      : REAL;         // Vzdálenost dvojice patek 1,2 a 5,6 [mm].
        fg                      : REAL;         // Dekladace tíhového zrychlení

END_VAR

VAR_OUTPUT

        fFopt           : DUT_Sily_optimalni;           // Pole vypočtených optimálních sil na rohové patky 1,2,5,6.

END_VAR

// Funkční blok zajišťuje výpočet optimálního zatížení rohových patek (1,2,5,6).
// Výpočet se provádí z vypočítané hmotnosti, která je volána z FB bloku (FB_Hmotnost_stroje).
// Dále jsou používány vypočtené proměnné A,B.
// A hodnoty zadané obsluhou X1,X2,Y.
// Je uvažováno tíhové zrychlení g.

// Výpočet optimálních sil, které by měly zatěžovat rohové patky.

// Patka 1:

fFopt[1] := fbHmotnostStroje.fHmotnost*fg*((fY-fbVypocetPolohyCOG.fB)*(fX1-fbVypocetPolohyCOG.fA))/(fY*fX1);

// Patka 2:

fFopt[2] := fbHmotnostStroje.fHmotnost*fg*((fY-fbVypocetPolohyCOG.fB)*fbVypocetPolohyCOG.fA)/(fY*fX1);
```

```
// Patka 5:

fFopt[5] := fbHmotnostStroje.fHmotnost*fg*(fbVypocetPolohyCOG.fB*(fX1-fbVypocetPolohyCOG.fA+(fX2-fX1)/2))/(fY*fX2);

// Patka 6:

fFopt[6] := fbHmotnostStroje.fHmotnost*fg*(fbVypocetPolohyCOG.fB*(fbVypocetPolohyCOG.fA+(fX2-fX1)/2))/(fY*fX2);
```

FB_Optimalizace_siloveho_zatizeni_rohovych_patek:

```
FUNCTION_BLOCK FB_Optimalizace_siloveho_zatizeni_rohovych_patek
VAR_INPUT

        fFa             : DUT_Sily;             // Definuje sily neboli reakce na ustavovacích patkách.
        fFopt           : DUT_Sily_optimalni;   // Definuje sily neboli reakce, které by měly zatěžovat ustavovací patky.

END_VAR

VAR_OUTPUT

        fFdeltaopt      : DUT_Sily_delta_opt;   // Definice diference mezi aktuální silou a její optimální velikostí patek 1,2,5,6.

END_VAR

// Funkčním blokem jsou vypočítávány diference mezi aktuální silou a její optimální velikostí.

// Diference mezi aktuální silou a její optimální velikostí patky 1.
fFdeltaopt[1] := fFa[1]+fFopt[1];

// Diference mezi aktuální silou a její optimální velikostí patky 2.
fFdeltaopt[2] := fFa[2]+fFopt[2];

// Diference mezi aktuální silou a její optimální velikostí patky 5.
fFdeltaopt[5] := fFa[5]+fFopt[5];
```

```
// Diference mezi aktuální silou a její optimální velikostí patky 6.
fFdeltaopt[6] := fFa[6]+fFopt[6];
```

FB_Kompletni_vyrovnani_stroje:

```
FUNCTION_BLOCK FB_Kompletni_vyrovnani_stroje

VAR_INPUT

// Definování DUT proměných:

        fFa                 : DUT_Sily;             // Pole zatěžujících sil [N].
        fLaktualni_vp2      : DUT_Laktualni;        // Aktuální hodnoty nodu 1 až 4.
        fLaktualni          : DUT_Laktualni;        // Aktuální hodnoty nodu 1 až 4.
        fPaktualni          : DUT_PolohyA;          // Pole aktuálních poloh ustavovacích patek [mm(um)].
        fLdelta             : DUT_Libely;           // Pole definující Hodnoty z Nodu 1. až 4.

// Definování FB:

        fbHmotnostStroje                                : FB_Hmotnost_stroje;       // Výpočet hmotnostistroje.
        fbVypocetPolohyCOG                              : FB_Poloha_COG;            // Výpočet polohy COG
        fbOptimalniSily                                 : FB_Optimalni_sily;        // Výpočet optimálních sil.
        fbSklonXY                                       : FB_SklonXY;               // Výpočet sklonu v osách X a Y.
        fbOptimalizaceSilovehoZatizeniRohovychPatek     : FB_Optimalizace_siloveho_zatizeni_rohovych_patek;     // Výpočet diference mezi aktuální silou a její
optimální velikostí.

// Proměné combobox:

        cbfVolbaPolohyAT    %Q*     : LREAL;        // Proměná přiřazující hodnotu 0 až 3 bloku combobox_4 pro zvolení polohy stolu v ose Y.
        sVolbaPolohy                : STRING;       // Má obsluhu upozornit na změnu polohy stolu.

// Vstupní proměná TOTAL STOP:

        bTotalStop          : BOOL  := GVL_Vstupy_a_vystupy.bgTotalStop;
```

```
// Proměné dále používané:

        fToleranceSklonuPuvodni                 : INT;
        nToleranceOptimalnihoZatizeniPatky      : INT;

END_VAR

VAR_OUTPUT

        fPzadana        : DUT_PolohyZ;          // Pole žádaných poloh ustavovacích patek.
        falfa           : DUT_alfa;             // Pole vypočtených sklonů v ose X.
        fbeta           : DUT_beta;             // Pole vypočtených sklonů v ose Y.
        fFopt           : DUT_Sily_optimalni;   // Pole vypočtených optimálních sil na rohové patky 1,2,5,6.

