ZÁPADOČESKÁ UNIVERZITA V PLZNI

**FAKULTA STROJNÍ**

Studijní program: B 2341 Strojírenství

Studijní zaměření: Programování NC strojů

# BAKALÁŘSKÁ PRÁCE

Podprogram pro vytvoření zkosení s využitím pro čtyřosé frézovací centrum.

Autor: **Zdeněk Jaroš**

Vedoucí práce: **Ing. Jiří Vyšata, Ph.D.**

Akademický rok 2013/2014

## Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů, uvedených v seznamu, který je součástí této bakalářské práce.

V Plzni dne: …….  . . . . . .

podpis autora

## Poděkování

V prvé řadě bych chtěl poděkovat vedoucímu mé práce Ing. Jiřímu Vyšatovi, Ph.D. za odborné rady, cenné připomínky a ochotu poradit při psaní této bakalářské práce. Dále bych chtěl poděkovat své rodině za podporu při studiu na VŠ. V neposlední řadě své přítelkyni za snášení hlasitého bědování při grafické a slohové úpravě této práce.

# ANOTAČNÍ LIST BAKALÁŘSKÉ PRÁCE

| | | | |
|---|---|---|---|
| **AUTOR** | **Příjmení**<br>Jaroš | **Jméno**<br>Zdeněk | |
| **STUDIJNÍ OBOR** | B 2341 „Programování NC strojů“ | | |
| **VEDOUCÍ PRÁCE** | **Příjmení (včetně titulů)**<br>Ing. Vyšata, Ph.D. | **Jméno**<br>Jiří | |
| **PRACOVIŠTĚ** | ZČU - FST - KTO | | |
| **DRUH PRÁCE** | ~~**DIPLOMOVÁ**~~ | **BAKALÁŘSKÁ** | **Nehodící se škrtněte** |
| **NÁZEV PRÁCE** | Podprogram pro vytvoření zkosení s využitím pro čtyřosé frézovací centrum | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **FAKULTA** | strojní | | **KATEDRA** | KTO | | **ROK ODEVZD.** | 2014 |

**POČET STRAN (A4 a ekvivalentů A4)**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **CELKEM** | 40 | | **TEXTOVÁ ČÁST** | 33 | | **GRAFICKÁ ČÁST** | - |

| | |
|---|---|
| **STRUČNÝ POPIS (MAX 10 ŘÁDEK)**<br>**ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY** | Bakalářská práce obsahuje postup při vytváření podprogramu pro zkosení hrany vykloněným nástrojem. Podprogram je parametrizován a odladěn v systému Heidenhain. |
| **KLÍČOVÁ SLOVA** | podprogram, Q-parametr, parametrické programování, cyklus |

# SUMMARY OF BACHELOR SHEET

| | | | |
|---|---|---|---|
| **AUTHOR** | **Surname**<br>Jaroš | **Name**<br>Zdeněk | |
| **FIELD OF STUDY** | B 2341 "Programming of NC Machines" | | |
| **SUPERVISOR** | **Surname (Inclusive of Degrees)**<br>Ing. Vyšata, Ph.D. | **Name**<br>Jiří | |
| **INSTITUTION** | ZČU - FST - KTO | | |
| **TYPE OF WORK** | ~~**DIPLOMA**~~ | **BACHELOR** | **Delete when not applicable** |
| **TITLE OF THE WORK** | A subroutine for creating a chamfer using a four-axis milling centre. | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **FACULTY** | Mechanical Engineering | | **DEPARTMENT** | Technology of Metal Cutting | **SUBMITTED IN** | 2014 |

**NUMBER OF PAGES (A4 and eq. A4)**

| | | | | | | |
|---|---|---|---|---|---|---|
| **TOTALLY** | 40 | | **TEXT PART** | 33 | **GRAPHICAL PART** | - |

| | |
|---|---|
| **BRIEF DESCRIPTION**<br><br>**TOPIC, GOAL, RESULTS AND CONTRIBUTIONS** | The Bachelor thesis contains a procedure of creating a subroutine for chamfering an edge using a tilted chamfer tool. The subroutine is parameterized and tuner in Heidenhain system. |
| **KEY WORDS** | Subroutine, Q-parameter, parametric programming, cycle |

# Obsah

# 1 Úvod

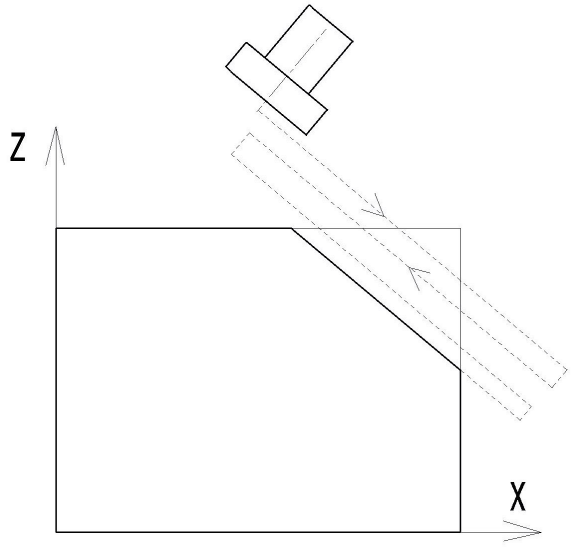
Podprogramy usnadňují tvorbu části programu a zároveň zachovávají možnost, aby tato tvorba byla ruční, nebo dílenská. Zmíněné části fungují samostatně a řeší zpravidla nějakou z elementárních úloh, vrtání otvorů, frézování kapsy, geometrická transformace atd. Elementárnost je myšlena v tom smyslu, že se z nich skládá program, ale jsou do jisté míry komplexní, což spočívá v tom, že každý z nich by bylo nutno zapsat více řádky programu. Jsou to vlastně krátké programy specializované na určitou charakteristickou, obecně se opakující činnost. Jde tedy především o časovou úsporu konečného uživatele podprogramu, který zadá pouze vstupní geometrické, technologické a případně další informace a systém si sám dopočte dráhu nástroje.

V praxi se vyskytují situace, které se v principu opakují a lze je vyjádřit oním principem konkretizovaným pomocí parametrů. Právě pro takové situace jsou vytvořené podprogramy a pokud nejsou vytvořené, je vhodné je vytvořit. Jednou z takových situací může být zkosení hrany vykloněným nástrojem. Aby toto bylo možné, musí být podprogram parametrizován a musí pracovat jako cyklus. Právě tomu se věnuje tato práce.

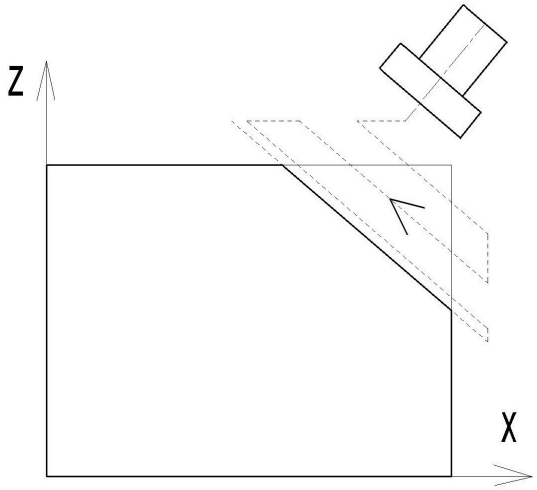
# 2 Specifikace problému

Podstatou této práce je vytvoření podprogramu na hrubování zkosení hrany vykloněnou čelní frézou. Úkolem podprogramu je vyklonit nástroj o předem definovaný úhel, vypočítat body dráhy nástroje a vyhrubovat zkosení do zadané hloubky. Podstatné ovšem je, že hrubování proběhne v požadované hloubce třísky zadané pomocí parametru. Může se tedy hrubovat s více přísuvy. Dráhu podprogram přepočítá tak, aby minimalizoval přejezdy a tím zkracoval čas obrábění. Vstupní informace budou tedy úhel, hloubka zkosení a vnější rozměry. Tyto informace jsou dány výkresovou dokumentací. Další informace, jako je hloubka třísky, bezpečnostní vzdálenost, způsob a směr obrábění atd. si uživatel určí sám. Podprogram bude parametrizován pomocí Q-parametrů a odladěn v systému Heidenhain.

Smyslem je v první řadě zefektivnit práci stroje tím, že se odstraní zbytečný pohyb pracovním posuvem v prostoru mimo obrobek. Dosud bylo možno zkosení rohu provádět cyklem 25 pro obrábění podél kontury, kdy dráha nástroje je v každé hloubce třísky stejná, viz (Obr. 1). V případě použití tohoto cyklu se tedy kopíruje dráha poslední třísky a pouze se mění hloubka obrábění. To vede k ztrátovým časům, kdy například v první hloubce obrábění jsou přejezdy zbytečně velké, a nástroj se pohybuje pracovním pohybem mimo záběr ve vzduchu. Tyto přejezdy jsou v případě použití zde navrhovaného podprogramu odstraněny. Vzdálenosti vyjetí frézy mimo obrobek jsou tak vždy konstantní vůči obrysu obrobku, a to v každé hloubce třísky, jak je vidět na obrázku (Obr. 2).




*Obrázek. 1: Dráha cyklu 25*




*Obrázek. 2: Dráha navrhovaného podprogramu*

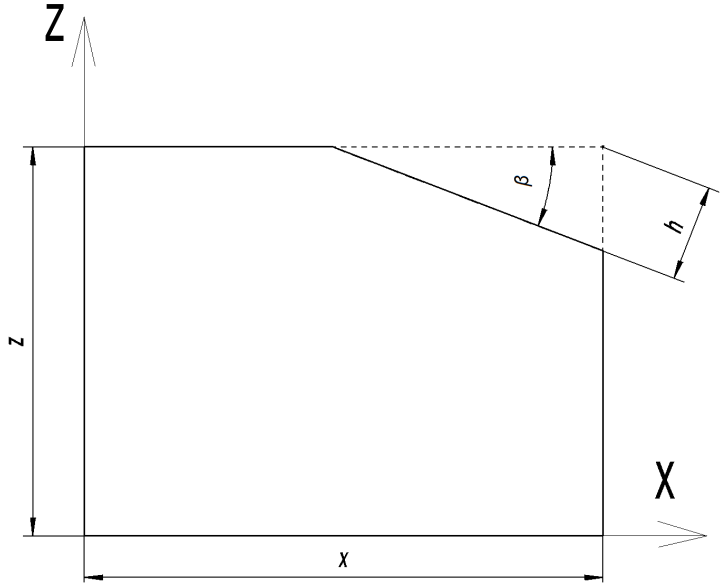
Dalším smyslem je zefektivnit práci programátora, respektive koncového uživatele podprogramu. Čas věnovaný tvorbě programování programu pro konkrétní součást bývá zpravidla kratší než čas pro vytváření obecného podprogramu. Nicméně právě tento čas se vrátí v opakovatelnosti používání podprogramu. Jednorázový program bývá daleko delší a tím je i méně přehledný. Další jeho nevýhodou je použití pouze u jednoho konkrétního případu, pro který je napsán. Při změně jakéhokoliv parametru se musí program předělat. V případě použití cyklu pro zkosení hrany stačí přepsat jedinou hodnotu parametru a podprogram bude

vykonávat proces upravený podle změněného parametru. Například při změně průměru nástroje se musí program přepsat, v případě použití navrhovaného cyklu stačí pouze přepsat hodnotu v tabulce nástrojů. Podprogram má tudíž skutečně přínos i pro obsluhu stroje.

Dráha nástroje v navrhovaném podprogramu je řešena pouze ve dvou osách, a to v ose *x* a *z*. Předpokládá se tedy, že průměr nástroje bude větší, než je šířka obrobku. Pokud by tomu tak nebylo, podprogram se v programu může vyvolat vícekrát, a to vždy s posunutím v ose *y*. Tomu musíme přizpůsobit stav podprogramu, ve smyslu používání vstupních parametrů. Těmto parametrům se proto nesmí v průběhu cyklu přiřazovat jiné hodnoty, než jsou na vstupu. Pro vnitřní výpočty je tedy nutno používat parametry jiné. Pokud by bylo přesto zapotřebí některému parametru přidat jinou hodnotu, musela by se mu na konci cyklu vrátit původní hodnota. V opačném případě by cyklus po opětovném vyvolání pracoval se změněnou hodnotou, a tudíž by pracoval odlišně.

# 3 Matematický model

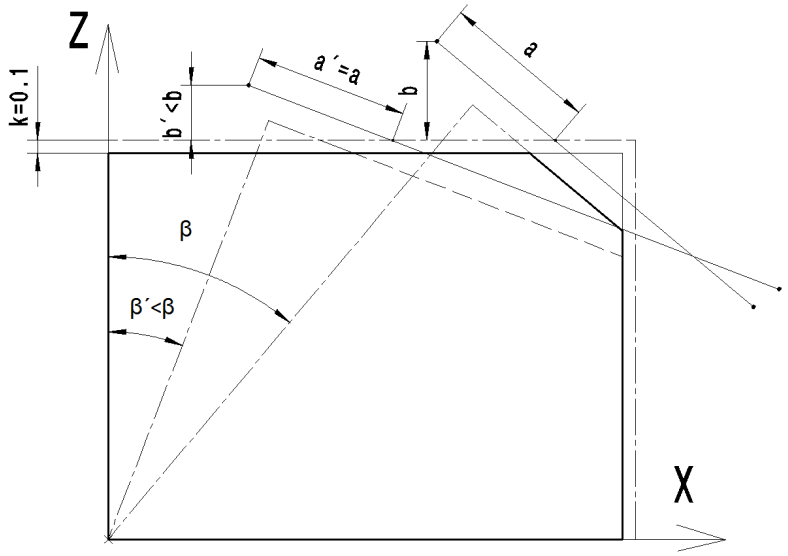
Pro správné určení drah nástroje při obrábění je třeba vytvořit matematický model. V tomto modelu se budou řešit další kroky potřebné k vytvoření podprogramu. Jde o natočení souřadného systému, vypočtení výchozího bodu, vypočtení drah přejezdů a všech dalších bodů potřebných k parametrizaci podprogramu. Z geometrického hlediska jsou pro matematický model podstatné hodnoty: poloha hrany pro zkosení, souřadnice *x* a *z*, úhel zkosení *β* a velikost zkosení daná jeho hloubkou *h*, jako vzdáleností roviny zkosení od hrany součásti. Tyto hodnoty jsou dány výkresovou dokumentací, jak je vidět na obrázku (Obr. 3).




*Obrázek. 3: Výkresová dokumentace*

K matematickému modelu součásti je také přiřazen takzvaný ochranný obal. Tento ochranný obal je konstantní vzdálenost *k*, která je rovna jedné desetině milimetru, okolo obrysu obrobku. Tato konstanta není závislá na velikosti úhlu zkosení, jak je vidět na obrázku (Obr.4). Její význam si nyní objasníme. Kdyby okolo obrobku nebyla, nejmenší vzdálenost nástroje od obrobku v nájezdové poloze by byla dána pouze hodnotou bezpečnostní vzdálenosti *a* a úhlem zkosení *β*. Pokud se bude úhel zkosení *β* zmenšovat, bude se také zmenšovat kolmá vzdálenost nájezdového bodu od obrobku $b' < b$, i přesto, že bezpečnostní

vzdálenost *a* bude dostatečně velká. Pro velmi malé hodnoty úhlu *β* se může stát, že nástroj najede rychloposuvem do plného materiálu v důsledku drobné nepřesnosti při odměření nástroje, drobné nepřesnosti odměření obrobku, drobné nerovnosti povrchu, drobné nepřesnosti najetí stroje rychloposuvem (dané parametrem stroje „přesnost opakovaného najetí“) a případného rizika překmitnutí posuvových regulátorů. A právě proto je zde tato konstantní vzdálenost, která je zohledněna v podprogramu, která by měla postačit pro vyloučení všech zmíněných nebezpečí. Při počítání bodů nájezdu je také zohledněna ve výpočtech. Nijak tedy neovlivní přesnost obrábění.




