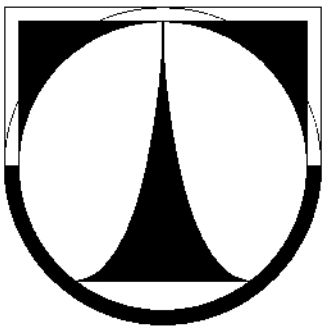
TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Katedra mechaniky, pružnosti a pevnosti



DISERTAČNÍ PRÁCE

Simulace a realizace ovládání robotizovaného podvozku

Simulation and Realization of Control of a Robotized Chassis

Liberec 2011 Ing. Miroslav Denk

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Disertační práce

k získání akademického titulu Ph.D.

ve studijním oboru

Aplikovaná mechanika – Inženýrská mechanika

Ing. Miroslav Denk

Simulace a realizace ovládání robotizovaného podvozku

Školitel: doc. Ing. Miroslav Šír, CSc.

Studijní program: P2301 Strojní inženýrství
Studijní obor: 3901V003 Aplikovaná mechanika
Zaměření: Inženýrská mechanika

Datum konání státní doktorské zkoušky: 23. června 2009

Poděkování

Rád bych touto cestou poděkoval všem, kteří mi s vypracováním disertační práce pomohli.

Zejména bych chtěl poděkovat svému vedoucímu disertační práce Doc. Ing. Miroslavu Šírovi, CSc z Katedry mechaniky, pružnosti a pevnosti, který mi poskytnul svůj čas, odborný dohled a mnohé cenné rady. V neposlední řadě bych rád poděkoval své manželce Markétě, která mi byla vždy oporou.

Anotace

Tato disertační práce se zabývá simulací a realizací ovládání robotizovaného podvozku vozíku pro sociálně zdravotní aplikace s cílem přispět k vývoji zařízení, které usnadní pohyb handicapovaných osob a ležících pacientů v obtížném terénu. Podvozek je opatřen čtyřmi nohami zakončenými koly. Každá noha má pět stupňů volnosti. Řídící program je vytvořen v programovacím jazyku C/C++ s využitím knihovny pro ovládání elektromotorů od firmy Maxon `EposCmd.dll` a knihovny OpenGL pro vizualizace podvozku. Dále jsou na simulačním modelu v MSC ADAMS ověřeny základní manévry.

Klíčová slova: robotizovaný podvozek, C/C++, OpenGL, Maxon.

Annotation

This thesis deals with simulation and realization of control of a robotized chassis for social/medical applications with the aim to help handicapped people to move on a rough terrain. The chassis is equipped with four shanks ended with wheels. Each shank has five degrees of freedom. The control program was created in the programming language C/C++ using the library intended for the control of the Maxon motors `EposCmd.dll` and OpenGL which was used for the visualization of the chassis. The basic maneuvers were verified on the simulation model in MSC ADAMS.

Key words: robotized chassis, C/C++, OpenGL, Maxon.

Obsah

1 Úvod

Tato práce vznikla v rámci výzkumného záměru *Optimalizace vlastností strojů v interakci s pracovními procesy a člověkem* MSM 4674788501 v sekci *Vibroizolační a vibroakustické systémy* a zabývá se robotizovaným podvozkem vozíku pro sociálně zdravotní aplikace s cílem přispět k vývoji zařízení, které usnadní pohyb handicapovaných osob a ležících pacientů v obtížném terénu. Navazuje na diplomové práce [7], [8] a [11], které vznikly v rámci výzkumného záměru.

1.1 Rešerše

Základním problémem je samotná koncepce podvozku, proto vlastnímu návrhu předcházela rešeršní činnost, která proběhla již před samotným započetím diplomových prací, na které tato práce navazuje. Cílem bylo najít analogická řešení prezentovaná v otevřených informačních zdrojích.

Invalidních vozíků určených do terénu je možno nalézt celou řadu. Žádný z nich však nemá uspokojivě řešenou stabilizaci prostoru pro uživatele a průchodnost a manévrovatelnost terénem řeší spíše hrubou silou, jak ukazují dále uvedené příklady.

1.1.1 Nabízená řešení pro pohyb handicapovaných osob v otevřeném terénu

Čtyřkolky ATV

Jedním z možných řešení je klasická terénní čtyřkolka (ATV - All Terrain Vehicle). V současné době se nabízejí stovky typů od desítek výrobců. Koncepčně jsou však tato vozidla prakticky identická. Mají spalovací motor a náhon 4x4 s rozvodem hnacího momentu prostřednictvím uzamykatelných nápravových a mezinápravových diferenciálů. Jednotlivé cenové kategorie se přitom liší mírou automatizace ovládání těchto diferenciálů. Přední nápravy jsou u většiny typů provedeny jako dvě nezávislá rovnoběžníková zavěšení se zvýšeným zdvihem a zadní nápravy bývají většinou tuhé, zavěšené na zkrutných ramenech.

Z hlediska našeho záměru je základním problémem skutečnost, že vozidla ATV nejsou primárně určena pro handicapované osoby. Částečně se používají pro hospodářské účely, většinou jsou to však prostředky pro provozování „adrenalinových" sportů. Navíc, zejména v Evropě, nemají povolen přístup do většiny turisticky zajímavých a ekologicky chráněných oblastí. Také zdaleka neřeší veškeré potřeby pohybu handicapovaných osob, například v urbanizovaných pěších územích, kde se běžně vyskytují překážky ve formě schodů, obrubníků chodníků a zúžených profilů.

SuperFour, ***odkaz na animaci [1]***

Vozidlo SuperFour nabízí firma OttoBock. Pohon tohoto vozidla je řešen čtyřmi nezávisle elektricky poháněnými koly, ale systém náprav žádné mimořádné řešení nevykazuje. Jedná se o čtyři klasická nezávislá rovnoběžníková zavěšení, pouze zdvih je výrazně zvětšen. Co se týče vodorovné stabilizace prostoru pro cestujícího, je zde možnost při sjezdu nebo výjezdu

kopce naklonit sedačku, což je pro pohyb v opravdu složitém terénu nedostačující. Maximální rychlost vozítka je cca 15 km/h.

Obr. 1.1 Vozidlo SuperFour

Tankchair*[2]*

Tankchair je pojízdné křeslo. Parametry tohoto vozidla nejsou na webových stránkách uvedeny, ale z obrázku je patrné, že toto řešení je vhodné jen pro venkovní použití, protože díky svým robustním rozměrům není vozík schopen projet úzkým místem, jako jsou např. zárubně dveří, což vylučuje jeho použití v bytě. Navíc zde není vůbec řešena vodorovná stabilita sedadla.

Obr. 1.2 Tankchair

6x6 Explorer*[3]*

6x6 Explorer je vozík vybavený 6 koly. Maximální uváděná rychlost vozítka je 4,5 - 6,5 km/h. Toto vozítko také nemá řešenu vodorovnou stabilitu sedadla a navíc pevné uložení kol neposkytuje dostatek komfortu při jízdě v terénu.

Obr. 1.3 Vozidlo 6x6 Explorer

Predator 4x4 ***[4]***

Predator 4x4 je vozík poháněný čtyřmi elektromotory, každý o výkonu 250 W. Dosahuje rychlosti 7 - 8 km/h. Toto řešení představuje na první pohled klasický elektrický vozík, pouze má větší kola a pohon s vyšším výkonem.

Obr. 1.4 Vozidlo Predator 4x4

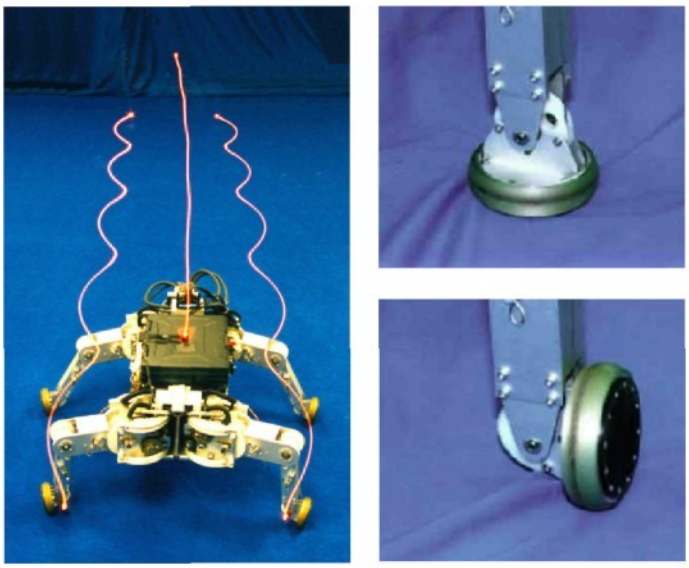
1.1.2 Příklady současných konstrukcí mobilních robotů s hybridním podvozkem

V následujících odstavcích budou zmíněny některé existující experimentální konstrukce mobilních robotů s podvozky, které lze charakterizovat jako kombinace kolového a (omezeně) kráčivého. Při tvorbě této podkapitoly bylo čerpáno z [12].

Roller-Walker*[5]*

Tento robot se po nerovném terénu pohybuje jako klasický čtyřnohý robot kráčením. Využívá k tomu přední stranu natáčecích pasivních koleček. Při pohybu po rovném a pevném terénu využívá natočení koleček pro valení. Vzhledem k tomu, že kolečka jsou pasivní, pro rozpohybování po rovině využívá „bruslení“, pro pohyb z kopce pak pouze pasivní jízdu. Brzdění probíhá vytvořením sbíhavosti koleček („plužení“). Každá noha má 3 ovládané stupně volnosti a 1 stupeň volnosti neřízený.

Robot dokáže vyvinout rychlost až 2,9 km/h, stoupavost je 10°. Výška robotu je 25 cm a celková váha je 24kg.



Obr. 1.5 Robot Rolle-Walker

Azimut*[6]*

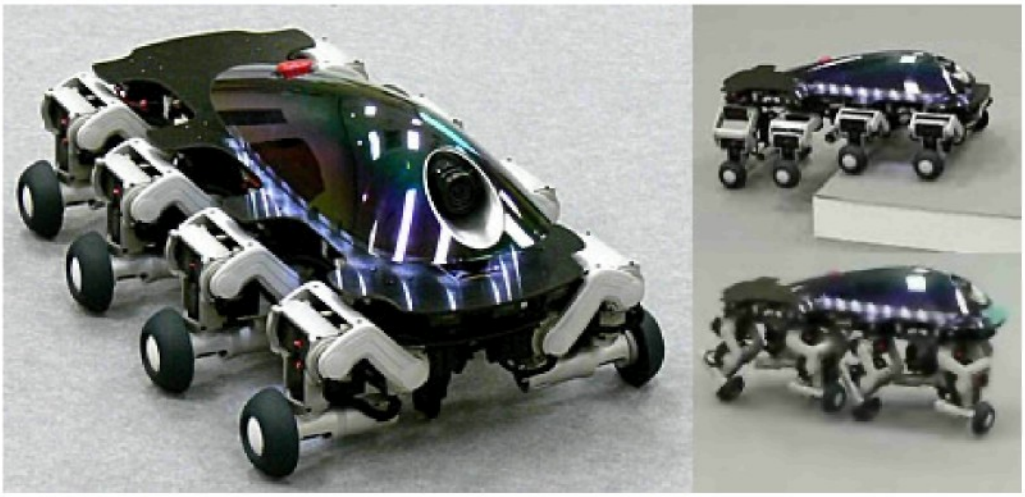
Robot se může pohybovat pomocí chůze i jízdou. K chůzi využívá koncovou plochu nohy, pro jízdu má robot dvě možnosti pohonu. První je jízda pomocí koleček, druhou je jízda pomocí pásu.

K pohybu využívá celkem 12 motorů pro 12 stupňů volnosti. Stoupavost robotu je 28° a dokáže uzvednout zátěž o hmotnosti 10,4 kg, přičemž jeho vlastní váha je 63 kg. Maximální rychlost, kterou dokáže vyvinout, je 4,3 km/h na rovném povrchu a 1,25 km/h při maximálním stoupání.

Obr. 1.6 Robot Azimut

Halluc II*[7]*

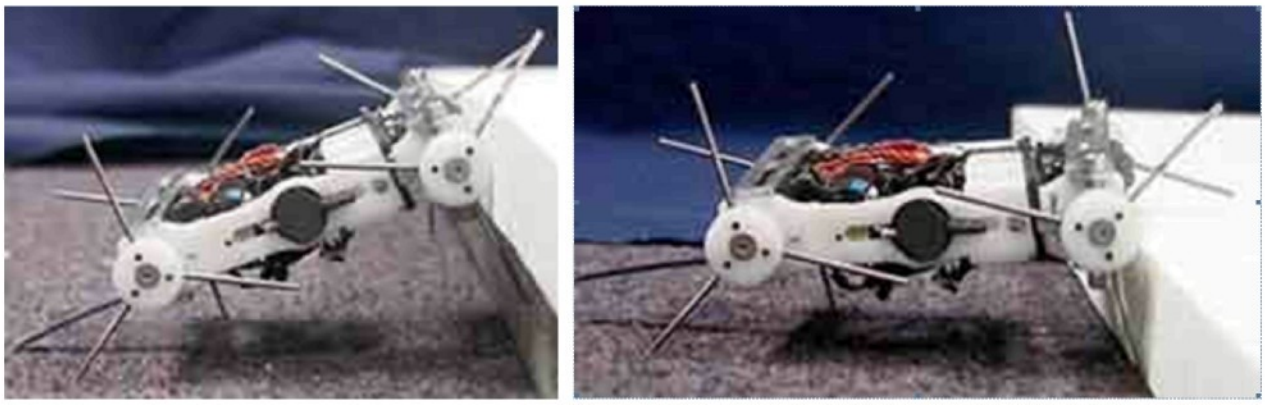
Tento velmi propracovaný robot pochází z Japonska. Operátor (robot je částečně autonomní) si může vybrat ze tří typů pohybu. Prvním je jízda za pomoci aktivních koleček. Druhým je hmyzí chůze, kdy se robot pohybuje po přední straně kolečka. Třetím typem je chůze zvířecí, při které se jako nášlapná plocha používá zadní strana motoru. Každé z osmi koleček je řízené nezávisle na ostatních. Celkově má robot 32 stupňů volnosti (3 stupně volnosti má noha + pohon kolečka) a 56 motorů. Na každý kloub nohy jsou použity 2 motory.



Obr. 1.7 Robot Halluc II

Whegs a Mini-Whegs*[8]*

Konstrukce robotů vychází ze studie pohybu švába, který se v klidu běžně pohybuje klasickou chůzí, při běhu ale končetinami pohybuje v podstatě kolem kyčelního kloubu a využívá je jako kola, která mu umožňují velmi rychlý pohyb a to jak po rovině, tak i při překonávání překážek. Robot má jeden stupeň volnosti na každou končetinu.

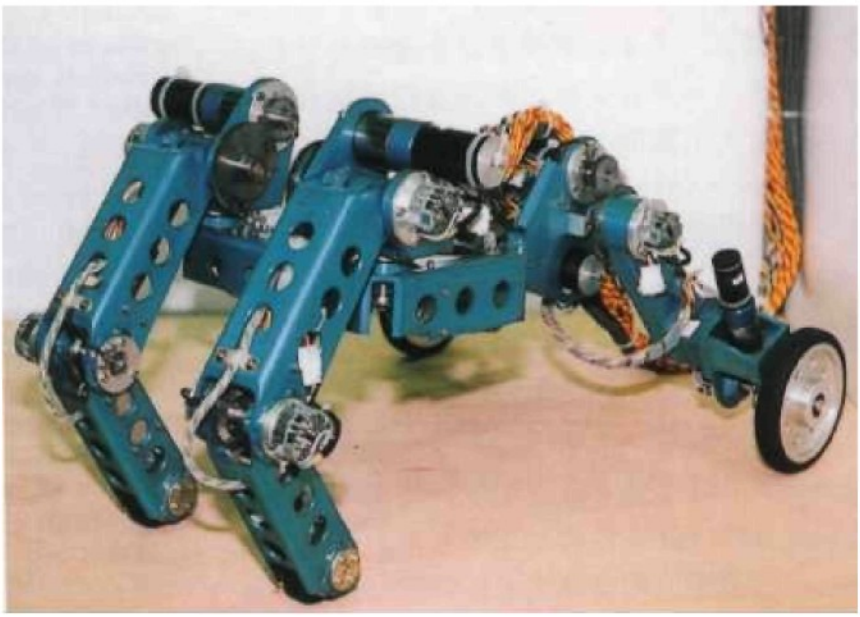


Obr. 1.8 Robot Mini-Whegs

Walk'n roll*[9]*

Robot z Japonska využívá kombinace jízdy a přitahování se pomocí končetin. Končetiny jsou vybaveny kolečkem s brzdou, takže se může po rovném terénu pohybovat pomocí jízdy a na

nerovném terénu se kolečka zabrzdí a robot využívá končetin k přitahování zadní části těla a ke šplhání na překážku.



Obr. 1.9 Robot Walk'n roll

Paw*[10]*

Robot je vybaven čtyřmi individuálně hnanými koly (pohánějí je motory Maxon s výkonem 20 W) umístěnými na otáčecích nohách (ovládané motory Maxon o výkonu 90 W). Přestože celková hmotnost robotu přesahuje 20 kg, je schopen dobré dynamiky pohybu jak na rovině, tak na mírně skloněném terénu i případně na travnatém terénu. Při zatáčení a brzdění využívá polohování noh.

Obr. 1.10 Robot Paw

X-VEAAT*[11]*

Robot X-VEAAT (X-perimental Vehicle Explorer Adaptable All Terrain) je určen pro demonstraci možnosti přizpůsobit podvozek vozidla terénu. Jediným pohybem robotu je v

současnosti jízda. Každá ze čtyř noh robotu je poháněna třemi servomotory, které jsou s kolenem a kolečkem propojeny pomocí několika ozubených kol. Je tak možné otáčet nohu s kolem i okolo svislé osy a realizovat tak holonomní platformu.

Obr. 1.11 Robot X-VEAAT

1.2 Organizace řešení zadané úlohy

Řešení úlohy bylo v minulosti rozděleno do více diplomových prací. První část, jejíž jsem autorem, se zabývala matematickým modelem [11], obr. 2, druhá část se zabývala konstrukcí základní podvozkové skupiny - kombinované podvozkové nohy [8], obr. 1. Zatím poslední z řady diplomových prací se zabývala vytvořením funkčního standu zjednodušené robotizované nohy a jeho simulací v prostředí MSC.ADAMS [7].

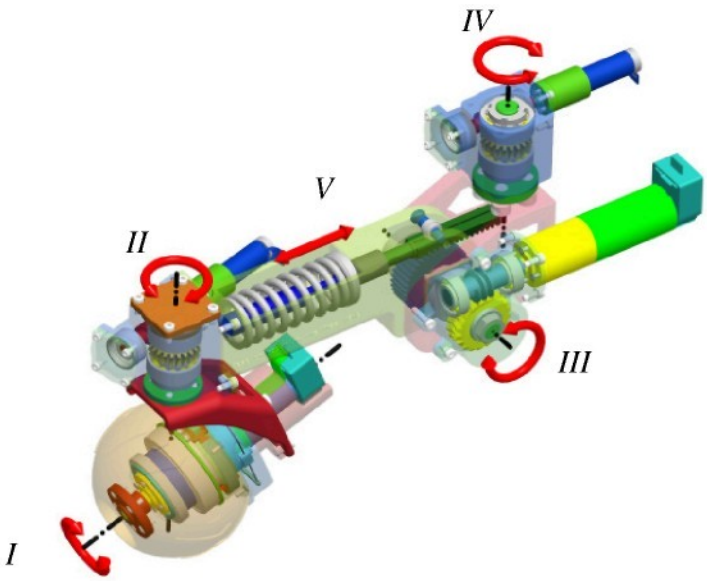
V současnosti je řešení této úlohy rozděleno do dvou disertačních prací. Jednou z částí je tato a druhá část řešení se zabývá regulací vodorovné polohy podvozku robotu.

1.3 Volba koncepce

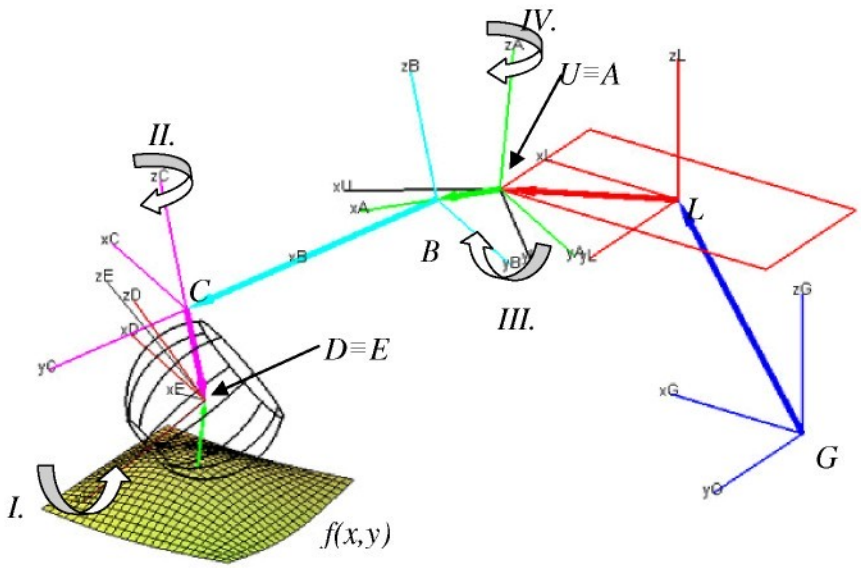
Aby se uživatel vozíku mohl volně pohybovat v urbanizovaném prostředí i ve volné přírodě bez pomoci jiné osoby, měla by být koncepce robotizovaného podvozku taková, aby byl schopen následujících manévrů při zachování sedačky ve vodorovné poloze:

- jízda v přímém i proměnném směru po rovném i zvlněném terénu,
- změna světlé výšky podvozku,
- průjezd úzkým profilem bez ztráty stability,
- překonání překážky překročením,
- chůze,
- pohyb po schodištích různých parametrů.

Z těchto důvodů byla zvolena konfigurace se čtyřmi nohami, z nichž každá je opatřena kolem. Kolo bylo zvoleno kulového tvaru, protože takové kolo je schopné jízdy i při větším odklonu osy rotace od pojezdové plochy. Každá noha má pět stupňů volnosti, z nichž čtyři jsou přímo ovládány samostatnými elektromotory a pátý se vymezuje působením síly v pružině a tlumiči. Stupně volnosti jsou vyznačeny na následujících obrázcích.



Obr. 1.12 Skutečná geometrie podvozkové nohy



Obr. 1.13 Matematický model podvozkové nohy

I. rotace kola

II. pivotace kola

III. vyrovnávání terénu

IV. rejd

V. pružení a tlumení

Rotací kola se uvádí celé vozidlo do pohybu. Změnou úhlu pivotace se dosahuje změny směru jízdy. Stupeň volnosti označený jako rejd (*IV*) slouží ke změně rozvoru a rozchodu kol a bude také využíván při překonávání překážek. Stupeň volnosti na obr. 1.12 a obr. 1.13 vyznačený jako *III* je úhel, který umožňuje vyrovnávání nerovností terénu a změnu světlé výšky podvozku. Ve fyzické realizaci bude ovládání řešeno součinností elektromotoru a tlačné pružiny (*V*) tak, že elektromotor prostřednictvím šnekové převodovky bude ovládat předpětí pružiny.

1.4 Parametry robotizovaného podvozku

První fází vývoje je tvorba měřítkového modelu, jehož základní parametry jsou uvedeny v tabulce 1. Pokud se model osvědčí, další fází bude tvorba prototypu, jehož rozměry nebudou přímo násobky modelu, ale jednotlivé části budou zvětšeny v požadovaném měřítku, které bude vycházet z finálních rozměrových požadavků. Rozvor a rozchod jsou vzhledem k pohybovým možnostem podvozku značně variabilní, proto jsou v tabulce uvedeny rozměry v mezních polohách.

Parametry modelu	
max. rychlost	8 km/h
pohotovostní hmotnost	20 kg
celková hmotnost	30 kg
rozvor	(100-500)mm
rozchod	(200-600)mm

Parametry konečného provedení budou přibližně odpovídat rozměrům běžně prodávaných vozíků. Celková hmotnost bude cca 200kg, rozvor a rozchod v základní poloze přibližně 1m. Rychlost pohybu bude asi 8km/h.

1.5 Jednotlivé části úkolu

Zadání řešené úlohy je rozděleno do dvou široce souvisejících částí. Cílem té části zadání, která se zabývá tvorbou ovládacího softwaru, je vytvořit soubor programů, kterými se budou ovládat veškeré požadované pohyby robotizovaného podvozku. Soubor programů musí být dostatečně otevřený, aby bylo možno vkládat další moduly, např. modul stabilizace příčného a podélného náklonu podvozku.

Druhá část úlohy, tedy simulace základních manévrů, slouží k ověření řídících programů a pohybových schopností podvozku. Je nutné ověřit, zda jsou reálně proveditelné základní manévry, které jsou uvedeny v kapitole **1.3**. Dále je nutné potvrdit či vyvrátit správnost dimenzování elektrických pohonů. Selekce pohonných jednotek byla provedena v diplomové práci [8].

2 Realizace ovládání

Základní idea realizace ovládání je taková, že 16 z 20 stupňů volnosti celého podvozku bude přímo nebo nepřímo ovládáno elektromotorem.

> *Poznámka*: Přímým ovládáním je myšleno, že silový účinek vyvozený elektromotorem bude přenášen na poháněný element pomocí převodovky k danému motoru příslušející, jak je tomu u pohonu kola. V nepřímém ovládání je přidána šneková převodovka, která zajišťuje samosvornost. Toho je využito u pohonů stupňů volnosti nazvaných rejd, pivotace a vyrovnávání terénu.

Ke každému motoru přísluší řídící jednotka. Kontrolní jednotky jsou zapojeny do série s využitím sběrnice CAN. Dále je každý elektromotor opatřen převodovkou a snímačem polohy. Řídící jednotky jsou ovládány prostřednictvím osobního počítače, ke kterému je bezdrátově připojen joystick. Schéma této řídící struktury je na obr. 2.1.

Obr. 2.1 Řídící struktura

2.1 Akční prvky ve struktuře řízení

Jak je uvedeno výše, všechny prvky potřebné k realizaci ovládání byly vybrány v diplomové práci [8]. Ve výběru však došlo ke změnám, a proto zde následuje popis jednotlivých prvků ve struktuře řízení. Použity byly výhradně výrobky firmy Maxon Motor (www.maxonmotor.com). Obrázky a technická data použité v této kapitole byly převzaty z katalogu firmy Maxon Motor.

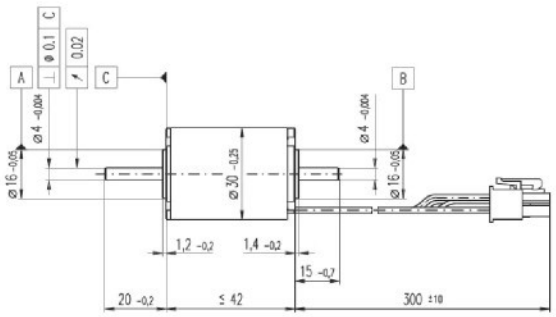
2.1.1 Motory

Pro pohon robotizovaného podvozku byly zvoleny stejnosměrné elektronicky komutované elektromotory z řady ***EC*** a ***EC*-max**. Jak z obr. 1.12 a obr. 1.13 vyplývá, na každou nohu podvozku připadají čtyři elektromotory. Jedná se tedy celkem o šestnáct motorů.

Obr. 2.2

Pro pohon rotace kola a pohon rejdu byl z katalogu firmy Maxon Motor vybrán stejný motor ***EC*-max 30** 272766 o výkonu 40 W, obr. 2.2.

***EC*-max 30** ∅30 mm, brushless, 40 Watt

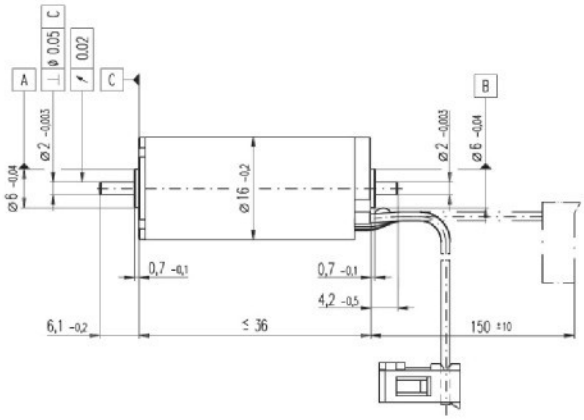


			272766
Motor Data			
	Values at nominal voltage		
1	Nominal voltage	V	12.0
2	No load speed	rpm	8700
3	No load current	mA	202
4	Nominal speed	rpm	6640
5	Nominal torque (max. continuous torque)	mNm	35.0
6	Nominal current (max. continuous current)	A	2.85
7	Stall torque	mNm	153
8	Starting current	A	11.8
9	Max. efficiency	%	76
	Characteristics		
10	Terminal resistance phase to phase	Ω	1.01
11	Terminal inductance phase to phase	mH	0.088
12	Torque constant	mNm / A	12.9
13	Speed constant	rpm / V	738
14	Speed / torque gradient	rpm / mNm	57.8
15	Mechanical time constant	ms	6.66
16	Rotor inertia	gcm²	11.0

Rozměry a technické parametry motoru *EC*-max 30 272766

Pro pohon stupně volnosti nazvaného pivotace byl vybrán nejméně výkonný motor ze všech čtyř. Jedná se o motor ***EC*-max 16** 283833 o výkonu 8 W.