END_VAR

VAR

        loop2           : INT     := 1;         // Definice proměnné v příkazu CASE.
        nVolba          : INT;                  // Vybírá se mód.
        fbCasovac1      : TON;                  // Definování časovače 1.
        bStartTime1     : BOOL;                 // Proměnná, která spouští časovače 1.
        tTimeValue1     : TIME := T#0.5S;       // Definice času pro odpočet.
        bStopTime1      : BOOL;                 // Proměnná, která spouští další funkce po doběhnutí časovače 1.
        fbCacovac2      : TON;                  // Definování časovače 2.
        bStartTime2     : BOOL;                 // Proměnná, která spouští časovače 2.
        tTimeValue2     : TIME := T#30S;        // Definice času pro odpočet.
        bStopTime2      : BOOL;                 // Proměnná, která spouští další funkce po doběhnutí časovače 2.
        mcStop          : MC_Stop;              // Zastavení osy.
        bStopAxis1      : BOOL  := FALSE;       // Proměnná, která bude spouštět zastavení osy.
        bStopAxis2      : BOOL  := FALSE;       // Proměnná, která bude spouštět zastavení osy.
        bStopAxis3      : BOOL  := FALSE;       // Proměnná, která bude spouštět zastavení osy.
        bStopAxis4      : BOOL  := FALSE;       // Proměnná, která bude spouštět zastavení osy.
        bStopAxis5      : BOOL  := FALSE;       // Proměnná, která bude spouštět zastavení osy.
        bStopAxis6      : BOOL  := FALSE;       // Proměnná, která bude spouštět zastavení osy.
END_VAR
```

// *****Funkční blok, který provádí kompletní ustavování OS*****

// Seznam CASE:

// První část funkčního bloku:
// 1: Provedení odlehčení vnitřních patek.
// 2: Rovnání sklonu ve směru (ose) Y.
// 3: Rovnání sklonu ve směru (ose) Y kontrola.
// 4: Rovnání sklonu ve směru (ose) X.
// 5: Rovnání sklonu ve směru (ose) X kontrola.
// 6: Kontrola obou sklonů 1.
// 7: Optimalizace silového zatížení rohových patek.
// 8: Logika 1.
// 9: Logika 2.
// 10: Logika akce 1.
// 11: Logika akce 2.
// 12: Logika akce 3.
// 13: Logika akce 4.
// 14: Kontrola silového zatížení rohových patek.
// 15: Kontrola obou sklonů 2.

// Druhá část funkčního bloku:
// 16: Kontrola kontaktu všech patek vp2.
// 17: Změna polohy k obsluze.
// 18: Kontrola kontaktu všech patek vp1.
// 19: Změna polohy ke stojanu.
// 20: Kontrola kontaktu všech patek vp3.
// 21: Změna polohy výchozi.
// 22: Znovu kontrola kontaktu všech patek vp2.
// 23: Změna polohy k obsluze 1.
// 24: Znovu kontrola kontaktu všech patek vp1.
// 25: Změna polohy ke stojanu 1.
// 26: Znovu kontrola kontaktu všech patek vp3.
// 27: Čtení sklonu vp3.
// 28: Změna polohy k obsluze 2.
// 29: Čtení sklonu vp1.

```
//                          30: Změna polohy ke stojanu 2.
//                          31: Čtení sklonu vp1.
//                          32: Změna polohy ke stojanu 2.

// ***Definice časovačů***

// *Časovač 1*

// Slouží k přechodu mezi příslušnými CASE po 0.5s.

        fbCasovac1(IN:= bStartTime1, PT:= tTimeValue1, Q=> bStopTime1, ET=> );

// *Časovač 2*

// Po uplinutí tohoto časovače nastane ERROR. Pokud není splněna jiná podmínka.

        fbCacovac2(IN:= bStartTime2, PT:= tTimeValue2, Q=> bStopTime2, ET=> );

// ***Definice Stop motion pro patky 1 až 6***

        MC_Stop(                                    //Řízení osy 1.
        Axis:= GVL_Axis.axis1,          // Přiřazení, která osa se bude řídit. ,
        Execute:= bStopAxis1,
        Error=> );

        MC_Stop(                                    //Řízení osy 2.
        Axis:= GVL_Axis.axis2,          // Přiřazení, která osa se bude řídit. ,
        Execute:= bStopAxis2,
        Error=> );

        MC_Stop(                                    //Řízení osy 3.
        Axis:= GVL_Axis.axis3,          // Přiřazení, která osa se bude řídit. ,
        Execute:= bStopAxis3,
        Error=> );

        MC_Stop(                                    //Řízení osy 4.
```

```
        Axis:= GVL_Axis.axis4,          // Přiřazení, která osa se bude řídit. ,
        Execute:= bStopAxis4,
        Error=> );

        MC_Stop(                                //Řízení osy 5.
        Axis:= GVL_Axis.axis5,          // Přiřazení, která osa se bude řídit. ,
        Execute:= bStopAxis5,
        Error=> );

        MC_Stop(                                //Řízení osy 6.
        Axis:= GVL_Axis.axis6,          // Přiřazení, která osa se bude řídit. ,
        Execute:= bStopAxis6,
        Error=> );


//*****USTAVOVÁNÍ*****

// Funkční blok je rozdělen do dvou částí:

// *****Začátek CASE*****

CASE loop2 OF

// První část se zaobívá vyrovnáním stroje na rohové patky s dosazením jejich optimálního zatížení
// (Vyrovnání_stroje_na_rohové_patky_s_dosazením_jejich_optimálního_zatížení).

// První úkon který se provede v této části je odlehčení vnitřních patek a výpočet hmotnosti, polohy COG s optimálními reakcemi

        1:
// *****Provedení odlehčení vnitřních patek*****

                IF fFa[3]>0 OR fFa[4]>0 THEN // Požaduje se, aby síly F3 a F4 na ustavovacích patkách byly nulové.

                        IF fLaktualni[1] < fLaktualni[3] AND fLaktualni[2] < fLaktualni[4] THEN // Jestli je splněna tato posmínka hodnot z nodu 1 až 4 libel.
                                fPzadana[6] := fPaktualni[6] + Krok_zdvihu_patky; // Provede se pohyb ustavovací patkou 6.
```

```
            ELSIF fLaktualni[1] < fLaktualni[3] AND fLaktualni[2] > fLaktualni[4] THEN // Jestli je splněna tato posmínka hodnot z nodu 1 až 4 libel.
                fPzadana[5] := fPaktualni[5] + Krok_zdvihu_patky; // Provede se pohyb ustavovací patkou 5.

            ELSIF fLaktualni[1] > fLaktualni[3] AND fLaktualni[2] < fLaktualni[4] THEN // Jestli je splněna tato posmínka hodnot z nodu 1 až 4 libel.
                fPzadana[2] := fPaktualni[2] + Krok_zdvihu_patky; // Provede se pohyb ustavovací patkou 2.

            ELSIF fLaktualni[1] > fLaktualni[3] AND fLaktualni[2] > fLaktualni[4] THEN // Jestli je splněna tato posmínka hodnot z nodu 1 až 4 libel.
                fPzadana[1] := fPaktualni[1]  + Krok_zdvihu_patky; // Provede se pohyb ustavovací patkou 1.

            END_IF

// Pak se provede výpočet hmotnosti, polohy COG a výpočet optimálních sil.

        ELSE
            fbHmotnostStroje(fFa:= fFa, fHmotnost=> );      // Volá se funkční blok a provede se vypočtení hmotnosti.

            fbVypocetPolohyCOG(                                     // Volá se funkční blok a provede se vypočtení polohy COG.
                fFa:= fFa ,
                fX1:= GVL_Konstanty.fgVzdalenost_X1,
                fX2:= GVL_Konstanty.fgVzdalenost_X2,
                fY:= GVL_Konstanty.fgVzdalenost_Y,
                fA=> ,
                fB=> );

            fbOptimalniSily(                            // Volá se funkční blok a provede se vypočtení optimální síly.
                fbHmotnostStroje:= ,
                fbVypocetPolohyCOG:= ,
                fX1:= GVL_Konstanty.fgVzdalenost_X1,
                fX2:= GVL_Konstanty.fgVzdalenost_X2,
                fY:= GVL_Konstanty.fgVzdalenost_Y,
                fg:= GVL_Konstanty.fgTihoveZrychleni,
                fFopt=> fFopt);

            loop2 := 2;                         // Dá se příkaz na odbavení kódu v CASE 2.