*Obrázek. 4: Ochranný obal*

V matematickému modelu jde o určení dráhy bodů potřebných k vytvoření podprogramu. Tyto body jsou popsány pomocí matematických funkcí. V těchto funkcích jsou matematické proměnné v podobě Q-parametrů. Podprogram si sám body vypočte na základě těchto parametrů.

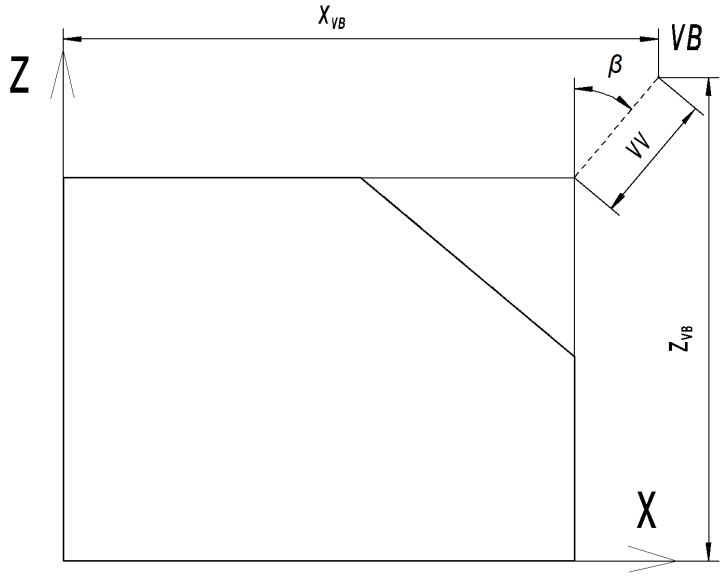
## 3.1 Výchozí bod

Výchozím bodem podprogramu není myšlen nulový bod obrobku. Význam nulového bodu je popsán v [2] jako zvolený bod počátku souřadného systému. Tento bod je dle autorů [2] volen programátorem na základě vhodnosti přepočítávání k tomuto bodu souřadnice, vhodnosti upnutí obrobku, vhodnosti při nastavování na stroji. Uživatel podprogramu tedy pracuje s nulovým bodem, který je definován v hlavním programu. Programátor před vyvoláním podprogramu na zkosení musí pouze nadefinovat polohu rohu vůči nulovému bobu obrobku, se kterým pracuje hlavní program. Poloha tohoto rohu je dána výkresovou dokumentací na obrázku (Obr. 3), a právě k tomuto rohu součásti jsou vztaženy souřadnice výchozího bodu podprogramu.

Výchozí bod ***VB*** je dán velikostí výchozí vzdálenosti ***VV***, která je řízena parametrem, vůči rohu součásti, jak je vidět na obrázku (Obr. 5). Tento parametr je definován uživatelem pomocí Q-Parametru. Vzdálenost výchozího bodu je kolmá na plochu zkosení. Na základě této výchozí vzdálenosti je potřeba v podprogramu nadefinovat souřadnice výchozího bodu.

Do těchto souřadnic na začátku podprogramu najede nástroj, ve kterém bude provedeno pootočení souřadného systému a také naklopení vřeteníku stroje. Následně z tohoto bodu bude nástroj najíždět na první hloubku obrábění.

Jak již bylo řečeno, v tomto bodě se provede naklopení vřeteníku, proto je potřeba být při definování velikosti parametru výchozí vzdálenosti ***VV*** obezřetný, aby se předešlo kolizím způsobeným při natáčení vřeteníku stroje.




*Obrázek. 5: Souřadnice výchozího bodu*

Souřadnice výchozího bodu jsou pak dány rovnicemi (1) a (2).

$$x_s = x + x_{VV} \cdot \sin(\beta) \tag{1}$$

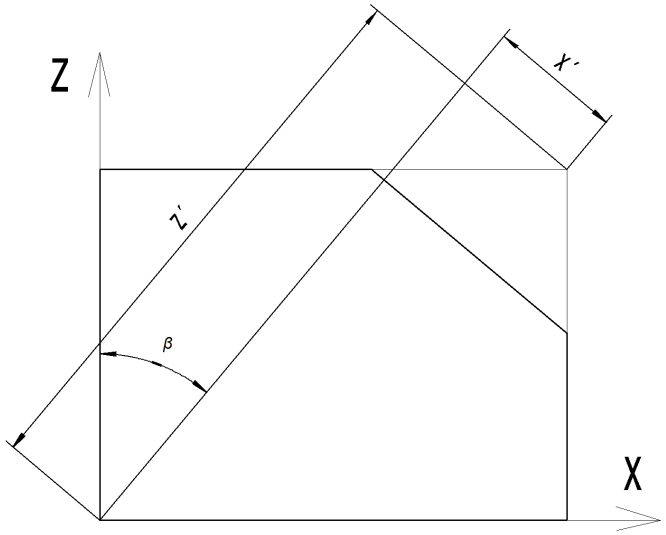
$$z_s = z + z_{VV} \cdot \cos(\beta) \tag{2}$$

## 3.2 Natočení souřadného systému

Frézování úkosu bude probíhat čelním způsobem při natočení nástroje vůči obrobku o úhel totožný s úhlem úkosu. Jde tedy o natočení souřadného systému o úhel $\beta$ okolo osy *y*, jak je vidět na obrázku (Obr. 6). Na základě natočení se vypočítají nové souřadnice, které budou závislé na tomto úhlu a vnějších rozměrech obrobku. Další body použité v tomto programu budou vztažené k těmto novým souřadnicím $z'$ a $x'$. Souřadnice $z'$ a $x'$ jsou pak dány rovnicemi (3) a (4).

$$x' = x\cos(\beta) - z\sin(\beta) \tag{3}$$

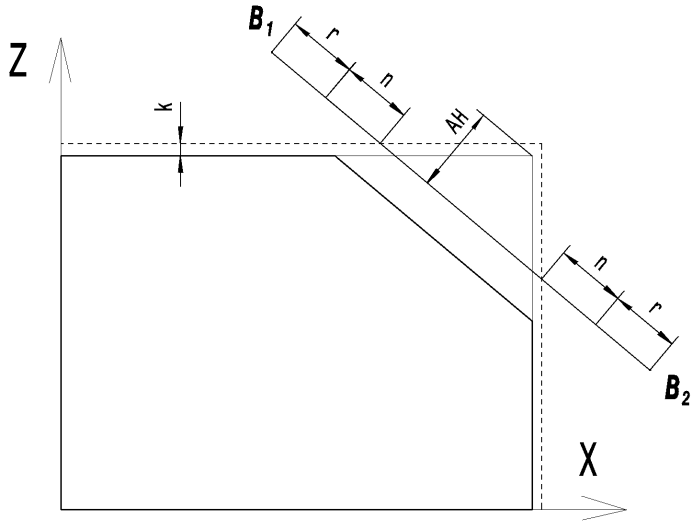
$$z' = x\sin(\beta) + z\cos(\beta) \tag{4}$$




*Obrázek. 6: Natočení souřadného systému*

## 3.3 Body přejezdů

V rámci nového souřadného systému se musí vypočítat body $\boldsymbol{B_1}$ a $\boldsymbol{B_2}$, do kterých se bude najíždět rychloposuvem a od nichž bude nástroj obrábět pracovním pohybem viz. (Obr. 7). Je potřeba vytvořit rovnice, které budou tyto body automaticky počítat. V těchto rovnicích je zohledněná aktuální hloubka $h_a$, která v každé hloubce třísky bude odlišná, a podprogram v každém cyklu vypočte tyto souřadnice nové, vztažené právě k aktuální hloubce. Tato aktuální hloubka zkosení $h_a$ není totožná s celkovou hloubkou zkosení $h$ ani s hloubkou přísuvu $a_p$. Jde o vzdálenost obráběné plochy od rohu součásti. Tedy vzdálenost v které bude následně součást obráběná. Dále je zde zohledněna velikost najetí $n$, tato velikost je definována uživatelem, poloměr nástroje $r$ a ochranný obal $k$. Souřadnice bodu přejezdu $\boldsymbol{B_1}$, $\boldsymbol{B_2}$ jsou pak dány rovnicemi (4), (6), (7) a (8).




*Obrázek. 7: Body přejezdů*

$$x_1 = x' - \left(r + n + \frac{k}{\sin(\beta)} + \frac{h_a}{\tan(\beta)}\right) \tag{5}$$

$$z_1 = z' - h_a \tag{6}$$

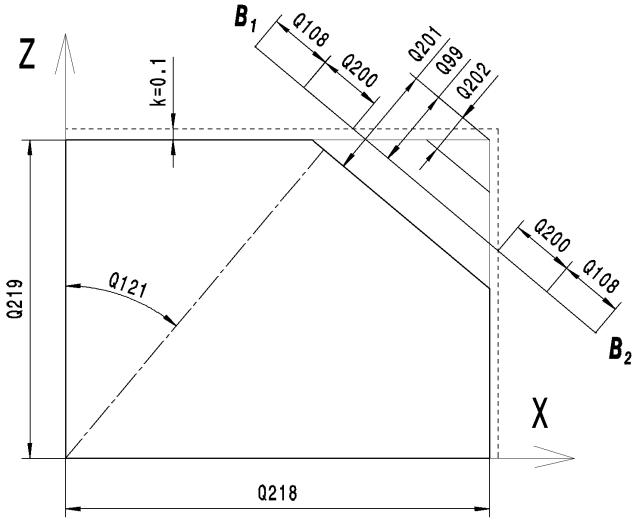
$$x_2 = x' + \left(r + n + \frac{k}{\cos(\beta)} + h_a \tan(\beta)\right) \tag{7}$$

$$z_2 = z_1 \tag{8}$$

# 4 Q-parametry

Při vytváření podprogramu jsou použity namísto číselných hodnot Q-parametry. Konečný uživatel podprogramu přiřadí příslušným Q-parametrům konkrétní číselné hodnoty. Tyto hodnoty můžou obsahovat hodnoty souřadnic, posuvů, otáček, goniometrické funkce atd. Všechny funkce jsou závislé na matematickém modelu. Jednotlivé veličiny matematického modelu jsou zadány pomocí parametrů, které se budou využívat v matematických rovnicích. Označení Q parametru se skládá z písmene Q a číselného indexu např. Q1, Q2 atd. Q-parametr může být nositelem logické informace, například způsobu obrábění. V závislosti na splnění těchto logických podmínek podprogram řídí technologické kroky a způsob obrábění. Použité Q-Parametry se budou rozlišovat na vnější a vnitřní [1].

Vnější parametry slouží pro zadávání číselných hodnot, jimiž uživatel podprogramu popisuje konkrétní požadavky pro výrobu konkrétního zkosení. Tyto parametry jsou voleny tak, aby odpovídaly již zavedené konvenci z jiných podprogramů, z příručky Heidenhain. Tak například parametr Q 108 je předvolen hodnotou z TNC a náleží aktuálnímu rádiusu nástroje, který je definován v tabulce nástrojů. Obráběcí cykly s čísly od 200 používají Q-parametry jako předávací parametry. Jednotlivým rysům specifických tvarů obráběních pomocí podprogramů přísluší odpovídající, vždy stejný parametr. Typicky to bývá celková hloubka obrábění, které odpovídá parametr Q 201. Parametr Q 200 je přiřazen bezpečnostní vzdálenosti (jde o vzdálenost mezi hrotem nástroje a povrchem obrobku ve směru pohybu nástroje – do tohoto parametru se zadává vždy kladná hodnota). Tyto a další parametry jsou použity ve většině cyklů vždy ve stejném významu. Tato konvence proto bude zachována i v podprogramu pro zkosení. Parametr Q 201 je definován jako hloubka zkosení. Je to kolmá vzdálenost mezi povrchem obrobku a dnem zkosení, tento parametr je rovněž v mnoha cyklech, například u cyklu vrtání je definován jako hloubka díry. Hloubka přísuvu $a_p$ je definována parametrem Q 202, je to vzdálenost, o kterou se nástroj pokaždé přisune, jde tedy o hloubku třísky. Tato hodnota je zadávána koncovým uživatelem podprogramu podle zvolených řezných podmínek. Tato hodnota je v každé hloubce zkosení stejná, je to tedy konstanta. Pouze v poslední hloubce třísky bude hodnota menší nebo stejná v závislosti na zbytku po obrábění v těchto hloubkách třísky. Nejde tedy o podíl k celkovému zkosení. Parametry Q 218 a 219 jsou hodnoty vnějších souřadnic *x* a *z*. Posledním vnějším parametrem Q 121, který definuje obrysy součásti, je úhel natočení souřadnicového systému kolem osy *y*. Tento úhel natočení se rovná úhlu zkosení, jak je viděl na obrázku (Obr. 8).




*Obrázek. 8: Přiřazení Q-Parametrů*

Do vnitřních parametrů nejsou zadávány hodnoty uživatelem podprogramu. Podprogram do nich sám ukládá výsledky vnitřních výpočtů na základě rovnic odpovídajících matematickému modelu v závislosti na uživatelem definovaných vnějších parametrech. Některé tyto Q-parametry také odpovídají zavedené konvenci z jiných cyklů a jsou tedy také voleny podle této konvence. Hodnota ovšem odpovídá specifikám frézování zkosení. Konkrétně například parametr Q 203 je hodnota *z*-ové souřadnice rohu obrobku při naklopení souřadného systému, a tím pádem je to také souřadnice povrchu obrobku. Tento parametr figuruje ve stejném významu ve všech vrtacích cyklech i v cyklech pro kapsování, výrobu drážek a dalších. Uživatel jej však v podprogramu frézování zkosení nezadává ani se nedozví

jeho hodnotu, ale podprogram jej realizuje při obrábění. Podobných parametrů je ještě několik. Indexy ostatních vnitřních parametrů jsou voleny libovolně, a to od čísla 99 níže. Parametr Q 99 kupříkladu definuje aktuální hloubku zkosení, jedná se o hodnotu polohy ve směru osy nástroje. ve které se právě nástroj nachází. Tato hodnota bývá násobkem hloubky přísuvu nebo při poslední třísce je rovna hloubce zkosení. V závislosti na této aktuální hloubce podprogram spočítá body přejezdů.