EC-max 16 ∅16 mm, brushless, 8 Watt

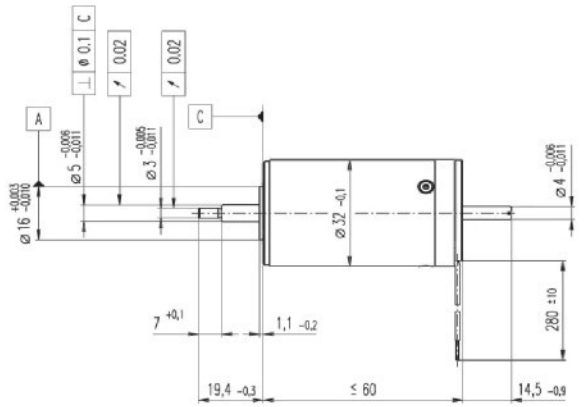


	Motor Data		283833
	Values at nominal voltage		
1	Nominal voltage	V	12.0
2	No load speed	rpm	12000
3	No load current	mA	58.5
4	Nominal speed	rpm	7310
5	Nominal torque (max. continuous torque)	mNm	8.04
6	Nominal current (max. continuous current)	A	0.907
7	Stall torque	mNm	21.1
8	Starting current	A	2.27
9	Max. efficiency	%	71
	Characteristics		
10	Terminal resistance phase to phase	Ω	5.30
11	Terminal inductance phase to phase	mH	0.140
12	Torque constant	mNm / A	9.32
13	Speed constant	rpm / V	1020
14	Speed / torque gradient	rpm / mNm	582
15	Mechanical time constant	ms	5.18
16	Rotor inertia	gcm²	0.850

Rozměry a technické parametry motoru *EC*-16 283833

Stupeň volnosti, který zajišťuje vyrovnávání terénu a změnu světlé výšky podvozku, pohání motor ***EC* 32** 118889, který má ze čtveřice motorů nejvyšší výkon a to 80 W.

EC 32 ∅32 mm, brushless, 80 Watt, CE approved



	Motor Data		118891
	Values at nominal voltage		
1	Nominal voltage	V	12.0
2	No load speed	rpm	15000
3	No load current	mA	901
4	Nominal speed	rpm	13600
5	Nominal torque (max. continuous torque)	mNm	37.5
6	Nominal current (max. continuous current)	A	5.82
7	Stall torque	mNm	428
8	Starting current	A	57.2
9	Max. efficiency	%	77.0
	Characteristics		
10	Terminal resistance phase to phase	Ω	0.21
11	Terminal inductance phase to phase	mH	0.03
12	Torque constant	mNm / A	7.48
13	Speed constant	rpm / V	1280
14	Speed / torque gradient	rpm / mNm	35.8
15	Mechanical time constant	ms	7.49
16	Rotor inertia	gcm²	20.0

Rozměry a technické parametry motoru *EC*-16 283833

2.1.2 Planetové převodovky

Výstupní otáčky motorů z předchozí kapitoly jsou poměrně vysoké a nominální momenty nedostačují pro základní manévry, a proto byly motory osazeny převodovkami s různými převodovými poměry.

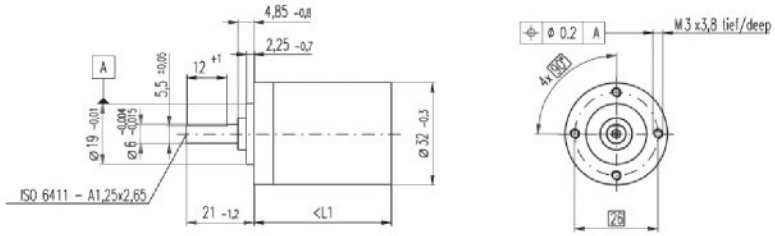
Firma Maxon Motor nabízí různé druhy převodovek. Jedná se o převodovky s čelním ozubením, převodovky planetové a převodovky planetové s keramickými komponenty.

Pro naše účely byly vybrány planetové převodovky s a bez keramických komponentů s převodovými poměry od 21:1 až do 86:1, obr. 2.3.

Obr. 2.3 Planetová převodovka od firmy Maxon

K motorům, které pohání kolo podvozku a rejd, přísluší převodovka **GP 32 C** 166935 s převodovým poměrem 21:1. Písmeno **C** v označení převodovky označuje, že se jedná o převodovku s keramickými komponenty.

Planetary Gearhead GP 32 C ∅32 mm, 1.0 - 6.0 Nm
Ceramic Version

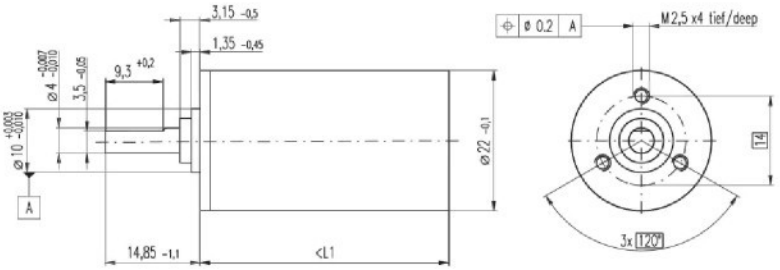


Technical Data	
Planetary Gearhead	straight teeth
Output shaft	stainless steel
Shaft diameter as option	8 mm
Bearing at output	ball bearing
Radial play, 5 mm from flange	max. 0.14 mm
Axial play	max. 0.4 mm
Max. radial load, 10 mm from flange	140 N
Max. permissible axial load	120 N
Max. permissible force for press fits	120 N
Sense of rotation, drive to output	=
Recommended input speed	< 8000 rpm
Recommended temperature range	-20 ... +100°C
Extended area as option	-35 ... +100°C

Rozměry a technické parametry převodovky GP 32 C 166935

Motory pohánějící pivotaci jsou osazeny planetovými převodovkami **GP 22 C** 143979 s převodovým poměrem 29:1. Jedná se také o převodovky s keramickými komponenty.

Planetary Gearhead GP 22 C ∅22 mm, 0.5 - 2.0 Nm
Ceramic Version

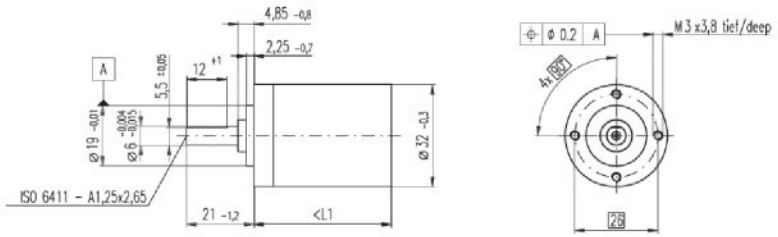


Technical Data	
Planetary Gearhead	straight teeth
Output shaft	stainless steel, hardened
Bearing at output	ball bearing
Radial play, 10 mm from flange	max. 0.2 mm
Axial play	max. 0.2 mm
Max. radial load, 10 mm from flange	70 N
Max. permissible axial load	100 N
Max. permissible force for press fits	100 N
Sense of rotation, drive to output	=
Recommended input speed	< 8000 rpm
Recommended temperature range	-20 ... +100°C
Extended area as option	-35 ... +100°C

Rozměry a technické parametry převodovky GP 22 C 143979

Poslední z řady planetových převodovek je **GP 32 A** 166167 s převodovým poměrem 86:1. Tato převodovka přísluší motorům, které pohání stupně volnosti používané ke změně světlé výšky a hlavně k udržování vodorovné polohy podvozku. Jedná se o verzi převodovky bez keramických komponentů, což naznačuje písmeno **A** v označení převodovky.

Planetary Gearhead GP 32 A ∅32 mm, 0.75 - 4.5 Nm



Technical Data	
Planetary Gearhead	straight teeth
Output shaft	stainless steel
Shaft diameter as option	8 mm
Bearing at output	ball bearing
Radial play, 5 mm from flange	max. 0.14 mm
Axial play	max. 0.4 mm
Max. radial load, 10 mm from flange	140 N
Max. permissible axial load	120 N
Max. permissible force for press fits	120 N
Sense of rotation, drive to output	=
Recommended input speed	< 6000 rpm
Recommended temperature range	-20 ... +100°C
Extended area as option	-35 ... +100°C

Rozměry a technické parametry převodovky GP 32 A 166167

2.1.3 Inkrementální snímače

Každý z motorů obsahuje senzor polohy (Hall sensor), který, jak už název napovídá, pracuje na základě Hallova efektu.

> *Poznámka*: Vložíme-li vodivou destičku tloušťky d, kterou protéká řídící elektrický proud I, do magnetického pole s magnetickou indukcí B_y, kolmou na směr proudu, pak ve třetím směru, kolmém na směr proudu a zároveň na směr magnetického pole změříme potenciálový rozdíl U_H. Následkem Hallova jevu vzniká Hallovo napětí
>
> $$U_H = R_H \frac{IB_y}{d},$$
>
> kde R_H je Hallova konstanta [$\mathrm{m^3A^{-1}s^{-1}}$].

Tyto vestavěné senzory mají nízký počet pulsů na otáčku, a proto byly motory doplněny o inkrementální snímače (encodery), které mají několikanásobně více pulsů na otáčku, a je tedy možné stanovit polohu motoru přesněji.

Firma Maxon Motor dodává encodery pracující na různých principech. Encodery pracující na magneto-resistentním principu (Magneto-resistant MR principle) byly použity pro motory, které pohání rejd a pivotaci. Pro zbytek motorů byly použity encodery pracující na opticko-elektrickém principu, obr. 2.4.

Obr. 2.4

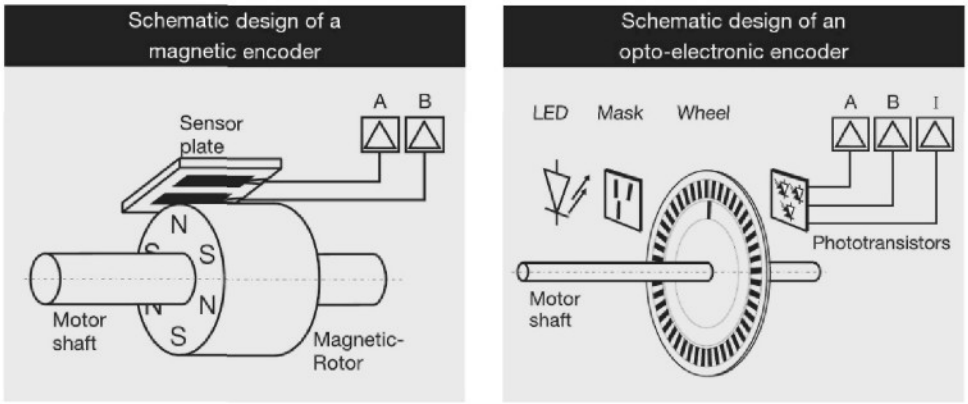
Magneto-resistentní encoder

U tohoto MR encoderu je pevně na hřídel namontován multipólový magnetický disk. Disk produkuje sinusové napětí v MR senzoru. Interpolací pak dostaneme typický signál encoderu.

Opticko-elektrický encoder

Na hřídeli je namontována mřížka, kterou prochází světlo. Změnu světelného toku zaznamenávají fototranzistory.

Tyto dva principy jsou schematicky naznačeny na obr. 2.5.

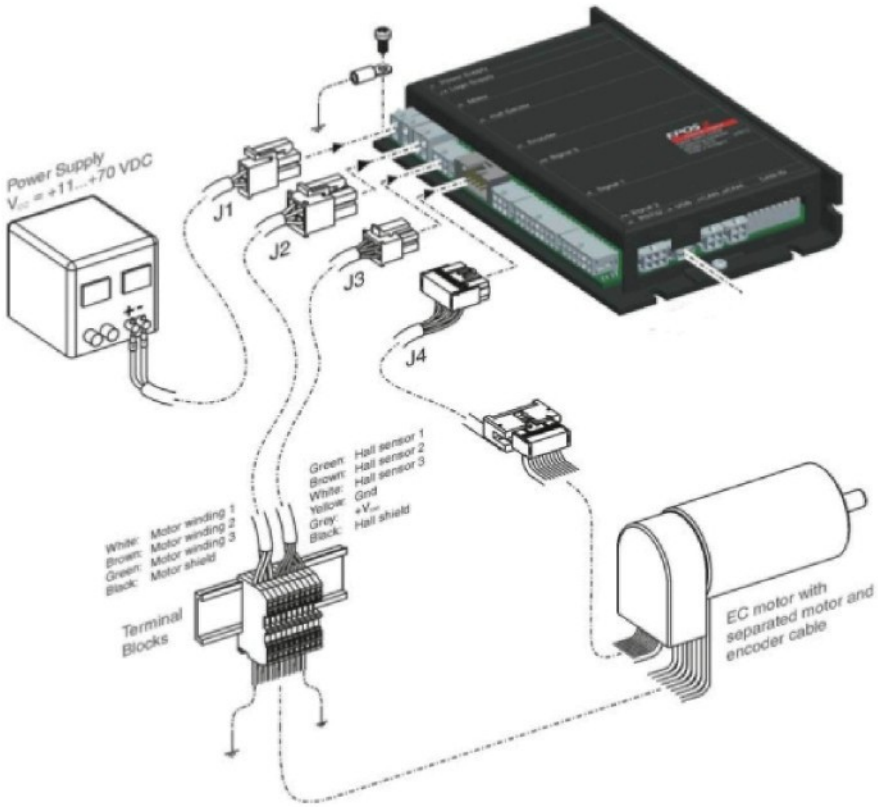


Obr. 2.5 Schéma principů práce encoderů

2.1.4 Řídící jednotky

Nezbytnou částí každého použitého elektromotoru je řídící jednotka. Pro naše účely byla zvolena řídící jednotka EPOS 70/10 300583 (obr. 2.6a). Ta v sobě obsahuje regulační obvod, což značně ulehčuje ovládání motorů. Jedna z řídících jednotek je připojena prostřednictvím sériového kabelu RS232 k počítači, ostatní řídící jednotky jsou mezi sebou propojeny sběrnicí CAN, obr. 2.1. Takto může být za sebou propojeno až 128 jednotek. Schéma zapojení motorů a řídících jednotek je znázorněno na obr. 2.6b.

Obr. 2.6a Řídící jednotka EPOS 70/10



Obr. 2.6b Zapojení řídící jednotky a motoru

Zkratka EPOS znamená Easy (to use) POSitioning System.

2.1.5 Brzda

Čtveřice motorů, která zajišťuje pohon kol, je navíc vybavena bezpečnostní brzdou. Brzda katalogového označení **AB 20** 301213 obsahuje permanentní magnet a udržuje hřídel motoru v zabrzděné poloze, pokud není systém napájen. V případě, že dojde k výpadku energie, bude tedy motor zabrzděn. U ostatních motorů není použita brzda, protože její funkci nahrazuje samosvornost šnekových převodovek.

2.2 Ovládání akčních prvků ve struktuře řízení

2.2.1 Způsoby ovládání (EposStudio vs. C/C++)

Je více způsobů, jak ovládání jednotlivých motorů realizovat. Jedním z nich je použití softwaru EPOS Studio, který je standardně dodáván s motory, nebo je volně ke stažení na stránkách firmy Maxon. EPOS Studio slouží k nastavení omezujících parametrů motorů, které jsou následovně uloženy v řídící jednotce. Jedná se například o nominální proud, povolené otáčky, atd. Dále EPOS Studio umožňuje nastavení parametrů regulace, přepínaní mezi jednotlivými pracovními módy motorů a v případě programovatelných řídících jednotek i jejich programování. K našemu účelu se ale EPOS Studio nehodí. Popis práce s EPOS Studiem je například uveden v [7].

Druhý způsob je využití knihoven pro různé programovací jazyky. Tyto knihovny umožňují využít všechny funkce EPOS Studia prostřednictvím zvoleného programovacího jazyka. Na stránkách firmy Maxon jsou k dispozici ke stažení například knihovny pro Borland C++, Borland Delphi, LabView, Visual C++, atd.

2.2.2 Důvody výběru stávajícího programového prostředí

Jedním z předmětů, které jsem absolvoval během doktorského studia, bylo programování C/C++. Proto i pro ovládání motorů byl zvolen tento programovací jazyk, který nám mimo jiné zajistí dostatečný výpočetní výkon.

2.2.3 Popis knihovny určené pro ovládání motorů prostřednictvím C/C++ (`EposCmd.dll`)

V této kapitole jsou popsány funkce z knihovny `EposCmd.dll`, které byly použity při tvorbě řídícího programu. Úplné funkční prototypy funkcí jsou definovány v hlavičkovém souboru `Definitions.h`. Zkratka `VCS` před každou z funkcí znamená „Virtual Commands Slave“.

Funkce z této knihovny jsou strukturovány do několika sekcí, zabývají se například inicializací, pomocnými funkcemi, konfigurací, operačními módy, stavem motoru, informacemi o pohybu motoru, atd.

Funkce z této knihovny pracují s vlastními datovými typy, jejichž přehled je v tabulce 2.1.

Tabulka 2.1

Name	Data type	Size bits	Size bytes	Range
char, __int8	signed integer	8	1	– 128 ... 127
BYTE	unsigned integer	8	1	0 ... 256
short	signed integer	16	2	– 32'768 ... 32'767
WORD	unsigned integer	16	2	0 ... 65'535
long	signed integer	32	4	– 2'147'483'648 ... 2'147'483'647
DWORD	unsigned integer	32	4	0 ... 4'294'967'295
BOOL	signed integer	32	4	TRUE =1 FALSE =0
HANDLE	pointer to an object	32	4	0 ... 4'294'967'295

Initialisation

První z částí, do kterých je soupis funkčních prototypů v souboru `Definitions.h` rozdělen, jsou funkce týkající se inicializace spojení mezi jednotkou a počítačem.

- `HANDLE VCS_OpenDevice(char* DeviceName, char* ProtocolStackName, char* InterfaceName, char* PortName, DWORD* pErrorCode);`

  Funkce `VCS_OpenDevice` otevře port pro přijímání a posílání příkazů a vrátí ukazatel na toto spojení. V našem případě, tedy při použití RS232, řídící jednotky EPOS a sériového portu COM1, byla funkce volána ve tvaru

  ```
  COM1 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_1,&pErrorCode);
  ```

- `BOOL VCS_SetProtocolStackSettings(HANDLE KeyHandle, DWORD Baudrate, DWORD Timeout, DWORD* pErrorCode);`

  Pomocí této funkce se nastavují parametry spojení. Při úspěšném provedení vrátí funkce nenulovou hodnotu, v opačném případě vrátí funkce nulu. Parametr

`KeyHandle`, je ukazatel na spojení, které vrátí funkce `VCS_OpenDevice` a je jedním z parametrů naprosté většiny funkcí z této knihovny.

Např.: `VCS_SetProtocolStackSettings(COM1,115200,500,&pErrorCode);`

State Machine

Další sekcí v `Definitions.h` jsou funkce, které pracují se stavem zařízení.

- `BOOL VCS_SetEnableState(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode);`
- `BOOL VCS_SetDisableState(HANDLE KeyHandle, WORD NodeId, DWORD* pErrorCode);`

Dvě výše uvedené funkce nastaví jednotky identifikované hodnotou `NodeId` do stavu `Enable`, resp. `Disable`. Ve stavu `Enable` drží motory svojí pozici za cenu přírůstku proudu, v `Disable` se mohou motory po překonání odporů spojených s konstrukcí motorů u převodovek otáčet.

Např.: `VCS_SetEnableState(COM1,pivo_1,&pErrorCode);`

- `BOOL VCS_SetOperationMode(HANDLE KeyHandle, WORD NodeId, __int8 Mode, DWORD* pErrorCode);`

Funkce nastaví operační mód u řídící jednotky identifikované hodnotou `NodeId` (hodnota v binární soustavě nastavená přímo na řídící jednotce) na `Mode`. Hodnoty, kterých může tento parametr nabývat, jsou uvedeny v tabulce 2.2.

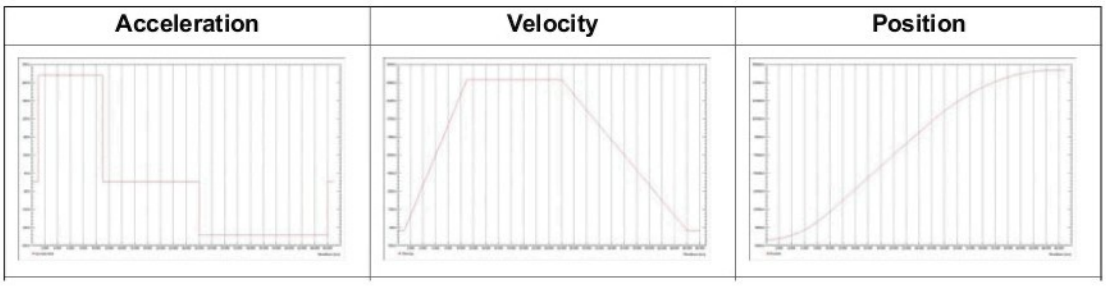
Tabulka 2.2.

Operation Mode	Description
6	Homing Mode
3	Profile Velocity Mode
1	Profile Position Mode
-1	Position Mode
-2	Velocity Mode
-3	Current Mode
-4	Diagnostic Mode
-5	MasterEncoder Mode
-6	Step/Direction Mode

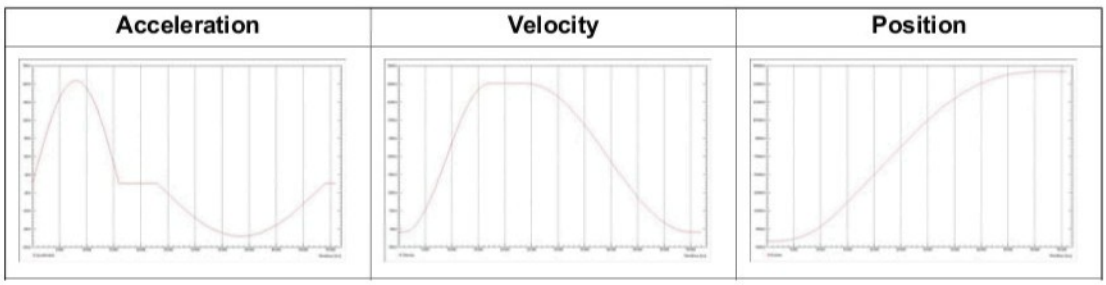
Např.: `VCS_SetOperationMode(COM1,rejd_1,0x01/*Profile Position Mode*/,&pErrorCode);`

Následuje popis operačních módů použitých v dosavadní práci.

Position and Velocity Mode slouží k přímému nastavení požadované polohy a rychlosti. Position Profile Mode definuje požadovanou pozici a trajektorii, po které bude tato pozice dosažena pomocí generátoru trajektorie. Generátor trajektorie podporuje různé druhy typů profilů. Cílové polohy může být dosaženo po lineární nebo sinusoidální rampě.

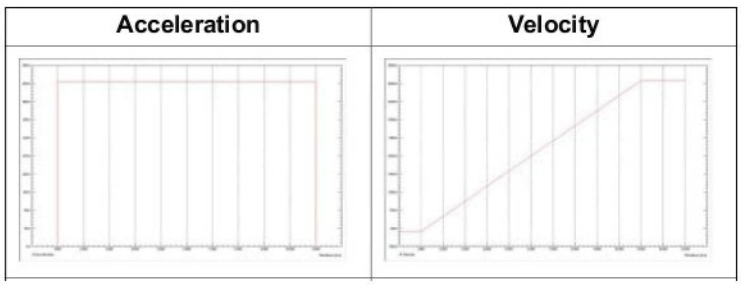


Profile Position Mode – lineární rampa

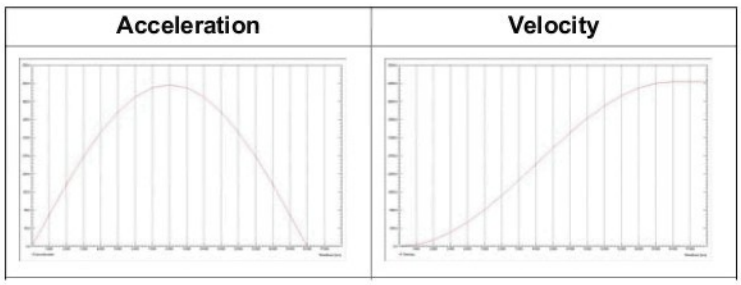


Position Mode – sinusoidální rampa

Profile Velocity Mode slouží k nastavení požadované rychlosti bez ohledu na aktuální pozici. Stejně jako u předchozího módu je profil přechodu na požadovanou rychlost vytvořen pomocí generátoru trajektorie po lineární, resp. sinusoidální rampě.

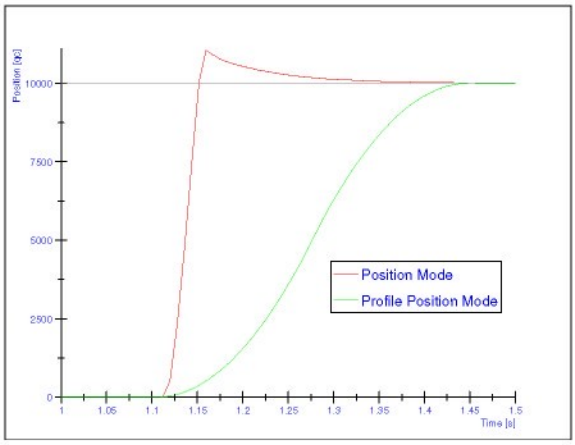


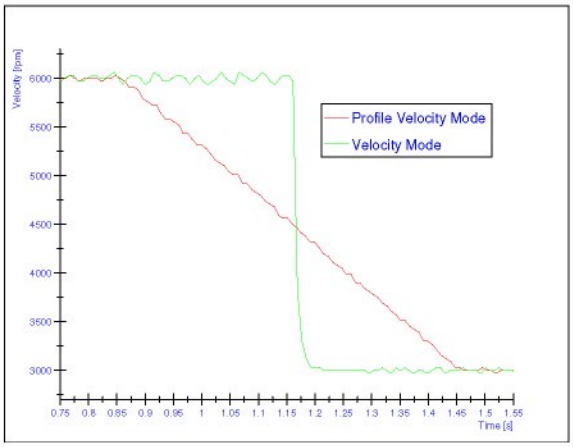
Profile Velocity Mode – lineární rampa



Profile Velocity Mode – sinusoidální rampa

Na následujících obrázcích jsou znázorněny rozdíly mezi Position a Profile Position a Velocity a Profile Velocity (lineární rampa) módy. Křivky jsou vykresleny na základě dat naměřených při reálné simulaci. Profile Position mód byl realizován s hodnotami `ProfileVelocity=8000rpm`, `ProfileAcceleration = ProfileDeceleration=10000rpm/s`. Profile Velocity mód byl realizován s hodnotami, `ProfileAcceleration = ProfileDeceleration= =5000rpm/s`.





Position and Profile Position Mode

- `BOOL VCS_SetPositionProfile(HANDLE KeyHandle, WORD NodeId, DWORD ProfileVelocity, DWORD ProfileAcceleration, DWORD ProfileDeceleration, DWORD* pErrorCode);`
- `BOOL VCS_MoveToPosition(HANDLE KeyHandle, WORD NodeId, long TargetPosition, BOOL Absolute, BOOL Immediately, DWORD* pErrorCode);`

První z funkcí slouží k nastavení parametrů profilu, podle kterého bude dosaženo dané polohy, druhá slouží k zahájení pohybu po daném profilu do požadované pozice (`TargetPosition`). `Absolute` nabývá hodnot `TRUE` nebo `FALSE` a udává, jestli se motor pohybuje o přírůstek nebo v rámci absolutních hodnot. Boolovský parametr `Immediately` značí, zda bude vyčkáno na dokončení předchozího pohybu, nebo jestli bude pohyb započat ihned. Obě funkce se týkají módu Profile Position Mode.

Např.:
```
VCS_SetPositionProfile(COM1,rejd_1,8700,20000,20000,&pErrorCode);
VCS_MoveToPosition(COM1,vyro_1,pozice,Absolute,Immediately,&pErrorCo);
```

- `BOOL VCS_SetPositionMust(HANDLE KeyHandle, WORD NodeId, long PositionMust, DWORD* pErrorCode);`

Funkce `VCS_SetPositionMust` se používá v operačním módu Position Mode a slouží k přímému přechodu na danou pozici (`PositionMust`).