        END_IF
```

```
// *****KONEC CASE 1****

        2:

// *****Rovnání sklonu ve směru (ose) Y*****

                IF ABS (GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni) < fToleranceSklonuPuvodni AND (GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni) < fToleranceSklonuPuvodni THEN
                        loop2 := 7;                             // Dá se příkaz na odbavení kódu v CASE 7.

                ELSE

                        IF fLaktualni_vp2[1] > fLaktualni_vp2[3] THEN
                                fPzadana[1] := fPaktualni[1] + Krok_zdvihu_patky;
                                fPzadana[2] := fPaktualni[2] + Krok_zdvihu_patky;

                        ELSE
                                fPzadana[5] := fPaktualni[5] + Krok_zdvihu_patky;
                                fPzadana[6] := fPaktualni[6] + Krok_zdvihu_patky;

                        END_IF
                END_IF

// Přechod do CASE 3 bude realizován po 0.5s.

                bStartTime1 := TRUE;            // Provede se zapnutí časovače 1.

                IF bStopTime1 THEN                      // Jestli že je hodnota bStopTime1 = TRUE, tak
                        bStartTime1     := FALSE;       // Provede se reset zapnutí časovače 1.
                        loop2 := 3;                             // Dá se příkaz na odbavení kódu v CASE 3.

                END_IF


// *****KONEC CASE 2****
```

```
        3:

// *****Rovnání sklonu ve směru (ose) Y kontrola*****

fbSklonXY(
        fSklon_v_ose_X_pocatecni:= GVL_Konstanty.fSklon_v_ose_X_pocatecni,
        fSklon_v_ose_Y_pocatecni:= GVL_Konstanty.fSklon_v_ose_Y_pocatecni,
        fLdelta:= fLdelta,
        nVzdalenostNodu:= GVL_Konstanty.ngVzdalenostNodu,
        nSklon:= GVL_Konstanty.ngSklon,
        nPrevod:= GVL_Konstanty.ngPrevod,
        fLaktualni=> fLaktualni,
        fSklon_v_ose_X_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni,
        fSklon_v_ose_Y_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni);

                bStartTime2 := TRUE;

                IF bStopTime2 THEN              // Slouží proto, aby se příkaz ukončil pokud nastana ERROR ("překročení počtu iterací")
                        bStartTime2 := FALSE;
                        loop2 := 33;                    // Dá se příkaz na odbavení kódu v CASE 33.

                        IF GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni > fToleranceSklonuPuvodni THEN
                        loop2 := 2;                     // Dá se příkaz na odbavení kódu v CASE 2.

                        ELSE
                                loop2 := 4;                     // Dá se příkaz na odbavení kódu v CASE 4.

                        END_IF
                END_IF

// *****KONEC CASE 3****

        4:

// *****Rovnání sklonu ve směru (ose) X*****
```

```
                IF ABS (GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni) < fToleranceSklonuPuvodni AND (GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni) < fToleranceSklonuPuvodni THEN
                        loop2 := 7;                             // Dá se příkaz na odbavení kódu v CASE 7.

                ELSE

                        IF fLaktualni_vp2[2] > fLaktualni_vp2[4] THEN
                                fPzadana[1] := fPaktualni[1] + Krok_zdvihu_patky;
                                fPzadana[5] := fPaktualni[5] + Krok_zdvihu_patky;

                        ELSE
                                fPzadana[2] := fPaktualni[2] + Krok_zdvihu_patky;
                                fPzadana[6] := fPaktualni[6] + Krok_zdvihu_patky;

                        END_IF
                END_IF

// Přechod do CASE 5 bude realizován po 0.5s.

                bStartTime1 := TRUE;            // Provede se zapnutí časovače 1.

                IF bStopTime1 THEN                      // Jestli že je hodnota bStopTime1 = TRUE, tak
                        bStartTime1     := FALSE;       // Provede se reset zapnutí časovače 1.
                        loop2 := 5;                             // Dá se příkaz na odbavení kódu v CASE 5.

                END_IF


// *****KONEC CASE 4****

        5:

// *****Rovnání sklonu ve směru (ose) X kontrola*****

fbSklonXY(
```

```
        fSklon_v_ose_X_pocatecni:= GVL_Konstanty.fSklon_v_ose_X_pocatecni,
        fSklon_v_ose_Y_pocatecni:= GVL_Konstanty.fSklon_v_ose_Y_pocatecni,
        fLdelta:= fLdelta,
        nVzdalenostNodu:= GVL_Konstanty.ngVzdalenostNodu,
        nSklon:= GVL_Konstanty.ngSklon,
        nPrevod:= GVL_Konstanty.ngPrevod,
        fLaktualni=> fLaktualni,
        fSklon_v_ose_X_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni,
        fSklon_v_ose_Y_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni);

                bStartTime2 := TRUE;

                IF bStopTime2 THEN              // Slouží proto, aby se příkaz ukončil pokud nastana ERROR ("překročení počtu iterací")
                        bStartTime2 := FALSE;
                        loop2 := 33;                    // Dá se příkaz na odbavení kódu v CASE 33.

                        IF GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni > fToleranceSklonuPuvodni THEN
                        loop2 := 4;                     // Dá se příkaz na odbavení kódu v CASE 4.

                        ELSE
                                loop2 := 6;                     // Dá se příkaz na odbavení kódu v CASE 6.

                        END_IF
                END_IF

// *****KONEC CASE 5****

        6:

// *****Kontrola obou sklonů 1*****

                bStartTime2 := TRUE;

                IF bStopTime2 THEN              // Slouží proto, aby se příkaz ukončil pokud nastana ERROR ("překročení počtu iterací")
                        bStartTime2 := FALSE;
                        loop2 := 33;                    // Dá se příkaz na odbavení kódu v CASE 33.
```

```
                    IF ABS(GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni) < fToleranceSklonuPuvodni AND ABS(GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni) < fToleranceSklonuPuvodni THEN
                        loop2 := 2;                 // Dá se příkaz na odbavení kódu v CASE 2.

                    ELSE
                    loop2 := 7;                 // Dá se příkaz na odbavení kódu v CASE 7.