Rovnice (1), (2), (3), (5), (4), (6) a (7) je proto vhodné přepsat s příslušnými parametry namísto symbolických proměnných.

$$Q98 = Q218 + Q204 \cdot \sin(Q121) \tag{9}$$

$$Q97 = Q219 + Q204 \cdot \cos(Q121) \tag{10}$$

$$Q230 = Q218 \cos(Q121) - Q219 \sin(Q121) \tag{11}$$

$$Q203 = Q218 \sin(Q121) + Q219 \cos(Q121) \tag{12}$$

$$Q91 = Q230 - (Q108 + Q200 + \frac{0{,}1}{\sin(Q121)} + \frac{Q99}{\tan(Q121)}) \tag{13}$$

$$Q93 = Q203 - Q99 \tag{14}$$

$$Q92 = Q230 + (Q108 + Q200 + \frac{0{,}1}{\cos(Q121)} + Q99 \tan(Q121)) \tag{15}$$

## 4.1 Tabulka Q-parametrů

Pro přehlednost, rychlejší orientaci v textu a zobrazení rozdělení Q-Parametrů je zde tabulka (tab. 1), která zobrazuje Q-parametry použité v podprogramu. V tabulce v levém sloupci jsou vidět vnější parametry, tyto parametry je potřebné definovat před vyvoláním podprogramu. V pravém sloupci jsou vnitřní parametry. Tyto parametry uživatel nedefinuje, podprogram si do nich ukládá hodnoty sám na základě výpočtů v závislosti na vnějších parametrech. Do těchto parametrů jsou také vloženy pomocné hodnoty, které budou následně doplňovat vnější parametry. Tyto hodnoty také nedefinuje uživatel, jsou definování přímo v podprogramu. Například do parametru Q99, který definuje aktuální hloubku zkosení, je na začátku podprogramu přiřazená 0. Tato hodnota proto, jelikož podprogram je na začátku a součást ještě není obrobena. Stejné tomu je i v parametru Q90, kde je přiřazena na začátku hodnota 1. Tento parametr bude pomáhat při určování pohybů nástroje. Tento parametry a všechny ostatní parametry budou v průběhu použití detailněji vysvětleny dále v textu.

| Vnější parametry | | Vnitřní parametry | |
|---|---|---|---|
| Q108 | Poloměr nástroje *r* | Q99 | Aktuální hloubka $h_a$ |
| Q200 | Bezpečnostní vzdálenost *n* | Q203 | Souřadnice povrchu *z´* |
| Q201 | Hloubka zkosení *h* | Q230 | Souřadnice povrchu *x´* |
| Q202 | Hloubka třísky $a_p$ | Q91 | Souřadnice přejezdu $B_{1x}$ |
| Q218 | Vnější souřadnice *x* | Q92 | Souřadnice přejezdu $B_{2X}$ |
| Q219 | Vnější souřadnice *z* | Q93 | Souřadnice přejezdu $B_{12Z}$ |
| Q121 | Úhel natočení $\beta$ | Q98 | Souřadnice výchozího bodu $X_{VV}$ |
| Q204 | Výchozí vzdálenost ***VV*** | Q97 | Souřadnice výchozího bodu $Z_{VV}$ |
| Q215 | Volba technologie | Q94 | Souřadnice přejezdu $X_p$ |
| | | Q95 | Souřadnice přejezdu $Z_p$ |
| | | Q96 | Výchozí úhel natočení |
| | | Q80 | Transformace Q215 |
| | | Q90 | Rozlišování dráhy |

*Tabulka 1: Tabulka Q-Parametrů*

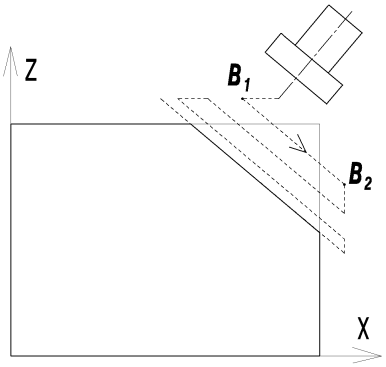
# 5 Volba technologie

V podprogramu je možnost zvolit si technologii obrábění, se kterou bude uživatel chtít danou součást obrábět. Celkem se naskytují tři možnosti. První a nejúčinnější možnost je způsob označovaný u obdobných metod vestavěných cyklů jako „pendlování“ viz. (Obr. 9). Česky to lze přeložit jako kývání. Nástroj po přejetí pracovním posuvem z jednoho bodu do druhého se v koncovém bodě posune rychloposuvem o hloubku třísky a jede zpět pracovním posuvem do předchozího bodu. Následuje opět posunutí rychloposuvem o hloubku třísky a přesun na následující bod.

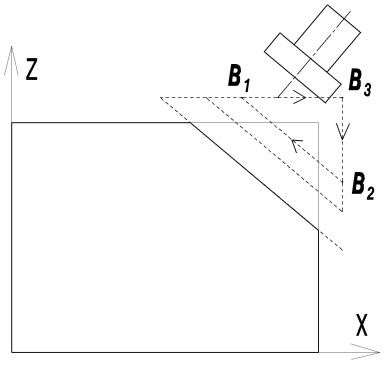
Další možností je obrábění vždy v jednom směru, a to buď z bodu $\boldsymbol{B_1}$ do bodu $\boldsymbol{B_2}$ viz (Obr. 11), nebo obráceně z bodu $\boldsymbol{B_2}$ do bodu $\boldsymbol{B_1}$ viz (Obr. 10). Po projetí pracovním posuvem z jednoho bodu do druhého v aktuální hloubce a odebrání třísky odjede nástroj rychloposuvem do přejezdového bodu $\boldsymbol{B_3}$ a z tohoto bodu zpět do prvního bodu posunutým o hloubku přísuvu. Tyto přejezdy jsou prováděny rychloposuvem a do bodu $\boldsymbol{B_3}$ se vyjíždí mimo materiál, tedy mimo obrys polotovaru. Dva posledně jmenované způsoby obrábění se asi budou používat především tehdy, když polotovar bude širší než je průměr nástroje a podprogram bude vyvoláván vícekrát za sebou, vždy posunutý v souřadnici *y*. Pokud bychom v tomto případě použili cyklus pendlování, tak při druhé a následně při každé další hloubce třísky by se boky zubů třely po již obrobeném materiálu, a to střídavě sousledně a nesousledně[1], což by mohlo být někdy nevhodné. Tomu se právě lze vyhnout použitím jednoho z oněch jednosměrných cyklů.

1 Při sousledném frézování směřuje vektor posuvové rychlosti na opačnou stranu než vektor řezné rychlosti. Tloušťka třísky je zde od maximální k minimální až k nulové. Při frézování nesousledném směřují oba vektory na stejnou stranu a tloušťka třísky má opačný průběh. Bližší informace jsou v literatuře [3].

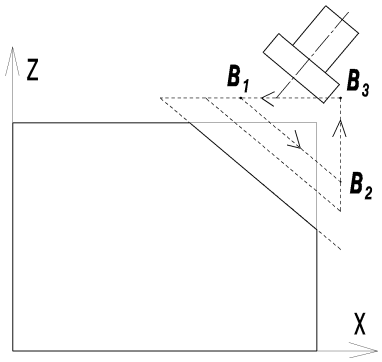
Volbu způsobu obrábění provádí uživatel podprogramu pomocí parametru Q215, který je tedy řazen mezi vnější parametry.




*Obrázek. 9: Q215=0*

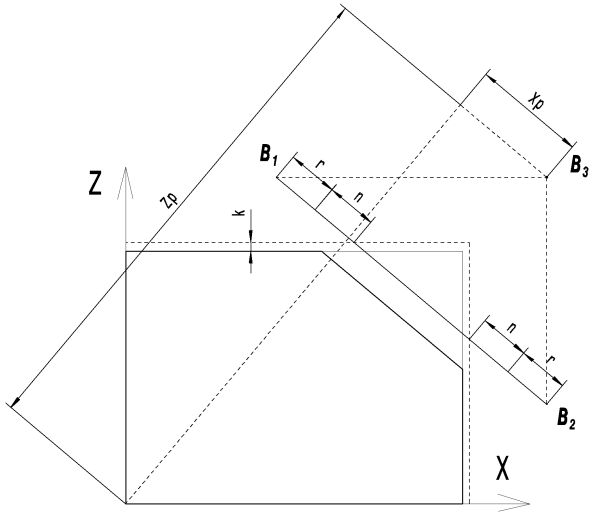



*Obrázek. 10: Q215=1*




*Obrázek. 11: Q215=-1*

Pro případ, že by se použil jeden z cyklů přejezdu pouze z jedné strany, musí se vypočítat souřadnice bodu přejezdu $B_3$, viz obrázek (Obr. 12), do kterého bude nástroj jezdit při přejezdech. Tento bod bude sloužit také jako zahajovací bod pro technologii pendlování. Je závislý pouze na vnějších rozměrech, poloměru nástroje a velikosti najetí. V tomto bodě je také zahrnuta konstanta ochranného obalu *k*. Tyto hodnoty jsou konstanty. Souřadnice bodu $B_3$ budou tedy v každé hloubce třísky stejné. Mohlo by se zdát, že tento bod má jednu souřadnici shodnou s bodem $B_1$ a druhou s bodem $B_2$. Je však třeba si uvědomit, že se počítají souřadnice v pootočeném souřadném systému, a nikoli v systému původním.




*Obrázek. 12: Bod přejezdu 3.*

Souřadnice Xp a Zp bodu $B_3$ jsou dány rovnicemi (16) a (17).

$$x_p = x' + (r+n)\cdot\cos(2\beta) + \sqrt{k^2+k^2}\cdot\sin(45-\beta) \tag{16}$$

$$z_p = z' + (r + n)\cdot\sin(2\beta) + \sqrt{k^2 + k^2}\cdot\cos(45 - \beta) \tag{17}$$

Dosazením příslušného Q-parametru do rovnic (16) a (17) se získají nové rovnice (18) a (19), které už lze použít v podprogramu.

$$Q94 = Q230 + (Q108 + Q200)\cdot\cos(2\mathrm{Q}121) + \sqrt{0{,}02}\cdot\sin(45 - Q121) \tag{18}$$

$$Q95 = Q203 + (Q108 + Q200)\cdot\sin(2\,Q121) + \sqrt{0{,}02}\cdot\cos(45 - Q121) \tag{19}$$

# 6 Podprogram

Nyní v této kapitole se věnujme samotnému podprogramu. Způsobů, jak dojít ke správnému cíli podprogramu, může být více. Začátek a konec podprogramu musí být v souladu s programem pro který je psán, nicméně logické kroky, které bude podprogram za sebou řešit, jsou plně v kompetenci programátora, který podprogram píše, a je na něm, aby dodržel zásady programovacího jazyka, technologie obrábění, přehlednost podprogramu, vyhnul se zbytečným oklikám a snažil se program psát co nejefektivněji, nejpřehledněji a nejsrozumitelněji.

V případě, kdy si uživatel volí mezi třemi způsoby obrábění, a to pendlování, obrábění z jedné strany nebo obrábění z druhé, byla první myšlenka naprogramovat tyto tři způsoby obrábění jako samostatné podprogramy, které by byly vyvolány v hlavním programu. Tyto tři způsoby obrábění by pak řešily zkosení samostatně včetně všech potřebných výpočtů. Po rozkreslení situace je vidět na obrázcích (Obr. 9, 10 a 11), jak všechny tyto tři způsoby pracují se stejnými body, respektive se stejnými pohyby, a to pohyb z bodu $\boldsymbol{B_1}$ do bodu $\boldsymbol{B_2}$ nebo obráceně. U technologie obrábění z jedné strany nebo z druhé navíc přibývá mezi tímto pohybem ještě pohyb do bodu $\boldsymbol{B_3}$. Tento bod je obsažen i v technologii pendlování, kde slouží jako výchozí bod, od kterého se najíždí do první hloubky záběru. Pokud bychom tedy řešili podprogram jako vyvolávací program pro následující tři podprogramy, které by samostatně řešily vždy jednu z předem definovaných technologií, byly by tyto podprogramy téměř podobné a většina bloků by se opakovala a celkový podprogram by byl zbytečně dlouhý. Namísto toho je vytvořen jeden podprogram, který je složen z pohybů. Tyto pohyby jsou řešeny jako samostatné rutiny vyvolávané cyklicky pracujícím řídicím jádrem. Vždy po odjetí jednoho pohybu (například z bodu $\boldsymbol{B_1}$ do bodu $\boldsymbol{B_2}$) se před samotným skončením této rutiny ověří, jaký pohyb má následovat, respektive jaký cyklus se má vyvolat, zda odjet do bodu $\boldsymbol{B_3}$, nebo následovat v pendlování a sjet o další přísuv.

Obě tyto metody definování podprogramu mají stejný výsledek a z hlediska technologie obrábění jsou ve výsledku od sebe nerozpoznatelné. Jde tedy pouze o přehlednou strukturu podprogramu a snížení taktů softwaru.

Dále je potřeba uvědomit si, jakým způsobem je třeba dostat se k bodům přejezdu. Z rovnic (5), (6) a (7) víme, že polohy přejezdových bodů jsou závislé na aktuální hloubce zkosení. Je zde navíc zohledněna velikost nájezdu a poloměr nástroje. Právě aktuální hloubka je pro program velice důležitá a způsobů, jak ji definovat, je opět několik. Zde je třeba brát velký zřetel na technologii obrábění. Aktuální hloubka bude vždy násobkem počtu již uskutečněných přísuvů a zvolené hloubky třísky $a_p$. Hloubka třísky může být rovna podílem,

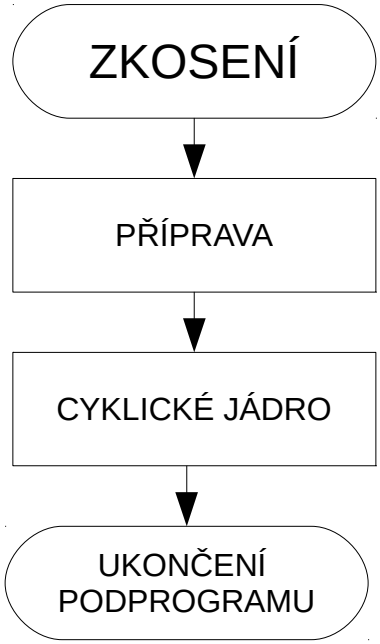
který zadá programátor formou počtu přísuvů ku celkové hloubce zkosení. V tomto případě bude každá hloubka třísky stejná, tedy včetně té poslední. Zde ale ztrácíme možnost si hloubku korigovat sami právě podle technologie obrábění, řezných podmínek, možnostmi stroje a nástroje a dalšími faktory. Byli bychom odkázáni pouze na možnosti korigovat hloubku přísuvu počtem přísuvů.

Proto se zde raději volí hloubka třísky pevně, respektive pevně volitelná vstupním parametrem, který si nadefinuje sám uživatel podprogramu pomocí vnějšího parametru. Od této hloubky se bude odvíjet počet přísuvů, který bude tedy celým podílem celkové hloubky zkosení a zbytkem. Poslední tříska bude tedy rovna hloubce třísky nebo bude menší. Může se tedy stát, že bude mít kvůli případné malé hloubce nevyhovující řezně podmínky, které zapříčiní špatný povrch obráběné plochy. Tento nedostatek se ale v této práci nebude řešit z následujících důvodů. Především podprogram je míněn a koncipován hlavně jako hrubovací a nic nebrání omezení jeho hloubky o tloušťku třísky potřebnou pro dokončování a následné vyvolání téhož podprogramu se zadáním parametrů pro práci přímo v plné hloubce načisto. Mimoto příliš tenká poslední vrstva jako výsledek náhodného libovolného zadávání přísuvu má nepříliš vysokou pravděpodobnost a zběhlejší uživatel je schopen si ji namátkou výpočtem překontrolovat, protože – a to je třetí důvod – všechny cykly vestavěné v systému pracují s celkovou hloubkou a přísuvem právě takovýmto způsobem. Podprogram je tedy v otázce přísuvů záměrně koncipován tak, aby zachoval chování jaké je obecně u cyklů v systému Heidenhain.

Podprogram je možno rozdělit na tři etapy, jak je vidět na obrázku (Obr. 13). Tato kapitola je rozdělena na příslušné podkapitoly, které jsou koncipovány stejně, jako je vývojový diagram na obrázku (Obr. 13). V těchto podkapitolách budou jednotlivé kroky blíže rozkresleny a vysvětleny.

Úvod podprogramu řeší kroky, které jsou potřebné pro správný chod podprogramu. Jde o definování vztahů, které na základě vstupních informací vypočítají a blíže specifikují jednotlivé body podprogramu. Tyto body jsou dále použity v cyklickém jádře podprogramu, kde na jejich základech se provádí jednotlivé pohyby stroje potřebné k obrobení součásti. V úvodu podprogramu jsou definovány pouze ty vztahy, které jsou pro zbytek podprogramu konstantami. Jako je například výchozí bod, natočení souřadného systému, k němu nové souřadnice, zvolená technologie obrábění a k ní příslušné parametry potřebné k správnému chodu oné technologie atd. Tato část je tedy vlastně přípravnou fází z hlediska běhu celého programu a bude vhodné ji tak i nadále nazývat.

V cyklickém jádru podprogramu jsou prováděny jednotlivé úkony, které jsou nezbytné pro obrobení součásti. Na začátku tohoto jádra jsou také definovány určité vstupní informace. Tyto definice jsou již v podobě proměnných funkcí, které v průběhu cyklického jádra mění svou hodnotu tak, jak je součást postupně obráběna. Jde především o výpočet bodů, mezi kterými se pohybuje nástroj při obrábění jednotlivými přísuvy. Tyto body jsou vztaženy k aktuální hloubce zkosení, která je zde také definována. Následně na této aktuální hloubce zkosení, budou v tomto cyklickém jádře realizovány jednotlivé pohyby nástroje, které budou součást obrábět.