Např.:
```
VCS_SetPositionMust(COM1,rejd_1,pozice,&pErrorCode);
```

Velocity and Profile Velocity Mode

- `BOOL VCS_SetVelocityProfile(HANDLE KeyHandle, WORD NodeId, DWORD ProfileAcceleration, DWORD ProfileDeceleration, DWORD* pErrorCode);`
- `BOOL VCS_MoveWithVelocity(HANDLE KeyHandle, WORD NodeId, long TargetVelocity, DWORD* pErrorCode);`

Podobně jako u Profile Position Mode slouží první z funkcí k nastavení parametrů profilu, podle kterého bude dosažena požadovaná rychlost (`TargetVelocity`) v módu Profile Velocity Mode. Druhou funkcí se zahájí přechod na požadovanou rychlost.

- `BOOL VCS_SetVelocityMust(HANDLE KeyHandle, WORD NodeId, long VelocityMust, DWORD* pErrorCode);`

Poslední použitá funkce z této části souboru `Definitions.h` je funkce `VCS_SetVelocityMust`. Ta pracuje ve Velocity Mode a přímo nastaví požadovanou rychlost (`VelocityMust`).

Např.: `VCS_SetVelocityMust(COM4,rota_4,87.0*xo,&pErrorCode);`

Motion Info

Funkce z Motion Info slouží ke zjištění informací o pohybu motorů.

- `BOOL VCS_GetPositionIs(HANDLE KeyHandle, WORD NodeId, long* pPositionIs, DWORD* pErrorCode);`

  Funkce uloží do `pPositionIs` aktuální pozici motoru identifikovaného hodnotou `NodeId`.

- `BOOL VCS_GetVelocityIs(HANDLE KeyHandle, WORD NodeId, long* pVelocityIs, DWORD* pErrorCode);`

  Funkce uloží do `pVelocityIs` aktuální rychlost motoru identifikovaného hodnotou `NodeId`.

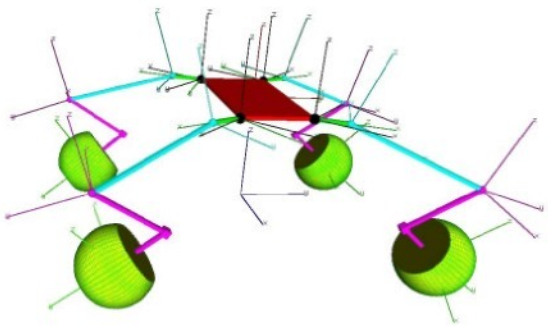
- `BOOL VCS_GetMovementState(HANDLE KeyHandle, WORD NodeId, BOOL* pTargetReached, DWORD* pErrorCode);`

  Boolovský parametr `pTargetReached` nabývá hodnoty `TRUE`, když motor identifikovaný `NodeId` dosáhne požadované pozice. Této funkce se využívá hlavně v případě několika po sobě následujících pohybů, kdy je nutné počkat na dokončení předchozího pohybu.

  Např.:
```
VCS_GetPositionIs(COM1, rejd_1, &maxon_phi_Az1,&pErrorCode);
VCS_GetMovementState(COM1, pivo_1,&pivo_1_done,&pErrorCode);
```

2.3 Vizualizace podvozku v softwarovém prostředí pomocí knihovny OpenGL (`gl.h`, `glu.h`, `glut.h`)

K vizualizaci podvozku robotu v softwarovém prostředí byla zvolena otevřená grafická knihovna OpenGL, obr. 2.7. Konfigurace podvozku v OpenGL je vytvořena na základě dat získaných z encoderů a poskytuje tak informaci o polohách jednotlivých částí podvozku, které by nemusely být patrné ze samotných motorů namontovaných v držáku, obr. 3.1.



Obr. 2.7 Model podvozku v OpenGL

2.3.1 OpenGL

OpenGL je programové rozhraní grafického hardwaru. Toto rozhraní se skládá z asi 250 příkazů (cca dvě stě v samotném jádru OpenGL a dalších padesát v knihovně utilit OpenGL), které se používají pro určení objektů a operací potřebných k vytvoření interaktivních trojrozměrných aplikací.

OpenGL je rozhraní nezávislé na hardwaru, které lze použít na mnoha platformách. Proto OpenGL neobsahuje žádné příkazy pro práci s okny či příkazy pro získávání uživatelských vstupů. Tyto příkazy je nutné použít v závislosti na aktuálně používaném hardwaru, viz níže (GLUT). Dále samotné OpenGL neobsahuje příkazy pro tvorbu složitých trojrozměrných těles. S OpenGL se požadovaná geometrie vytváří pomocí základních tvarů, jako je bod, přímka, polygon.

Standardní součástí OpenGL je knihovna utilit pro OpenGL (GLU). Tato knihovna poskytuje mnoho modelovacích možností, které jsou postaveny na bázi základních příkazů OpenGL. S použitím GLU je možné vytvořit matice projekcí, provádět teselaci polygonů (převod složitých polygonů na trojúhelníky) a generovat povrchy, zejména kvadriky. Procedury GLU mají předponu **`glu`**.

Další důležitou knihovnou využívanou při práci s OpenGL je knihovna pro práci s grafickými utilitami. Tato knihovna se nazývá Graphics Library Utility Toolkit (GLUT, předpona **`glut`**), jejím autorem je Mark J. Kilgard. GLUT slouží ke zjednodušení práce s okny a práce se vstupy a výstupy. Knihovna GLUT také obsahuje procedury pro generování složitějších geometrií.

Základy práce s OpenGL byly čerpány z různých internetových zdrojů, viz příloha INTERNETOVÉ ZDROJE a dále z knihy „*OPENGL, Průvodce programátora*“ [9].

2.3.2 Funkce využité z knihovny OpenGL pro tvorbu modelu podvozku

V této kapitole jsou vysvětleny funkce a procedury potřebné pro vytvoření modelu vizualizace podvozku. Kdyby byl popis jednotlivých funkcí a procedur uveden níže s popisem vlastního programu v C/C++, vedlo by to k nepřehlednosti, a proto jsou zde vysvětleny funkce a procedury, které se týkají specifických částí v sestavování modelu.

Práce s okny a práce se vstupy

Pro práci s okny a práci se vstupy se používají funkce a procedury ze spřízněné knihovny GLUT. Mezi ně patří a v programu byly použity tyto:

- `void glutInit(int *argcp, char **argv);`

  Procedura, která inicializuje GLUT má dva argumenty. První argument `argcp` představuje ukazatel na proměnnou obsahující počet parametrů programu, druhý argument `argv` je pole řetězců reprezentujících jednotlivé parametry.

- `void glutInitDisplayMode(unsigned int mode);`

  Procedura, která určí, zda bude použit barevný model RGBA či indexované barvy.

- `void glutInitWindowSize(int width, int height);`

Určuje velikost okna v pixelech.

- `void glutInitWindowPosition(int x, int y);`

  Určuje pozici levého horního rohu okna na obrazovce.

- `int glutCreateWindow(const char *title);`

  Tato funkce má jediný argument `title` představující řetězec zobrazený v titulkovém pruhu okna. Vrací identifikátor pro toto okno.

- `void glutDisplayFunc(void (*func)(void));`

  Registrace funkce volané při požadavku na překreslení. Tato funkce má jako jediný argument ukazatel na vytvořenou funkci, jež nemá žádné parametry.

Dále bylo potřeba registrovat funkce, které jsou volány v případě určitých událostí. Jsou to například tyto funkce:

- `void glutReshapeFunc(void (*func)(int width, int height));`

  Funkce registrovaná touto procedurou se volá při každé změně velikosti okna.

- `void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));`
- `void glutSpecialFunc(void (*func)(int key, int x, int y));`
- `void glutMouseFunc(void (*func)(int button, int state, int x, int y));`

  Tři výše uvedené funkce slouží k registraci funkcí, které jsou volány v případě stisku klávesy na klávesnici, resp. speciální klávesy na klávesnici (např. šipky, PageDown, atd.), resp. tlačítka myši.

- `void glutMotionFunc(void (*func)(int x, int y));`

  Procedura registrující funkci volanou při pohybu myši se současně stisknutým tlačítkem.

- `void glutJoystickFunc(void (*func)(unsigned int buttonMask, int x, int y, int z), int pollInterval);`

  Funkce `glutJoystickFunc` má dva parametry. První je ukazatel na funkci, která se volá při manipulaci s joystickem a druhý parametr je interval v [ms] mezi voláními této funkce.

Zjišťování informací o běžícím programu

Pro zjištění informací o právě běžícím programu slouží funkce `glutGet()`.

- `int glutGet(GLenum type);`

Funkce `glutGet` má pouze jeden parametr `type`. V závislosti na tomto parametru vrací funkce hodnotu `int`. Například je-li parametr `GLUT_ELAPSED_TIME`, vrátí funkce počet milisekund od volání funkce `glutInit`, tedy od počátku programu. Použijeme-li parametr `GLUT_WINDOW_WIDTH`, vrátí šířku aktuálního okna v pixelech.

Funkce pro kreslení geometrických tvarů

Model vizualizace podvozku robotu se skládá ze základních geometrických tvarů. Pro jejich tvorbu byly použity procedury z OpenGL (předpona **gl**) a procedury z knihovny GLUT (předpona **glut**).

Jelikož je většina modelu tvořena kvádry a knihovna GLUT funkci pro vytvoření kvádru, narozdíl od krychle, koule, kuželu apod., neobsahuje, byly kvádry vytvořeny pomocí základních geometrických primitiv v OpenGL z polygonů, které tvoří jednotlivé strany.

Vrcholy geometrických objektů se v OpenGL definují pomocí funkce `glVertex*()`.

```
void glVertex3f(GLfloat x, GLfloat y, GLfloat z);
```

V příkladu je uveden vrchol tvořený třemi souřadnicemi datového typu **float**.

Vykreslení daného tělesa tvořeného jednotlivými vrcholy bylo provedeno s použitím procedur **void** `glBegin(GLenum mode)` a **void** `glEnd(void)`.

Zde následuje příklad vykreslení čtverce v rovině (x, y), který je tvořen tvarem typu `GL_QUADS`, tedy parametr funkce `glBegin mode` má hodnotu `GL_QUADS`.

```
glBegin(GL_QUADS);
    glVertex3f(1.0,1.0,0.0);
    glVertex3f(1.0,-1.0,0.0);
    glVertex3f(-1.0,-1.0,0.0);
    glVertex3f(-1.0,1.0,0.0);
glEnd();
```

Seznam typů tvarů ve funkci `glBegin`, stejně tak jako různé způsoby definování vrcholů, je uveden v [9].

Procedury z knihovny GLUT byly hlavně použity při tvorbě modelu kola. Kolo je tvořeno částí kulové plochy, jak je patrné na obr. 1.12.

```
void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);
void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);
```

Funkce pro vytvoření plné kulové plochy, resp. drátěného modelu kulové plochy. Drátěný model kola byl přidán proto, aby byla patrná rotace kola.

Jak je uvedeno výše, kolo je tvořeno částí kulové plochy. Aby bylo dosaženo požadovaného tvaru, je nutné sféry vytvořené předchozími funkcemi oříznout rovinami. To se provede následující sekvencí procedur:

```
void glClipPlane(GLenum plane, const GLdouble *equation);
void glEnable(GLenum plane);
⋮
void glDisable(GLenum plane);
```

Parametr `plane` je `GL_CLIP_PLANEi`, kde `i` je celé číslo, jež určuje, která ořezová rovina je takto definována a následně používána. Parametr `equation` je ukazatel na pole čtyř hodnot datového typu **`double`**, což jsou koeficienty A, B, C, D v obecné rovnici ořezové roviny $Ax + By + Cz + D = 0$. Tato rovina se pak pomocí procedury `glEnable` a `glDisable` povolí, resp. zakáže. Příklad použití ořezu pomocí roviny je uveden dále v textu, viz kapitola soubor `kolo.cpp`.

Transformace

Poslední skupinou znalostí nutných k sestavení modelu podvozku jsou TRANSFORMACE. Transformace jsou v OpenGL široký pojem. Zde se omezíme na transformace tzv. `MODELVIEW_MATRIX` matice, což jsou modelovací transformace. Ostatními typy transformací (transformacemi projekčními a zobrazovacími) se zabývat nebudeme, protože pro sestavení modelu podvozku nebyly zapotřebí. Více informací v INTERNETOVÉ ZDROJE a [9].

Modelovací transformace

Transformací se rozumí změna transformační matice, která slouží k vykreslení daného tělesa (souřadnicového systému). Pro libovolné části modelu můžeme načíst různé transformační matice, nebo aktuální transformační matici vynásobit novou, která představuje další v řadě transformací. V podstatě jsou transformace v OpenGL analogií použití rozšířených transformačních matic v mechanice, viz [1]. V modelu podvozku se bude jednat o změnu `MODELVIEW_MATRIX` matice.

V OpenGL transformační matici představuje pole 16 hodnot, které může být definováno takto: **`float`** `Matice[16]`.

Základními funkcemi pro modelovací transformace jsou `glTranslatef*()` a `glRotatef*()`. Tyto funkce transformují objekt (souřadnicový systém) tak, že provedou posun nebo rotaci. Jsou ekvivalentem vytvoření vlastních matic a jejich načtení voláním funkce `glMultMatrixf*()`.

Nyní přejděme k vysvětlení jednotlivých procedur transformace `MODELVIEW_MATRIX`:

- **`void`** `glMatrixMode(GLenum mode);`

  Určuje, jestli bude upravována matice zobrazení, projekce, textur. V našem případě, kdy budeme měnit jen matici `MODELVIEW_MATRIX`, bude mít parametr `mode` hodnotu `GL_MODELVIEW`.

- **`void`** `glLoadIdentity(void);`

  Nastaví aktuální modifikovanou matici na jednotkovou. Tato matice bude v modelu podvozku reprezentovat základní rám (globální souřadnicový systém).

- **`void`** `glLoadMatrixf(const GLfloat *m);`

  Nastaví 16 hodnot aktuální matice na hodnoty, na které ukazuje `m`.

- **`void`** `glMultMatrixf(const GLfloat *m);`

  Násobí aktuální matici maticí šestnácti hodnot z `m`.

- `void glTranslatef(GLfloat x, GLfloat y, GLfloat z);`

  Násobí aktuální matici maticí, která posouvá daný objekt o vektor (x, y, z). Násobí tedy aktuální matici maticí

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

- `void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);`

  Násobí aktuální matici maticí, která otáčí objekt proti směru hodinových ručiček o úhel `angle` ve stupních okolo osy procházející počátkem a určenou směrovým vektorem (x, y, z). Pro `glRotatef(phi,0.0,0.0,1.0);` je výsledkem násobení aktuální matice maticí

$$\begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Použijeme-li tedy nejprve transformaci posunem (`glTranslatef`) a následně rotací (`glRotatef`) výsledkem bude součin výše uvedených matic

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

čímž dostaneme matici

$$\begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 & x \\ \sin(\varphi) & \cos(\varphi) & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

což je standardní rozšířená transformační matice pro transformaci souřadnic z lokálního souřadnicového systému, který je posunut o (x, y, z) a následně pootočen o úhel φ okolo osy z lokálního souřadnicového systému do globálního souřadnicového systému.

Uložení aktuální transformační matice a práce se zásobníkem matic

- `void glGetFloatv(GLenum pname, GLfloat *params);`

  Tato funkce slouží k získávání informací z OpenGL. Byla použita ve tvaru `glGetFloatv(GL_MODELVIEW_MATRIX, m);`, který uloží aktuální `MODELVIEW_MATRIX` do proměnné, kam ukazuje `m`.

Každá transformační matice doposud vytvořená, ať už násobením nebo načtením konkrétních hodnot, se nacházela na vrcholu zásobníku matic. Pro práci se zásobníkem matic nabízí OpenGL tyto funkce:

- `void glPushMatrix(void);`

  Funkce posune všechny matice ze zásobníku o jednu úroveň níže a do první úrovně zásobníku zkopíruje matici, která byla nejvýše před voláním této funkce. To znamená, že v prvních dvou vrstvách zásobníku je stejná matice.

- `void glPopMatrix(void);`

  Smaže vrchní matici ze zásobníku a druhá matice ze zásobníku se stává aktuální.

Příklad použití funkcí pro práci se zásobníkem je uveden dále v textu, viz kapitola soubor `main.cpp`.

2.4 Realizace v programovacím jazyku C/C++

Celý řídící program byl vytvořen v tomto programovacím jazyku. V jednotlivých částech řídícího programu je použito již dříve vysvětlených funkcí pro ovládání motorů a tvorbu vizualizačního modelu prostřednictvím OpenGL. Popis jednotlivých souborů programu je obsahem této kapitoly. Nebudou zde uvedeny kompletně celé části programů ale pouze výběr z nich. Celý výpis řídícího programu včetně všech podsouborů viz příloha. Názvosloví jednotlivých částí podvozku je uvedeno v tabulce 3.2.

2.4.1 Struktura programu (využití odděleného překladu souborů)

Při tvorbě řídícího programu bylo z důvodu přehlednosti použito odděleného překladu jednotlivých souborů. Oddělený překlad souborů znamená, že se každý soubor přeloží zvlášť (vznikne tedy několik `.obj` souborů) a ty se spojí do jednoho programu až pomocí sestavovacího programu (`linker`). V současnosti je řídící program rozdělen do 17 souborů, které jsou popsány v následujících kapitolách.

2.4.2 Význam a popis jednotlivých souborů

`main.cpp`

První a nejdůležitější součástí řídícího programu je soubor `main.cpp`. V tomto souboru dochází k inicializaci okna, ve kterém jsou podvozek a pomocné údaje vykresleny. Dále zde dochází k propojení jednotlivých částí ovládacího programu.

V souboru `main.cpp` jsou definovány booleovské proměnné (`start_jizda`, `start_ensabling`, atd), které určují, jaký manévr se bude provádět. Dále jsou zde definovány booleovské proměnné (`vyro_1_done`, `pivo_1_done`, atd.) informující o dokončení pohybů jednotlivých stupňů volnosti. Hodnoty proměnných `vyro_1_done`, `pivo_1_done`, atd. získáme pomocí funkce `VCS_GetMovementState` (viz výše).

Mezi další proměnné definované v souboru `main.cpp` patří proměnné typu **`int`**. Jsou to hodnoty převodových poměrů planetových a šnekových převodovek, počet pulsů na otáčku encoderů, hodnoty úhlů, které definují základní konfiguraci podvozku a hodnoty `NodeId`

identifikující jednotlivé řídící jednotky. Hodnoty úhlů jsou odečítány v tzv. quadcountech (ozn.: qc) a ty jsou přepočteny pomocí převodových poměrů na úhly jako takové a dosazeny do transformačních matic.

V souboru `main.cpp` jsou dále definovány funkce, které jsou volány při změně velikosti okna (`onResize`) a funkce volané při požadavku na překreslení okna (`init`, `onDisplay`).

Funkce `init()` nastavuje různé parametry zobrazení.

Funkce `onDisplay()` obsahuje příkazy pro výpis textu na obrazovku, funkce pro vytvoření transformačních matic (`vytvorMatice()`, viz dále), funkce pro výpočet úhlů natočení kol (`nastaveni()`, viz dále), funkce pro vykreslení jednotlivých částí podzovku, funkce `VCS_GetMovementState` a podmínky, podle kterých se aktivují různé manévry.

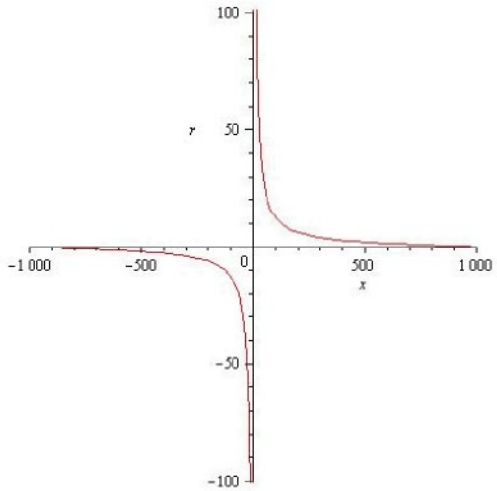
V neposlední řadě soubor `main.cpp` obsahuje funkci `main()`. Ve funkci `main()` jsou funkce z kapitoly Práce s okny a práce se vstupy, které vytvářejí okno, určují barevné schéma, registrují funkce volané v případě zvláštních událostí, atd.

`joystick.cpp`

Jak už název souboru napovídá, tato část řídícího programu slouží k ošetření práce s joystickem. Hlavní součástí tohoto souboru je funkce `void joystick(unsigned int joystick, int x, int y, int z)`, která je v `main.cpp` registrována pomocí funkce `glutJoystickFunc`, která je popsána výše. Proměnné `x`, `y`, `z` představují tři osy joysticku a nabývají hodnot ±1000. Proto je potřeba je přepočítat na vhodné hodnoty. Například hodnotu osy `x`, která v programu představuje požadovaný poloměr zatáčení, bylo nutné přepočítat podle vztahu

$$r = -2\mathrm{tg}\left(\frac{x}{1000}\frac{\pi}{2} - \frac{\pi}{2}\right), \tag{2.1}$$

kdy pro nulovou výchylku joysticku je nastaven „nekonečný“ poloměr zatáčky a pro maximální výchylku joysticku je nastaven poloměr zatáčky na nulu, viz graf.

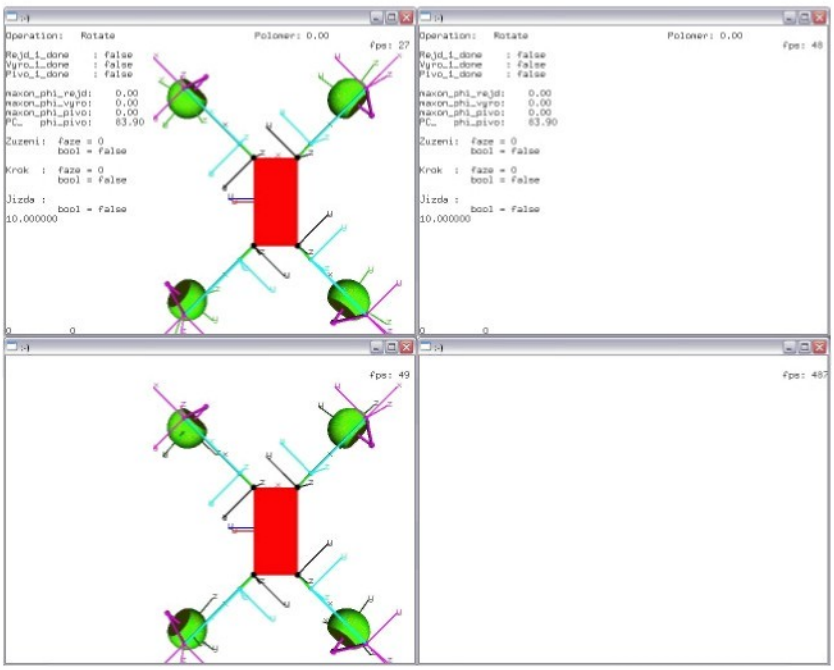


Graf 1: Průběh funkce pro výpočet poloměru zatáčení

Dále jsou zde ošetřeny události v případě stisknutí tlačítek na joysticku. Ty jsou čtyři a slouží k přepínání jednotlivých manévrů podvozku a navíc první z nich (v programu označeno `GLUT_JOYSTICK_BUTTON_A`) vytvoří spojení s počítačem, nastaví parametry spojení a nastaví požadované operační módy pro manévr *jízda*.

`keyboard.cpp`

Soubor `keyboard.cpp` sloužil k přepínání jednotlivých manévrů pomocí klávesnice. To však bylo nahrazeno pomocí tlačítek na joysticku a pomocí klávesnice se nyní přepíná mezi různými druhy zobrazení modelu (obr. 2.8) a zapíná/vypíná zápis do souboru.



Obr. 2.8 Různé druhy zobrazení modelu

Z obr. 2.8 je patrné, že se snižujícím počtem informací vykreslených v okně se zvyšuje frekvence překreslování snímků (`fps` vpravo nahoře) a tedy i frekvence zjišťování či zadávání informací z/do řídící jednotky.

`kvadr_ss.cpp, kvadr_pos.cpp, Sou_sys.cpp, te_pivotace.cpp, kolo.cpp`

Dále následuje popis nejjednodušších částí programu, proto jsou uvedeny v jedné podkapitole. Jsou to soubory `kvadr_ss.cpp`, `kvadr_pos.cpp`, `te_pivotace.cpp`, `Sou_sys.cpp`, `kolo.cpp`, které slouží k vykreslení částí podvozku v OpenGL.

Soubor `kvadr_ss.cpp` obsahuje funkci **`void`** `kvadr_ss_v_T(`**`float`** `a,` **`float`** `b,` **`float`** `c)`. Úkolem této funkce je vykreslit kvádr o rozměrech `a`, `b`, `c` s těžištěm v počátku aktuálního souřadnicového systému.

Podobný předchozímu souboru je soubor `kvadr_pos.cpp` obsahující funkci **`void`** `kvadr_pos(`**`float`** `a,` **`float`** `b,` **`float`** `c)`. Tato funkce také slouží k vykreslení kvádru o rozměrech `a`, `b`, `c` ale s těžištěm posunutým o `a/2` ve směru osy *x* aktuálního souřadnicového systému.

Jednotlivé souřadnicové systémy jsou vykresleny pomocí funkce `void sou_sys(void)`, která je součástí souboru `Sou_sys.cpp`. Podrobný popis souřadnicových systémů jednotlivých částí robotizovaného podvozku je uveden v mé diplomové práci [11].

Pro vykreslení tělesa pivotace bylo použito funkce `void te_pivotace(void)` ze souboru `te_pivotace.cpp`.

Posledním ze souborů určených ke kreslení geometrie je soubor `kolo.cpp` s funkcí `void kolo(void)`. Funkce `kolo()` využívá funkcí pro ořez obrazu, které byly vysvětleny výše, viz Funkce pro kreslení geometrických tvarů.

phi_Cz.cpp

Soubor `phi_Cz.cpp` slouží k nastavení rychlosti rotace kol a jejich polohy v souladu s požadovanou rychlostí pohybu a požadovaným poloměrem zatáčení, což jsou hodnoty ovládané samotným uživatelem prostřednictvím joysticku.

Teorie výpočtu požadovaných poloh a úhlových rychlostí jednotlivých kol je uvedena v [11]. Výsledkem výpočtu v matematickém softwaru Maple jsou funkce závislé jak na geometrii robotizovaného podvozku a jeho aktuální konfiguraci, tak na poloměru zatáčky a rychlosti průjezdu touto zatáčkou. Tyto funkce byly pomocí příkazu **`C()`** z knihovny **`CodeGeneration`** ze softwaru Maple exportovány do programovacího jazyka C/C++.

Aktuální konfigurací se rozumí aktuální hodnoty jednotlivých stupňů volnosti. Tato závislost nám zajišťuje, že kola jsou ve správné poloze (tedy osy rotace směřují to středu zatáčky) i u manévru *zúžení*, kdy dochází ke změně rozvoru a rozchodu podvozku.

matice.cpp

Další součástí řídícího programu je soubor `matice.cpp` a funkce `void vytvorMatice()` v něm obsažená. Funkce `vytvorMatice()` slouží k vytvoření transformačních matic, které popisují pohyb jednotlivých částí robotizovaného podvozku.

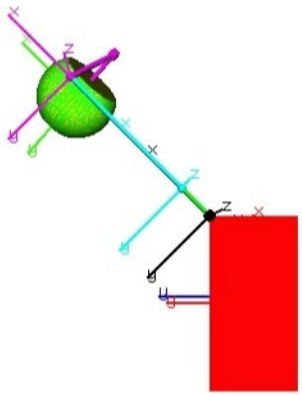
Základem pro vytvoření všech transformačních matic je informace o aktuální poloze motorů pohánějících rejdy, vyrovnávání a pivotace. K tomuto účelu byla použita funkce `VCS_GetPositionIs` popsaná v kapitole Motion Info.

Dále je hlavně s použitím funkcí `glTranslatef` a `glRotatef` popsán každý souřadnicový systém a jeho relativní pohyb vůči předchozímu souřadnicovému systému. Takto vzniklé transformační matice jsou uloženy prostřednictvím funkce `glGetFloatv` a opět načteny pomocí funkce `glMultMatrixf`.

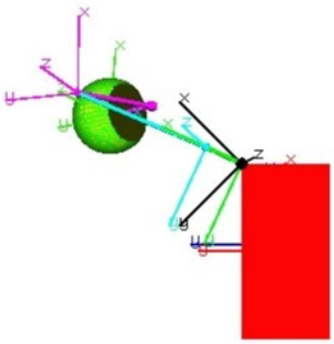
enabling.cpp

Soubor `enabling.cpp` je první ze souborů, které pracují převážně s funkcemi týkajících se ovládání motorů.

Po zapnutí zdroje, který prozatím slouží pro napájení řídících jednotek, jsou motory v poloze nula (obr. 2.9). Je tedy nutné nastavit základní konfiguraci podvozku, ve které se bude provádět nejvíce používaný manévr *jízda*.