                    END_IF
                END_IF

// *****KONEC CASE 6****

        7:

// *****Optimalizace silového zatížení rohových patek*****

fbOptimalizaceSilovehoZatizeniRohovychPatek(
        fFa:= ,
        fFopt:= ,
        fFdeltaopt=> );

fbSklonXY(
        fSklon_v_ose_X_pocatecni:= GVL_Konstanty.fSklon_v_ose_X_pocatecni,
        fSklon_v_ose_Y_pocatecni:= GVL_Konstanty.fSklon_v_ose_Y_pocatecni,
        fLdelta:= fLdelta,
        nVzdalenostNodu:= GVL_Konstanty.ngVzdalenostNodu,
        nSklon:= GVL_Konstanty.ngSklon,
        nPrevod:= GVL_Konstanty.ngPrevod,
        fLaktualni=> fLaktualni,
        fSklon_v_ose_X_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni,
        fSklon_v_ose_Y_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni);

            IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[1] < fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[2] THEN
                loop2 := 8;                 // Dá se příkaz na odbavení kódu v CASE 8.
```

```
            ELSE
                loop2 := 9;                 // Dá se příkaz na odbavení kódu v CASE 9.

            END_IF

// *****KONEC CASE 7****

        8:

// *****Logika 1*****

            IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[1] < fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[5] THEN
                loop2 := 10;                // Dá se příkaz na odbavení kódu v CASE 10.

            ELSE
                loop2 := 11;                // Dá se příkaz na odbavení kódu v CASE 11.

            END_IF

// *****KONEC CASE 8****

        9:

// *****Logika 2*****

            IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[2] < fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[6] THEN
                loop2 := 12;                // Dá se příkaz na odbavení kódu v CASE 12.

            ELSE
                loop2 := 13;                // Dá se příkaz na odbavení kódu v CASE 13.

            END_IF

// *****KONEC CASE 9****

        10:
```

```
// *****Logika akce 1*****

            IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[1] < fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[6] THEN
                fPzadana[1] := fPaktualni[1]  + Krok_zdvihu_patky *Zjemneni_kroku_1;

            ELSE
                fPzadana[6] := fPaktualni[6]  + Krok_zdvihu_patky * Zjemneni_kroku_1;

            END_IF

// Přechod do CASE 11 bude realizován po 0.5s.

            bStartTime1 := TRUE;            // Provede se zapnutí časovače 1.

            IF bStopTime1 THEN                  // Jestli že je hodnota bStopTime1 = TRUE, tak
                bStartTime1     := FALSE;       // Provede se reset zapnutí časovače 1.
                loop2 := 11;                    // Dá se příkaz na odbavení kódu v CASE 11.

            END_IF


// *****KONEC CASE 10****

        11:

// *****Logika akce 2*****

            IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[5] < fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[6] THEN
                fPzadana[5] := fPaktualni[5]  + Krok_zdvihu_patky *Zjemneni_kroku_1;

            ELSE
                fPzadana[6] := fPaktualni[6]  + Krok_zdvihu_patky * Zjemneni_kroku_1;

            END_IF
```

```
// Přechod do CASE 12 bude realizován po 0.5s.

                bStartTime1 := TRUE;            // Provede se zapnutí časovače 1.

                IF bStopTime1 THEN                      // Jestli že je hodnota bStopTime1 = TRUE, tak
                        bStartTime1     := FALSE;       // Provede se reset zapnutí časovače 1.
                        loop2 := 12;                    // Dá se příkaz na odbavení kódu v CASE 12.

                END_IF


// *****KONEC CASE 11****

        12:

// *****Logika akce 3*****

                IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[2] < fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[5] THEN
                        fPzadana[2] := fPaktualni[2]  + Krok_zdvihu_patky *Zjemneni_kroku_1;

                ELSE
                        fPzadana[5] := fPaktualni[5]  + Krok_zdvihu_patky * Zjemneni_kroku_1;

                END_IF

// Přechod do CASE 13 bude realizován po 0.5s.

                bStartTime1 := TRUE;            // Provede se zapnutí časovače 1.

                IF bStopTime1 THEN                      // Jestli že je hodnota bStopTime1 = TRUE, tak
                        bStartTime1     := FALSE;       // Provede se reset zapnutí časovače 1.
                        loop2 := 13;                    // Dá se příkaz na odbavení kódu v CASE 13.

                END_IF
```

```
// *****KONEC CASE 12****

        13:

// *****Logika akce 4*****

                IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[6] < fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[5] THEN
                        fPzadana[6] := fPaktualni[6]  + Krok_zdvihu_patky *Zjemneni_kroku_1;

                ELSE
                        fPzadana[5] := fPaktualni[5]  + Krok_zdvihu_patky * Zjemneni_kroku_1;

                END_IF

// Přechod do CASE 14 bude realizován po 0.5s.

                bStartTime1 := TRUE;            // Provede se zapnutí časovače 1.

                IF bStopTime1 THEN                      // Jestli že je hodnota bStopTime1 = TRUE, tak
                        bStartTime1     := FALSE;       // Provede se reset zapnutí časovače 1.
                        loop2 := 14;                    // Dá se příkaz na odbavení kódu v CASE 14.

                END_IF


// *****KONEC CASE 13****

        14:

// *****Kontrola silového zatížení rohových patek*****

                fbOptimalizaceSilovehoZatizeniRohovychPatek(
                        fFa:= ,
                        fFopt:= ,
                        fFdeltaopt=> );
```

```
                IF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[1] > nToleranceOptimalnihoZatizeniPatky THEN
                        loop2 := 7;                     // Dá se příkaz na odbavení kódu v CASE 7.

                ELSIF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[2] > nToleranceOptimalnihoZatizeniPatky THEN
                        loop2 := 7;                     // Dá se příkaz na odbavení kódu v CASE 7.

                ELSIF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[5] > nToleranceOptimalnihoZatizeniPatky THEN
                        loop2 := 7;                     // Dá se příkaz na odbavení kódu v CASE 7.

                ELSIF fbOptimalizaceSilovehoZatizeniRohovychPatek.fFdeltaopt[6] > nToleranceOptimalnihoZatizeniPatky THEN
                        loop2 := 7;                     // Dá se příkaz na odbavení kódu v CASE 7.

                ELSE
                        loop2 := 15;            // Dá se příkaz na odbavení kódu v CASE 15.

                END_IF

// *****KONEC CASE 14****

        15:

// *****Kontrola obou sklonů 2*****

                // Reset4 := Reset4 + 1;
                bStartTime2 := TRUE;

                IF bStopTime2 THEN              // Slouží proto, aby se příkaz ukončil pokud nastana ERROR ("překročení počtu iterací")
                        bStartTime2 := FALSE;
                        loop2 := 33;                    // Dá se příkaz na odbavení kódu v CASE 33.

                        IF  ABS(GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni)  <  fToleranceSklonuPuvodni  OR  ABS(GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni)  <
fToleranceSklonuPuvodni THEN
                                loop2 := 2;                     // Dá se příkaz na odbavení kódu v CASE 2.

                        ELSE
```

```
                        IF nVolba = 8 THEN
                                loop2 := 33; // => nahradit, že musí program skončit

                        ELSE
                                loop2 := 16;                    // Dá se příkaz na odbavení kódu v CASE 16.