*Obrázek. 13: Podprogram*

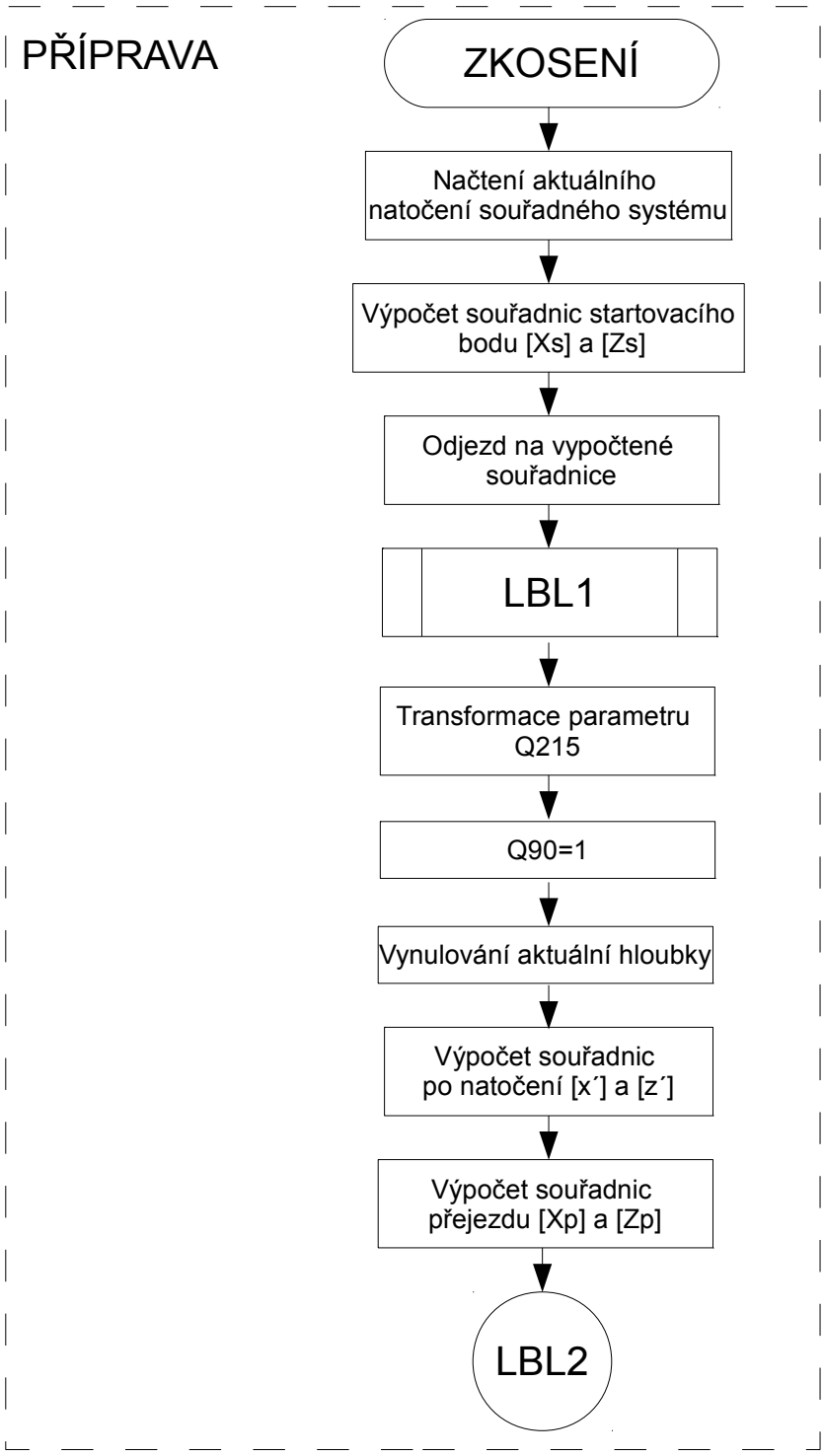
V poslední části podprogramu je jeho ukončení. Tato část je definována jako určitý proces, který provádí bezpečné ukončení podprogramu. Zde jsou definovány a realizovány jednotlivé kroky, které nasměrují nástroj do bodu, ve kterém bude bezpečné podprogram ukončit. Jde tedy o odjezd nástroje do bezpečné vzdálenosti, v které bude souřadný systém a tedy i vřeteník stroje pootočen do polohy, v které byl podprogram vyvolán.

## 6.1 Přípravná fáze

Podprogram, který je předmětem této práce, neřeší definici nástroje, jeho vyvolání, osu nástroje, otáčky vřetene, definici neobrobeného materiálu[2] a další informace obecnějšího charakteru potřebné k obrábění, které ovšem nejsou přímo charakteristické pro zkosení rohu. Tyto informace je nutné zadat v rámci hlavního programu.

Na obrázku (Obr. 14) je zobrazen vývojový diagram. Tento diagram zobrazuje přípravnou fázi podprogramu. Hned na začátku podprogramu se načte do parametru Q96 aktuální úhel natočení souřadného systému. Aktuální úhel natočení je úhel, který je před vyvoláním podprogramu. Načtení této hodnoty je z důvodu, aby tato hodnota mohla být použita na konci programu při natáčení souřadného systému a hlavně rotačních os stroje zpět. Tím se vrátí natočení do výchozí polohy, respektive do polohy, v které byl podprogram vyvolán. Tedy do polohy, která je uložena v parametru Q96.

2 Systém Heidenhain umožňuje zadávat rozměry výchozího polotovaru čistě jen pro účely zobrazování simulace chodu programu. Pro ten účel slouží definice „BLOCK FORM“ [1]. Nemá smysl polotovar definovat uvnitř podprogramů a cyklů.

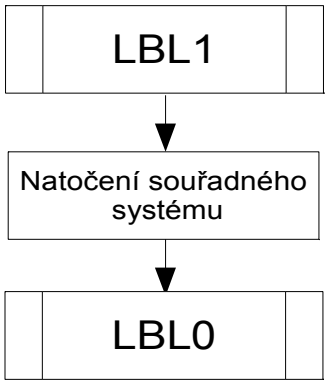



*Obrázek. 14: Diagram začátku programu*

V následujícím kroku jsou vypočteny souřadnice startovacího bodu podprogramu *Xs* a *Zs*. Tyto body jsou vypočteny pomocí rovnic (1) a (2), kde v programu jsou rovnice s již vyplněnými Q-parametry, jako je tomu v rovnicích (9) a (10).

Na základě těchto vypočtených bodů odjede nástroj do startovacího bodu. Na místo číselné hodnoty, která udává koncový bod pojezdu, se dosadí Q-parametr, jehož hodnotu si podprogram vypočítal v předchozím bloku. Pohyb do tohoto bodu je proveden rychloposuvem. Poté se vyvolá v podprogramu rutina s návěstím LBL 1.

V rutině LBL 1, který je vidět na obrázku (Obr. 15), bude provedeno pootočení souřadného systému okolo osy *y* podle definovaného úhlu parametrem Q121, který je tedy roven úhlu zkosení a je zadáván uživatelem podprogramu. Pootočením souřadného systému se natočí také fyzické osy obráběcího stroje.




*Obrázek. 15: LBL1*

Po pootočení souřadného systému se rutina LBL 1 ukončí a vrátí se zpět do programu, kde byla vyvolána a podprogram pokračuje ve čtení následujícího řádku.

Ukončení a návrat zpět na místo hlavního podprogramu, kde byla příslušná rutina vyvolána, se děje příkazem LBL 0. Dokud tomu tak není, podprogram pokračuje volně dál ve čtení následujícího řádku. Jelikož v hlavním podprogramu se na příslušné rutiny pouze odkazuje příkazem CALL LBL a indexem příslušné rutiny, jejich obsah je však psán za hlavním podprogramem. Pokud tedy rutina není ukončen příkazem LBL 0, podprogram dále pokračuje ve čtení následujícího řádku, kde může být například napsáná jiná rutina, což může být i v některém případě žádoucí.

Tento krok, natočení souřadného systému, je řešen pomocí samostatné rutiny z toho důvodu, aby byla možnost si tuto rutinu na konci programu opět vyvolat, a na místo parametru Q121, který definuje úhel natočení, dosadit parametr Q96, a tím natočit souřadný systém a vřeteník stroje zpět do polohy, v které byl natočen.

### 6.1.1 Transformace parametru Q215

Parametr Q-215 určuje, jakým způsobem se bude zkosení obrábět. Je to právě tento parametr, který rozhoduje o tom, zda se při postupném odfrézování jednotlivých vrstev bude najíždět vždy z jednoho směru, a to buď shora dolů – tedy směrem ke stolu stroje a nebo ve směru opačném s přejezdy přes třetí bod umístěný poblíž původní hrany a nebo zda se využije strategie označovaná často jako pendl – kývání. To, jaký způsob obrábění se má použít, se zadá pomocí čísel –1, 0 a 1, jak je vidět na obrázcích (Obr. 9, 10 a 11). Tato symbolika je

volena podle již zavedené konvence z jiných cyklů, aby byla pro uživatele co nejvstřícnější. Pro podprogram na zkosení hrany je však tato konvence nevyhovující z důvodů potřeby co nejelegantnějšího určení směru obrábění pro každou strategii a pro každou hloubku. Tento parametr je tudíž potřeba transformovat na jiné hodnoty, které budou vhodnější pro použití v podprogramu.

Program, jak již bylo psáno na začátku této kapitoly, pracuje vlastně pouze s dvěma pracovními pohyby, a to z bodu $B_1$ do bodu $B_2$ nebo obráceně. V podprogramu tedy mohou být zpracovány jen tyto dva pohyby a ty se mohou opakovaně nebo střídavě vyvolávat řízeně na základě vyhodnocení určité podmínky. Nyní je potřeba formulovat onu příslušnou podmínku, která bude schopna řídit který z pohybů se má použít. Pokud bude zvolena technologie obrábění v pendlování, nebo obrábění vždy v jednom směru, a to z bodu $B_1$ do bodu $B_2$, podprogram provede vždy první pohyb do bodu $B_1$ a z tohoto bodu pojede pracovním pohybem do bodu $B_2$. Proto je potřeba, aby tyto technologie měly shodný vyvolávací podnět. V případě použití symbolů z již zavedené konvence z jiných podprogramů mají tyto symboly pro pendlování hodnotu 0 a pro přejezd z bodu $B_1$ do bodu $B_2$ hodnotu 1. V tomto případě je stejným vyvolávacím podnětem skutečnost, že hodnota obou čísel je nezáporná a podmínku nesplňuje parametr -1, který slouží pro vyvolání technologie, která obrábí z bodu $B_2$ do bodu $B_1$. Tento příkaz splňuje potřebu shodnosti vyvolávacího podnětu. Nicméně je zapotřebí rozlišit také obrábění v pendlování a obrábění ve směru z bodu $B_1$ do bodu $B_2$. V první hloubce přísuvu budou mít obě tyto strategie stejný pohyb, a to pohyb z bodu $B_1$ do bodu $B_2$. Ovšem po ukončení tohoto prvního pohybu se již budou lišit. Cyklus obrábění v jednom směru musí odjet do bodu $B_3$ a cyklus pendlování musí dále obrábět v opačném směru posunut o hloubku přísuvu. Každý z těchto cyklů má jiný symbol pro vyvolání (pendlování 0 a přejezd z bodu $B_1$ do bodu $B_2$ hodnotu 1) a je tedy možnost použít další příkaz, který tyto technologie rozliší. To je ovšem zbytečné vkládání dalších příkazů. Proto je vhodné najít jeden příkaz, který dokáže rozlišit všechny pohyby. Minimalizace příkazů totiž bude jednak činit program elegantnější, ale také bude šetřit čas pro vykonávání výpočtových a programovacích operací[3].

Při volbě technologie obrábění v pendlování je zpětný chod v tomto cyklu stejný pohyb, jako při volbě technologie obrábění z bodu $B_2$ do bodu $B_1$. Proto je zapotřebí, aby také tyto pohyby měly shodný vyvolávací podnět. Je tedy nutné, aby technologie pendlování měla při jednom pohybu stejný vyvolávací podnět s jedním cyklem, a při zpětném pohybu s cyklem druhým. Proto právě hodnota 0, která vyvolává cyklus pendlování, je nevyhovující, jelikož s nulou se špatně operuje (nedá se měnit znaménko, násobit atd. aniž by změnila svou hodnotu) a proto pro podprogram je vhodnější použít hodnotu -1, kdy po každém pohybu bude podprogram měnit této hodnotě znaménko. Původní hodnota 0 bude přiřazena technologii obrábění z bodu $B_1$ do bodu $B_2$. Tak bude možno nalézt elegantní jedno-příkazovou podmínku pro všechny tři strategie, a to zda zmiňované hodnoty nebudou větší než 0. Takže hodnoty parametru pro jednotlivé strategie budou 1 pro obrábění zdola, 0 pro obrábění shora a -1 pro kývání. Hodnotou tohoto parametru se před vykonáním dráhy vynásobí hodnota uložená v dalším zvláštním parametru Q90. S tímto parametrem se provádí vyhodnocení v rozhodovacím bloku, zda je jeho hodnota nekladná. Jeho výchozí hodnota je 1. Po libovolném počtu vynásobení se získá taková hodnota, která v případě obrábění v jednom směru bude i nadále stále splňovat (nebo nesplňovat) podmínku jako před násobením. Opakovaným násobením jedničkou totiž se hodnota parametru Q90 nemění a již po prvním vynásobení nulou nabude

3 Slovo operace se zde chápe nikoli ve vztahu k technologickému postupu, ale ve vztahu k výpočetní technice a vykonávání programových příkazů.

nulové hodnoty. Při opakovaném násobení hodnotou -1 bude ovšem parametr Q90 střídavě nabývat hodnoty 1 a -1. V případě strategie kývání se tedy po každém vynásobení bude střídat splnění a nesplnění podmínky, což bude mít za následek střídavé vyvolávání dráhy v jednom a v druhém směru.

Při zpětném pohybu pendlování, který je stejný jako volba technologie z bodu $B_2$ do bodu $B_1$ (jejíž hodnota je 1), se cyklu pendlování změní znaménko. Hodnota parametru Q90 bude mít tedy hodnotu 1 totožnou právě s touto technologii, a tím je pro tento pohyb také zaručen stejný vyvolávací podnět.

Podprogram bude provádět pohyb z bodu $B_1$ do bodu $B_2$ v případě, kdy nebude hodnota parametru větší než nula; tedy v případě, že bude zvolena technologie obrábění z bodu $B_1$ do bodu $B_2$, přičemž je hodnota parametru 0, anebo při pendlování, kdy v první třísce bude hodnota -1. Pokud bude hodnota parametru větší než 0, podprogram bude provádět pohyb z bodu $B_2$ do bodu $B_1$. To splňuje právě cyklus v bodu $B_2$ do bodu $B_1$, kdy je hodnota parametru 1, a cyklus pendlování, kdy tento cyklu bude mít opačné znaménko, tedy také 1.