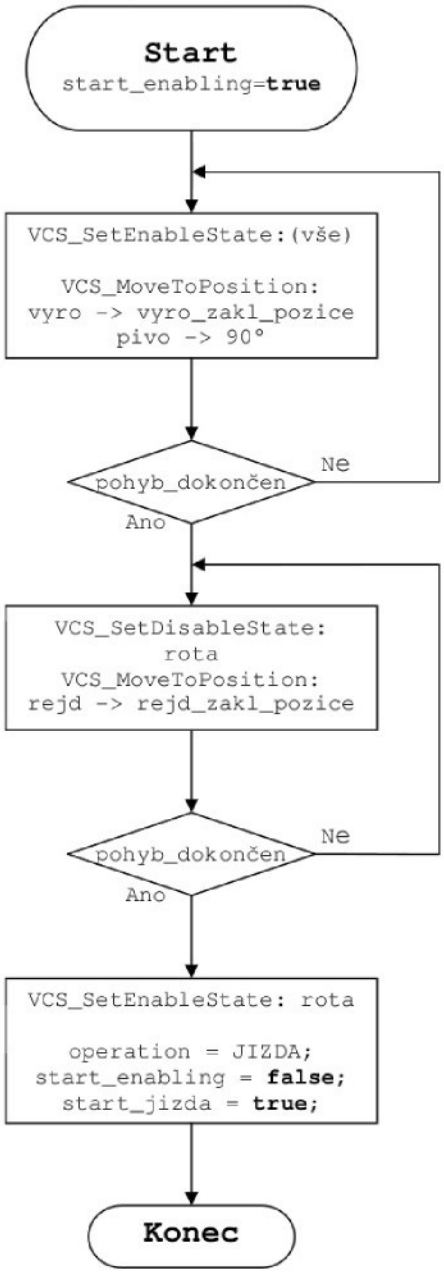


Obr. 2.9 Nulová pozice podvozku



Obr. 2.10 Základní konfigurace podvozku

Základní konfigurace podvozku je dosaženo následovně. Nejprve se všechny pohony nastaví do stavu `Enable`. Poté se stupeň volnosti nazvaný vyrovnávání nastaví na hodnotu `vyro_zakl_pozice`, což je prozatím stanovená základní pozice a zároveň pivotace na hodnotu 90°. To umožní v další fázi po nastavení motoru pohonu kol do stavu `Disable` a současné změně rejdu na hodnotu `rejd_zakl_pozice` najetí do základní konfigurace podvozku (obr. 2.10). Samozřejmostí je, že vykonání další fáze funkce `enabling` je podmíněno dokončením fáze předchozí. V poslední řadě se nastaví motor pohonu do stavu `Enable`, aktuální manévr na `JIZDA` a booleovské proměnné, které definují, který manévr se provádí na `start_enabling = false` a `start_jizda = true`. Celá procedura je znázorněna v diagramu 2.1.



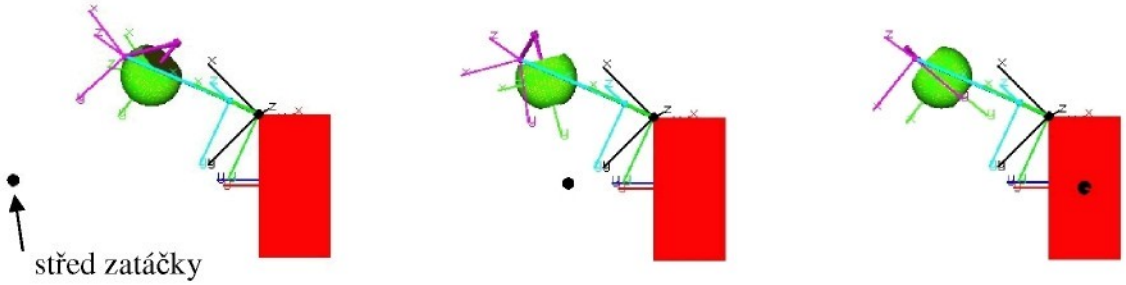
Diag. 2.1 Fáze *enabling*

`jizda.cpp`

Tím se dostáváme k souboru `jizda.cpp` a funkci **`void`** `jizda(`**`void`**`)`. Jak už bylo uvedeno, *jízda* bude u tohoto robotizovaného podvozku nejvíce používaný manévr.

V průběhu tohoto manévru se joystickem ovládá rychlost jízdy a poloměr zatáčení. Prostřednictvím funkce `nastaveni()` ze souboru `phi_Cz.cpp` se nastaví správná poloha kol tak, aby se podvozek pohyboval po dané trajektorii, obr. 2.11. Střed zatáčení je v modelu a na obrázcích vyznačen černým bodem.

V budoucnu bude tento soubor doplněn o systém regulace vodorovné polohy podvozku, který je v současné době ve vývoji.

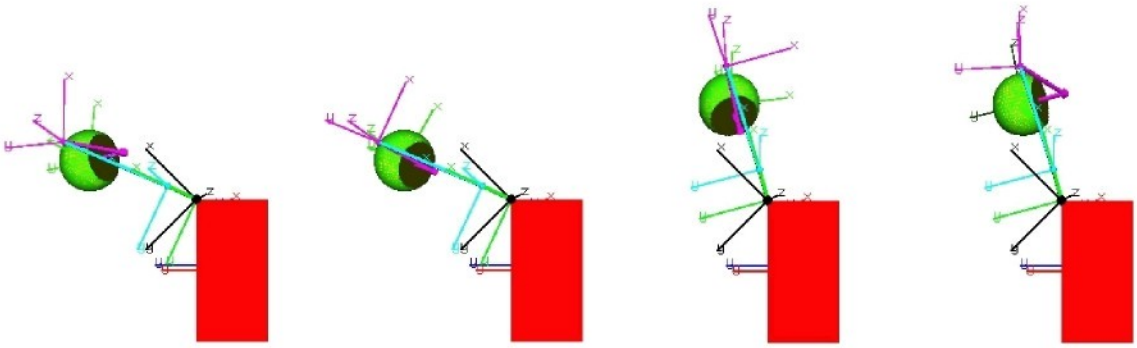


Obr. 2.11 Poloha kola při zatáčení

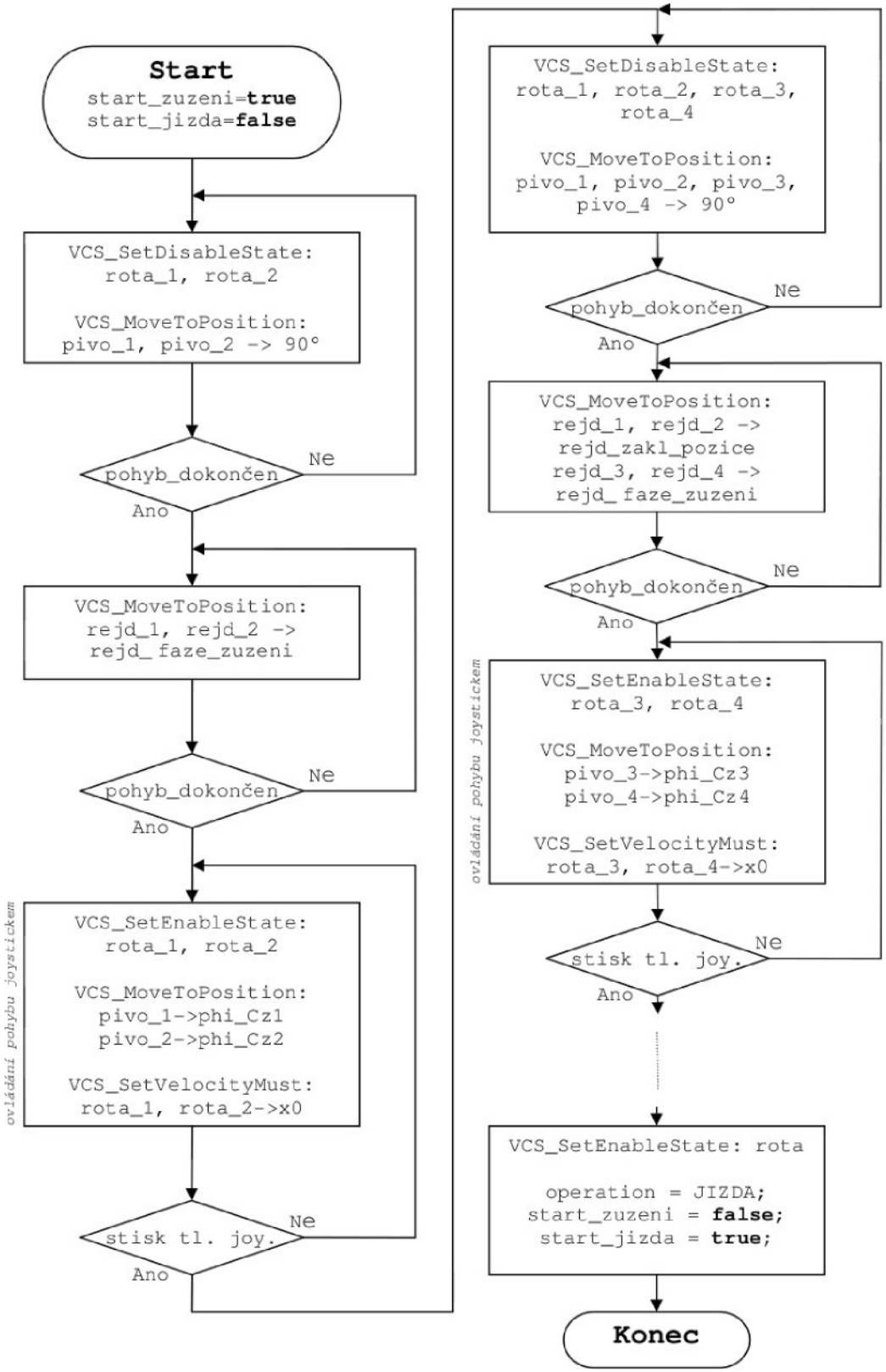
`zuzeni.cpp`

Další velice užitečný manévr je průjezd zúženým místem. Manévr *zúžení* obstarává soubor `zuzeni.cpp` a funkce **`void`** `zuzeni(`**`void`**`)`. Funkce `zuzeni` má podobnou strukturu jako funkce `enabling`. Každá z fází manévru je podmíněna úspěšným dokončením předchozí fáze a po dokončení manévru se opět nastaví jako aktuální prováděný manévr `JIZDA`, `start_jizda = `**`true`** a `start_zuzeni = `**`false`**.

Manévr začíná v základní konfiguraci a probíhá, jak je uvedeno na obr. 2.12 a diag. 2.2. V první fázi jsou nastaveny pohony předních kol do `Disable` a pivotace natočeny na hodnotu 90° (stejná pozice jako u `enabling`). Následně je rejd u předních kol nastaven na hodnotu `rejd_faze_zuzeni` a pohony předních kol nastaveny do `enable`. V této fázi se opět zapne řízení polohy kol v závislosti na poloze joysticku, aby bylo možné i v průběhu manévru podvozek ovládat. Když přední kola projedou zúžené místo, vrací se podle podobného předpisu do základní konfigurace a zbytek manévru se současně začíná opakovat pro kola zadní.



Obr. 2.12 Jednotlivé fáze manévru *zúžení*



Diag. 2.2 Manévr *zúžení*

krok.cpp

Dalším ze základních manévrů je pohyb robotizovaného podvozku chůzí. Struktura souboru `krok.cpp` je podobná jako u předchozích manévrů a i funkce `void krok(void)` je rozdělena do několika fází. Zda bude vykonání další fáze závislé na úspěšném dokončení předchozí fáze, nebo zda se budou fáze překrývat, bude upřesněno později na základě vybraného a na fyzickém modelu vyzkoušeného způsobu chůze. Viz kapitola Návrh manévru *chůze* - různé provedení.

text.cpp

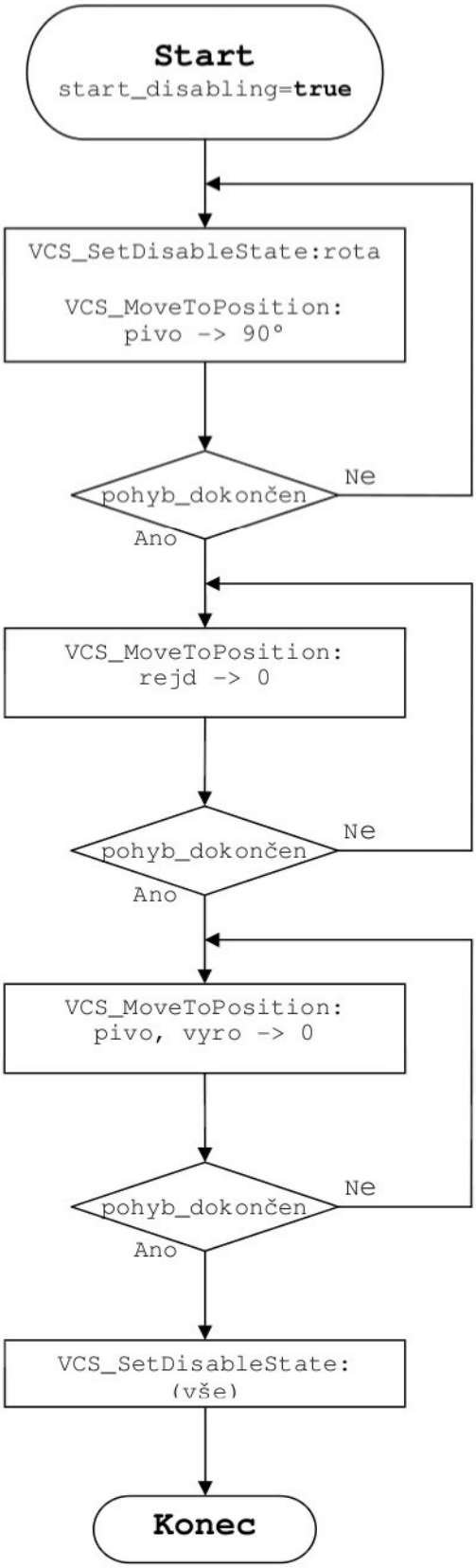
Ryze informativní charakter má soubor `text.cpp`. Ten a funkce v něm obsažené (`void setPerspectiveProjection()`, `void setOrthogonalProjection()`, `void printString(int x, int y, char *text)` a `void drawInfoText()`) slouží k výpisu informativního textu na obrazovku řídícího programu. Těmito informacemi například jsou: počet snímků za vteřinu, aktuální fáze aktuálního manévru, informace o dokončení pohybu, atd.

disabling.cpp

Soubor `disabling.cpp` a s ním související funkce `void disabling(void)` slouží k nastavení jednotlivých stupňů volnosti do nulové polohy a ukončení spojení mezi počítačem a řídící jednotkou, resp. řídícími jednotkami. Nastavení všech motorů do nulové polohy probíhá v opačném sledu než u funkce `enabling.cpp` podle diag. 2.3. Zkráceně tedy: motory pivotace se nastaví na hodnotu 90°, následně pak motory rejdu najedou na hodnotu nula a po úspěšném dokončení i motory vyrovnávání najedou na hodnotu nula. Po dokončení všech těchto pohybů se motory nastaví do `Disable` a dojde k ukončení řídícího programu.

extern.h

Poslední z řady souborů, ze kterých se skládá řídící program je soubor `extern.h`. Je to tzv. hlavičkový soubor, ve kterém jsou definovány globální proměnné používané ve více souborech řídícího programu.



Diag. 2.3 Fáze *disabling*

3 Simulace základních manévrů

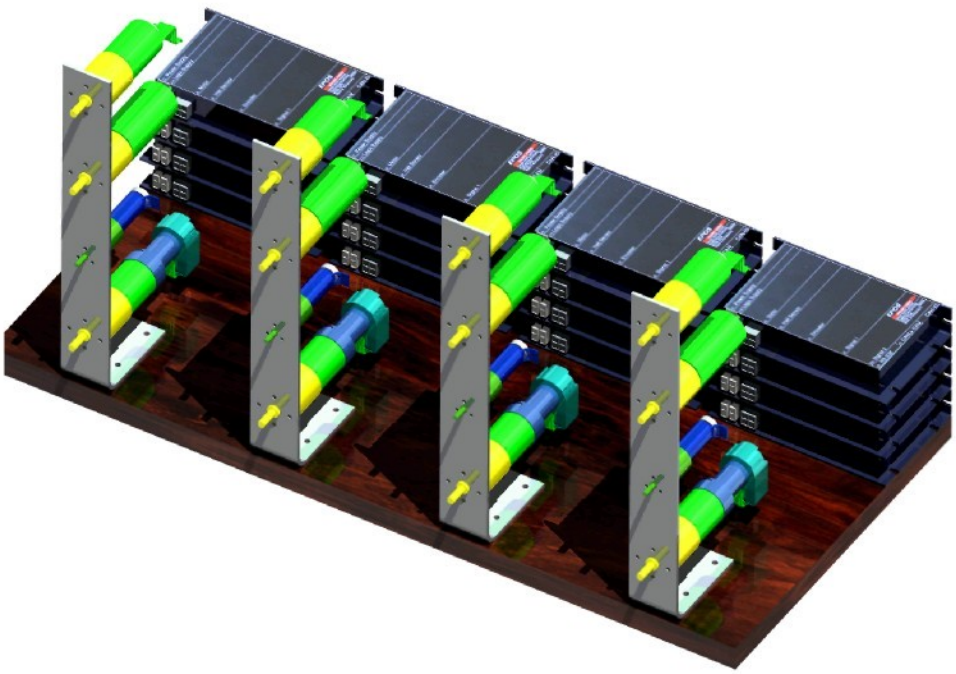
Tato část práce se zabývá tvorbou zjednodušeného modelu robotizovaného podvozku v softwaru ProEngineer a jeho exportem do prostředí ADAMS. V programu ADAMS byly následně provedeny simulace základních manévrů, které slouží k ověření pohybových schopností podvozku. Dále bylo nutné potvrdit či vyvrátit správnost dimenzování elektrických pohonů. Výběr pohonných jednotek byl proveden v diplomové práci [8].

Animace jednotlivých manévrů jsou zveřejněny na internetových stránkách katedry mechaniky, pružnosti a pevnosti na adrese www.kmp.tul.cz/lide/denk.

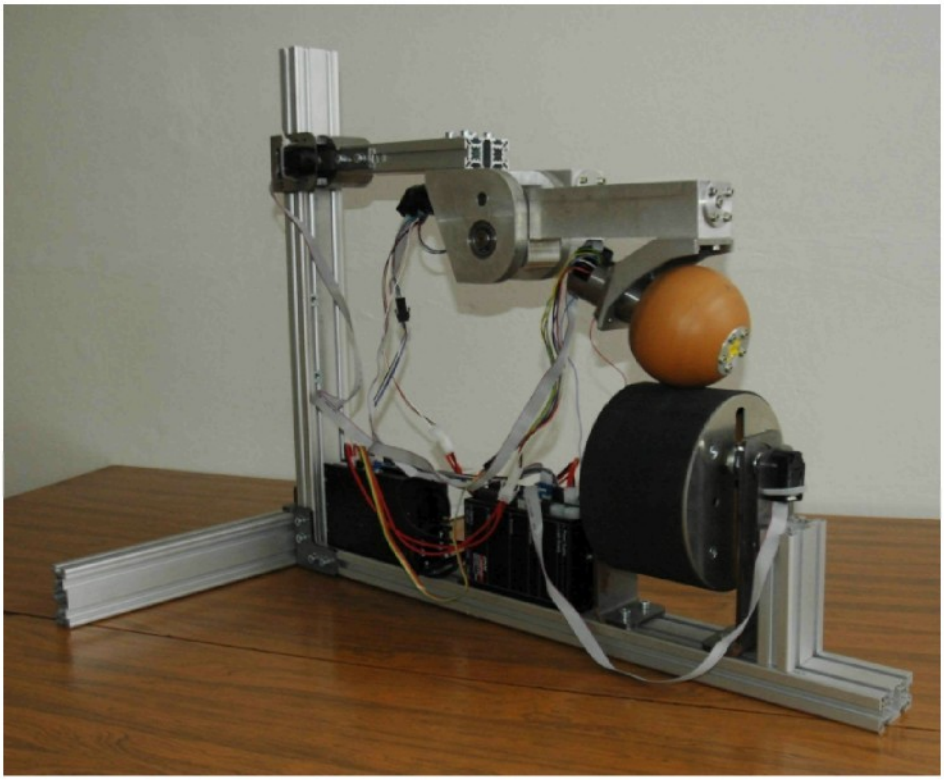
3.1 Ověření pohybu motoru s a bez zátěže

V období tvorby této práce nebyl k dispozici reálný model robotizovaného podvozku. Byly k dispozici pouze pohony pro všechny stupně volnosti a měřící stand - zjednodušený model jedné nohy podvozku, viz obr. 3.2. Bylo tedy možné prověřit vztah mezi pohybem motorů bez zátěže a se zátěží.

Ověření pohybu s a bez zátěže bylo provedeno na motoru poháněním stupeň volnosti, který ovládá vyrovnávání terénu. Motor, který měl simulovat pohyb bez zátěže, byl nainstalovaný v držáku, obr. 3.1. Druhý motor byl namontován do reálného zjednodušeného modelu nohy podvozku, obr. 3.2. Na část standu, která představuje rám podvozku, bylo připevněno postupně 0, 3 a 5 závaží o hmotnosti 0,75kg a byl realizovaný stejný pohyb, viz dále v tabulce 3.1.



Obr. 3.1 Držák motorů

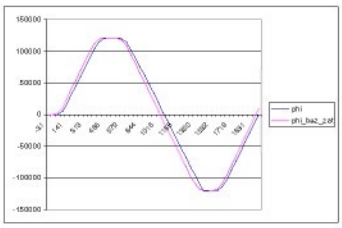
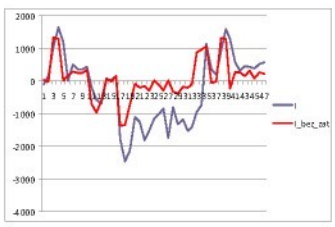
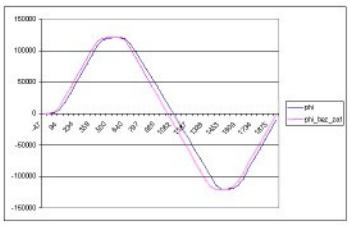
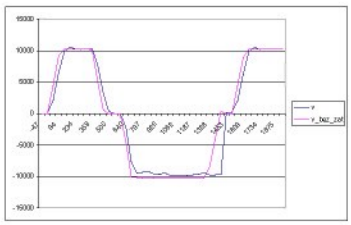
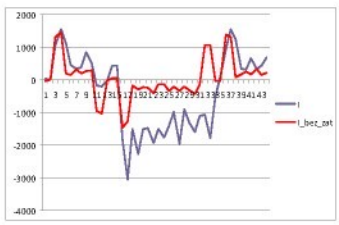


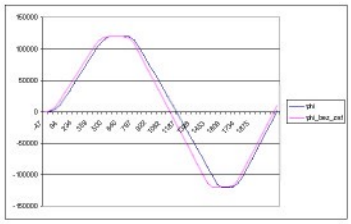
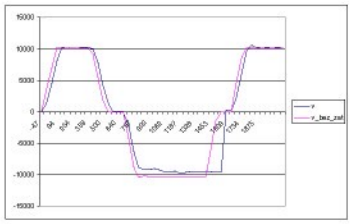
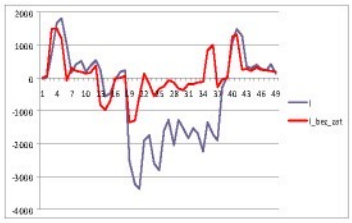
Obr. 3.2 Měřící stand

Z tabulky 3.1 vyplývá, že díky vlastní regulační smyčce motoru není mezi pohybem motoru se zátěží a bez zátěže velký rozdíl. To však platí za předpokladu, že nejsou překročeny nějaké omezující parametry pohonu, např.: maximální moment, maximální proud, atd.

Výchylka 120000 qc přestavuje krajní polohu pohybu nohy a rychlost motoru 10250 min^{-1} je maximální rychlost motoru ***EC* 32** zatíženého odporem planetové převodovky.

Tabulka 3.1 Porovnání pohybu motorů s a bez zátěže

záv.	Poloha [qc]	Rychlost [1/min]	Proud [mA]
0			
3			

5			

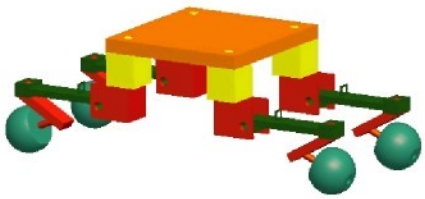
Z výše uvedených obrázků je patrné, že při zvyšování zátěže motoru roste proud, kterým je daný motor prostřednictvím řídící jednotky napájen. Tento proud je u motoru ***EC-32*** omezen hodnotou maximálního kontinuálního proudu 5,82 A uvedeného v katalogových listech.

3.2 Tvorba zjednodušeného modelu podvozku

Pro potřeby simulací základních manévrů podvozku musel být vytvořen simulační model. Pro tvorbu geometrie tohoto modelu byl použit software ProEngineer, ze kterého byl model následně exportován do MBS ADAMS.

3.2.1 Vytvoření geometrie v ProE, export do MBS ADAMS

V softwaru ProEngineer byl vytvořen zjednodušený model robotizovaného podvozku, obr. 3.3. Model byl vytvořen na základě skutečné geometrie uvedené v [8], obr. 3.4.

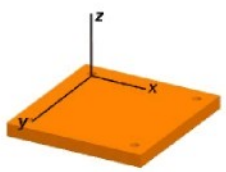
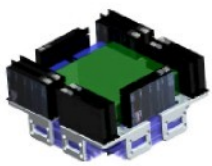


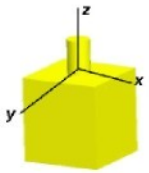
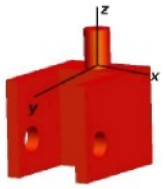
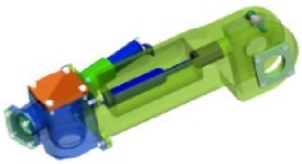
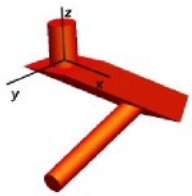
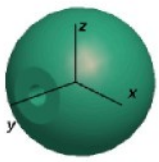
Obr. 3.3 Zjednodušený model podvozku v SW ProEngineer

Obr. 3.4 Skutečná geometrie virtuálního modelu podvozku

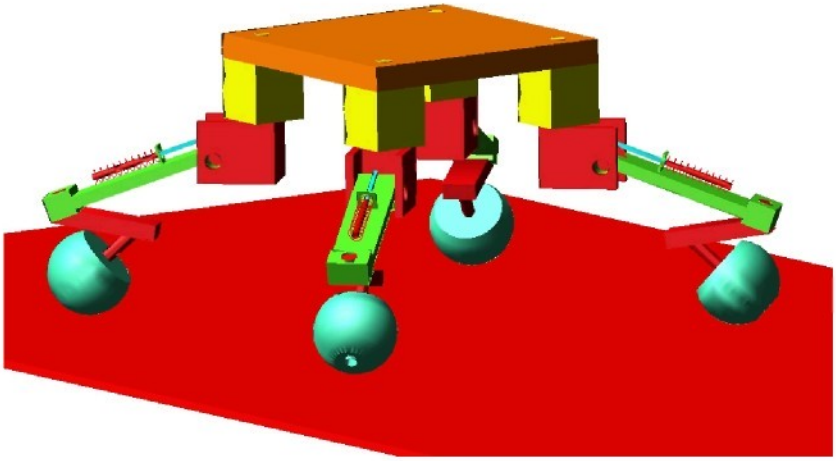
Jednotlivé části jak zjednodušeného modelu, tak i virtuálního podvozku jsou znázorněny v tabulce 3.2. V tabulce 3.2 jsou dále uvedeny základní hmotnostní charakteristiky, hmotnosti a souřadnice těžišť vzhledem k zobrazeným vztažným souřadnicovým systémům.