                        END_IF
                END_IF
        END_IF

// *****KONEC CASE 15****

// Druhá část se zaobívá vyrovnáním stroje s aktivací zbylích ustavovacích patek a sledování kontaktu napříč všemi diskrétními polohami (polohováním se stolem v ose Y).
// (Aktivace_vnitrnich_patek_a_kontrola_kontaktu_vsech_patek_napric_vsemi_diskretnimi_polohami).

        16:

// *****Kontrola kontaktu všech patek vp2*****

        IF fFa[1] = 0 OR fFa[2] = 0 OR fFa[3] = 0 OR fFa[4] = 0 OR fFa[5] = 0 OR fFa[6] = 0 THEN

                IF fFa[1] = 0 THEN
                        fPzadana[1] := fPaktualni[1] + Krok_zdvihu_patky;

                END_IF

                IF fFa[2] = 0 THEN
                        fPzadana[2] := fPaktualni[2] + Krok_zdvihu_patky;

                END_IF

                IF fFa[3] = 0 THEN
                        fPzadana[3] := fPaktualni[3] + Krok_zdvihu_patky;

                END_IF
```

```
                IF fFa[4] = 0 THEN
                        fPzadana[4] := fPaktualni[4] + Krok_zdvihu_patky;

                END_IF

                IF fFa[5] = 0 THEN
                        fPzadana[5] := fPaktualni[5] + Krok_zdvihu_patky;

                END_IF

                IF fFa[6] = 0 THEN
                        fPzadana[6] := fPaktualni[6] + Krok_zdvihu_patky;

                END_IF

        ELSE
                loop2 := 17;            // Dá se příkaz na odbavení kódu v CASE 17.

        END_IF

// *****KONEC CASE 16****

        17:

// *****Změna polohy k obsluze*****

        sVolbaPolohy := 'ZADEJTE VOLBU POLOHY 1';    // Upozornění pro obsluhu.

        IF cbfVolbaPolohy = 1 THEN      // Když bude navolena hodnota 1, tak
                loop2 := 18;                    // Dá se příkaz na odbavení kódu v CASE 18.

        ELSE
                loop2 := 17;                    // Dá se příkaz na odbavení kódu v CASE 17.

        END_IF
```

```
// *****KONEC CASE 17****

        18:

// *****Kontrola kontaktu všech patek vp1*****

                IF fFa[1] = 0 OR fFa[2] = 0 OR fFa[3] = 0 OR fFa[4] = 0 OR fFa[5] = 0 OR fFa[6] = 0 THEN

                        IF fFa[1] = 0 THEN
                                fPzadana[1] := fPaktualni[1] + Krok_zdvihu_patky;

                        END_IF

                        IF fFa[2] = 0 THEN
                                fPzadana[2] := fPaktualni[2] + Krok_zdvihu_patky;

                        END_IF

                        IF fFa[3] = 0 THEN
                                fPzadana[3] := fPaktualni[3] + Krok_zdvihu_patky;

                        END_IF

                        IF fFa[4] = 0 THEN
                                fPzadana[4] := fPaktualni[4] + Krok_zdvihu_patky;

                        END_IF

                        IF fFa[5] = 0 THEN
                                fPzadana[5] := fPaktualni[5] + Krok_zdvihu_patky;

                        END_IF

                        IF fFa[6] = 0 THEN
                                fPzadana[6] := fPaktualni[6] + Krok_zdvihu_patky;
```

```
                END_IF

            ELSE
                loop2 := 19;            // Dá se příkaz na odbavení kódu v CASE 19.

            END_IF

// *****KONEC CASE 18****

        19:

// *****Změna polohy ke stojanu*****

            sVolbaPolohy := 'ZADEJTE VOLBU POLOHY 3';    // Upozornění pro obsluhu.

            IF cbfVolbaPolohy = 3 THEN      // Když bude navolena hodnota 3, tak
                loop2 := 20;                    // Dá se příkaz na odbavení kódu v CASE 20.

            ELSE
                loop2 := 19;                    // Dá se příkaz na odbavení kódu v CASE 19.

            END_IF

// *****KONEC CASE 19****

            20:

// *****Kontrola kontaktu všech patek vp3*****

            IF fFa[1] = 0 OR fFa[2] = 0 OR fFa[3] = 0 OR fFa[4] = 0 OR fFa[5] = 0 OR fFa[6] = 0 THEN

                IF fFa[1] = 0 THEN
                    fPzadana[1] := fPaktualni[1] + Krok_zdvihu_patky;

                END_IF
```

```
                IF fFa[2] = 0 THEN
                        fPzadana[2] := fPaktualni[2] + Krok_zdvihu_patky;

                END_IF

                IF fFa[3] = 0 THEN
                        fPzadana[3] := fPaktualni[3] + Krok_zdvihu_patky;

                END_IF

                IF fFa[4] = 0 THEN
                        fPzadana[4] := fPaktualni[4] + Krok_zdvihu_patky;

                END_IF

                IF fFa[5] = 0 THEN
                        fPzadana[5] := fPaktualni[5] + Krok_zdvihu_patky;

                END_IF

                IF fFa[6] = 0 THEN
                        fPzadana[6] := fPaktualni[6] + Krok_zdvihu_patky;

                END_IF

        ELSE
                loop2 := 21;                    // Dá se příkaz na odbavení kódu v CASE 21.

        END_IF

// *****KONEC CASE 20****

        21:

// *****Změna polohy výchozi*****
```

```
                sVolbaPolohy := 'ZADEJTE VOLBU POLOHY 2';    // Upozornění pro obsluhu.

                IF cbfVolbaPolohy = 2 THEN      // Když bude navolena hodnota 2, tak
                        loop2 := 22;                    // Dá se příkaz na odbavení kódu v CASE 22.

                ELSE
                        loop2 := 21;                    // Dá se příkaz na odbavení kódu v CASE 21.

                END_IF

// *****KONEC CASE 21****

        22:

// *****Znovu kontrola kontaktu všech patek vp2*****

                bStartTime2 := TRUE;

                IF bStopTime2 THEN              // Slouží proto, aby se příkaz ukončil pokud nastana ERROR ("překročení počtu iterací")
                        bStartTime2 := FALSE;
                        loop2 := 33;                    // Dá se příkaz na odbavení kódu v CASE 33.

                        IF fFa[1] = 0 OR fFa[2] = 0 OR fFa[3] = 0 OR fFa[4] = 0 OR fFa[5] = 0 OR fFa[6] = 0 THEN
                                loop2 := 16;            // Dá se příkaz na odbavení kódu v CASE 16.

                        ELSE
                                loop2 := 23;            // Dá se příkaz na odbavení kódu v CASE 23.

                        END_IF
                END_IF

// *****KONEC CASE 22****

        23:

// *****Změna polohy k obsluze 1*****
```

```
                sVolbaPolohy := 'ZADEJTE VOLBU POLOHY 1';    // Upozornění pro obsluhu.