Je tedy potřeba aby se hodnoty zaměnily následovně:

$$-1 \rightarrow 0 \qquad 0 \rightarrow -1 \qquad 1 \rightarrow 1$$

Potřebujeme vlastně funkci, která při zadání vstupních hodnot -1, resp. 0, resp. 1 poskytne požadované výstupní hodnoty 0, resp. -1, resp. 1. Je to tedy funkce procházející body [-1, 0], [0, -1], [1, 1]. Jednou z možných funkcí je parabola, jejíž předpis udává kvadratická rovnice. Na to, abychom stanovili konkrétní hodnoty jejích tří koeficientů (konstantního, lineárního a kvadratického), vyřešíme trojici kvadratických rovnic, která vystihuje požadavek transformace parametru Q215 [3].

$$y_{\{-1\}} = k\,x^2 + l\,x + c = 0 \tag{20}$$

$$y_{\{0\}} = k\,x^2 + l\,x + c = -1 \tag{21}$$

$$y_{\{1\}} = k\,x^2 + l\,x + c = 1 \tag{22}$$

Po dosazení do rovnic (20), (21) a (22) dostaneme tři rovnice o třech neznámých.

$$k - l + c = 0 \tag{23}$$

$$c = -1 \tag{24}$$

$$k + l + c = 1 \tag{25}$$

Jejich vyřešením získáme hodnoty tří koeficientů $k$, $l$, $c$ a výsledný tvar transformační rovnice (26).

$$y=\frac{3}{2}x^2+\frac{1}{2}x-1 \tag{26}$$

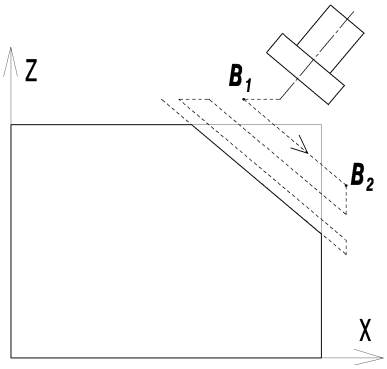
Proměnnou *x* v rovnici nahradí Q-parametr, a tak získáme rovnici, kterou následně použijeme v podprogramu.

$$Q80=\frac{3}{2}Q215^2+\frac{1}{2}Q215-1 \tag{27}$$

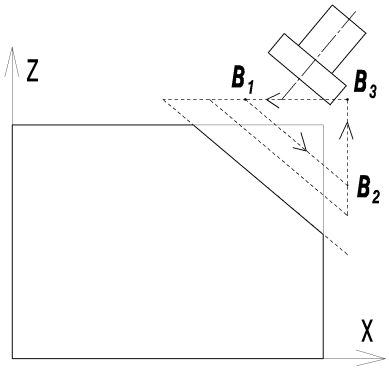
Tuto transformaci lze opět provést několika způsoby. Pokud bychom nechtěli z nějakého důvodu řešit rovnici paraboly, jde to provést pomocí skoků v programu. Nicméně tato metoda je přehlednější a v programu zabírá pouze jeden řádek.

V rovnici (27) je vidět, že hodnota parametru Q215 se nemění, ale uloží do nového parametru Q80. Pokud by tomu tak nebylo a parametr Q215 by se přeuložil na tvrdo, tak při opětovném vyvolání podprogramu, například posunutý v ose *y*, by podprogram pracoval již s touto změněnou hodnotou, která by se ještě jednou transformovala a podprogram by nepracoval správně.

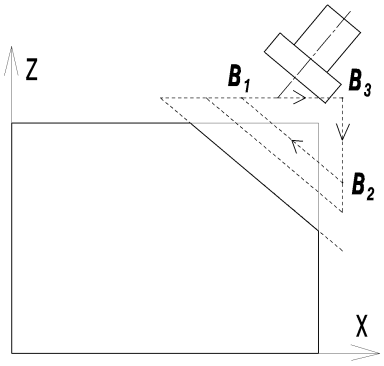
Nyní řídicí systém pro vytváření podprogramu bude pracovat s volbou technologie podle obrázků (16), (17) a (18), a to pouze s parametrem Q80. Ještě jednou je potřeba zopakovat že, tento parametr si podprogram sám vypočte a uživatel ho nijak nedefinuje, ten pracuje pouze s parametrem Q215 a volí technologii podle obrázků (9), (10) a (11).




*Obrázek. 16: Q80 = –1*




*Obrázek. 17: Q80 = 0*




*Obrázek. 18: Q80 = 1*

Po provedení transformace Q-parametru podprogram přiřadí parametru Q90 hodnotu 1, jak je vidět na obrázku (Obr. 14). Tento vnitřní parametr bude sloužit jako nástroj, který bude rozlišovat, v jakém směru obrábění se má nástroj pohybovat, a také bude sloužit jako nástroj pro změnu směru obrábění při obrábění v pendlování. Parametr se bude dále v cyklu násobit s parametrem Q80, který určuje, jaká je zvolena technologie. Výsledná hodnota se po násobení uloží zpět do parametru Q90, jak je vidět v rovnici (28). Podprogram bude následně při rozhodování o směru obrábění pracovat s parametrem Q90.

Při prvním projetí této rovnice (28) se nijak nezmění směr obrábění, jelikož v hodnotě je uložena jednička, výsledná hodnota bude tedy buďto 1 nebo 0. Tyto hodnoty určují obrábění v jednom nebo v druhém směru, a při dalším a následujícím projetí této rovnice se parametr Q90 nezmění, bude tedy 1 nebo 0, podle toho jaká technologie bude zvolena na začátku podprogramu.

Pokud bude zvolena volba technologie obrábění v pendlování, kde hodnota parametru Q80 je -1, tak při prvním projetí rovnice se do parametru Q90 uloží -1. Ta zaručí obrábění v první hloubce přísuvu z bodu $B_1$ do bodu $B_2$. Při opětovném vyvolání cyklu, kde na začátku je rovnice (28), bude podprogram násobit dvě -1. Výsledná hodnota parametru Q90 je tedy 1, změnila tedy původní hodnotu, a tím určila, že nástroj při pendlování jede opačným směrem. Při každém dalším projetí této rovnice se bude tedy v hodnotě Q90 měnit hodnota z -1 na 1, a tím se bude při pendlování měnit i směr obrábění.

$$Q90 = Q90 \cdot Q80 \tag{28}$$

Rovnice (28) bude do podprogramu vložena až na začátku samotného cyklického jádra podprogramu. Zde je pouze přiřazená hodnota parametru Q90, jak je vidět na obrázku (14). Rovnice je zde vysvětlena z důvodu objasnění parametru Q90.

Dalším krokem v podprogramu je vložit do vnitřního parametru Q99, který slouží pro výpočet aktuální hloubky zkosení, hodnotu nula (rovnice 29). Jelikož se podprogram nachází na začátku a nástroj ještě neprovedl žádný úkos, proto je hodnota nula.

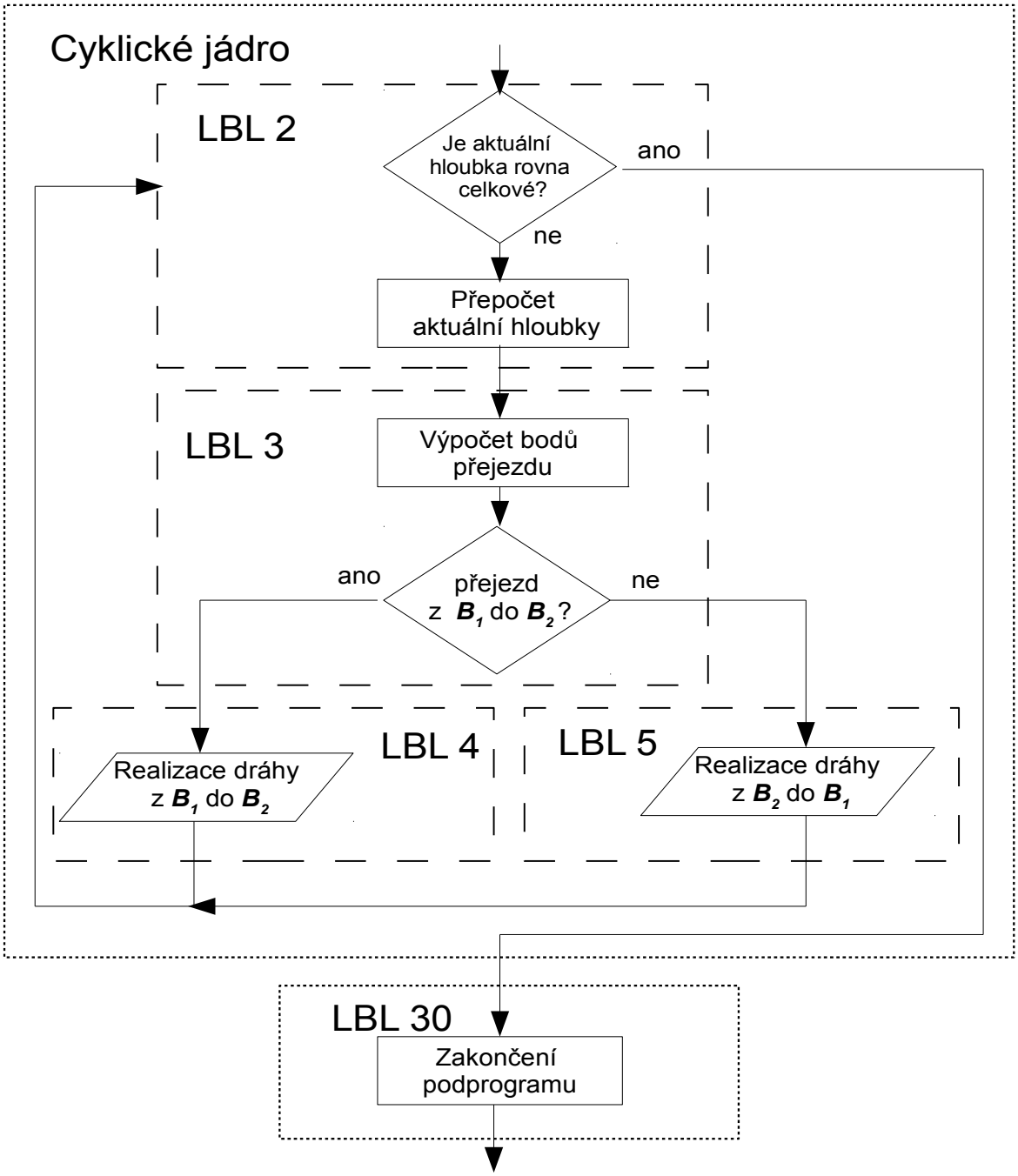
$$Q99 = 0 \tag{29}$$

Následně jsou v programu vyvolány rovnice (11) a (12), které vypočtou nové souřadnice po natočení souřadného systému, znázorněné na obrázku (Obr. 6). Také rovnice (18) a (19), které vypočtou souřadnice přejezdu, které jsou vidět na obrázku (Obr. 12). Všechny tyto rovnice jsou závislé pouze na vnějších souřadnicích a úhlu natočení, nejsou závislé na aktuální hloubce ani na způsobu technologie, jakým bude podprogram obrábět. Jsou pro zbytek podprogramu konstantami.

## 6.2 Cyklické jádro podprogramu

Doposud má podprogram natočený souřadný systém, vypočtený bod nájezdu a bod přejezdu. Tyto kroky, jak již bylo řečeno, jsou konstantami. Následovat budou jednotlivé pohyby nástroje, které mají charakter cyklu, budou se v podprogramu opakovaně vyvolávat až do doby, dokud nebude součást plně obrobena. Tyto pohyby pracují s body, které již nejsou konstantami a jsou vždy závislé na aktuální hloubce zkosení. Před vyvolání pohybu, respektive příslušné rutiny, která obsahuje jednotlivé pohyby, je nutné aby podprogram měl vždy aktuální souřadnice jeho koncového bodu vztažené právě k aktuální hloubce zkosení.

Na obrázku (Obr. 19) je vývojový diagram, na kterém je vidět samotné cyklické jádro podprogramu, kde se systém vrací na začátek na návěstí LBL 2. Tento vývojový diagram volně navazuje na vývojový diagram na obrázku (Obr. 14). Podprogram tedy při prvním kroku najede volně na návěstí LBL 2 a kroky, které následují za tímto návěstím, se budou opakovat až do doby, dokud nebude obrobek plně obroben.




*Obrázek. 19: Jádro podprogramu*

Cyklické jádro dále pracuje s pomocí jednotlivých dílčích rutin. Tyto rutiny mají své návěští označené jako LBL, na které se v průběhu cyklického jádra podprogram odkazuje či jen na ně podprogram volně navazuje. V nich se dále provádí jednotlivé úkony. Tyto rutiny nepracují jako jednotlicé samostatné podprogramy. Návěstí zde má význam pouze jako řádek, na který se v cyklickém jádru podprogram odkazuje a provádí úkony psané v následujícím řádku.

Jednotlivé příkazy vývojového diagramu a podprogramu samotného budou vysvětleny v následující kapitole, kde bude diagram i detailněji rozkreslen.

### 6.2.1 Řízení aktuální hloubky

Aktuální hloubka se řídí prostřednictvím rutiny pod návěštím LBL 2. V této samostatné rutině, která je vidět na obrázku (Obr. 20), je potřeba nadefinovat, v jaké fázi obrábění se podprogram nachází, jaká je aktuální hloubka zkosení a zda již není materiál plně odebrán.

Pokud bude materiál plně odebrán, bude se aktuální hloubka zkosení rovnat celkové hloubce zkosení. Právě tato podmínka je na začátku rutiny LBL 2. Pokud bude tedy tato podmínka splněna, aktuální hloubka zkosení se bude rovnat té celkové, nastane ukončení tohoto podprogramu. Za ukončení podprogramu je myšleno ukončení cyklického jádra podprogramu, ve kterém se provádí obrábění. Toto ukončení je provedeno vyvoláním rutiny LB 30. Vyvolání LBL 30 není ukončení celkového podprogramu, ale pouze podmíněný skok na toto návěstí, které je umístěno za cyklickým jádrem podprogramu. V této rutině se budou provádět ještě nějaké úkony, které budou vysvětleny v následujících kapitolách.

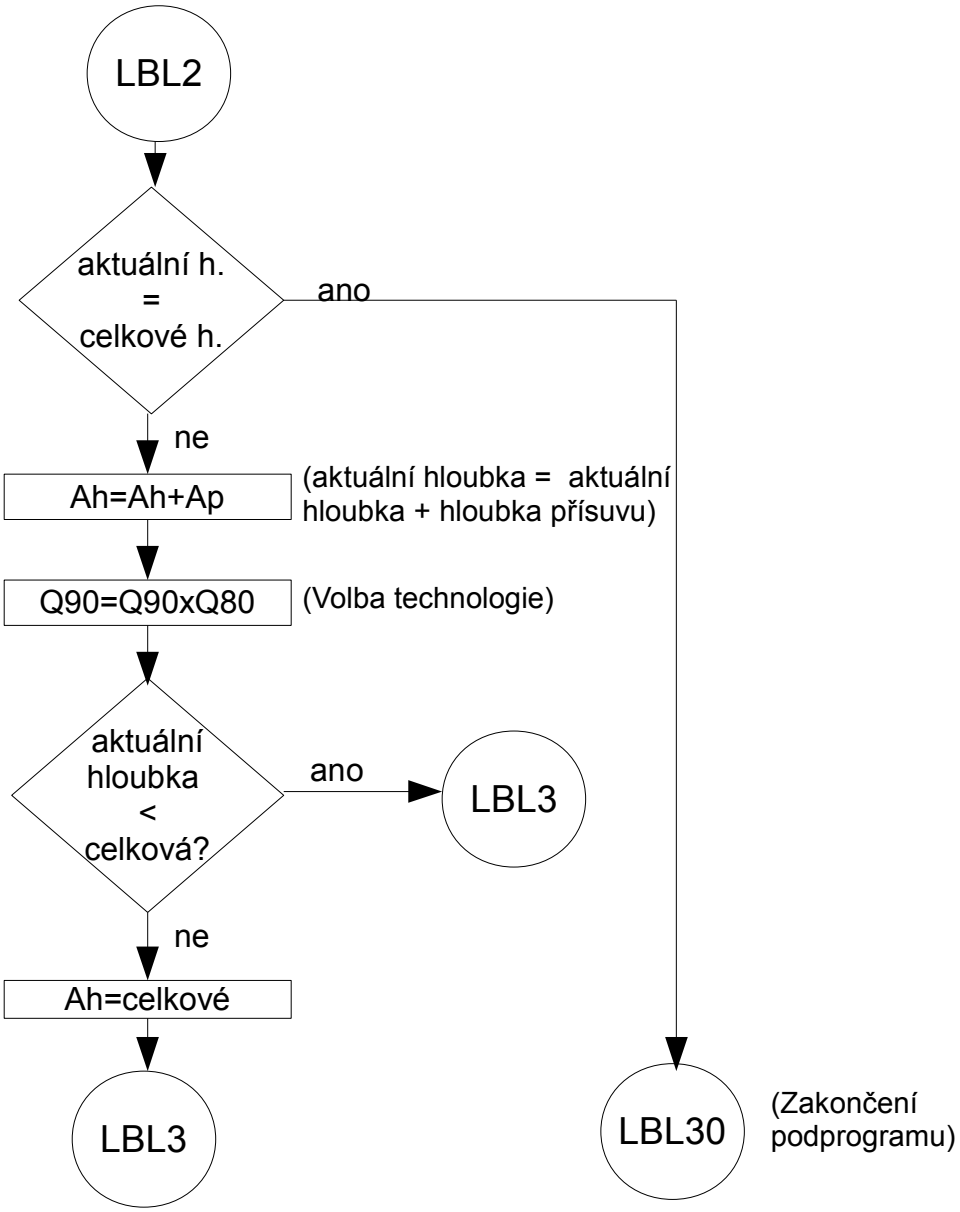
Pokud není splněna podmínka, je tedy aktuální hloubka zkosení menší a je zapotřebí její hodnotu správně definovat. To je provedeno rovnicí (31), která vypočte aktuální hloubku zkosení. Aktuální hloubka zkosení se tedy rovná aktuální hloubka zkosení plus hloubka přísuvu. Při prvním najetí cyklického jádra, kdy je aktuální hloubka rovna nule, která jí byla přiřazena v úvodu podprogramu, se k této nule přičte hloubka přísuvu. Výsledná hodnota se přeuloží zpět do parametru Q99, tedy aktuální hloubce zkosení. Výsledná hodnota bude tedy rovna hloubce přísuvu. Při dalším projetí tohoto cyklického jádra, kdy už aktuální hloubka bude rovna hloubce přísuvu a k ní se přičte opět hloubka přísuvu, bude výslední hodnota dvě hloubky přísuvu. Takto se bude následně přičítat při každém dalším projetí cyklu, respektive při každém projetí cyklického jádra programu.

$$Q99 = Q99 + Q202 \tag{30}$$

V následující rovnici je násoben parametr Q80 (rovnice 28), která udává volbu technologie pomocným parametrem Q90. Výsledek této rovnice bude použit v následujícím cyklu a bude určovat, jaký pohyb má nástroj realizovat, a tím do jakého bodu má nástroj jet. Tato rovnice byla vysvětlena v kapitole 6.1.1 (Transformace parametru Q215) a její použití dále v textu. Časová posloupnost tohoto příkazu není nijak závislá, musí být pouze uložena v této rutině LBL 2.