Tabulka 3.2 Porovnání virtuálního a zjednodušeného modelu podvozku

	Zjednodušený model	Virtuální model	Matice setrvačnosti $[\mathrm{kg\,mm^2}]$, hmotnost, souř. těžiště
Podvozek a baterie			
1			$\mathbf{I} = \begin{bmatrix} 3.23\cdot10^5 & -2.07\cdot10^5 & -3.23\cdot10^4 \\ -2.07\cdot10^5 & 3.26\cdot10^5 & -3.10\cdot10^4 \\ -3.23\cdot10^4 & -3.10\cdot10^4 & 6.05\cdot10^5 \end{bmatrix}$ $m = 12.40\,\mathrm{kg}$, $\mathbf{r}_T = [15.3, 16.2, 5.5]$

Těleso rejdu			
2			$\mathbf{I} = \begin{bmatrix} 770.76 & -161.94 & -279.02 \\ -161.94 & 1332.01 & -156.29 \\ -279.02 & -156.29 & 969.55 \end{bmatrix}$ $m = 0.49\,\text{kg}$, $\mathbf{r}_T = [0.3, 1.2, -34.5]$
Těleso vyrovnávání			
3			$\mathbf{I} = \begin{bmatrix} 9728.43 & -2943.56 & 1067.83 \\ -2943.56 & 5675.22 & 2493.39 \\ 1067.83 & 2493.39 & 1.13 \cdot 10^4 \end{bmatrix}$ $m = 2.33\,\text{kg}$, $\mathbf{r}_T = [0.6, 12.4, -35.5]$
Noha			
			$\mathbf{I} = \begin{bmatrix} 2.88 \cdot 10^4 & -1039.39 & 190.37 \\ -1039.39 & 1619.53 & 4478.75 \\ 190.37 & 4478.75 & 2.81 \cdot 10^4 \end{bmatrix}$ $m = 1.50\,\text{kg}$, $\mathbf{r}_T = [-1.5, 102.5, 6.6]$
Těleso pivotace			
5			$\mathbf{I} = \begin{bmatrix} 6925.77 & -4.69 & -7.01 \\ -4.69 & 4638.30 & -1634.00 \\ -7.01 & -1634.00 & 2711.54 \end{bmatrix}$ $m = 1.13\,\text{kg}$, $\mathbf{r}_T = [0.0, -40.3, -29.2]$
Kolo			
6			$\mathbf{I} = \begin{bmatrix} 759.09 & -1.25 & 2.20 \\ -1.25 & 894.22 & 0 \\ 2.20 & 0 & 759.30 \end{bmatrix}$ $m = 0.66\,\text{kg}$, $\mathbf{r}_T = [0.0, 4.3, 0.0]$

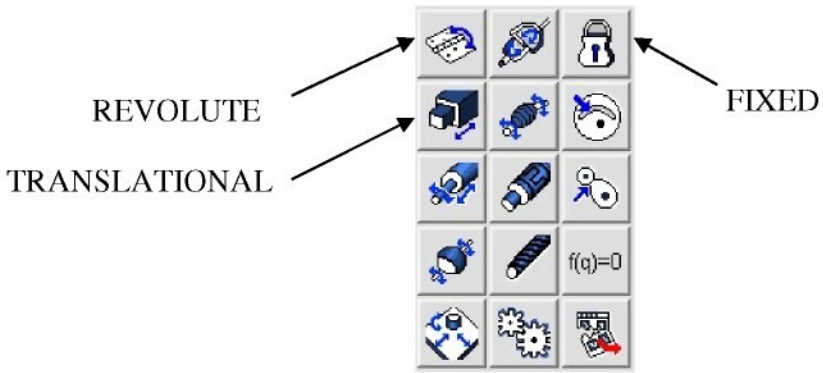
Sestava zjednodušeného modelu podvozku z ProEngineer byla uložena do formátu Parasolid CAD file (*.x_t). Tento soubor byl následně importován do MBS ADAMS. Zjednodušená sestava podvozku byla relativně jednoduchá, a proto bylo možné převést do MBS ADAMS celou sestavu najednou. V opačném případě by bylo nutné provést export po jednotlivých částech.



Obr. 3.5 Zjednodušený model v MSC ADAMS

Při tvorbě simulačního modelu v MBS ADAMS bylo postupováno následovně. Po importu sestavy do prostředí ADAMS view bylo nutné modifikovat hmotnostní charakteristiky v souladu s tabulkou 3.2 a dále byly modifikovány polohy těžišť jednotlivých částí.

K popsání vzájemných pohybů jednotlivých částí podvozku byly použity základní vazby programu ADAMS, jejich přehled je uveden na obr. 3.6.



Obr. 3.6 ADAMS základní vazby

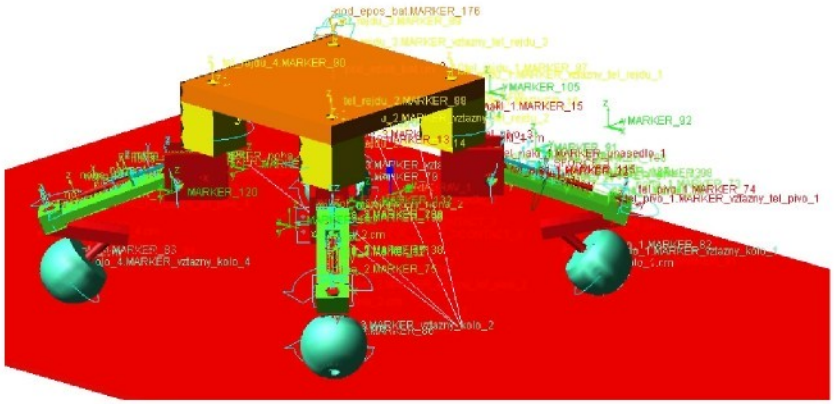
Základem pro sestavení modelu bylo těleso v tab. 3.2 označené číslicí 1, ke kterému bylo vazbou FIXED (nepohyblivou) připevněno těleso rejdu (v tab. 3.2 označeno jako 2). Vazba FIXED byla použita z důvodu, že nedochází ke vzájemnému pohybu těchto dvou částí. Pro připojení dalšího tělesa nazvaného těleso naklápění (v tab. 3.2, 3) k tělesu rejdu bylo použito rotační vazby REVOLUTE. Stejná vazba byla použita i pro připojení dalšího tělesa (noha, 4). Na konci tělesa nohy je připojeno těleso pivotace (5). Zde je opět použita rotační vazba REVOLUTE. Posledním tělesem každé nohy je kolo, které je také připojeno vazbou REVOLUTE.

Do každé rotační vazby byl přidán rotační pohon (ROTATIONAL JOIT MOTION), který představuje příslušný elektromotor.

V neposlední řadě byl definován kontakt mezi koly a pojezdovou plochou. Vazba, která definuje kontakt mezi dvěma tělesy, se nenachází mezi základními vazbami z obr. 3.6. Najdeme ji v sekci CONNECTORS, kde se dále nacházejí pružiny, tlumiče, atd. Vazba CONTACT byla použita s defaultními hodnotami.

Model byl dále doplněn o ozubený hřeben, pružinu a tlumič. Poloha ozubeného hřebenu vůči tělesu nohy je definována pomocí vazby TRANSLATIONAL a pomocí vazby RACK AND PINION, což představuje vazbu ozubeného hřebenu na ozubené kolo.

Na následujícím obrázku je znázorněn celý model v SW ADAMS. Jsou na něm také vyznačeny jednotlivé vazby, markery, které určují polohy těžišť a vztažné souřadnicové systémy a také pojezdová plocha.



Obr. 3.7 Model v SW ADAMS s vyznačenými vazbami, markery a pojezdovou plochou

3.3 Návrh manévru *chůze* – různá provedení

Manévr *chůze* je jedním z nejsložitějších manévrů, které má robotizovaný podvozek, v souladu s kapitolou 1.3 Volba koncepce, vykonávat.

V této fázi bylo nutné prostudovat možnosti pohybu čtyřnohého robotizovaného podvozku kráčením. Na internetu je k nalezení mnoho případů. Většina z nich popisuje pohyb robotů s více nohami. Jako příklad uveďme robot Halluc II, popsaný v kapitole 1.1.1. Robot Halluc disponuje osmi nohami, což činí manévr chůze jednodušším.

Náš robotizovaný podvozek disponuje jen čtyřmi nohami, což manévr chůze komplikuje. Absence dalších nohou může vést k tomu, že při zvednutí jedné nohy při nesprávné konfiguraci nohou ostatních může dojít ke ztrátě stability celého podvozku. Chceme-li tedy zajistit dostatečnou stabilitu při manévru *chůze*, je nutné, aby se těžiště podvozku nacházelo v trojúhelníku tvořeného body dotyku kol, které jsou v daném okamžiku v kontaktu s podložkou.

Další možnost pohybu podvozku je taková, kdy využijeme dynamických účinků pohybu podvozku, což nám dovolí zvednout do vzduchu až dvě nohy a tím výrazně chůzi zrychlit. To má své klady i zápory, viz dále.

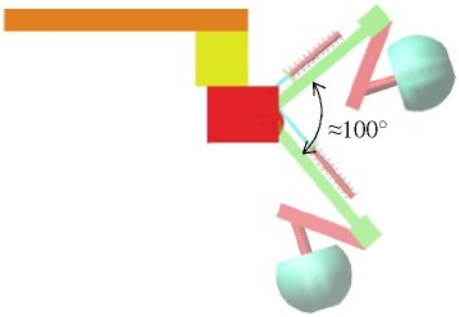
3.3.1 Změna převodovky u motoru vyrovnávání

Při simulacích manévru *chůze* vyvstal požadavek na změnu převodovky u motoru, který ovládá vyrovnávání terénu.

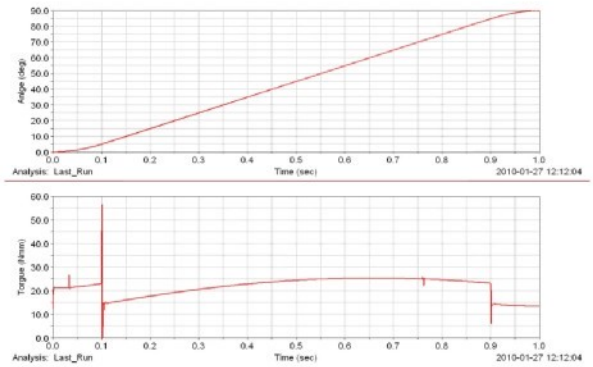
Jedná se o motor ***EC* 32** 118889 s maximální rychlostí přibližně $n_{\text{motor}} = 10250\ \text{min}^{-1}$. Požadavek, podle kterého měla být vybrána převodovka, byl následující: natočení podvozkové nohy v rozmezí $0 - 100°$ za 1s. To odpovídá otáčkám $n_{\text{teor.}} = \frac{100}{360} \doteq 0{,}278\ \text{s}^{-1} \doteq 16{,}667\ \text{min}^{-1}$. Mezi samotnou nohu a převodovku, u které má dojít ke změně, je vložena šneková převodovka s převodový poměrem $i_{\text{šnek}} = 20$. Výsledný maximální převodový poměr planetové převodovky i_{planet}, za předpokladu splnění výše uvedeného požadavku, je dán vzorcem:

$$i_{\text{planet}} \leq \frac{n_{\text{motor}}}{n_{\text{teor.}} n_{\text{šnek}}} \quad \Rightarrow \quad i_{\text{planet}} \leq 30{,}75\,.$$

Z katalogu firmy Maxon byla vybrána převodovka **GP 32 C** 166936 o převodovém poměru $i = 23$. S touto převodovkou byla následně provedena simulace požadovaného pohybu v MSC ADAMS, obr. 3.8 a obr. 3.9, s výsledkem, že maximální moment v průběhu pohybu nepřesahuje maximální kontinuální moment motoru ***EC* 32** 37,2 Nmm.



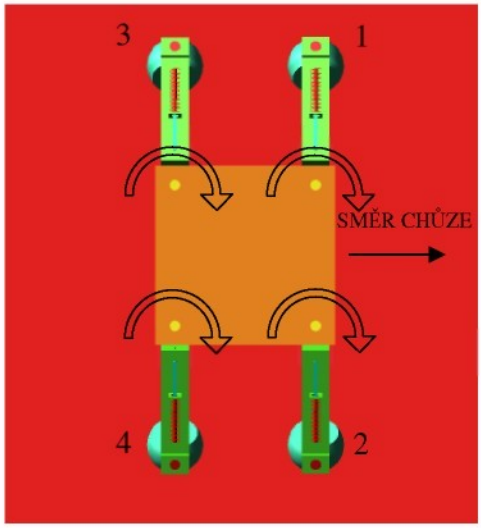
Obr. 3.8 Požadovaný rozsah pohybu



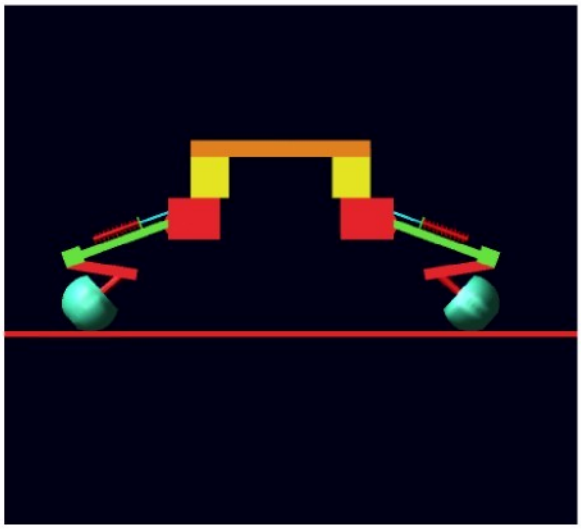
Obr. 3.9 Předpis a výsledek simulace

3.3.2 Simulace jednotlivých druhů chůze

Během testování byly provedeny simulace dvou výše zmíněných základních typů chůze. Pomalejšího, ale v každé fázi pohybu stabilního, a rychlejšího, který musí překonat fáze nestabilní.



Obr. 3.10 Rejd



Obr. 3.11 Vyrovnávání

Základní konfigurace, od které jsou během manévru chůze odměřovány úhly, je znázorněna na obr. 3.10 a 3.11. Úhel vyrovnávání je vzhledem k vodorovné poloze pootočen o 20°. Jednotlivé úhly jsou také znázorněny na obr. 1.12 a 1.13.

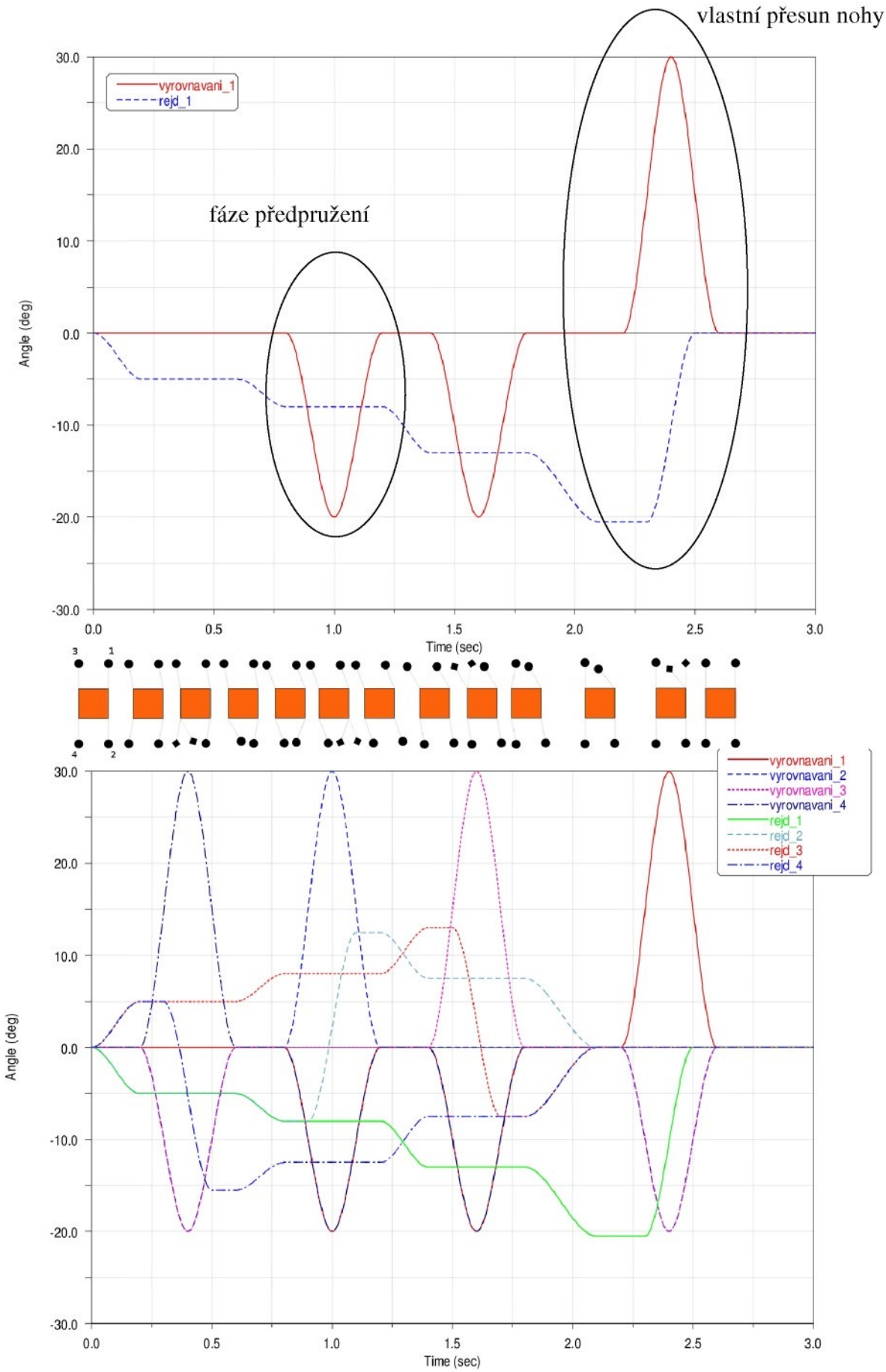
Pomalejší typ chůze

U tohoto typu manévru *chůze* se těžiště podvozku vždy nachází v trojúhelníku tvořeném třemi bodu dotyku kol s podložkou. Je tak docíleno toho, že nedochází k nadměrnému naklápění rámu podvozku během manévru.

Při ztrátě kontaktu jedné nohy s pojezdovou plochou dochází ke změně zatížení ostatních tří nohou. To má za následek jiné zatížení pružin spojených s ozubenými hřebeny, a tedy i tendenci ke změně vodorovné horizontální polohy. Tomu je bráněno cíleným předpružením ostatních pružin, vyznačeno na obr. 3.12. Když například dochází k přesunu nohy 1, změní se tím zatížení ostatních nohou, a proto u nohou 2 a 3 dochází k cílenému předpružení pružin, které má za následek udržení relativně vodorovné polohy.

Na obr. 3.12 jsou znázorněny průběhy rejdu a vyrovnávání, resp. rejdů a vyrovnávání během tohoto manévru.

Animace k prohlédnutí na www.kmp.tul.cz/lide/denk.



Obr. 3.12b Manévr *chůze*

Na dalších obrázcích jsou znázorněny výsledky simulací. Jedná se o hodnoty otáček a momentů potřebných během manévru.

První graf na obr. 3.13 znázorňuje průběh úhlu pro vyrovnávání terénu první nohy podvozku. Následující graf je derivací předchozího a znázorňuje tedy úhlovou rychlost natáčení nohy podvozku ve stupních za sekundu. Další graf udává průběh stejné rychlosti, ale v otáčkách za minutu, což jsou jednotky, v kterých jsou definovány katalogové otáčky motorů. Následuje převod otáček s ohledem na vložené převodovky (šneková, planetová). Jsou zde uvedeny dvě varianty: otáčky s uvažováním původní planetové převodovky (1:86) a otáčky s novou převodovkou (1:23).

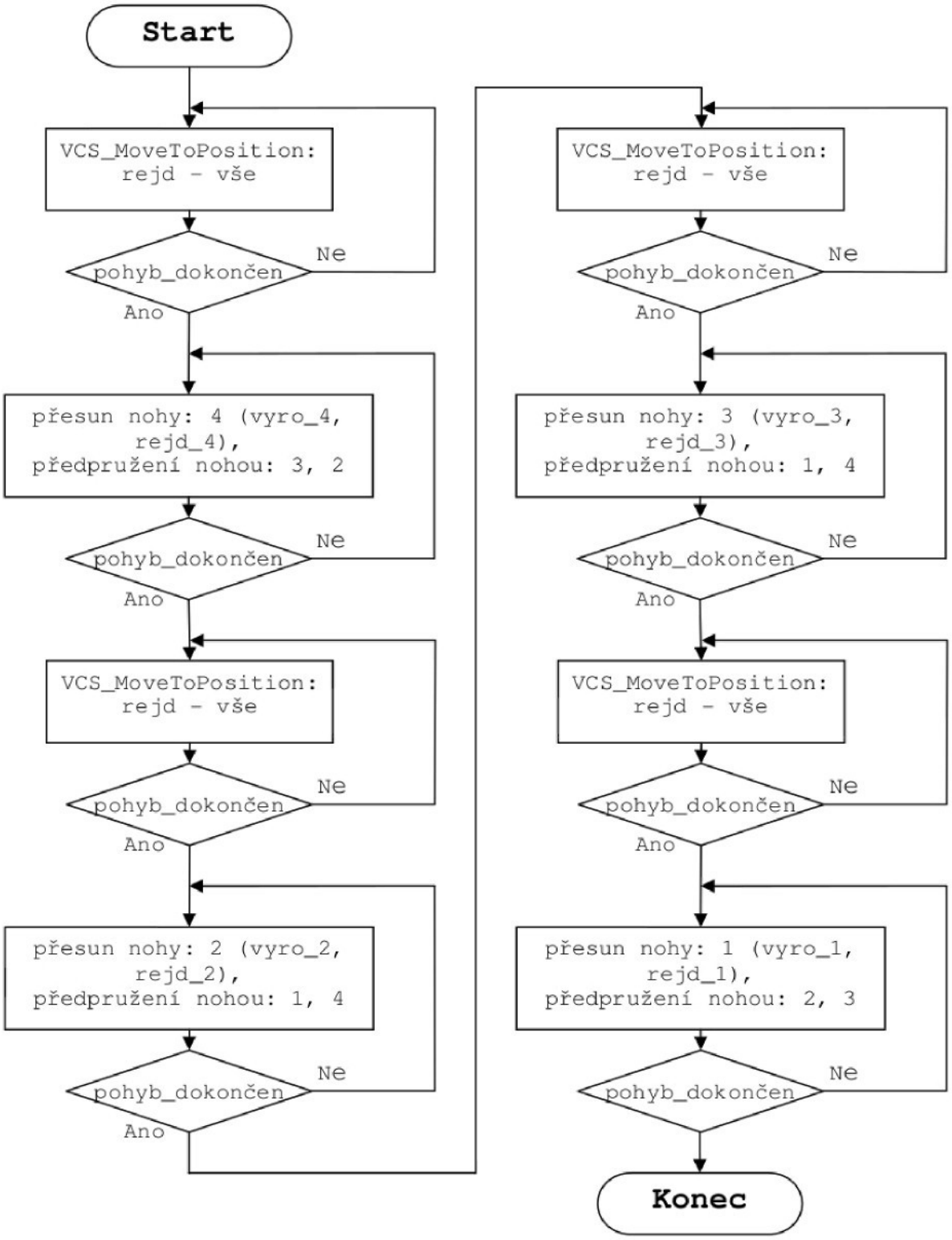
Maximální kontinuální otáčky motoru pohánějící vyrovnávání terénu jsou 13600 ot/min. Je tedy zřejmé, že motor je schopen, co se týče otáček, tento manévr vykonat pouze s novou převodovkou. To však neznamená, že je původní převodovka nedostačující. Tento druh chůze má výhodu v tom, že jednotlivé fáze následují až po skončení fází předchozích. Je tedy vcelku jednoduché upravit algoritmus tak, že bude vyhovující i převodovka s větším převodovým poměrem.

Na obr. 3.14 je totéž co na 3.13 ale pro motor rejdu. Z grafu vyplývá, že z hlediska otáček je motor rejdu dimenzován správně.

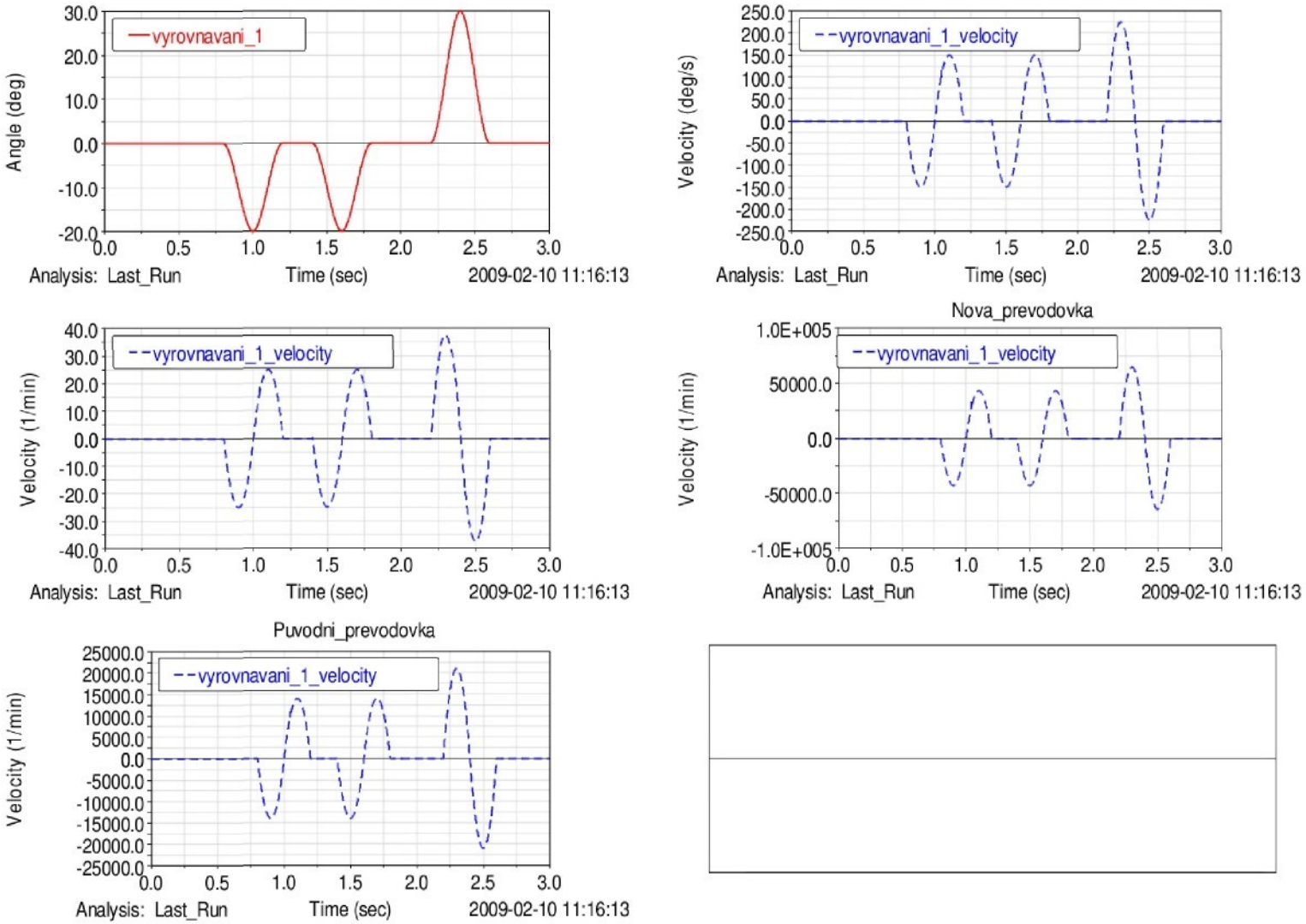
Na obr. 3.15 jsou převážně znázorněny průběhy momentů potřebných pro pohyb nohou podvozku. Moment je opět přepočítán pro původní a novou převodovku. Z průběhu momentu je vytvořen výřez v čase největšího zatížení, tedy v době, kdy dochází k přemístění daného kola. Maximální moment je v obou případech pod hranicí 37,5 Nmm (mNm), což je maximální kontinuální moment motoru ***EC* 32**.

Moment potřebný pro změnu rejdu při manévru *chůze* znázorňuje obr. 3.16. Z průběhu je opět vytvořen výřez. Ani v tomto případě nedochází k překročení omezující hodnoty momentu motoru rejdu 35 Nmm (35 mNm). Z grafu byl vytvořen výřez v času od 2,25s do 2,55s, i když se moment přibližně v čase 1,6s jeví vyšší. Tato špička momentu nebyla uvažována, protože je způsobena třením kola po podložce a na motor se z důvodu samosvornosti šnekové převodovky nepřenáší.