                IF cbfVolbaPolohy = 1 THEN      // Když bude navolena hodnota 1, tak
                        loop2 := 24;                    // Dá se příkaz na odbavení kódu v CASE 24.

                ELSE
                        loop2 := 23;                    // Dá se příkaz na odbavení kódu v CASE 23.

                END_IF

// *****KONEC CASE 23****

        24:

// *****Znovu kontrola kontaktu všech patek vp1*****

                IF fFa[1] = 0 OR fFa[2] = 0 OR fFa[3] = 0 OR fFa[4] = 0 OR fFa[5] = 0 OR fFa[6] = 0 THEN
                        loop2 := 18;            // Dá se příkaz na odbavení kódu v CASE 18.

                ELSE
                        loop2 := 25;            // Dá se příkaz na odbavení kódu v CASE 25.

                END_IF

// *****KONEC CASE 24****

        25:

// *****Změna polohy ke stojanu 1*****

                sVolbaPolohy := 'ZADEJTE VOLBU POLOHY 3';    // Upozornění pro obsluhu.

                IF cbfVolbaPolohy = 3 THEN      // Když bude navolena hodnota 3, tak
                        loop2 := 26;                    // Dá se příkaz na odbavení kódu v CASE 26.
```

```
                ELSE
                        loop2 := 25;                    // Dá se příkaz na odbavení kódu v CASE 25.

                END_IF

// *****KONEC CASE 25****

        26:

// *****Znovu kontrola kontaktu všech patek vp3*****

                IF fFa[1] = 0 OR fFa[2] = 0 OR fFa[3] = 0 OR fFa[4] = 0 OR fFa[5] = 0 OR fFa[6] = 0 THEN
                        loop2 := 20;            // Dá se příkaz na odbavení kódu v CASE 20.

                ELSE
                        loop2 := 27;            // Dá se příkaz na odbavení kódu v CASE 27.

                END_IF

// *****KONEC CASE 26****

        27:

// *****Čtení sklonu vp3*****

fbSklonXY(
        fSklon_v_ose_X_pocatecni:= GVL_Konstanty.fSklon_v_ose_X_pocatecni,
        fSklon_v_ose_Y_pocatecni:= GVL_Konstanty.fSklon_v_ose_Y_pocatecni,
        fLdelta:= fLdelta,
        nVzdalenostNodu:= GVL_Konstanty.ngVzdalenostNodu,
        nSklon:= GVL_Konstanty.ngSklon,
        nPrevod:= GVL_Konstanty.ngPrevod,
        fLaktualni=> fLaktualni,
        fSklon_v_ose_X_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni_vp3,
        fSklon_v_ose_Y_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni_vp3);
```

```
                loop2 := 28;                // Dá se příkaz na odbavení kódu v CASE 28.

// *****KONEC CASE 27****

        28:

// *****Změna polohy k obsluze 2*****

                sVolbaPolohy := 'ZADEJTE VOLBU POLOHY 1';    // Upozornění pro obsluhu.

                IF cbfVolbaPolohy = 1 THEN      // Když bude navolena hodnota 1, tak
                        loop2 := 29;                        // Dá se příkaz na odbavení kódu v CASE 29.

                ELSE
                        loop2 := 28;                        // Dá se příkaz na odbavení kódu v CASE 28.

                END_IF

// *****KONEC CASE 28****

        29:

// *****Čtení sklonu vp1*****

fbSklonXY(
        fSklon_v_ose_X_pocatecni:= GVL_Konstanty.fSklon_v_ose_X_pocatecni,
        fSklon_v_ose_Y_pocatecni:= GVL_Konstanty.fSklon_v_ose_Y_pocatecni,
        fLdelta:= fLdelta,
        nVzdalenostNodu:= GVL_Konstanty.ngVzdalenostNodu,
        nSklon:= GVL_Konstanty.ngSklon,
        nPrevod:= GVL_Konstanty.ngPrevod,
        fLaktualni=> fLaktualni,
        fSklon_v_ose_X_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni_vp1,
        fSklon_v_ose_Y_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni_vp1);

                loop2 := 30;                // Dá se příkaz na odbavení kódu v CASE 30.
```

```
// *****KONEC CASE 29****

        30:

// *****Změna polohy k obsluze 2*****

                sVolbaPolohy := 'ZADEJTE VOLBU POLOHY 2';    // Upozornění pro obsluhu.

                IF cbfVolbaPolohy = 2 THEN      // Když bude navolena hodnota 2, tak
                        loop2 := 31;                    // Dá se příkaz na odbavení kódu v CASE 31.

                ELSE
                        loop2 := 30;                    // Dá se příkaz na odbavení kódu v CASE 30.

                END_IF

// *****KONEC CASE 30****

        31:

// *****Čtení sklonu vp2*****

fbSklonXY(
        fSklon_v_ose_X_pocatecni:= GVL_Konstanty.fSklon_v_ose_X_pocatecni,
        fSklon_v_ose_Y_pocatecni:= GVL_Konstanty.fSklon_v_ose_Y_pocatecni,
        fLdelta:= fLdelta,
        nVzdalenostNodu:= GVL_Konstanty.ngVzdalenostNodu,
        nSklon:= GVL_Konstanty.ngSklon,
        nPrevod:= GVL_Konstanty.ngPrevod,
        fLaktualni=> fLaktualni,
        fSklon_v_ose_X_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni_vp2,
        fSklon_v_ose_Y_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni_vp2);

                loop2 := 32;            // Dá se příkaz na odbavení kódu v CASE 32.
```

```
// *****KONEC CASE 31****

        32:

// *****Čtení sklonu*****

                falfa[1] := GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni_vp1;
                fbeta[1] := GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni_vp1;
                falfa[2] := GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni_vp2;
                fbeta[2] := GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni_vp2;
                falfa[3] := GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni_vp3;
                fbeta[3] := GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni_vp3;

// *****KONEC CASE 32****

        33:

// *****Zastavení programu popřípadě ERROR*****

        bStopAxis1      := TRUE;        // Zastavení osy.
        bStopAxis2      := TRUE;        // Zastavení osy.
        bStopAxis3      := TRUE;        // Zastavení osy.
        bStopAxis4      := TRUE;        // Zastavení osy.
        bStopAxis5      := TRUE;        // Zastavení osy.
        bStopAxis6      := TRUE;        // Zastavení osy.