Nyní je potřeba předejít tomu, aby podprogram součást pod-frézoval. Například pokud by celkové zkosení bylo 10mm a hloubka přísuvu byla 4mm, tak při třetím projetí této rutiny a rovnice (28) by hodnota aktuální hloubky byla 12mm. Nástroj by tedy součást o 2mm pod-frézoval. Tomuto se zabrání podmínkou, kde se podprogram ptá, zda je aktuální hloubka menší než celková. Pokud nebude podmínka splněna, jako je v tomto případě, přiřadí se parametru aktuální hloubky Q99 hodnota celkové hloubky obrábění Q201 podle rovnice (31). V tomto příkladu tedy 10mm. Pokud podmínka splněna bude, nástroj pojede například druhý úkos, hodnota bude tedy 8mm, tato rovnice se přeskočí (31) a podprogram bude dál pracovat s hodnou 8mm.

$$Q99 = Q201 \tag{31}$$




*Obrázek. 20: LBL 2*

Přeskočení rovnice, tedy v případě kdy bude podmínka splněna, je provedeno pomocí podmíněného skoku na návěstní LBL 3, jehož tělo je umístěno za tělem rutiny LBL 2. Podprogram tedy, aniž realizuje přiřazení hloubky podle rovnice, přeskočí na další rutinu. Posledním úkonem v LBL 2, jak je také vidět na obrázku (Obr. 20), je právě rovnice (31). Za touto rovnicí se LBL 2 ukončí a jelikož nebyl vyvolán příkazem CALL LBL, není tedy ukončen ani příkazem LBL 0, nevrátí se do místa, kde byl vyvolán, a načte se následující řádek v podprogramu, jak již bylo vysvětleno dříve. Následující řádek je právě návěstí s rutinou LBL 3. Tím se tedy provede přeskočení rovnice (31).

Pokud podmínka splněna nebude, podprogram po zbytek cyklického jádra programu bude pracovat s hodnotou aktuální hloubky rovnou celkové hloubce zkosení. Pojede tedy svou poslední třísku. Po opětovném vrácení na začátek cyklického jádra na návěstí s rutinou LBL 2, kde v této rutině bude splněna podmínka o rovnosti hloubek zkosení, a právě tehdy bude cyklické jádro ukončeno a vyvolána rutina LBL 30.

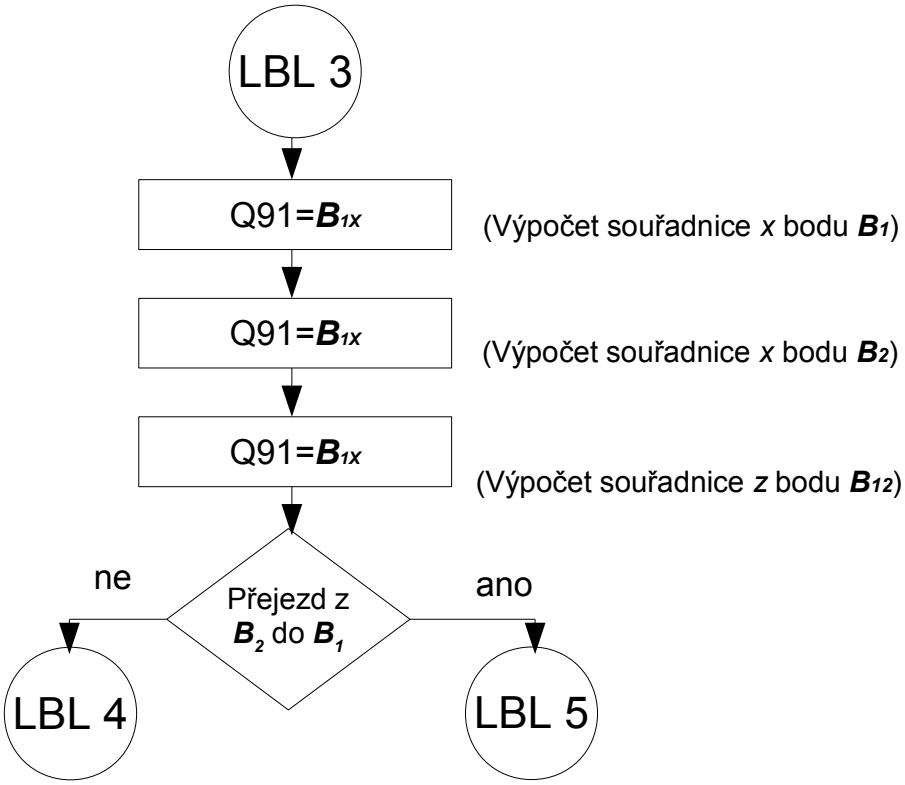
Rutina s návěstím LBL 2 má tedy tři výstupy. Proto také tuto rutinu není možné chápat jako podprogram. Podprogram by totiž měl být vůči zbytku programu uzavřený a měl by mít jeden vstup a jeden výstup. Po výstupu z podprogramu by pak měl proces pokračovat dál od místa, na kterém byl podprogram vyvolán. Části programu, které nejsou vyvolávány, ale na které běh programu přeskakuje podmíněně nebo bez podmínky se zde proto nazývají rutiny. V rutině LBL 2 je výstup na LBL 3, s případnou možností přeskočení závěrečné úpravy hloubky při splnění podmínky, že ještě nebyla dosažena celková hloubka. Mimoto je také možný výstup rovnou na závěrečnou část programu na návěstí LBL 30. Také z tohoto důvodu nejsou ve vývojovém diagramu tato návěstí graficky označena jako podprogram. Tento styl psaní programu zde není použit z vůle autora, ale je vynucen skutečností, že prostředky jazyka Heidenhain nedovolují podmíněné volání s návratem a tudíž nedovolují strukturované programování.

### 6.2.2 Souřadnice přejezdu

Souřadnice přejezdu se počítají v samostatné rutině pod návěštím LBL 3 (Obr. 21). Používání ucelených dílčích částí programu označených návěštím, které jsou v podprogramu umístěny na zvláštním místě, celý program zpřehledňuje. Jejich výhoda je i v tom, že jednotlivé dílčí problémy je při odlaďování programu možno řešit do značné míry odděleně. Pokud je například při odlaďování programu chybné řízení hloubky řezu, není nutno hledat chybu v celém programu, ale stačí se zaměřit na poměrně krátký úsek kódu. Rutina pod návěštím LBL 3 vypočte souřadnice přejezdu bodů $\boldsymbol{B_1}$ a $\boldsymbol{B_2}$ obrázek (Obr. 7), které budou již vztaženy k aktuální hloubce a které jsou v rámci pootočeného souřadného systému.

Body přejezdů budou vypočteny pomocí již výše uvedených rovnic (13), (14) a (15). Výpočet těchto souřadnic je řešen jako samostatná rutina pouze z důvodu, aby před tímto výpočtem bylo návěstí. Na toto návěstí je pak možné se v rutině LBL 2 odkázat, a tím přeskočit rovnici (31) o přiřazení aktuální hloubky zkosení celkové hloubky zkosení. Bylo vysvětleno v předchozí kapitole.

Rutina LBL 3 má dva výstupy. To, jaký výstup bude v rutině realizován, je závislé na výsledku vyhodnocení podmínky v rozhodovacím bloku, který určuje, jaká je zvolena technologie obrábění. Na zvolené technologii je závislá dráha nástroje. Tato dráha bude realizována v následujících rutinách. Rozhodování, jaká bude zvolena rutina, bude vysvětleno v následující samostatné kapitole.

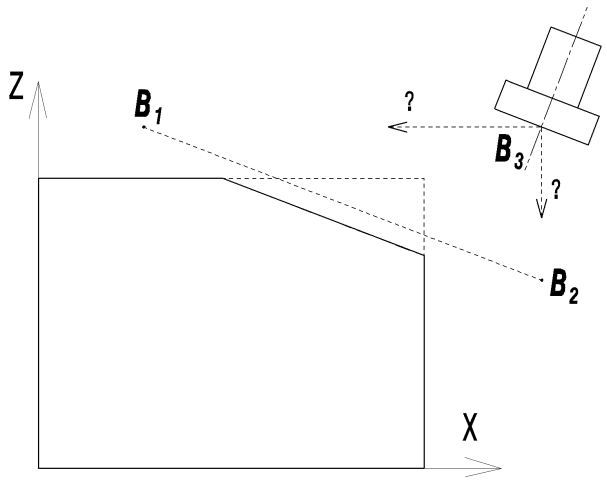



*Obrázek. 21: LBL 3*

### 6.2.3 Volba přejezdu

Program má nyní vypočteny všechny souřadnice potřebné k obrábění vztažené již k aktuální hloubce zkosení a je potřeba zadat, do jakého bodu má nástroj odjet. To, do jakého bodu nástroj pojede, je určeno tím, jakou technologií obrábění si programátor zvolil na začátku podprogramu. Právě do jakého bodu nástroj pojede, bude řešit následující příkaz.

Jak je vidět na obrázku (22), nástroj se nyní nachází v bodě $B_3$, a je tedy potřeba určit, zda pojede do bodu $B_1$ nebo do bodu $B_2$.




*Obrázek. 22: Volba přejezdu*

K určení, na jaký bod má nástroj jet, bude sloužit pomocný parametr Q90. Tento parametr je násoben parametrem Q80, ve kterém je uložena informace o způsobu obrábění. Tato rovnice byla vysvětlena v předchozích kapitolách. Pro zopakování je dobré připomenout jaké hodnoty může nabývat parametr Q80.

| | |
|---|---|
| Q80 = -1 | Obrábění v pendlování |
| Q80 = 0 | Obrábění z bodu $B_1$ do bodu $B_2$ |
| Q80 = 1 | Obrábění z bodu $B_2$ do bodu $B_1$ |

Parametr Q80 je tedy násoben parametrem Q90, ve kterém je uložena hodnota 1. Tato hodnota byla přidělena na úvodu podprogramu. Výsledná hodnota je přeuložena opět do parametru Q90. Výsledná hodnota tohoto parametru v závislosti na počtu odjetých cyklů, které podprogram projel, je zobrazena v tabulce(2).

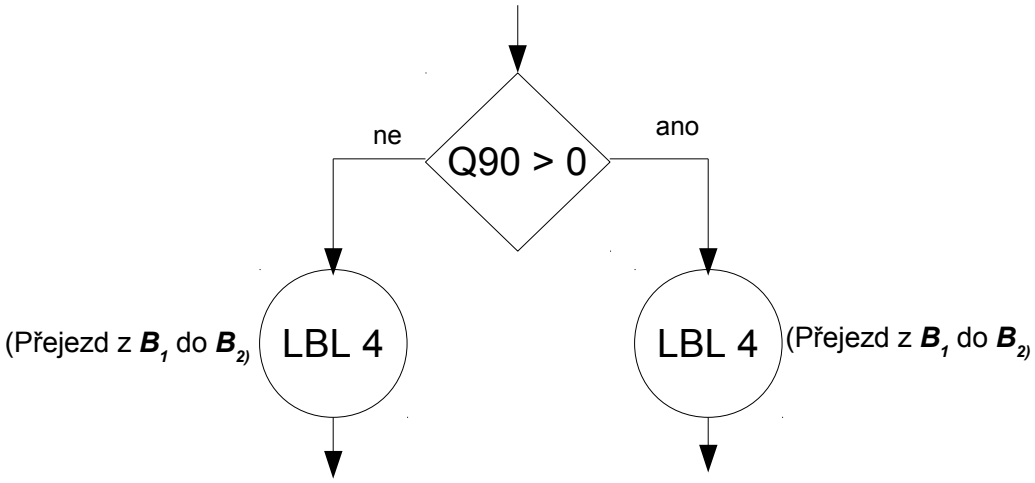
| Technologie | Rovnice | I cyklus | | II cyklus | | III cyklus | |
|---|---|---|---|---|---|---|---|
| | | dosazení | **Q90** | dosazení | **Q90** | dosazení | **Q90** |
| Q80=-1 | Q80xQ90=Q90 | -1x1=-1 | **-1** | -1x-1=1 | **1** | 1x1=-1 | **-1** |
| Q80=0 | Q80xQ90=Q90 | 0x1=0 | **0** | 0x0=0 | **0** | 0x0=0 | **0** |
| Q80=1 | Q80xQ90=Q90 | 1x1=1 | **1** | 1x1=1 | **1** | 1x1=1 | **1** |

*Tabulka 2: Hodnota parametru Q90*

Výsledná hodnota parametru Q90 pří prvním vynásobení, tedy na začátku cyklického jádra podprogramu, je stejná jako hodnota parametru Q80, ve kterém je uložena volba technologie obrábění. Pokud bude zvolena technologie obrábění, kde se bude obrábět pracovním pohybem pouze v jednom nebo druhém směru, výsledná hodnota se již měnit nebude, jak je vidět v tabulce na druhém a třetím řádku. Hodnota parametru bude stále 1 nebo 0 nezávisle na počtu opakování..

Pokud bude zvolena technologie obrábění v pendlování, tak ve druhém a každém dalším projetí rovnice, tedy celého cyklického jádra programu, bude hodnota parametru Q90 měnit svou hodnotu. Měnit se bude pouze znaménko hodnoty. Jak je vidět tučným písmem v prvním řádku tabulky.

Nyní přijde na řadu rozhodovací mechanizmus (Obr. 23), který bude na základě parametru Q90 rozhodovat, do jakého bodu nástroj odjede. Na základě rozhodnutí, do jakého bodu nástroj pojede, podprogram vyvolá samostatnou rutinu s návěstím LBL 4 nebo LBL 5, ve které jsou jednotlivé pohyby uloženy.