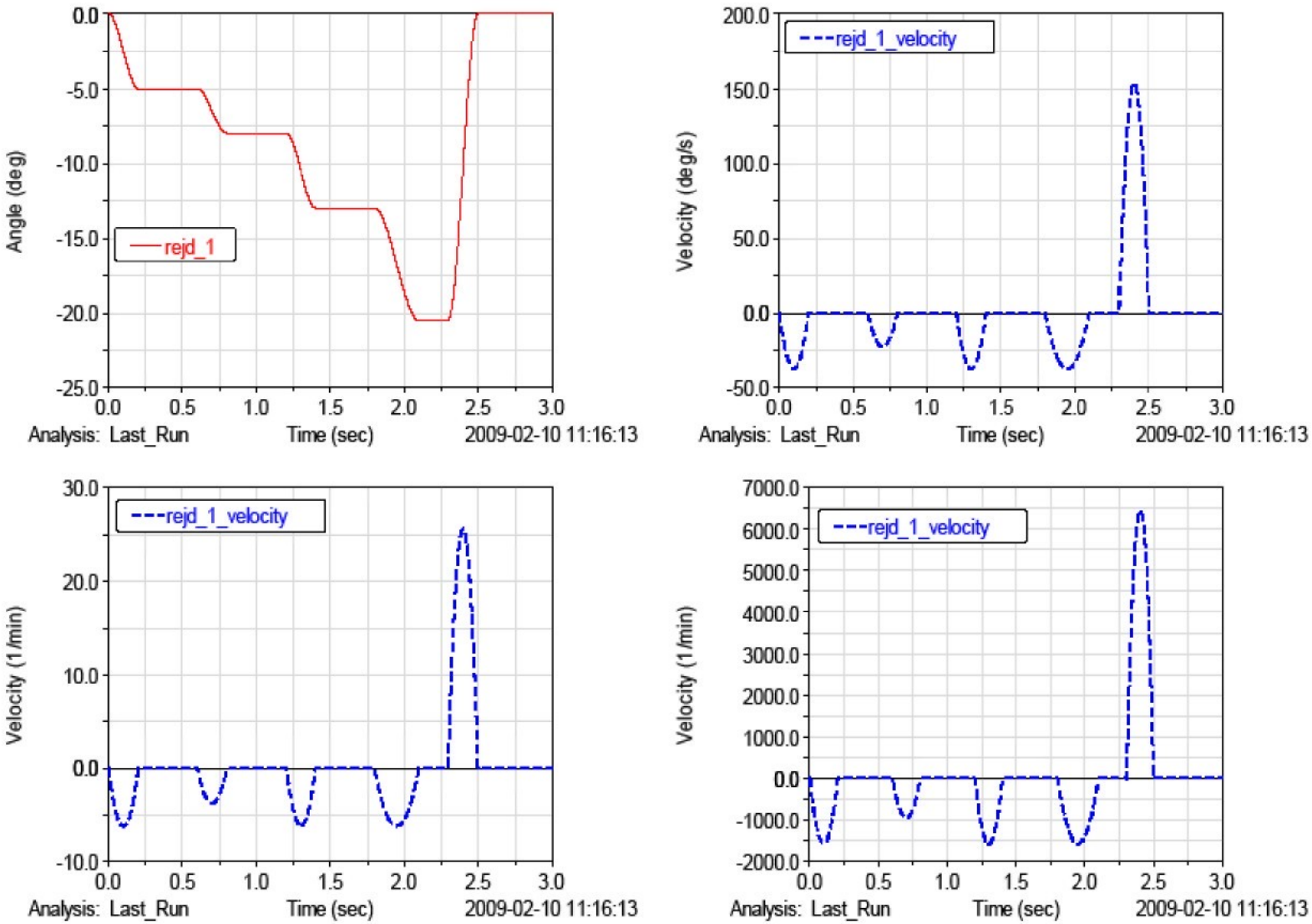
Z obr. 3.17 vyplývá jeden z nedostatků tohoto typu chůze a to, že dopředná rychlost chůze není příliš konstantní. Průměrná rychlost simulované chůze je $42{,}5\,\mathrm{mm\,s^{-1}} = 0{,}0425\,\mathrm{m\,s^{-1}} = 0{,}153\,\mathrm{km\,h^{-1}}$.



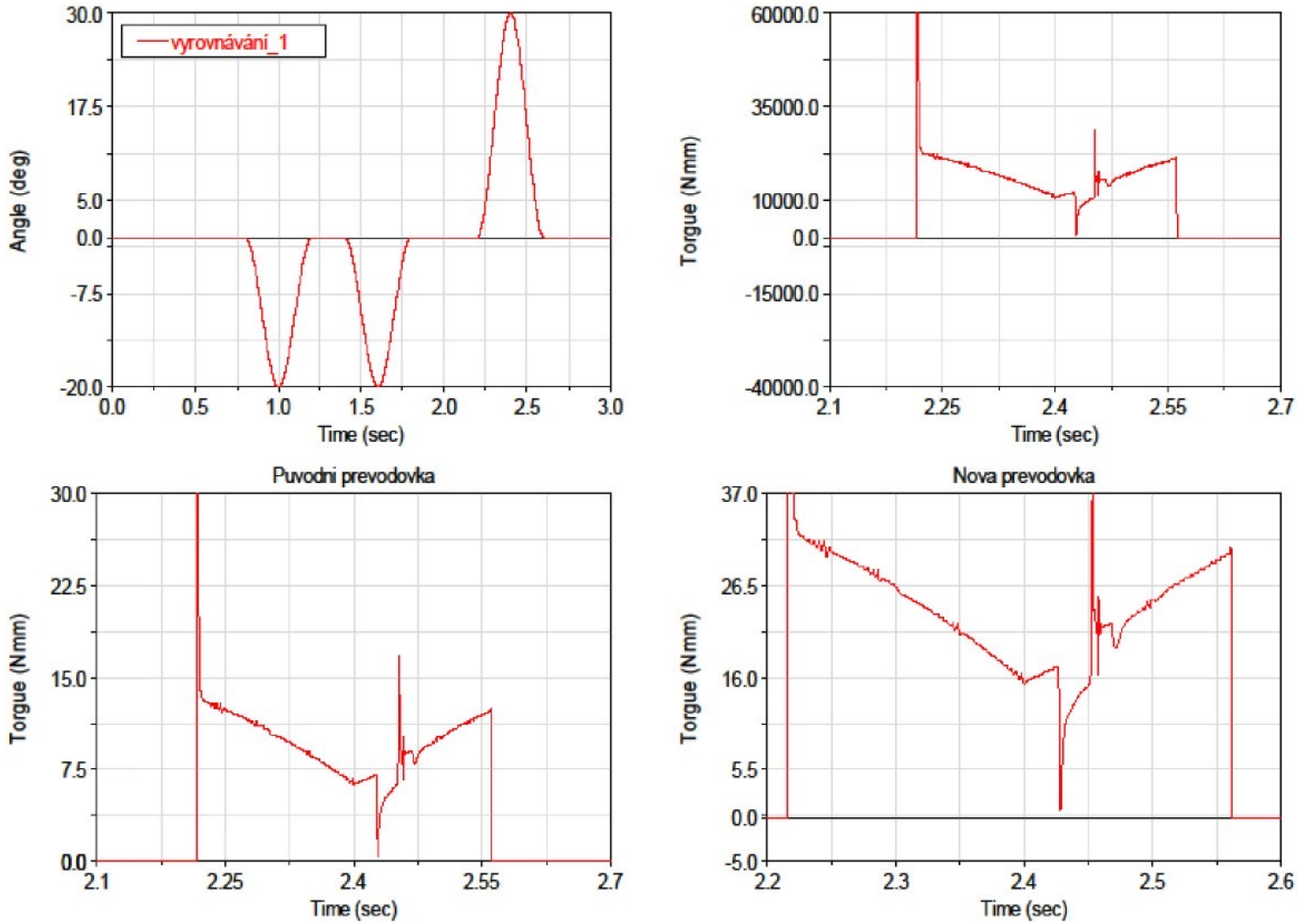
Diag. 3.3 Fáze pomalejší *chůze*



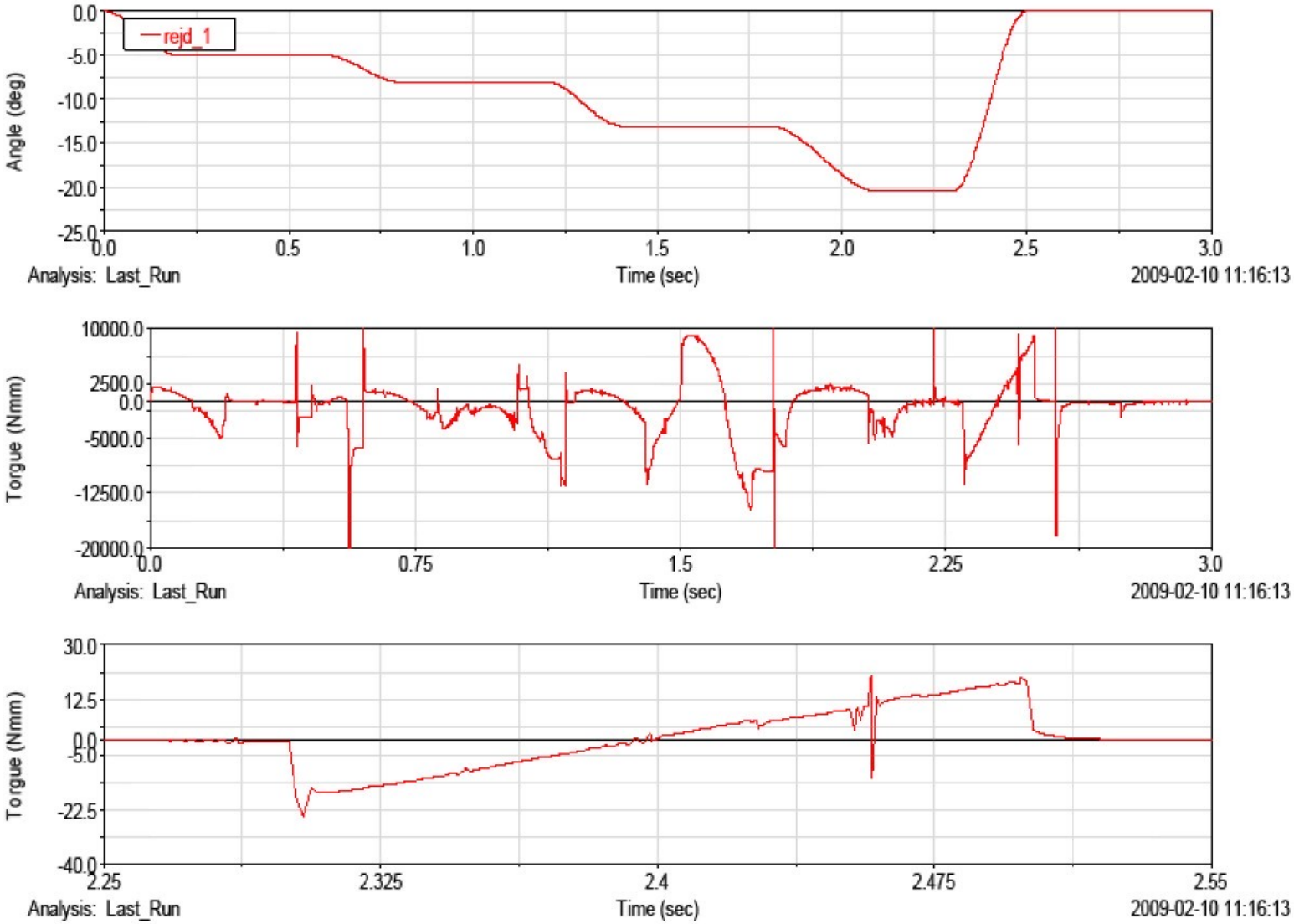
Obr. 3.13 Kinematika pohonu pro vyrovnávání během manévru



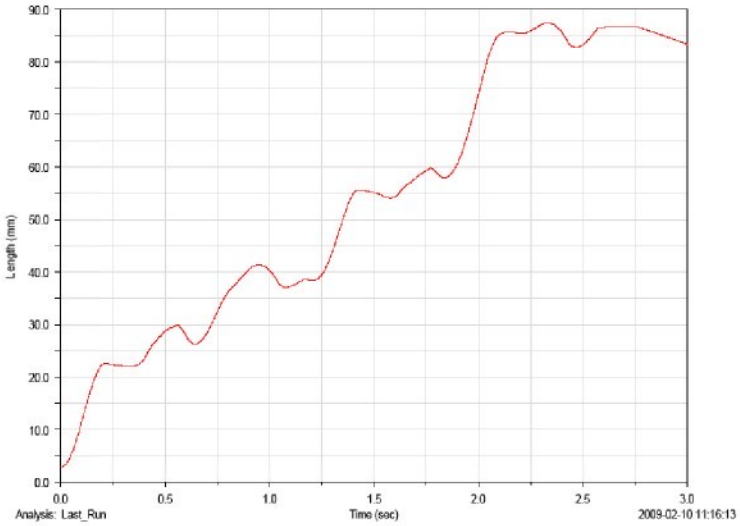
Obr. 3.14 Kinematika pohonu pro rejd během manévru



Obr. 3.15 Momentové zatížení pohonu pro vyrovnávání během manévru



Obr. 3.16 Momentové zatížení pohonu pro rejd během manévru

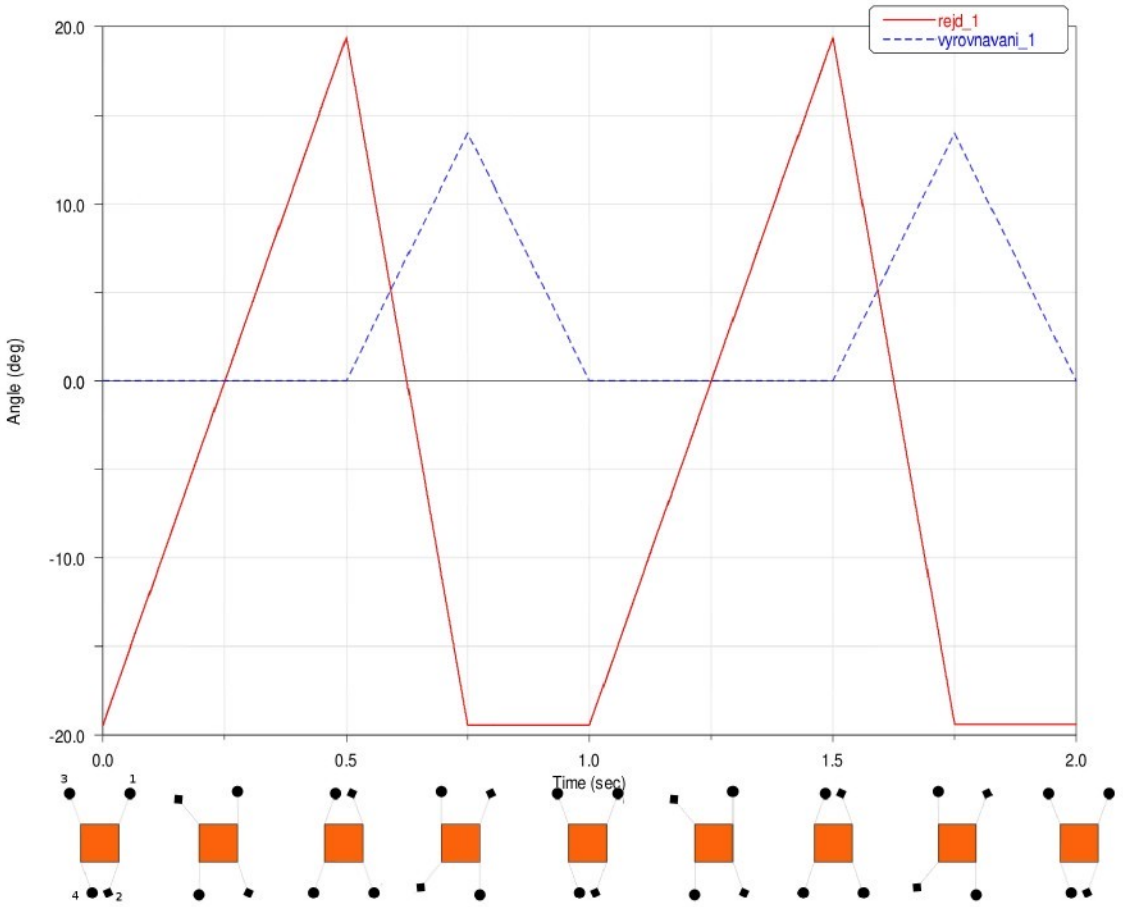


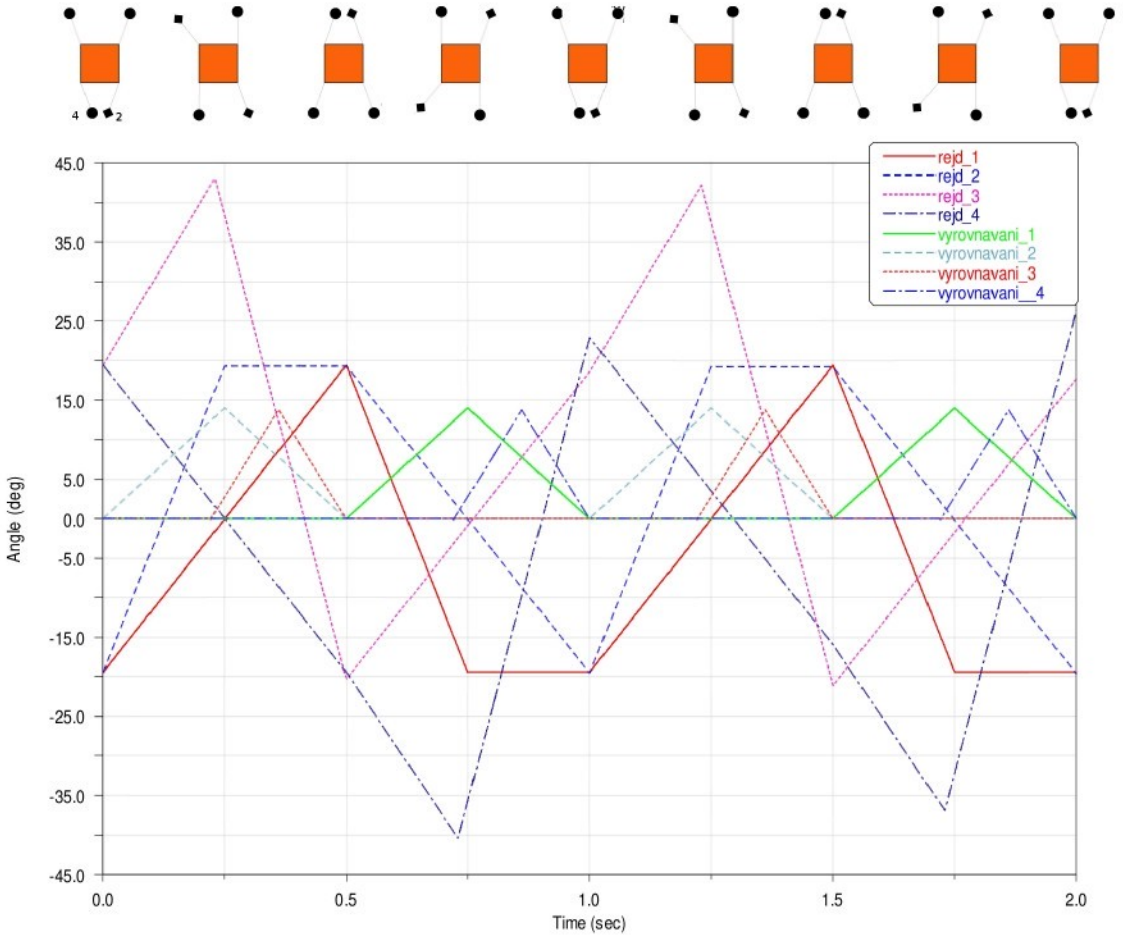
Obr. 3.17 Pozice těžiště podvozku ve směru *chůze*

Rychlejší typ chůze

U tohoto typu chůze už se nenachází v každém okamžiku těžiště uvnitř trojúhelníku tvořeného body dotyku ostatních kol, ale dochází k situacím, kdy až dvě kola nejsou ve stejný okamžik v dotyku s pojezdovou plochou.

Průběhy vyrovnávání a rejdu při tomto manévru jsou znázorněny na obr. 3.18.

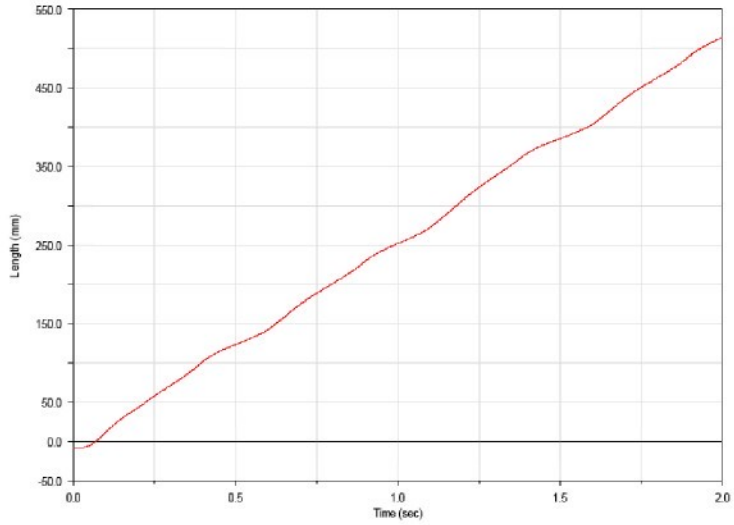




Obr. 3.18 Manévr *chůze*

V tomto případě nebylo (jako v případě předchozím) použito řízení polohy jednotlivých stupňů volnosti. Stupně volnosti jsou řízeny pomocí rychlostí. Způsob řízení rychlostí byl zvolen, protože se zde nečeká na dokončení předchozího pohybu, ale jednotlivé fáze kroku se překrývají. Další výhoda řízení manévru pomocí rychlostí je ve snadnější implementaci do řídícího programu.

Z obr. 3.18 je také patrné, že rejd u všech nohou je neustále v pohybu, z čehož vyplývá relativně konstantní dopředná rychlost, obr. 3.19, která činí $237{,}5\,\mathrm{mm\,s^{-1}} = 0{,}2375\,\mathrm{m\,s^{-1}} = 0{,}855\,\mathrm{km\,h^{-1}}$.



Obr. 3.19 Pozice těžiště podvozku ve směru *chůze*

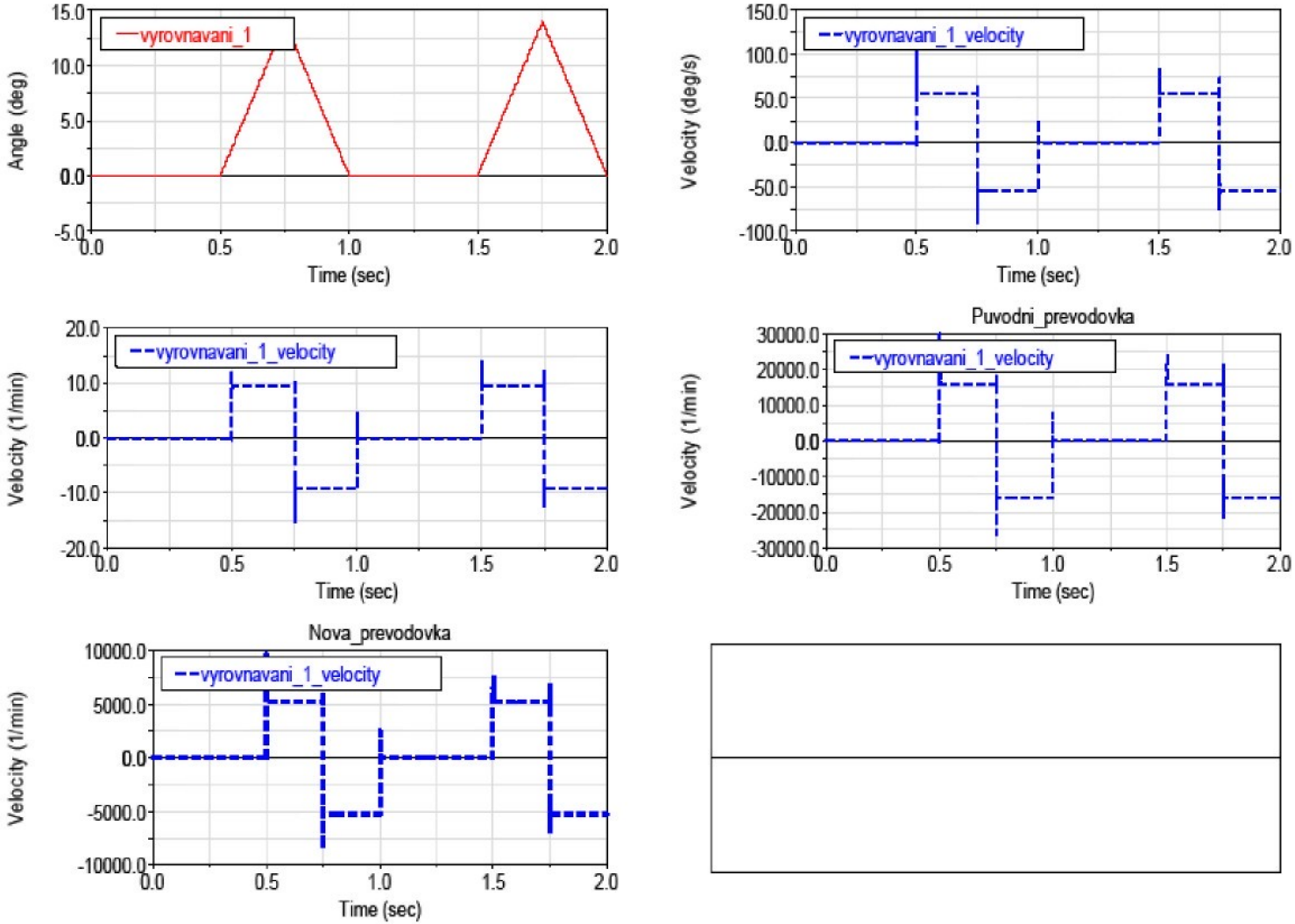
Z obr. 3.20 je patrné, že takto definovaný manévr, co se týče otáček, je motor vyrovnávání schopen provést pouze s novou planetovou převodovkou.

Otáčky motoru rejdu, obr. 3.21 jsou při tomto typu chůze relativně nízké, a proto nepředstavují omezení pro pohyb motoru.

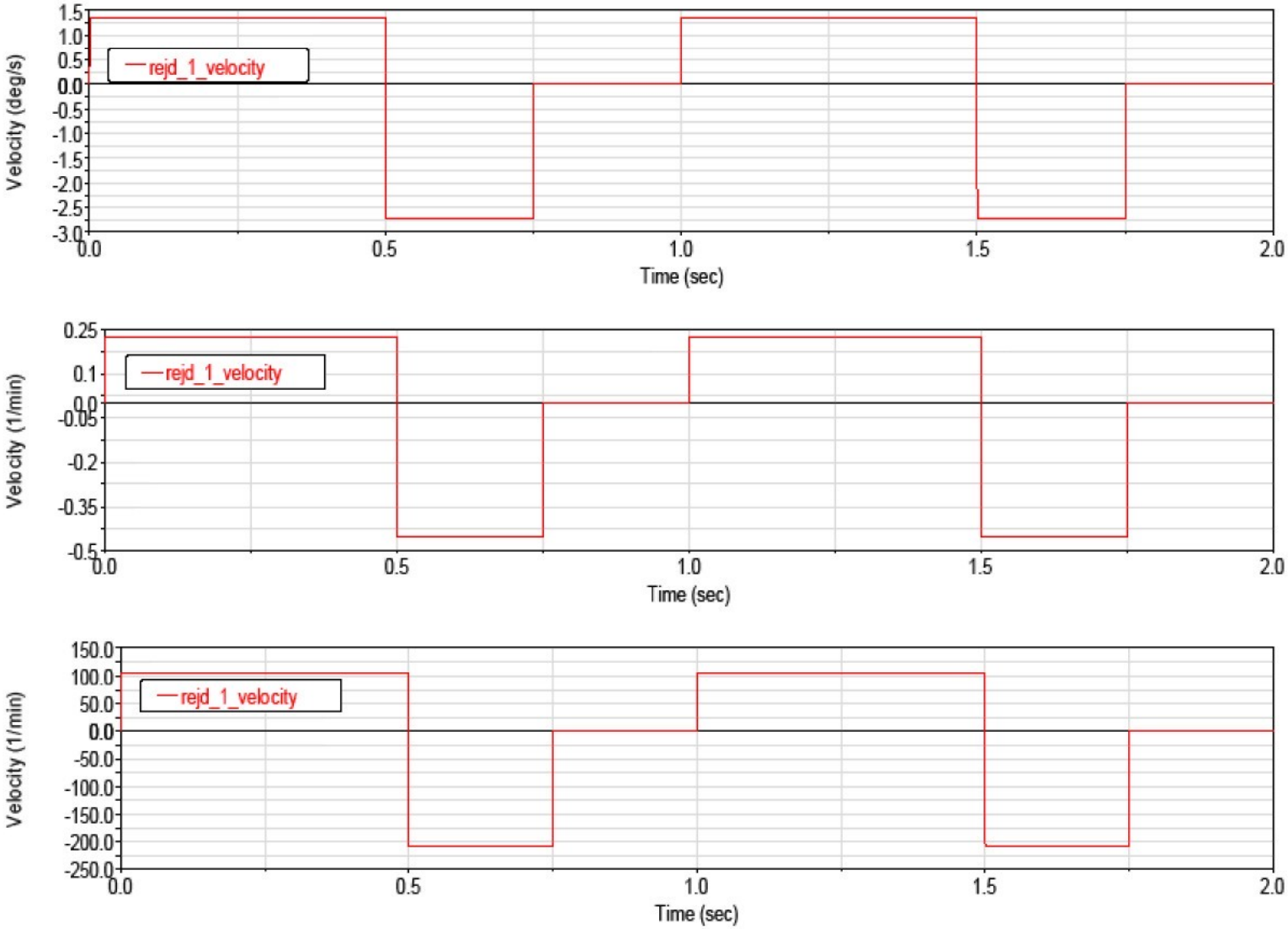
Co se týče momentu vyrovnávání a rejdu během rychlejšího typu chůze (obr. 3.22, 23), tak ani zde nedochází v nejkritičtějších případech k překročení již zmíněných momentů motorů 37,5 Nmm (mNm) pro vyrovnávání a 35 Nmm (35 mNm) pro rejd.