// *****KONEC CASE 33****

        34:

// *****TOTAL STOP*****

        bTotalStop := TRUE;             // Sepnutí tlačítka

// *****KONEC CASE 34****
END_CASE
```

```
// *****KONEC CASE CELKOVĚ*****
```

Hlavní program (PRG):

MAIN_Hlavni_program:

```
PROGRAM MAIN_Hlavni_program

// Programátor: Aleš Jiránek

VAR

// Definice proměných combobox prvků pro HMI panel:

    cbVyberDruh         : INT;    // Proměnná přiřazující hodnotu 0 až 2 bloku combobox pro zobrazení combobox_1 nebo combobox_2 nebo skrytí obou.
    cbVyberTypS         : INT;    // Proměnná přiřazující hodnotu 0 až 6 bloku combobox_1 pro zobrazení ručního nebo automatického režimu.
    cbVyberTypF         : INT;    // Proměnná přiřazující hodnotu 0 až 8 bloku combobox_2 pro zobrazení ručního nebo automatického režimu.
    cbnVolbaUkonu       : INT;    // Proměnná přiřazující hodnotu 0 až 9 bloku combobox_3 pro zobrazení volby úkonu, které požaduje obsluha.

// Definování vstupních a výstupních proměných:

    fPzadana            : DUT_PolohyZ;          // Přiřazení DUT_PolohyZ k proměnné fLdelta, kde se budou přiřazovat globální proměnné.
    fPaktualni          : DUT_PolohyA;          // Přiřazení DUT_PolohyA k proměnné fLdelta, kde se budou přiřazovat globální proměnné.
    fFa                 : DUT_Sily;             // Přiřazení DUT_Sily k proměnné fLdelta, kde se budou přiřazovat globální proměnné.
    fOdlehnutiPatky     : DUT_OdlehnutiPatek;   // Přiřazení DUT_OdlehnutiPatek k proměnné fLdelta, kde se budou přiřazovat globální proměnné.
    fLdelta             : DUT_Libely;           // Přiřazení DUT_Libely k proměnné fLdelta, kde se budou přiřazovat globální proměnné.
    fLaktualni          : DUT_Laktualni;        // Přiřazení DUT_Laktualni k proměnné fLaktualni.
    falfa               : DUT_alfa;             // Pole vypočtených sklonů v ose X.
    fbeta               : DUT_beta;             // Pole vypočtených sklonů v ose Y.

// Definice funkčních bloků:

    fbSklonXY2          : FB_SklonXY_vp2;       // Funkční blok definující výpočet sklonu polohy 2.
    fbSklonXY           : FB_SklonXY;           // Funkční blok definující výpočet sklonu.
```

```
        fbKompletniVyrovnaniStroje      : FB_Kompletni_vyrovnani_stroje;    // Funkční blok definující kompletní vyrovnání OS.
        fbPatka1                        : FB_Patka1;    // Funkční blok definující ruční řízení Axis1.
        fbPatka2                        : FB_Patka2;    // Funkční blok definující ruční řízení Axis2.
        fbPatka3                        : FB_Patka3;    // Funkční blok definující ruční řízení Axis3.
        fbPatka4                        : FB_Patka4;    // Funkční blok definující ruční řízení Axis4.
        fbPatka5                        : FB_Patka5;    // Funkční blok definující ruční řízení Axis5.
        fbPatka6                        : FB_Patka6;    // Funkční blok definující ruční řízení Axis6.

// Konec definice funkčních bloků.

// Proměné:

        loop1                   : INT  := 10;           // Definice proměnné v příkazu CASE.

END_VAR

// Název práce:
// Program pro automatické ustavení OS na základ
// Programem jsou řízeny chytré ustavovací patky

// Autor: Aleš Jiránek
// Programátor: Aleš Jiránek

// Datum změny programu (práce na programu):
// Datum: 1) 19.03.2023; 2)

// Kontrola vedoucím (Ing. Lukáš Novotný Ph.D):
// Datum: 1)

// *****Přiřazení globálních proměných, proměným v programu*****

// fPzadana:

GVL_Vstupy_a_vystupy.fP1zadana      := fPzadana[1]; // Přiřazení fPzadana globální proměnou.
GVL_Vstupy_a_vystupy.fP2zadana      := fPzadana[2]; // Přiřazení fPzadana globální proměnou.
GVL_Vstupy_a_vystupy.fP3zadana      := fPzadana[3]; // Přiřazení fPzadana globální proměnou.
```

```
GVL_Vstupy_a_vystupy.fP4zadana      := fPzadana[4]; // Přiřazení fPzadana globální proměnou.
GVL_Vstupy_a_vystupy.fP5zadana      := fPzadana[5]; // Přiřazení fPzadana globální proměnou.
GVL_Vstupy_a_vystupy.fP6zadana      := fPzadana[6]; // Přiřazení fPzadana globální proměnou.

// fPaktualni:

fPaktualni[1]    := GVL_Vstupy_a_vystupy.fP1aktualni; // Přiřazení fPaktualni globální proměnou.
fPaktualni[2]    := GVL_Vstupy_a_vystupy.fP2aktualni; // Přiřazení fPaktualni globální proměnou.
fPaktualni[3]    := GVL_Vstupy_a_vystupy.fP3aktualni; // Přiřazení fPaktualni globální proměnou.
fPaktualni[4]    := GVL_Vstupy_a_vystupy.fP4aktualni; // Přiřazení fPaktualni globální proměnou.
fPaktualni[5]    := GVL_Vstupy_a_vystupy.fP5aktualni; // Přiřazení fPaktualni globální proměnou.
fPaktualni[6]    := GVL_Vstupy_a_vystupy.fP6aktualni; // Přiřazení fPaktualniglobální proměnou.

// fFa:

fFa[1]                  := GVL_Vstupy_a_vystupy.fF1; // Signál síly, který dává čidlo na chytré patce 1.
fFa[2]                  := GVL_Vstupy_a_vystupy.fF1; // Signál síly, který dává čidlo na chytré patce 2.
fFa[3]                  := GVL_Vstupy_a_vystupy.fF1; // Signál síly, který dává čidlo na chytré patce 3.
fFa[4]                  := GVL_Vstupy_a_vystupy.fF1; // Signál síly, který dává čidlo na chytré patce 4.
fFa[5]                  := GVL_Vstupy_a_vystupy.fF1; // Signál síly, který dává čidlo na chytré patce 5.
fFa[6]                  := GVL_Vstupy_a_vystupy.fF1; // Signál síly, který dává čidlo na chytré patce 6.

// fOdlehnutiPatky:

fOdlehnutiPatky[1]      := GVL_Vstupy_a_vystupy.fOdlehnutiPatky1; // Signál indikující odlehnutí  patky 1 od lože.
fOdlehnutiPatky[2]      := GVL_Vstupy_a_vystupy.fOdlehnutiPatky2; // Signál indikující odlehnutí  patky 2 od lože.
fOdlehnutiPatky[3]      := GVL_Vstupy_a_vystupy.fOdlehnutiPatky3; // Signál indikující odlehnutí  patky 3 od lože.
fOdlehnutiPatky[4]      := GVL_Vstupy_a_vystupy.fOdlehnutiPatky4; // Signál indikující odlehnutí  patky 4 od lože.
fOdlehnutiPatky[5]      := GVL_Vstupy_a_vystupy.fOdlehnutiPatky5; // Signál indikující odlehnutí  patky 5 od lože.
fOdlehnutiPatky[6]      := GVL_Vstupy_a_vystupy.fOdlehnutiPatky6; // Signál indikující odlehnutí  patky 6 od lože.