*Obrázek. 23: Rozhodovací mechanizmus*

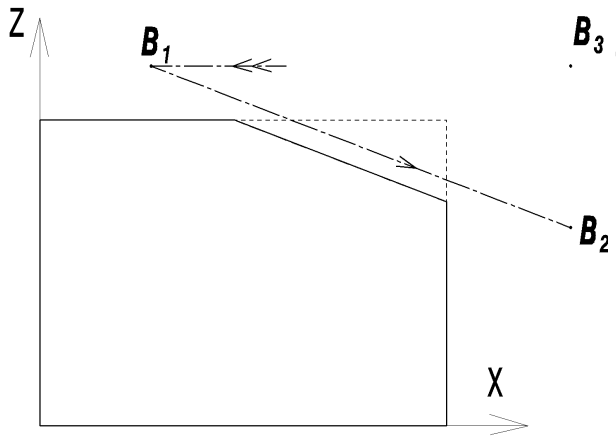
Nástroj tedy pojede do bodu $B_1$ nebo do bodu $B_2$, jak je vidět na obrázku (Obr. 22). Do bodu $B_1$, pojede v případě, kdy je zvolena technologie přejezdu z tohoto bodu nebo při první a následně každým lichým najetím při pendlování. Hodnoty parametru Q90 jsou u těchto technologií -1 a 0. Společný vyvolávací podnět těchto dvou čísel je, že nejsou větší než 0. Zároveň tento podnět také nesmí vyloučit hodnotu 1, která je pro opačný pohyb nástroje. Příkaz se tedy ptá, pokud nejsou hodnoty parametru Q90 větší než 0, odskok na LBL 4. Tedy na pohyb z bodu $B_1$ do bodu $B_2$.

Pokud bude podmínka splněna a hodnota parametru bude větší než 0, program tedy odskočí na návěstí LBL 5. Zde je realizován pohyb z bodu $B_2$ do bodu $B_1$. Právě tento pohyb je realizován při volbě technologie mezi těmito body. Hodnota parametru Q90 je zde rovna 1, podmínka je tedy splněna. Tato podmínka bude také splněna, pokud bude zvolena technologie pendlování, kdy nástroj pojede svojí druhou třísku a následně každou další sudou. Hodnota v tomto případě je také rovna 1, jak je vidět v tabulce (2).

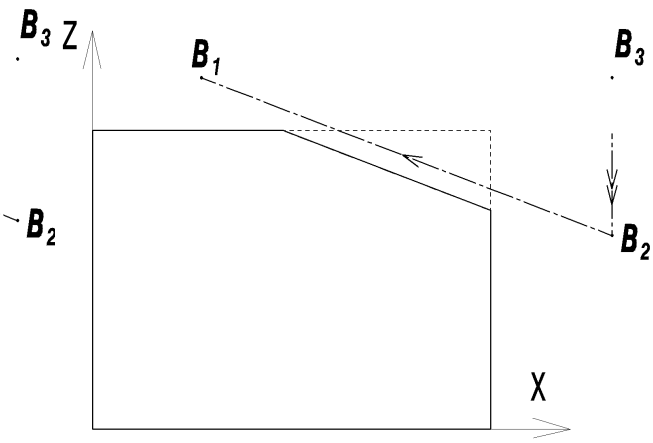
Pro upřesnění je dobré zmínit, že systém realizuje podmíněné skoky pouze v případě, kdy je některá podmínka splněna. Pokud tato podmínka splněna není, načítá se následující řádek. Systém neumí tedy odkázat program na jedno místo programu, pokud podmínka splněna je, a zároveň na jiné místo v programu, pokud podmínka splněna není. Vyřešit se to dá například následující podmínkou, která bude mít obrácený charakter. V případě této práce je za podmínkou v následujícím řádku samotné tělo rutiny LBL 4. Podprogram tedy buď odskočí na rutinu s návěstím LBL 5, pokud je splněna podmínka, nebo načte rutinu LBL 4, když splněna není. Na konci každé z rutin pro vykonání dráhy nástroje je nepodmíněný skok na začátek cyklického jádra – na návěští LBL 2.

### 6.2.4 Vlastní dráha nástroje

Když je učiněna volba směru dráhy nástroje, realizuje se tato dráha rutinou s návěstím LBL 4 nebo LBL 5 podle toho, který směr byl zvolen. Realizace této dráhy je zobrazena na obrázku (Obr. 24 a 25). Nástroj jede tedy rychloposuvem do jedno z bodů a následně pracovním pohybem do druhého bodu.




*Obrázek. 24: Dráha LBL 4*

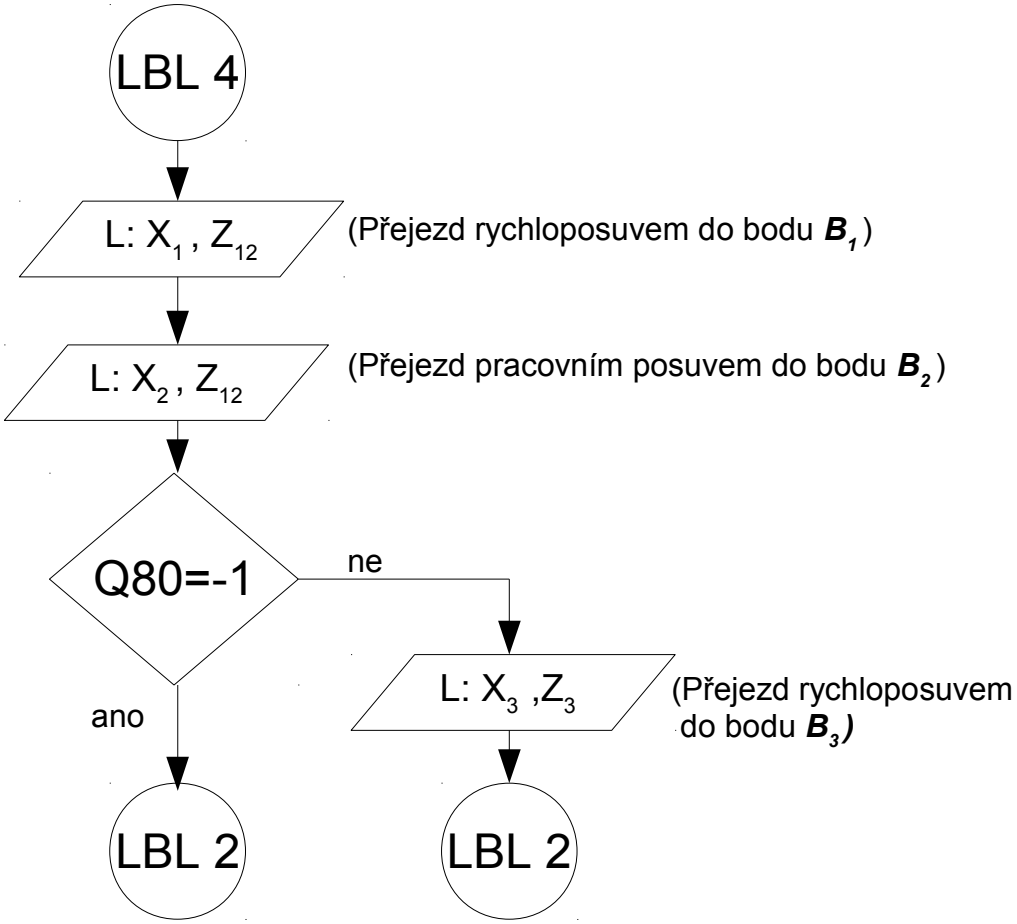



*Obrázek. 25: Dráha LBL 5*

V koncovém bodě tohoto pohybu, ať v bodě $B_2$ při pohybu pomocí rutiny LBL 4, nebo v bodě $B_1$ při pohybu pomocí rutiny LBL 5, přijde rozhodující cyklus, který musí určit, kam bude nástroj dále pokračovat. Tento rozhodující cyklus je zobrazen na vývojovém diagramu (Obr. 26), který znázorňuje pohyb pomocí rutiny LBL 4.

Rozhodnutí, na jaký bod nástroj pojede, je závislé na způsobu, jakým je součást obráběna. Především jde o to zjistit, zda je zvolena technologie v pendlování. Pokud by nebyla tato metoda zvolena, realizovala by následně tato rutina ještě pohyb do bodu $B_3$. Pokud by například byla zvolena volba technologie obrábění v jednom směru z bodu $B_1$ do bodu $B_2$. Pohyby by se realizovaly pomocí rutiny LBL4 , ve které by se prováděly následující pohyby. Pohyb rychloposuvem do bodu $B_1$, následně pracovním pohybem do bodu $B_2$ a dále by nástroj odjel rychloposuvem do bodu $B_3$. Poté by se rutina ukončila a program by se vrátil na začátek cyklického jádra podprogramu na návěstí LBL 2. Tyto pohyby by se opakovaly, včetně projetí celého cyklického jádra se všemi potřebnými výpočty do doby, než by byla součást plně obrobena.

Stejně tomu bude i v případě, pokud bude zvolena volba technologie obrábění v druhém směru pomocí rutiny LBL 5.




*Obrázek. 26: LBL 4*

Pokud by ovšem byla zvolena volba technologie v pendlování, nástroj by po projetí pracovním pohybem mezi body $B_1$ a $B_2$ zůstal na koncovém bodě tohoto pohybu. Rutina by tedy nerealizovala pohyb do bodu $B_3$. Pokud tedy podprogram pojede například svou první třísku a bude zvolena právě technologie v pendlování, vyvolá se první pohyb pomocí rutiny LBL 4, který se skládá z rychloposuvu do bodu $B_1$ a následně v pracovním pohybu do bodu $B_2$ a v tomto bodě bude rutina ukončena. Zde by byl návrat na začátek cyklického jádra podprogramu, následné projetí tohoto jádra, kde by byla změněna hodnota parametru Q90, a tím vyvolán pohyb pomocí rutina LBL 5. Tato rutina by se opět skládala z pohybu rychloposuvem do bodu $B_2$. Začátek tohoto pohybu rychloposuvem by byl v původním koncovém bodě $B_2$, kde byla ukončena rutina LBL 4, a koncový bod tohoto rychloposuvu by byl nový bod $B_2$, který je jež posunut o hloubku přísuvu. Následně by nástroj pokračoval pracovním pohybem do bodu $B_1$, kde by opět rutina LBL 5 byla ukončena před pohybem do bodu $B_3$.

Rozhodovací mechanizmus se tedy ptá, zdali je zvolena technologie obrábění v pendlování. Tomu bude v případě, kdy se parametr Q80=-1. Pokud je podmínka splněna, nástroj tedy nebude provádět pohyb rychloposuvem do bodu $B_3$ a následně bude podprogram nasměrován na návěstí LBL 2. Pokud podmínka nebude splněna, následují blok bude obsahovat příkaz přejezd rychloposuvem do bodu $B_3$. Až následně bude tato rutina ukončena a program odkázán na návěstí LBL 2. Toto odkázání v programu je realizováno pomocí rozhodovacího

mechanizmu, kde je již předem známý výsledek. Jelikož systém neumí tyto skoky realizovat, pokud není splněna nějaká podmínka. V tomto podprogramu je podmínka pokud se 1=1 odskok na návěstí LBL 2.

Nepodmíněný skok jde realizovat také pomocí funkce CALL LBL, jako je tomu například na začátku podprogramu při vyvolávání rutiny LBL 1, kde je provedeno natočení souřadného systému. Při vyvolávání pomocí této funkce musí být na konci příslušné rutiny blok s LBL 0. Tento blok vrátí podprogram na místo vyvolání. Na konci LBL 4 a LBL 5, kde je vyvolána rutina LBL 2, není žádoucí aby se program na toto místo vracel. Proto je zde tato předem splněná podmínka.

### 6.2.5 Ukončení cyklického jádra podprogramu

V samotném cyklickém jádře se provádí veškeré operace, které se cyklicky opakují v závislosti na aktuální hloubce zkosení. Také tedy obrábění. Obrobení je ukončeno, když je splněna podmínka v rutině LBL 2, tedy aktuální hloubka je rovna té celkové a v podprogramu je vyvolána rutina LBL 30. Ukončením se rozumí právě ukončení cyklického jádra, které bylo vysvětleno v podkapitole 6.2 (Cyklické jádro podprogramu).

## *6.3 Ukončení podprogramu*

Ukončení podprogramu, je proces, který nastane, pokud se ukončí cyklické jádro programu, materiál je plně obroben a budou následovat kroky, které jsou potřebné k bezpečnému ukončení celého podprogram. Jde především o odjetí do bezpečnostní vzdálenosti, jelikož nástroj je momentálně někde na koncovém bodě jedné z rutiny, která řešila jednotlivé pohyby. Dále zpětné natočení souřadného systému a vřeteníku stroje.

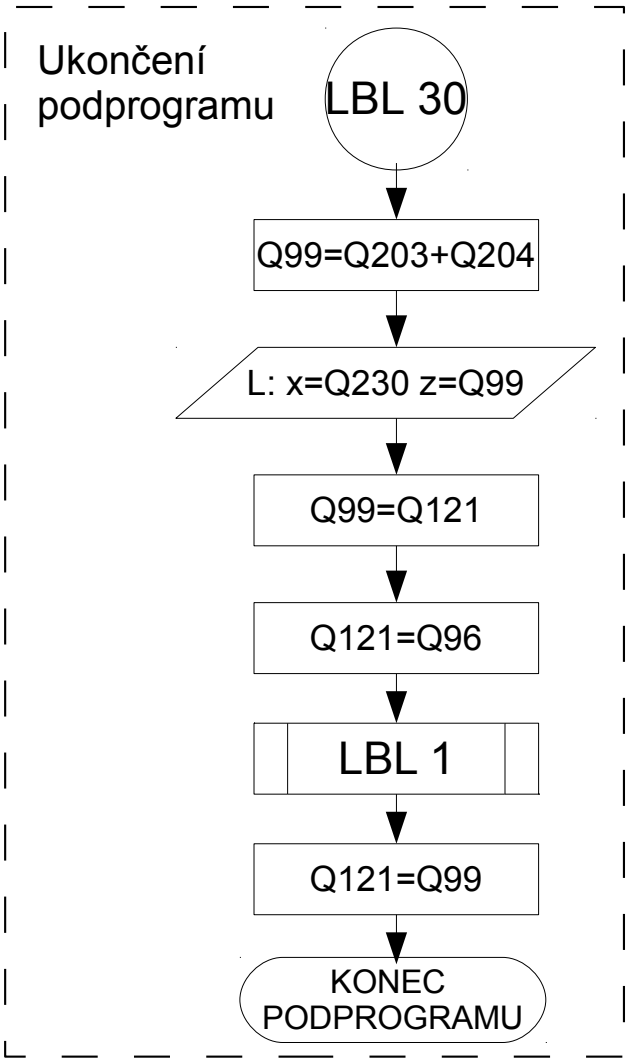
Rutina označena jako LBL 30 je umístěna před samotným koncem programu. V tomto cyklu a následně za ním není žádný podmíněný skok, který by podprogram vrátil na začátek nebo na nějaké místo vně podprogramu. Pokud se tedy program dostane do tohoto cyklu, je již zkosení hotové. Tato rutina musí být právě na samotném konci podprogramu, jelikož se podprogram musí vrátit do hlavního programu, kde byl vyvolán. Není tedy možné tento podprogram ukončit příkazem M30 ani M02 a je nutné nechat projít tento podprogram jeho přirozeným koncem.