3.4 Manévr *chůze do schodů*

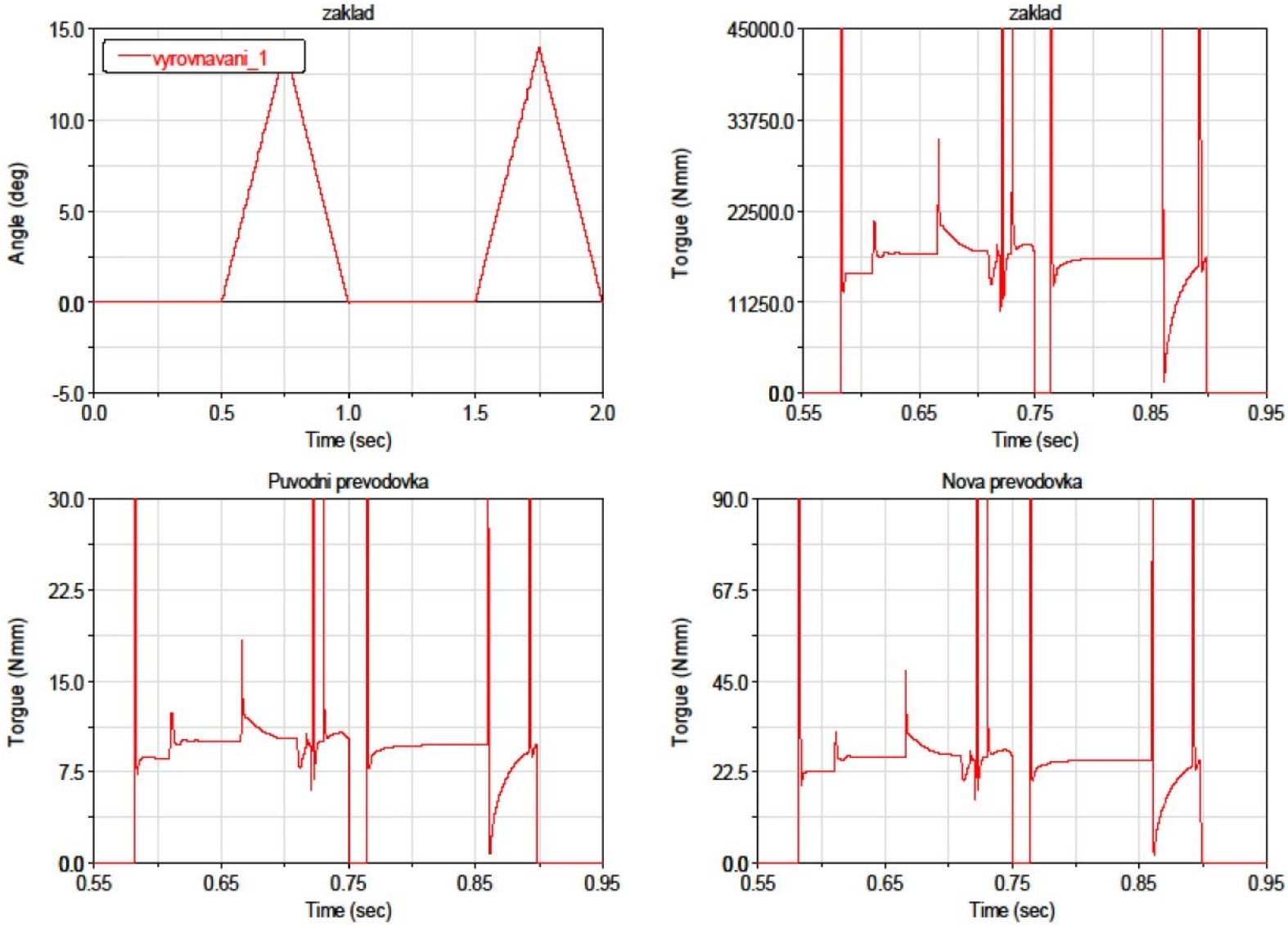
Manévr *chůze do schodů* je vůbec nejsložitější manévr, který bude tento robotizovaný podvozek provádět. Z důvodu veliké variability různých druhů schodišť není možné manévr realizovat pomocí jednoho algoritmu. Podvozek bude muset být osazen několika senzory, které určí parametry schodiště a následně upraví algoritmus tak, aby vyhovoval daným parametrům schodiště. Takto složitý manévr by byl možná sám o době tématem další disertační práce, a proto tato kapitola neobsahuje kompletní algoritmus, ale spíše jen ověření, zda je takovýto manévr při dané konfiguraci podvozku reálně proveditelný.



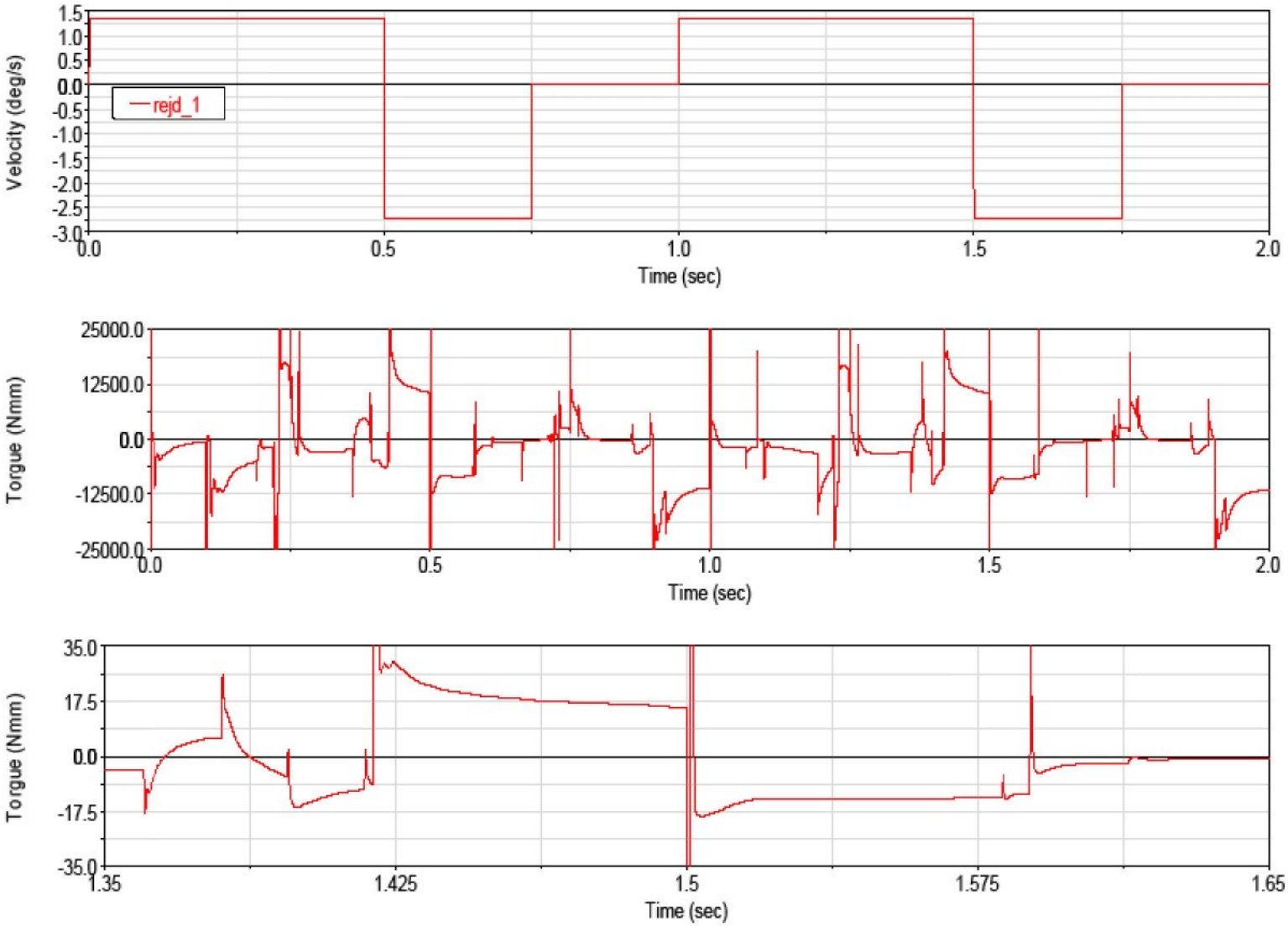
Obr. 3.20 Kinematika pohonu pro vyrovnávání během manévru



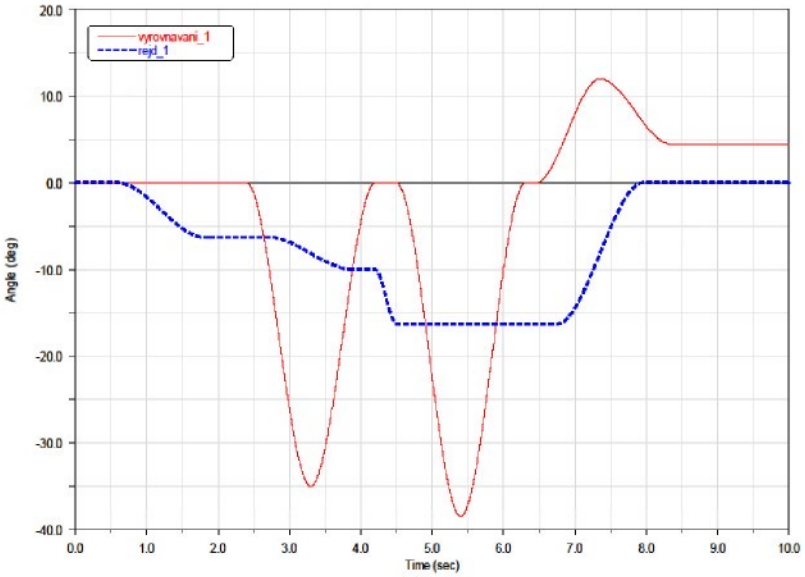
Obr. 3.21 Kinematika pohonu pro rejd během manévru



Obr. 3.22 Momentové zatížení pohonu pro vyrovnávání během manévru



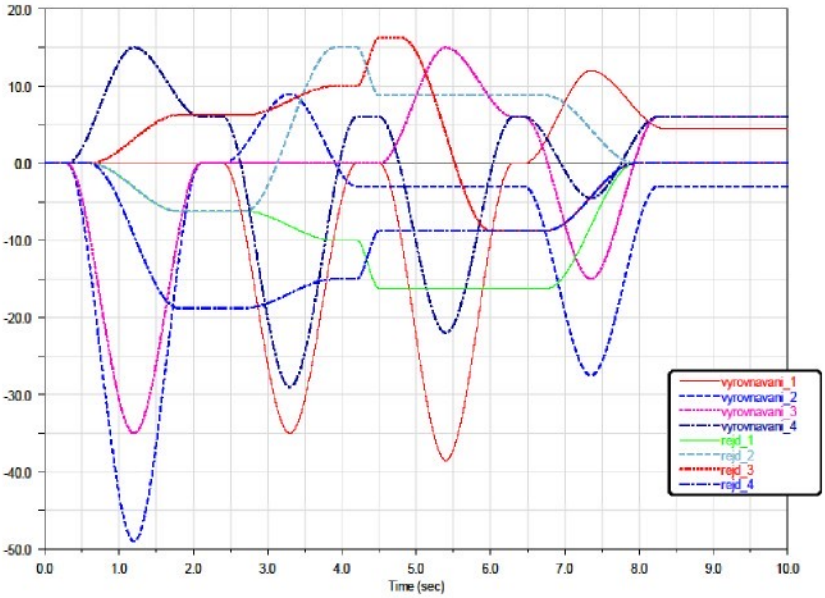
Obr. 3.23 Momentové zatížení pohonu pro rejd během manévru



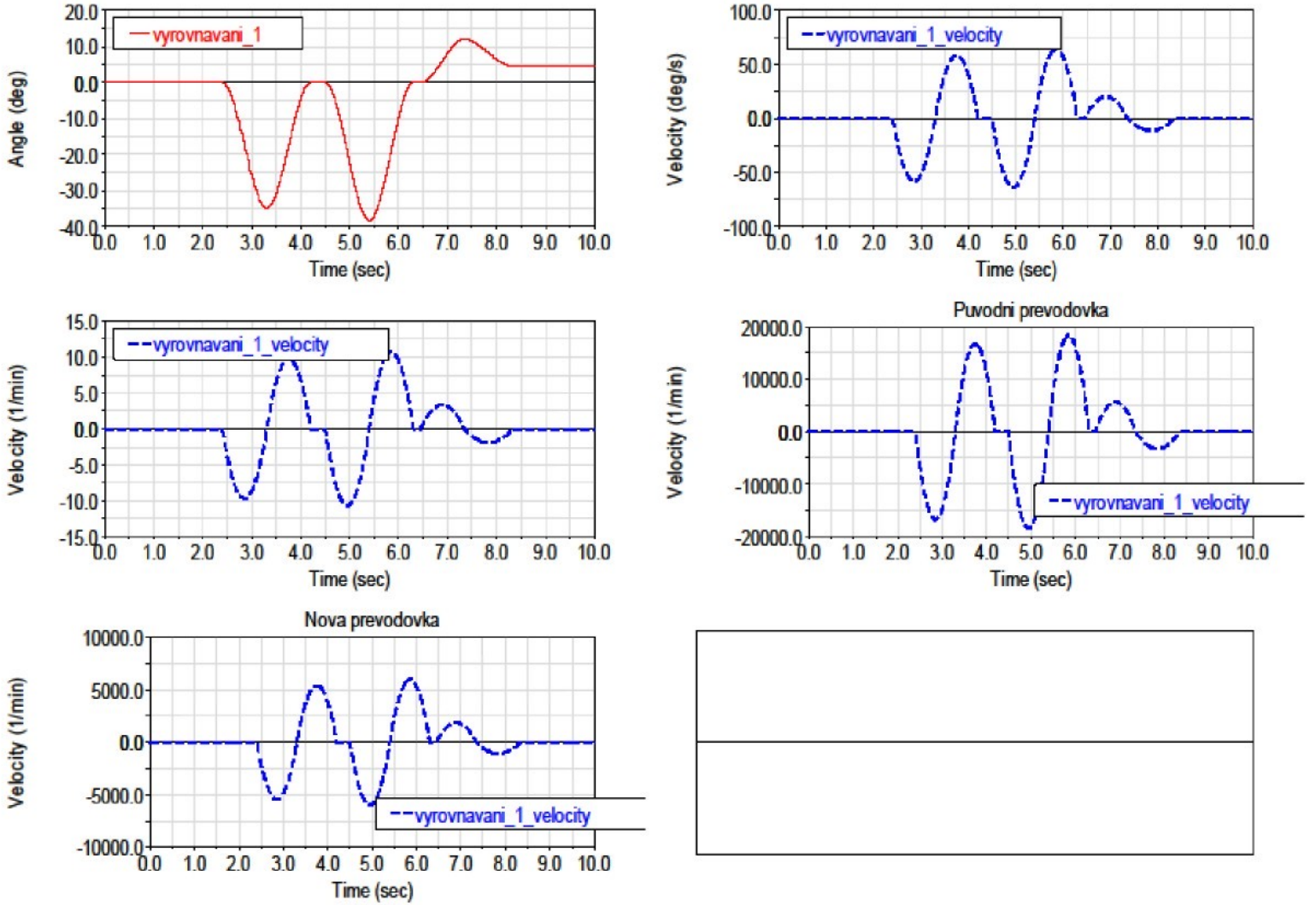
Obr. 3.24 Manévr *chůze do schodů* 1. noha

Na obr. 3.24 a 3.25 jsou znázorněny změny vyrovnávání a rejdu jedné, resp. všech nohou během chůze do schodů. Dochází zde také jako v případě chůze k předpružení nohou, které jsou v kontaktu s podložkou, což zabraňuje ztrátě stability. Na rozdíl od manévru chůze zde však nedochází k návratu vyrovnávání na výchozí hodnotu, což je dáno sklonem schodiště.

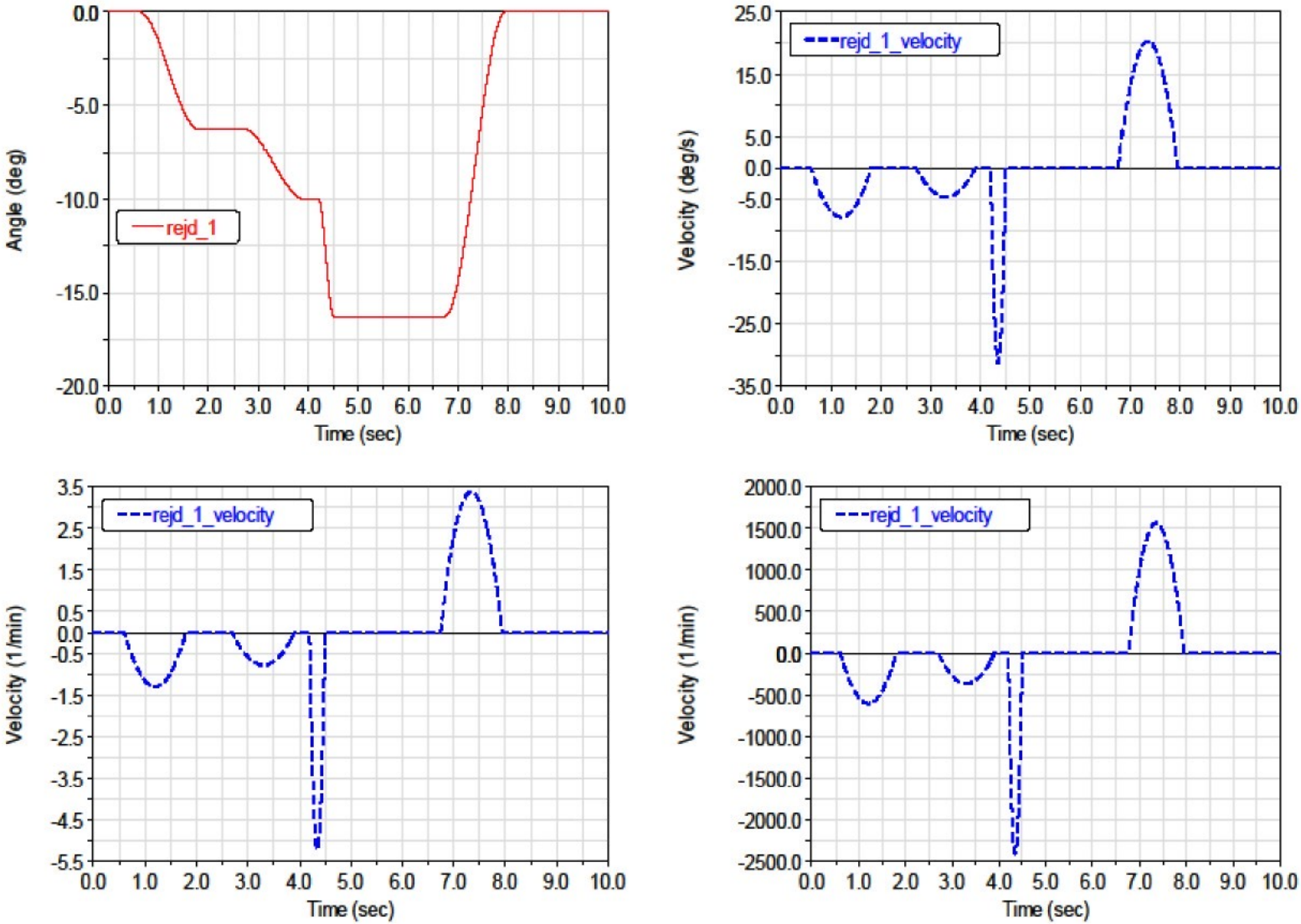
Z obr. 3.26 až 3.29 vyplývá podobně jako u rychlejšího způsobu chůze, že zde, s výjimkou otáček vyrovnávání, nedochází k překročení omezujících hodnot motorů. Otáčky motoru vyrovnávání jsou dostatečné pouze s použitím nové planetové převodovky.



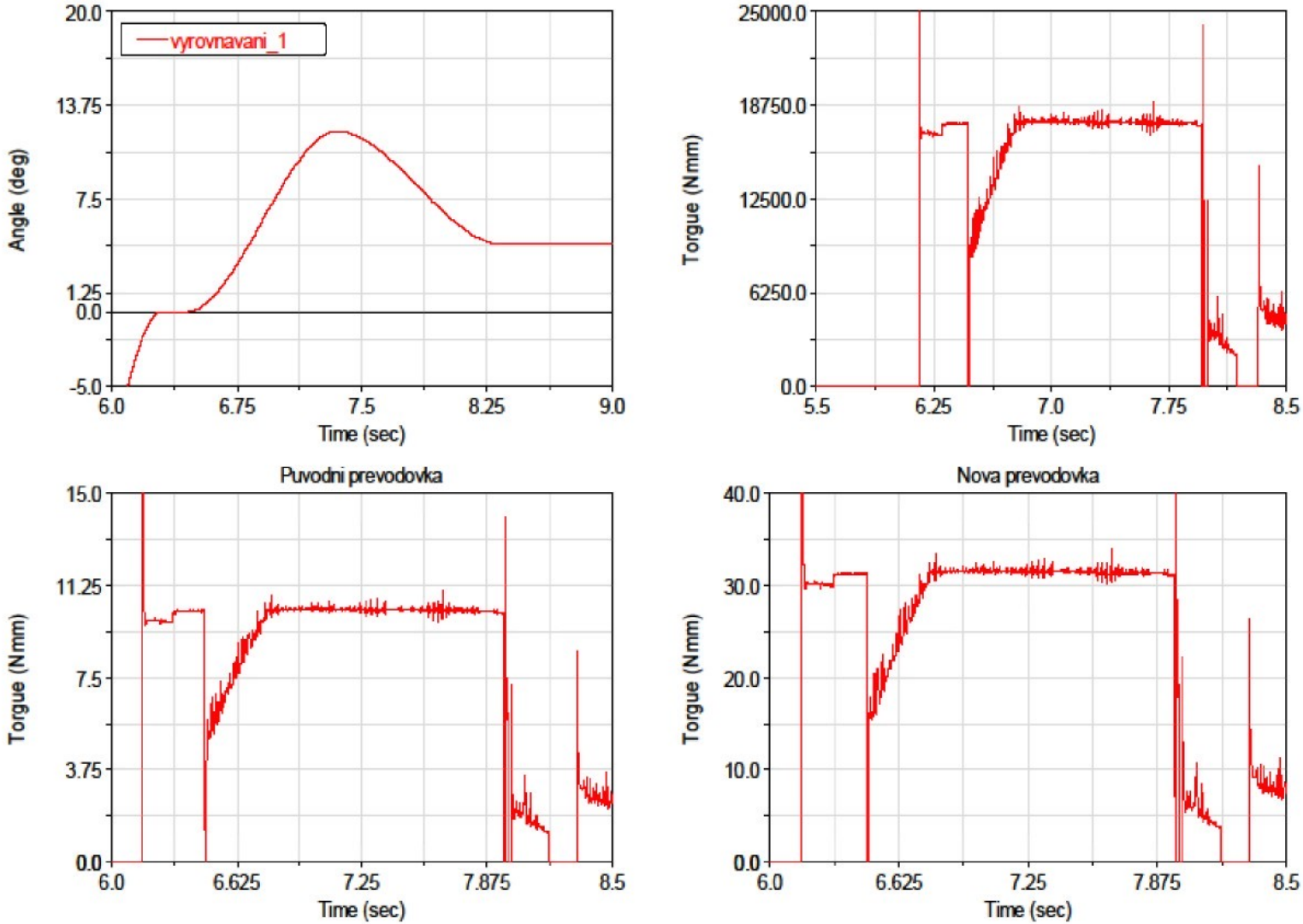
Obr. 3.25 Manévr *chůze do schodů*



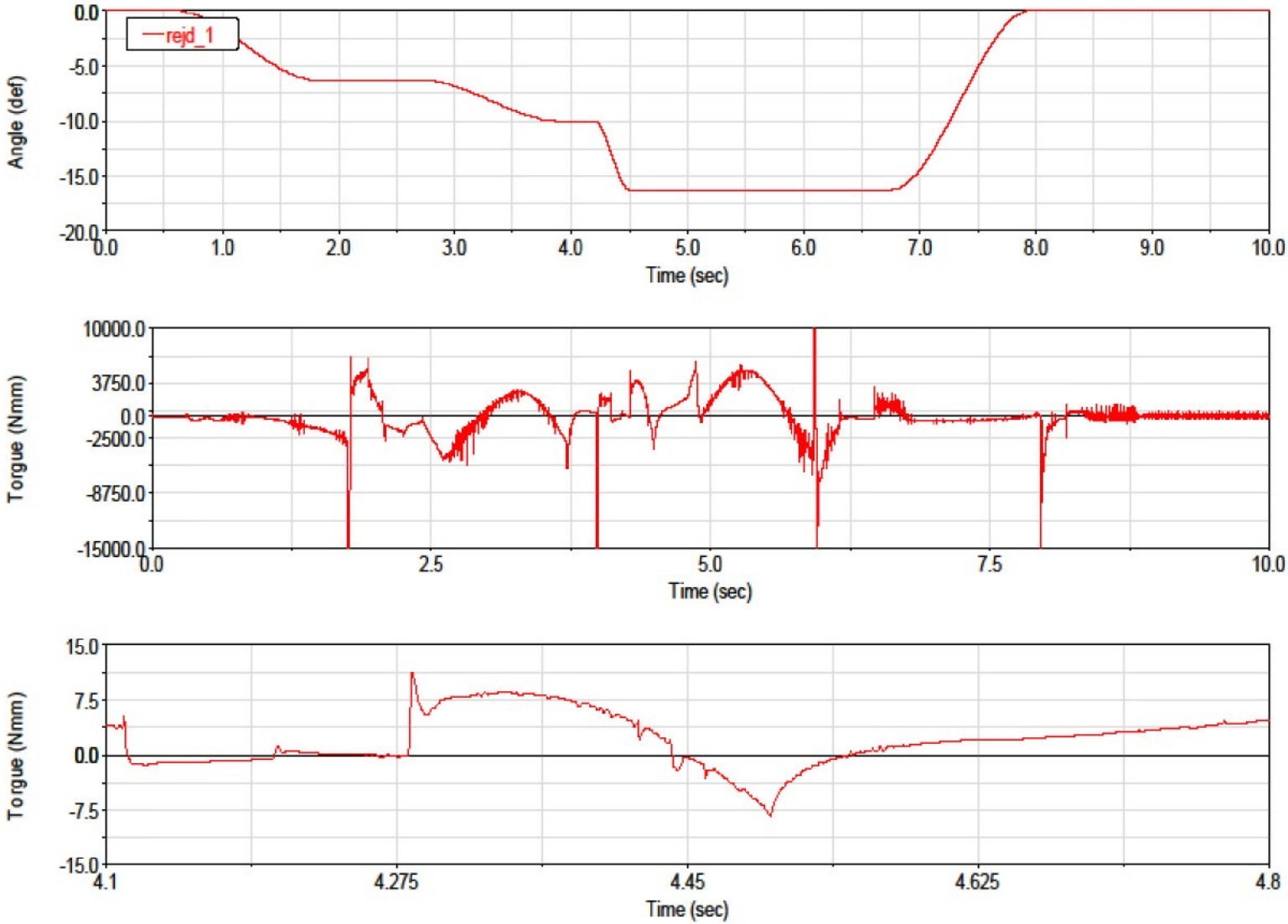
Obr. 3.26 Kinematika pohonu pro vyrovnávání během manévru



Obr. 3.27 Kinematika pohonu pro rejd během manévru



Obr. 3.28 Momentové zatížení pohonu pro vyrovnávání během manévru



Obr. 3.29 Momentové zatížení pohonu pro rejd během manévru

3.5 Manévr průjezd zúženým místem (*zúžení*)

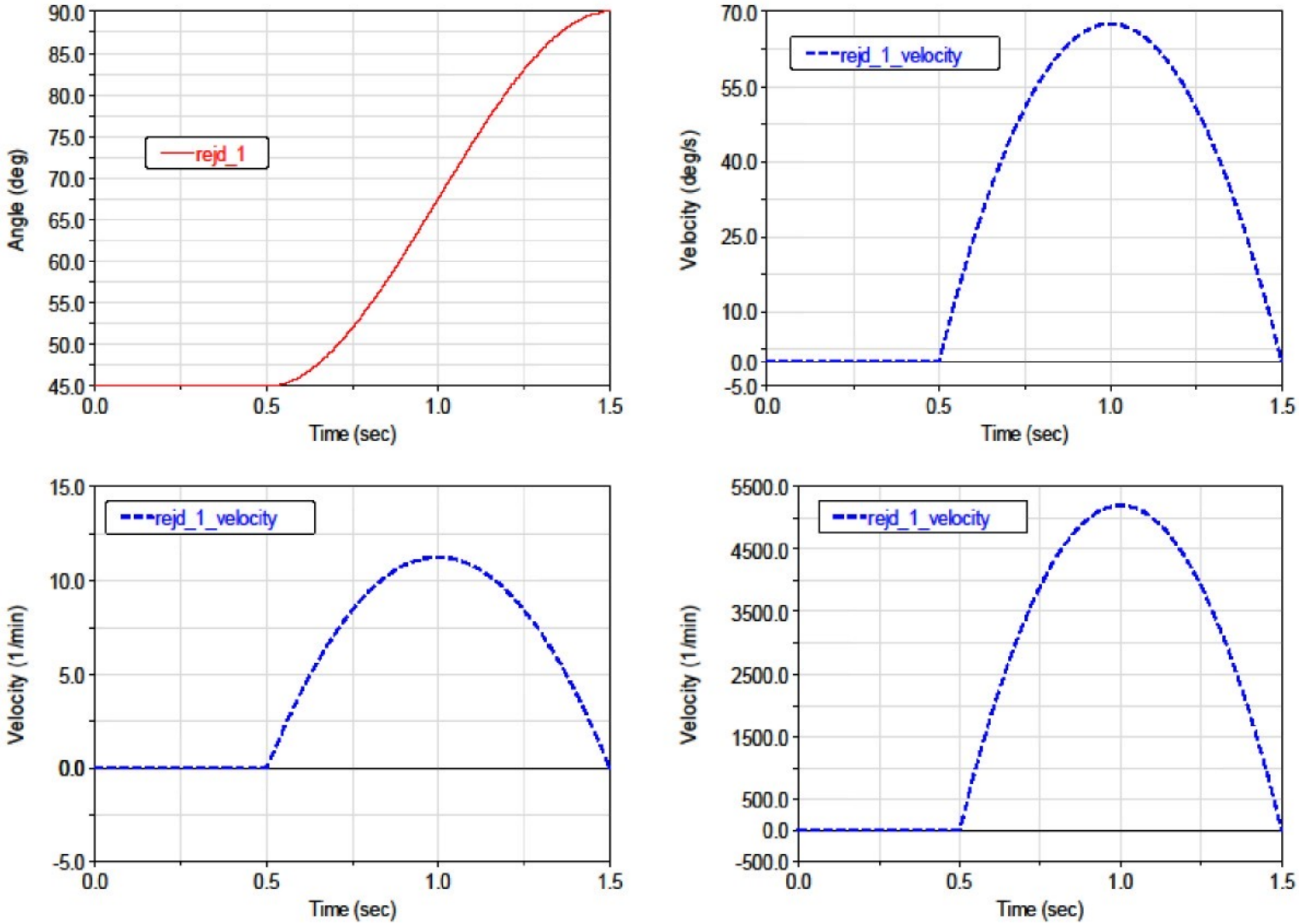
Průjezd zúženým místem je dalším ze základních manévrů podvozku popsaných v kapitole 1.3. Při provozu robotizovaného podvozku, ať už v urbanizovaném prostředí nebo ve volné přírodě, může dojít k tomu, že podvozek díky svému rozchodu nedokáže projet zúženým místem. Takové místo mohou představovat třeba zárubně dveří, které v mnoha případech bývají užší než rozchod běžně prodávaných invalidních vozíků. V přírodě se těchto míst vyskytuje daleko více, např. úzké místo tvořené dvěma stromy, úzká lávka, apod.