// fLdelta:

fLdelta[1]      := GVL_Vstupy_a_vystupy.fL1; // Přiřazení fLdelta globální proměnou (signál, který dává nod 1)
fLdelta[2]      := GVL_Vstupy_a_vystupy.fL2; // Přiřazení fLdelta globální proměnou (signál, který dává nod 2)
fLdelta[3]      := GVL_Vstupy_a_vystupy.fL3; // Přiřazení fLdelta globální proměnou (signál, který dává nod 3)
```

```
fLdelta[4]        := GVL_Vstupy_a_vystupy.fL4; // Přiřazení fLdelta globální proměnou (signál, který dává nod 4)

// falfa:

falfa[1] := GVL_Vstupy_a_vystupy.falfa1;          // Hodnota vypočteného sklonu v ose X.
falfa[2] := GVL_Vstupy_a_vystupy.falfa2;          // Hodnota vypočteného sklonu v ose X.
falfa[3] := GVL_Vstupy_a_vystupy.falfa3;          // Hodnota vypočteného sklonu v ose X.

// fbeta:

fbeta[1]        := GVL_Vstupy_a_vystupy.fbeta1;          // Hodnota vypočteného sklonu v ose Y.
fbeta[2]        := GVL_Vstupy_a_vystupy.fbeta2;          // Hodnota vypočteného sklonu v ose Y.
fbeta[3]        := GVL_Vstupy_a_vystupy.fbeta3;          // Hodnota vypočteného sklonu v ose Y.


// *****Konec přiřazení globálních proměných, proměným v programu*****

// ********** PROGRAM PRO RUČNÍ A AUTOMATICKÉ USTAVOVÁNÍ VOLNĚ LOŽENÝCH OS NA USTAVOVACÍCH PATKÁCH**********

// *****Hlavní progam servisního(ručního) ustavování OS*****

// ***Volání funkčních bloků, kde jsou definovány jednotlivé ustavovací patky***

// **Ustavovací patka 1:**

fbPatka1(
        bEnable1:= ,
        bStartV1:= ,
        bStartZ1:= ,
        bAxisVysunuti1:= ,
        bAxisZasunuti1:= ,
        lrOverride1:= ,
        lrVelo:= ,
        lrHodnotaVysunuti1=> ,
        lrNulovani1=> );
```

```
// **Ustavovací patka 2:**

fbPatka2(
        bEnable2:= ,
        bStartV2:= ,
        bStartZ2:= ,
        bAxisVysunuti2:= ,
        bAxisZasunuti2:= ,
        lrOverride2:= ,
        lrVelo:= ,
        lrHodnotaVysunuti2=> ,
        lrNulovani2=> );

// **Ustavovací patka 3:**

fbPatka3(
        bEnable3:= ,
        bStartV3:= ,
        bStartZ3:= ,
        bAxisVysunuti3:= ,
        bAxisZasunuti3:= ,
        lrOverride3:= ,
        lrVelo:= ,
        lrHodnotaVysunuti3=> ,
        lrNulovani3=> );

// **Ustavovací patka 4:**

fbPatka4(
        bEnable4:= ,
        bStartV4:= ,
        bStartZ4:= ,
        bAxisVysunuti4:= ,
        bAxisZasunuti4:= ,
        lrOverride4:= ,
        lrVelo:= ,
```

```
        lrHodnotaVysunuti4=> ,
        lrNulovani4=> );

// **Ustavovací patka 5:**

fbPatka5(
        bEnable5:= ,
        bStartV5:= ,
        bStartZ5:= ,
        bAxisVysunuti5:= ,
        bAxisZasunuti5:= ,
        lrOverride5:= ,
        lrVelo:= ,
        lrHodnotaVysunuti5=> ,
        lrNulovani5=> );

// **Ustavovací patka 6:**

fbPatka6(
        bEnable6:= ,
        bStartV6:= ,
        bStartZ6:= ,
        bAxisVysunuti6:= ,
        bAxisZasunuti6:= ,
        lrOverride6:= ,
        lrVelo:= ,
        lrHodnotaVysunuti6=> ,
        lrNulovani6=> );

// *****Konec servisního (ručního) programu*****

// ------------------------------------------------------------------------------------------

// *****Hlavní progam automatického ustavování OS*****

// ***Volání funkčního bloku, kde jsou vypočítány sklony v osách X a Y (naklopení OS)***
```

```
fbSklonXY(
	fSklon_v_ose_X_pocatecni:= GVL_Konstanty.fSklon_v_ose_X_pocatecni,
	fSklon_v_ose_Y_pocatecni:= GVL_Konstanty.fSklon_v_ose_Y_pocatecni,
	fLdelta:= fLdelta,
	nVzdalenostNodu:= GVL_Konstanty.ngVzdalenostNodu,
	nSklon:= GVL_Konstanty.ngSklon,
	nPrevod:= GVL_Konstanty.ngPrevod,
	fLaktualni=> fLaktualni,
	fSklon_v_ose_X_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_X_aktualni,
	fSklon_v_ose_Y_aktualni=> GVL_Sklon_aktualni.fgSklon_v_ose_Y_aktualni);

// *****Začátek CASE*****

CASE loop1 OF

	10:

		IF (bStartAuto = TRUE) AND NOT (bStopAuto = TRUE) THEN	// Jestliže bude bStartAuto = TRUE a nebude bStopAuto = TRUE tak proveď.
			loop1 := 20;				// Dá se příkaz na odbavení kódu v CASE 20.

		END_IF

// *****KONEC CASE 10****

	20:

// Volání funkčního bloku, kde se provádí kompletní vyrovnání stroje.

fbKompletniVyrovnaniStroje(
	fFa:= fFa,
	fLaktualni_vp2:= ,
	fLaktualni:= fLaktualni,
	fPaktualni:= fPaktualni,
	fLdelta:= fLdelta,
	fbHmotnostStroje:= ,
```

```
        fbVypocetPolohyCOG:= ,
        fbOptimalniSily:= ,
        fbSklonXY:= ,
        fbOptimalizaceSilovehoZatizeniRohovychPatek:= ,
        cbfVolbaPolohy:= ,
        sVolbaPolohy:= ,
        fToleranceSklonuPuvodni:= ,
        nToleranceOptimalnihoZatizeniPatky:= ,
        fPzadana=> ,
        falfa=> ,
        fbeta=> ,
        fFopt=> );

// *****KONEC CASE 20****

END_CASE

// *****KONEC CASE CELKOVĚ*****

// *****Konec Hlavního programu*****
```