V rutině LBL 30, která je vidět na obrázku (Obr. 27), se řeší tedy přejezd do bezpečnostní vzdálenosti, kde se provede natočení souřadného systému. Před samotným odjezdem do tohoto bodu je potřeba nadefinovat jeho souřadnice. Na začátku tohoto podprogramu byly vypočteny souřadnice startovacího bodu, v kterém se provádělo natočení souřadného systému. Tento bod byl ovšem nadefinován právě před natočení souřadného systému, měl tedy původní souřadnice. Nyní je tedy vhodné nadefinovat nový bezpečnostní bod, který pracuje s natočeným souřadným systémem. Hodnota souřadnice na ose $x$ je definována parametrem Q 230. Tento parametr obsahuje souřadnici povrchu polotovaru v ose $x$ (hodnotu po natočení). V této souřadnici není potřeba odjíždět daleko od materiálu. Dále od materiálu je potřeba hlavně v ose nástroje $z$. Tato souřadnice je vypočtena podle rovnice (32). Hodnota této souřadnice je rovna součtem parametru Q 203, který je roven souřadnicím povrchu polotovaru v ose $z$ a parametrem Q 204, který je definován jako vzdálenost startovacího bodu.

$$Q99=Q203+Q204 \tag{32}$$

Hodnota parametru je uložena do parametru Q 99, tento parametr sloužil v podprogramu jako schránka pro hodnotu aktuální hloubky zkosení. Jelikož je polotovar obroben, tuto hodnotu nebude podprogram již potřebovat, proto je možné jí přepsat. Tomuto parametru je také na začátku podprogramu přidělena správná hodnota, která je potřebná pro správnou funkci tohoto podprogramu, případné opětovné vyvolání bude tedy pracovat se správnou hodnotu.

Nástroj je nyní v bezpečnostní vzdálenosti a je možné provést naklopení souřadného systému a vřeteníku stroje do výchozí polohy, v které byl podprogram vyvolán. Zpětné natočení se provádí podle stejné rutiny v které bylo natočeno, tedy pomocí rutiny LBL 1. Tato rutina pracuje s parametrem Q 121, v kterém je uložen úhel zkosení hrany, tedy i úhel natočení. Před vyvoláním rutiny je potřeba parametru Q 121 přidělit hodnotu Q 96, v které byl na začátku podprogramu uložen aktuální úhel natočení. Rutina bude následně pracovat právě s touto hodnotou a natočí systém do výchozí hodnoty.




*Obrázek. 27: LBL30*

Parametr Q 121 se řadí mezi vnější parametry. Tyto parametry jsou definovány uživatelem, který používá tento podprogram. Není tedy možné tyto parametry přepisovat, jelikož uživatel může tento podprogram opětovně vyvolat v hlavním programu, aniž by tyto hodnoty znova definoval. Pokud by podprogram přepsal hodnotu Q 121 na jiný úhel, opětovné vyvolání tohoto podprogramu by pracovalo s touto změněnou hodnotou.

V podprogramu je tedy před změněním parametru Q 121 tato hodnota uložena do parametru Q99. Hodnotu v tomto parametru již podprogram nebude potřebovat, je tedy možné ji opět přeuložit. Až následně je hodnota v parametru Q 121 změněna na hodnotu z parametr Q 96. Po natočení souřadného systému podprogram vrátí parametru Q 121 původní hodnotu definovanou uživatelem, která je prozatím uschována v parametru Q 99.

Po natočení souřadného systému se program vrátí zpět do rutiny LBL 30, kde následně projde svým přirozeným koncem a bude ukončen. Program bude již pokračovat v hlavním programu, kde byl tento podprogram vyvolán.

# 7 Závěr

Účelem této práce bylo vytvořit podprogram na zkosení hrany vykloněným čelním nástrojem. Zkosení se provádí na základě vstupních informací definovaných uživatelem. Za vstupní informace jsou považovány poloha rohu součásti, kde bude zkosení provedeno, úhel zkosení a hloubka zkosení. Tyto informace bývají definovány výkresovou dokumentací. Uživatel také definuje další doplňující vstupní informace. Tyto informace slouží k doplnění technologických podmínek potřebných k obrábění součásti, jako je velikost najetí, hloubka přísuvu, způsob technologie.

Podprogram je plně parametrizován pomocí Q-parametrů. Podprogram je tedy univerzální pro jakýkoliv úhel a hloubku zkosení. Jednotlivé body potřebné k pohybu nástroje se počítají pomocí matematických funkcí. Tyto funkce jsou definovány na matematickém modelu, který je popsán v samostatné kapitole.

Podprogram pracuje pouze ve dvou osách. Předpokládá se velikost průměru nástroje větší než šířka obráběného materiálu. Pokud by tomu tak nebylo, je možnost vyvolat podprogram v hlavním programu vícekrát za sebou, vždy posunutý ve třetí ose.

Uživatel podprogramu má možnost vybrat si, jakou metodou bude součást obrábět. Nejúčinnější metoda je takzvané pendlování. Nástroj zde obrábí v obou směrech. Další možnost je obrábění vždy v jednom směru. Buďto shora směrem ke stolu obráběcího stroje nebo obráceně. Způsob obrábění volí uživatel pomocí příslušného Q-parametru.

Podprogram je vytvořen pouze pro jednu hranu, tedy pouze v jednom kvadrantu. Pokud je požadavek zkosit hranu také na jiné straně, je možnost polotovar upnout na vícekrát, nebo použít otočný stůl. Stroj, na kterém byl podprogram odladěn, umožňuje naklopení vřeteníku pouze ve směru, pro jaký je tato práce řešena. Tak se případně otevírá do budoucna možnost program rozšířit pro frézování zkosené plochy větších rozměrů. Předložená práce ale neměla takový požadavek v zadání.

Podprogram je odladěn v systému Heidenhain. Přílohou této práce je návod k podprogramu. Ten je vytvořen stejnou formou, jaká se používá v příručce systému Heidenhain pro vysvětlování ostatních cyklů. Spolu s návodem je v příloze umístěn i samotný podprogram.

Podprogram, který byl zadán jako téma Bakalářské práce, byl odladěn ve firmě Pilsentools s.r.o. na stroji Hermle U 1130.

# Použitá literatura

[1] Heidenhain iTNC 530 – uživatelská příručka.
[2] JANDEČKA, K., ČESÁNEK, J., KOŽMÍN, P.: *Programování NC strojů.* Plzeň : ZČU, 2000.
[3] RERKTORYS K.: *Přehled užité matematiky.* Praha : SNTL, 1981.
[4] SOVA, F.: *Technologie obrábění a montáže.* Plzeň : ZČU, 2001.
[5] STANĚK, J., NĚMEJC, J.: *Metodika zpracování a úprava diplomových prací.* Plzeň : ZČU, 2005.
[6] VRABEC, M., MÁDL, J.: *NC programování v obrábění.* Praha : ČVUT, 2004.

# Seznam použitých obrázků

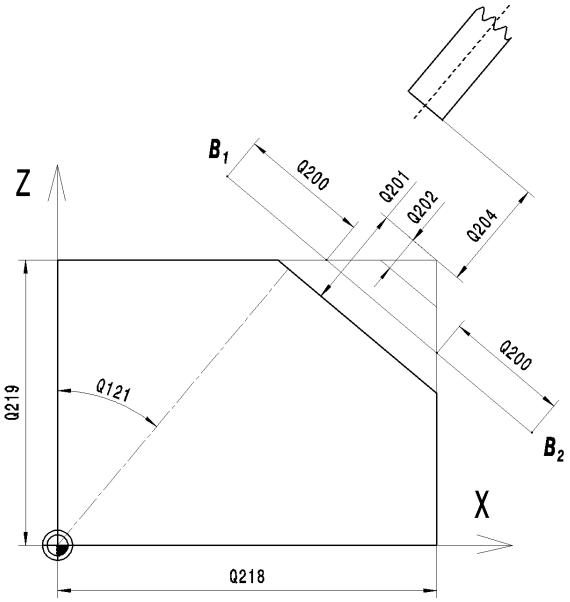
# Seznam příloh

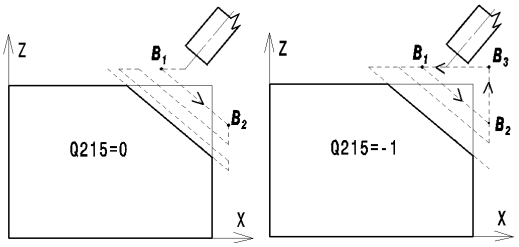
# PŘÍLOHA č. 1

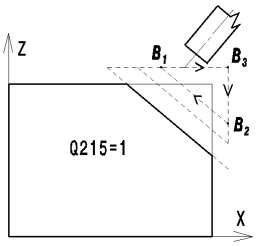
## Návod k podprogramu pro zkosení hrany

# Zkosení hrany

1. Nástroj najede rychloposuvem do výchozího bodu a pootočí se o zvolený úhel okolo osy Y.
2. Rychloposuvem se nástroj napolohuje do jednoho z bodů $B_1$ a $B_2$.
3. Pracovním pohybem přejede nástroj do bodu $B_2$ respektive $B_1$ (ten druhý, než do kterého napolohoval)
4. Pokud není zadána technologie „pendl“, přesune se nástroj rychloposuvem do bodu $B_3$ a pak ještě do bodu $B_1$ , respektive $B_2$ jako při první dráze, ale o hloubku přísuvu níže.
5. Pokud je zadána technologie „pendl“, přesune se nástroj z bodu dosaženého pracovním pohybem do nového bodu stejného označení, ale umístěného o hloubku přísuvu níže. Přesun se provede rychloposuvem.
6. Kroky 4 až 6 se opakují, dokud není dosaženo celkové hloubky obrábění
7. Nakonec nástroj přejede rychloposuvem do výchozího bodu a dojde k obnovení původního natočení.




Před programováním dejte pozor na zadání dostatečné vzdálenosti výchozího bodu, v tomto bodě bude provedeno natočení vřeteníku stroje.

- **Bezpečnostní vzdálenost Q200** (inkrementálně): vzdálenost nástroje – povrch obrobku
- **Hloubka zkosení Q201** (inkrementálně): vzdálenost roh obrobku – dno zkosení
- **Hloubka přísuvu Q202** (inkrementálně): rozměr, o nějž se nástroj vždy přisune; zadejte hodnotu větší než 0
- **Vnější souřadnice Q218 a 219** (inkrementálně): souřadnice rohu zkosení
- **Úhel zkosení Q121** (absolutně): Úhel, o který se natočí nástroj
- **Výchozí bod Q204** (inkrementálně): Vzdálenost od rohu součásti – nástroje. V tomto bodě bude natočení nástroje
- **Volba technologie Q215 –** způsob obrábění

Podprogram pro zkosení rohu není možné vyvolávat stejně jako cykly vestavěné v systému. Je však možno použít cyklus 12 – vyvolání podprogramu. Podprogram musí být před vyvoláním umístěn buď ve stejném adresáři s programem, ze kterého je vyvolán a nebo se musí jako adresa zadat celá cesta k němu.

| | |
|---|---|
| **CYCL DEF 12.0 PGM CALL** | |
| **CYCL DEF 12.1 TNC:\ZKOSENI** | |
| **Q200=20** | **Bezpečnostní vzdálenost** |
| **Q201=10** | **Hloubka zkosení** |
| **Q202=4** | **Hloubka přísuvu** |
| **Q218=100** | **Vnější souřadnice** |
| **Q219=80** | **Vnější souřadnice** |
| **Q121=30** | **Úhel zkosení** |
| **Q204=25** | **Výchozí bod** |
| **Q215=0** | **Volba technologie** |
| **L Y+20 F MAX M99** | |

# PŘÍLOHA č. 2

## Podprogram pro zkosení hrany

```
0  BEGIN PGM zkoseni MM
1  FN 18: SYSREAD Q96 = ID270 NR1 IDX5                ; nacteni aktualniho natoceni
2  Q98 = Q218 + Q204 * SIN Q121               ;vypocet souradnice startovaciho bodu
3  Q97 = Q219 + Q204 * COS Q121
4  L  X+Q98  Z+Q97 FMAX                                ;odjezd do startovaciho bodu
5  CALL LBL 1
6  Q80 = 1.5 * Q215 ^ 2 + 0.5 * Q215 - 1                       ; transformace Q215
7  Q90 = 1                                  ;pomocny parametr pro volbu technologie
8  Q99 = 0                                             ;vynulovani aktualni hloubky
9  Q203 = Q218 * SIN Q121 + Q219 * COS Q121               ;souradnice po natoceni Z
10 Q230 = Q218 * COS Q121 - Q219 * SIN 121                ;souradnice po natoceni X
11 Q94 = Q230 + ( Q108 + Q200 ) * COS ( 2 * Q121 ) + SQRT 0.02 * SIN (45-Q121)
                                                          ;souradnice prejezdu Xp
12 Q95 = Q203 + ( Q108 + Q200 ) * SIN ( 2 * Q121 ) + SQRT 0.02 * COS ( 45-Q121 )
                                                          ;souradnice prejezdu Zp
13 FN 9: IF +1 EQU +1 GOTO LBL 2 ;podmíněný skok na LBL2
14 ;
15 ; konec
16 FN 9: IF +1 EQU +1 GOTO LBL 30  ;konec programu, podmineny skok na konec, sem se
                                   podrpogram nikdy nedostane
17 LBL 1
18 M126 M129
19 CYCL DEF 19.0 ROVINA OBRABENI
20 CYCL DEF 19.1
21    B+Q121 FMAX M128 F15000                           ;natoceni souradneho systemu
22 M129
23 CYCL DEF 19.0 ROVINA OBRABENI
24 CYCL DEF 19.1  B+Q121 VZDAL.100
25 LBL 0
26 STOP M30                                    ;M30 je pouze bezpecnostniho charakteru,
                                                sem se podprogram nikdy nedostane
27 LBL 2
28 FN 9: IF +Q99 EQU +Q201 GOTO LBL 30      ;pokud aktualni hloubka je rovna
                                           celkove hloubce skok na konec programu
29 Q90 = Q90 * Q80                         ;volba technologie
30 Q99 = Q99 + Q202                        ;aktualni hloubka + hloubka prisuvu
31 FN 12: IF +Q99 LT +Q201 GOTO LBL 3      ;pokud aktualni hloubka je mensi nez~
                                           celkova, preskoc nasledujici blok
32 Q99 = Q201                  ;aktualni hloubka se rovna celkove hloubce srazeni
33 LBL 3
34 Q91 = Q230 - ( Q108 + Q200 + 0.1 / SIN Q121 + Q99 / TAN Q121 ) ;souradnice~
                                                          vypocet bodu najezdu X1
35 Q92 = Q230 + ( Q108 + Q200 + 0.1 / COS Q121 + Q99 * TAN Q121 ) ;souradnice~
                                                          vypoctu bodu najezdu X2
36 Q93 = Q203 - Q99                                ;souradnice vypoctu najezdu Z12
37 FN 11: IF +Q90 GT +0 GOTO LBL 5          ;pokud parametru Q90 je vetsi nez 0
                                            skok na lbl 5
38 LBL 4
39 L  X+Q91  Z+Q93 FMAX                                 ;prejezd na souradnice najezdu
40 L  X+Q92  Z+Q93 F AUTO           ;prejezd na souradnice vyjezdu pracovnim posuvem
41 FN 9: IF +Q80 EQU -1 GOTO LBL 2          ;pokud je pendl preskoc nasledujici blok
42 L  X+Q94  Z+Q95 FMAX                        ;prejezd na souradnice prejezdu Xp,Zp
43 FN 9: IF +1 EQU +1 GOTO LBL 2                          ;podmineni skok na LBL 2
44 LBL 5
45 L  X+Q92  Z+Q93 FMAX                             ;prejezd na souradnice najezdu B2
46 L  X+Q91  Z+Q93 F AUTO                           ;prejezd na souradnice vyjezdu B1
47 FN 9: IF +Q80 EQU -1 GOTO LBL 2          ;pokud je pendl preskoc nasledujici blok
48 L  X+Q94  Z+Q95 FMAX                               ;prejezd na souradnice prejezdu
49 FN 9: IF +1 EQU +1 GOTO LBL 2                          ;podmineni skok na LBL 2
50 LBL 30                                                  ; zakoncení podprogramu
51 Q99 = Q203 + Q204         ;vypocet souradnice na natoceni souradneho systemu zpet
52 L  X+Q230  Z+Q99 R0 FMAX                                ;odjezd na tyto souradnice
53 Q99 = Q121                                              ;zalohovani uhlu zkoseni
54 Q121 = Q96
55 CALL LBL 1                ;nepodminene vyvolani LBL1 v kterem se provadi natoceni
56 Q121 = Q99                                         ;vraceni parametru uhel zkoseni
57 END PGM zkosení MM
```