Tento manévr je prováděn podle algoritmu uvedeného v kapitole `zuzeni.cpp`, diag. 2.2. Během manévru nedochází k pohybu stupně volnosti nazvaného vyrovnávání, a proto jsou na následujících obrázcích znázorněny pouze průběhy otáček a momentů týkajících se rejdu, který je během tohoto manévru nejvíce namáhán.

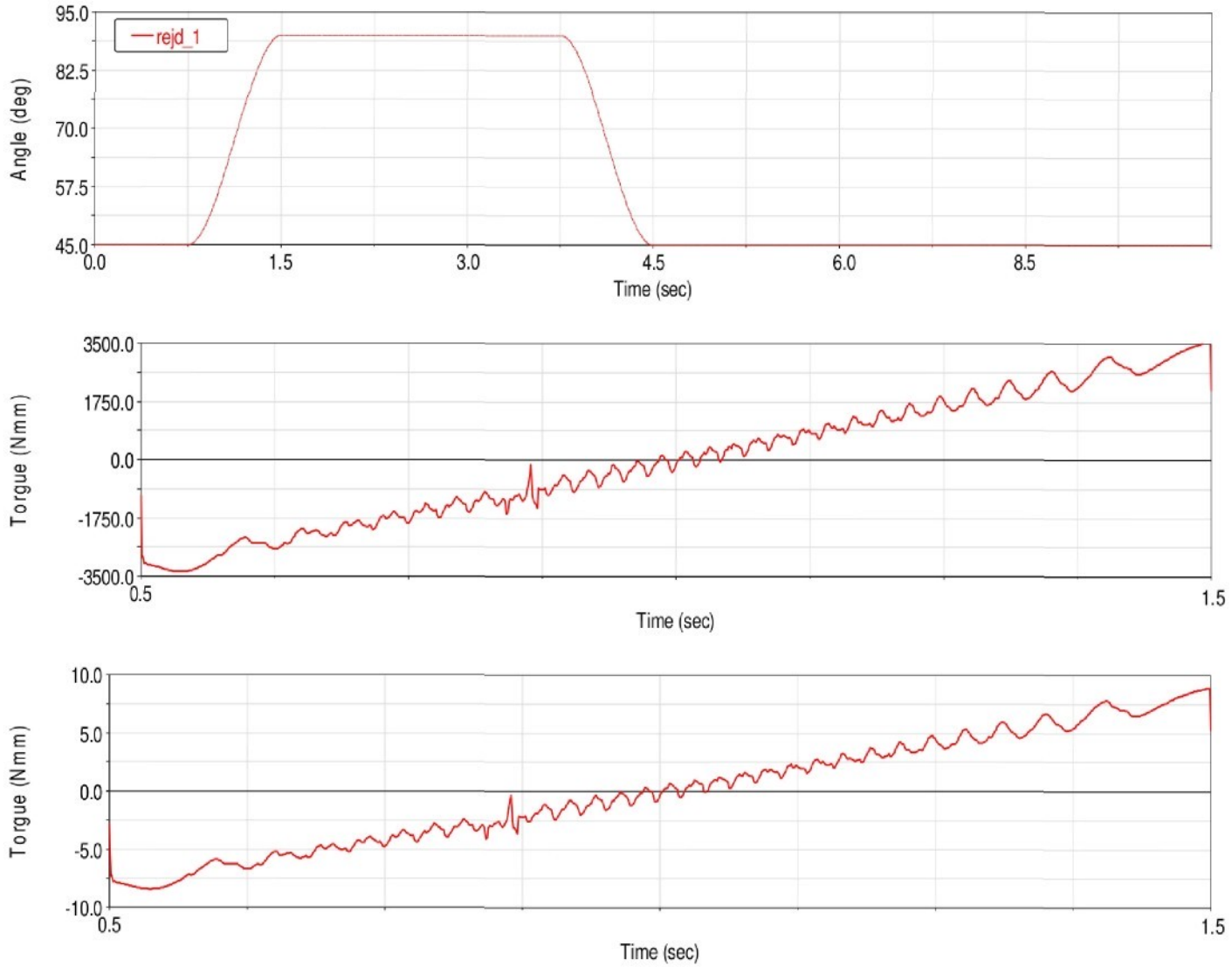
Pořadí grafů na obrázcích je stejné jako u předchozích manévrů. Na obr. 3.26 jsou znázorněny otáčky motorů poháněnějících rejd v následujícím pořadí. První graf popisuje úhel rejdu, druhý graf (derivace prvního) otáčky v °/s, následuje přepočet na ot/min a dále na otáčky elektromotoru. Maximální hodnota otáček elektromotoru opět nepřesahuje hodnotu uvedenou v katalogu 6640 min^{-1}. Na obr. 3.27 jsou průběhy momentů rejdu. Maximální hodnota také nepřesahuje hodnotu 35 Nmm.

3.6 Manévr změna světlé výšky podvozku

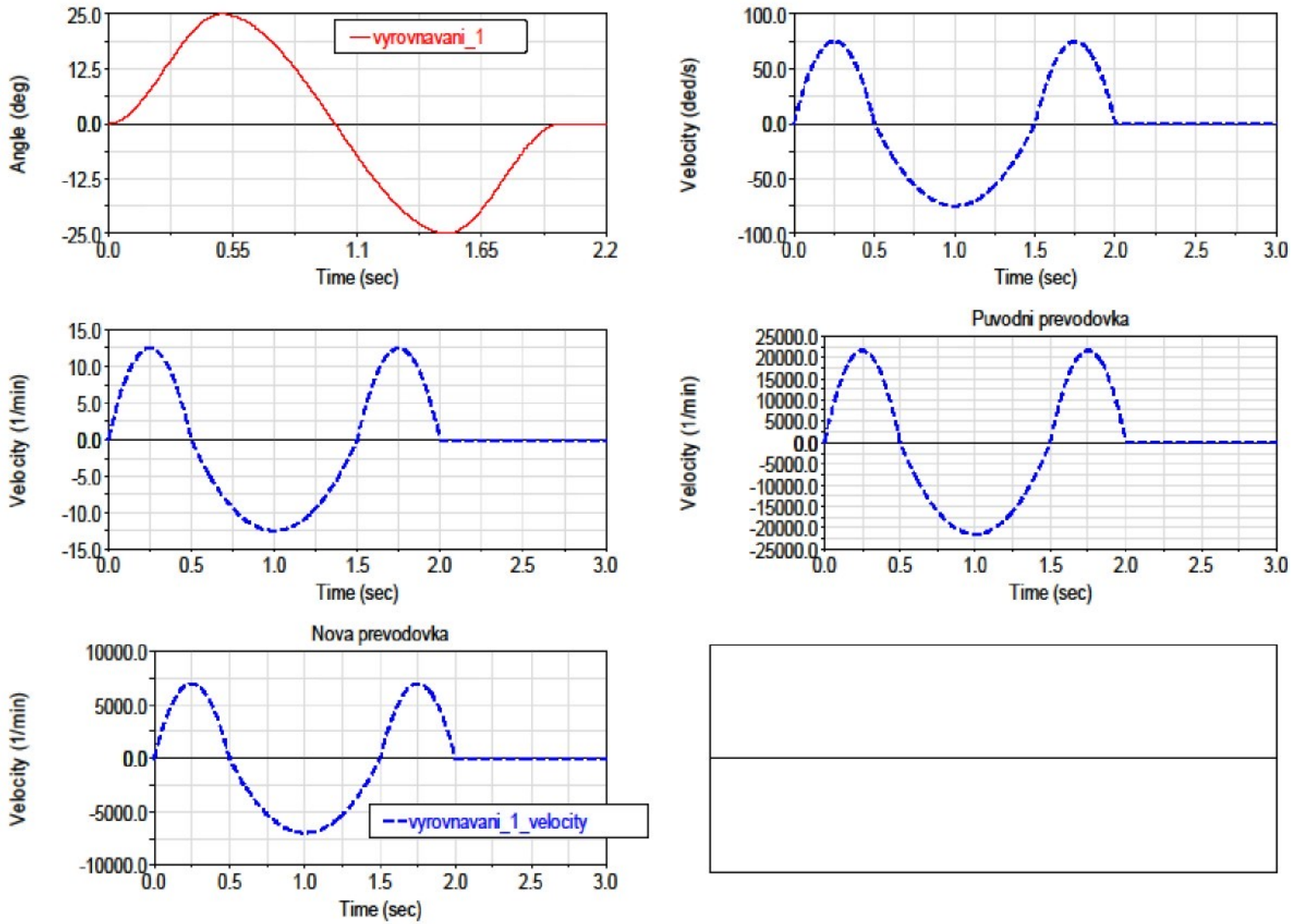
Posledním a nejjednodušším pohybem podvozku je změna světlé výšky. Změna světlé výšky je docílena synchronizovanou změnou stupňů volnosti nazvaných vyrovnávání. Na obr. 3.28 je uveden průběh otáček během změny sv. výšky odpovídající změně úhlu vyrovnávání v rozsahu ±25°. Maximální rychlost motoru při změně světlé výšky je přibližně 7500 min^{-1}. Maximální hodnota momentu motoru se pohybuje pod 10 Nmm. Ani jedna maximální hodnota nepřevyšuje omezující parametry motoru ***EC* 32**.



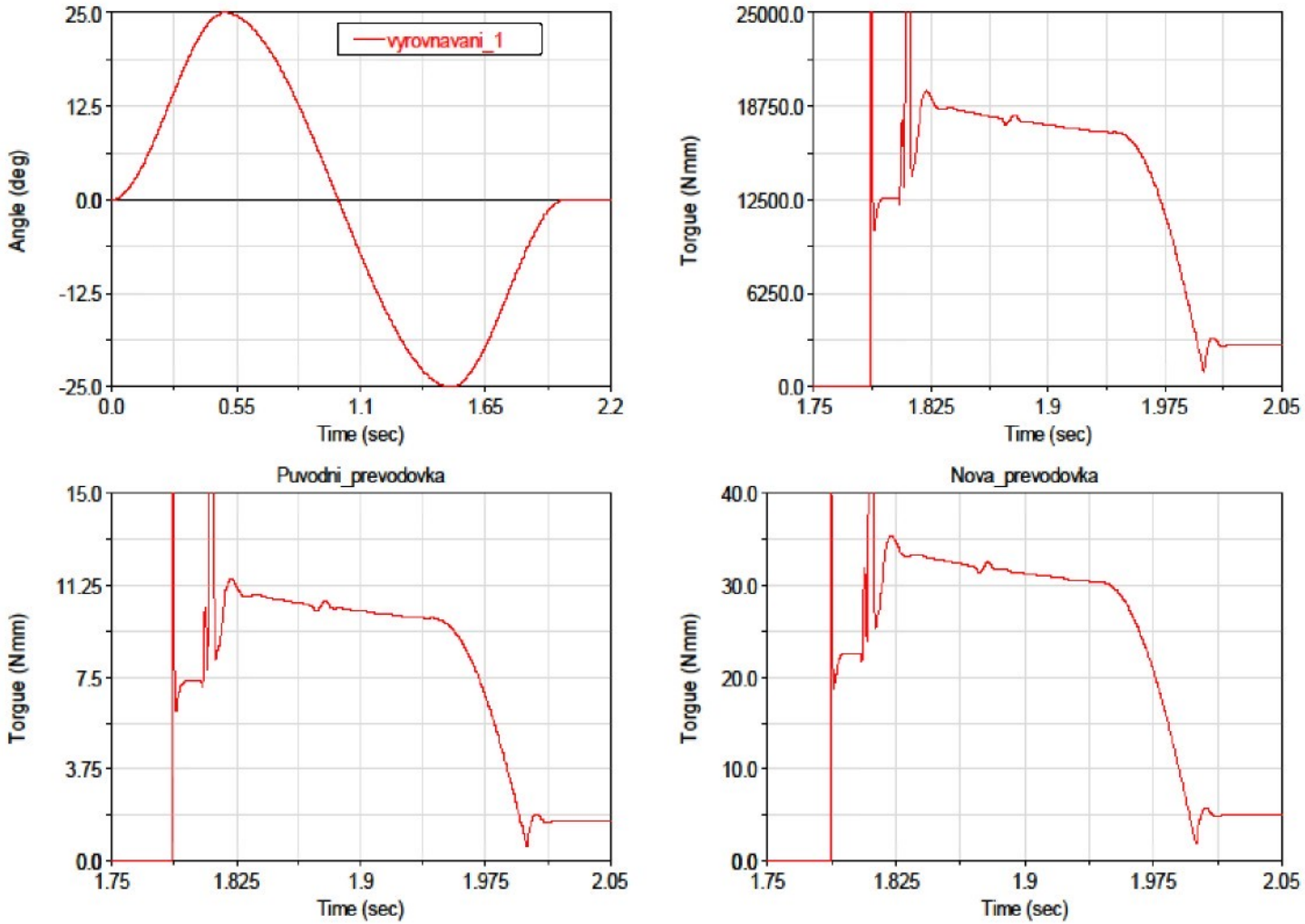
Obr. 2.26 Kinematika pohonu pro rejd během manévru zúžení



Obr. 3.27 Momentové zatížení pohonu pro rejd během manévru zúžení



Obr. 3.28 Kinematika pohonu pro vyrovnávání během manévru změna sv. výšky



Obr. 3.29 Momentové zatížení pohonu pro vyrovnávání během manévru změna sv. výšky

4 Závěr

Práce se zabývala simulacemi a realizací ovládání robotizovaného podvozku určeného pro sociálně zdravotní aplikace s cílem přispět k vývoji zařízení, které usnadní pohyb handicapovaných osob a ležících pacientů v obtížném terénu.

Cílem bylo vytvořit strukturu řízení pomocí stejnosměrných bezkartáčových motorů a řídících jednotek od firmy Maxon, program vhodný pro jejich ovládání a simulace základní manévrů, které měly potvrdit správnost dimenzování pohonů a způsobilost robotizovaného podvozku vykonávat manévry popsané v kapitole 1.3 **Volba koncepce**.

Práce obsahuje kompletní specifikaci pohonů, příslušných řídících jednotek, dalšího vybavení (jako jsou encodery a brzdy), schéma řídící struktury a vzájemné zapojení jednotlivých prvků této struktury.

V úvodní části jsou dále dvě kapitoly klíčové pro samotnou realizaci ovládání robotizovaného podvozku. Jsou to kapitoly 2.2 **Ovládání akčních prvků ve struktuře řízení** a 2.3 **Vizualizace podvozku v softwarovém prostředí pomocí knihovny OpenGL**. Kapitola 2.2 detailně popisuje knihovnu určenou pro programovací jazyk C/C++ použitou pro ovládání elektromotorů prostřednictvím zmíněného programovacího jazyka. Kapitola 2.3 přináší přehled funkcí z knihovny OpenGL využité pro vizualizaci podvozku v softwarovém prostředí. Přínos těchto dvou kapitol je nejen to, že byly základním stavebním kamenem pro tvorbu řídícího softwaru, ale i v tom, že mohou být použity další osobou pracující na pokračování tohoto projektu. Předměty vyučované během doktorského studia se touto tématikou podrobně nezabývaly. Pomocí kapitol 2.2 a 2.3 je tedy možné si osvojit znalosti práce s knihovnami určenými pro ovládání motorů a vizualizaci podvozku.

Následující kapitola obsahuje popis řídícího programu. Tento program je rozdělen do několika podsouborů, z nichž každý obstarává jinou z funkcí podvozku nebo jinou část běhu programu. Popis souborů je doplněn vývojovými diagramy, podle kterých jsou dané pohyby realizovány. Výpis celého řídícího programu je uveden v příloze práce. Konečný model podvozku zatím není k dispozici, a proto byl řídící program otestován na samotných motorech, jak bude předvedeno u obhajoby.

Simulace základních manévrů je obsahem poslední kapitoly. Je zde uveden kompletní popis tvorby zjednodušeného modelu robotizovaného podvozku, jeho následný export do MSC ADAMS. V MSC ADAMS byly do modelu vloženy vazby, představující jednotlivé stupně volnosti a pohony, které nahrazují v reálu použité elektromotory. Následně byly provedeny simulace základních manévrů. Na základě výsledků simulací byla posouzena vhodnost zvolených pohonů z pohledu otáček a momentů, což jsou omezující parametry každého z motorů.

Výsledkem této práce je souhrn teorie ovládání knihoven potřebný pro tvorbu řídícího programu, vlastní ovládací program, který je připraven na testování na skutečném modelu a

v neposlední řadě simulační model v MSC ADAMS určený a využitý pro ověření pohybů podvozku a proveditelnosti těchto pohybů z hlediska výkonnosti elektromotorů.

Všechny cíle vytyčené pro tuto práci byly splněny.

Další prací, kterou by logicky mělo toto snažení pokračovat, je výroba funkčního modelu, na kterém se otestuje řídící software, algoritmy všech manévrů a který bude doplněn o další nezbytnosti jako je například senzor polohy a natočení základní platformy podvozku. Data ze senzoru polohy a natočení, který je již také zakoupen, budou dalším parametrem vstupujícím do řídícího programu. Tento vstup bude realizován pomocí tzv. evaluační desky, která bude vytvořena v rámci jedné z diplomových prací na fakultě mechatroniky.

5 Použitá literatura

[1] Brát, V., Rosenberg, J., Jáč.,V.: *Kinematika*, SNTL, Praha 1987

[2] Juliš, K., Brepta, R. a kol.: *Mechanika I. díl, Statika a mechanika*, SNTL, Praha 1986

[3] Rektorys, K. a kol.: *Přehled užité matematiky*, SNTL, Praha 1981

[4] Medůna, O., *Mechanika podvozku se čtyřmi nezávisle zavěšenými koly s tlumiči a pružinami s řízeným předpětím*, Liberec, Diplomová práce, 2002

[5] Dudek, G, Jenkin, M., *Computational Principles of Mobile Robotics*, Cambridge University Press

[6] kolektiv autor: *Mechatronika - vybrané problémy*, VUT, Brno 2008

[7] Košťál Miroslav: *Experimentální náprava robotizovaného podvozku*, Liberec, 2010

[8] Korf Jaroslav: *Konstrukce podvozkové nohy robotizovaného podvozku*, Liberec, 2007

[9] D. Shreiner, M. Woo, J. Neider, T. Davis: *OPENGL, Průvodce programátora*, Computer Press, 2006

[10] P. Herout, *Učebnice jazyka C*, Kopp, 2004

[11] Denk Miroslav: *Matematický model kinematiky robotizovaného podvozku se šestnácti stupni volnosti*, Liberec, 2007

[12] Milan Bezdíček, Pavel Čoupek, Robert Grepl, Jakub Hrabec, Jiří Konvičný, Martin Krejčiřík, Jiří, Krejsa, Jiří Radoš, Jan Rajlich, František Šolc, Stanislav Věchet: *Mechatronika, vybrané problémy*, Brno 2008

6 Internetové zdroje

- Oficiální internetové stránky firmy Analog Devices http://www.analog.com/en/index.html
- Seriál OPENGL, http://programujte.com
- Seriál Tvorba přenositelných grafických aplikací využívajících knihovnu GLUT, http://www.root.cz/
- Seriál Grafická knihovna OpenGL, http://www.root.cz/
- OpenGL Reference Manual, http://www.glprogramming.com/blue/
- OpenGLUT API Reference, http://openglut.sourceforge.net/group__api.html
- Seriál o programování v C a C++, http://www.linuxsoft.cz
- Učíme se C, http://www.builder.cz
- Poznámky k jazykům C a C++, http://k-prog.wz.cz/
- Programování v jazyku C/C++, http://www.sallyx.org/

Odkazy na animace z kapitoly rešerše:

[1] http://www.ottobock.com

[2] http://www.tankchair.com/gallery.htm

[3] http://www.kemcare.co.nz

[4] http://www.radicalmobility.com/products.htm

[5] http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=1185

[6] http://www.gel.usherbrooke.ca/laborius/projects/AZIMUT/index.html

[7] http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=1141

[8] http://biorobots.cwru.edu/projects/whegs/miniwhegs.html

[9] http://www.botjunkie.com/2007/09/04/hybrid-robot-can-walk-n-roll

[10] http://www.cim.mcgill.ca/~jasmith

[11] http://www.robots-dreams.com/2006/02/robot_builder_p.html

7 Publikace

Vzhledem k patentovému řízení a nebezpečí předzvěřejnění byla možnost publikací výrazně omezena.

Články z konferencí týkající se tématu disertační práce:

M. Denk, M. Šír: KINEMATICS OF 16 DOF ROBOTIZED VEHICLE CHASSIS. In: National Conference with International Participation ENGINEERING MECHANICS 2008, Svratka, Czech Republic, May 12 - 15, 2008, pp. 48-49 ISBN 978-80-87012-11-6.

M. Denk, J. Korf. M. Šír: ROBOTIZED CHASSIS, Modeling of the Mechanical and Mechatronics Systems MMaMS'2009, Zemplínska Šírava, Slovakia, September 22. -24. 2009, page: 86, ISBN 978-80-553-0288-1,

Články z konferencí netýkající se tématu disertační práce:

Bohdana Marvalová, Ludvík Prášil, Miroslav Denk, Measurement of Mechanical Properties of Cord-Rubber Composites, 25 Danubia-Adria Symp, Sept. 2008, Daniel, M., Holy, S., Ruzicka, M (Eds), pp. 167-8, ISBN 978-80-01-04162-8

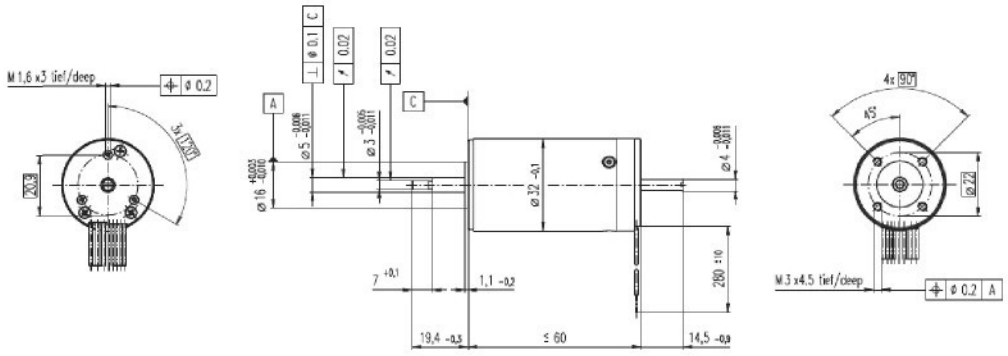
Sivčák M., Denk M., Škoda J.: Verification of mathematical model of the gyroscopic stabilizer, COMPUTATIONAL MECHANICS 2009, Hrad Nečtiny, Czech Republic, November 9 - 11, 2009, ISBN 978-80-7043-824-4

8 Přílohy

Katalogové listy

maxon EC motor

EC 32 ∅32 mm, brushless, 80 Watt, CE approved



M 1:2

Stock program
Standard program
Special program (on request)

Order Number

			118891	118892	118888	118889	118893	118890
Motor Data								
	Values at nominal voltage							
1	Nominal voltage	V	12.0	18.0	18.0	24.0	36.0	48.0
2	No load speed	rpm	15000	14300	13000	11000	14700	11300
3	No load current	mA	901	555	487	286	289	148
4	Nominal speed	rpm	13600	12800	11600	9510	13200	9790
5	Nominal torque (max. continuous torque)	mNm	37.5	40.1	41.2	43.6	39.7	42.6
6	Nominal current (max. continuous current)	A	5.82	3.88	3.61	2.37	1.98	1.19
7	Stall torque	mNm	428	443	407	355	454	353
8	Starting current	A	57.2	37.4	31.4	17.3	19.7	8.84
9	Max. efficiency	%	77.0	77.6	77.1	76.4	77.7	76.2
	Characteristics							
10	Terminal resistance phase to phase	Ω	0.21	0.481	0.573	1.39	1.83	5.43
11	Terminal inductance phase to phase	mH	0.03	0.0752	0.09	0.226	0.285	0.856
12	Torque constant	mNm / A	7.48	11.8	13.0	20.5	23.1	40.0
13	Speed constant	rpm / V	1280	806	737	465	414	239
14	Speed / torque gradient	rpm / mNm	35.8	32.7	32.6	31.5	32.8	32.5
15	Mechanical time constant	ms	7.49	6.86	6.82	6.59	6.87	6.8
16	Rotor inertia	gcm²	20.0	20.0	20.0	20.0	20.0	20.0

Specifications

	Thermal data	
17	Thermal resistance housing-ambient	5.4 K / W
18	Thermal resistance winding-housing	2.5 K / W
19	Thermal time constant winding	15.4 s
20	Thermal time constant motor	1180 s
21	Ambient temperature	-20 ... +100°C
22	Max. permissible winding temperature	+125°C
	Mechanical data (preloaded ball bearings)	
23	Max. permissible speed	25000 rpm
24	Axial play at axial load < 8 N	0 mm
	> 8 N	max. 0.14 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	5.6 N
27	Max. force for press fits (static)	110 N
	(static, shaft supported)	1200 N
28	Max. radial loading, 5 mm from flange	28 N
	Other specifications	
29	Number of pole pairs	1
30	Number of phases	3
31	Weight of motor	270 g

Values listed in the table are nominal.

Connection motor (Cable AWG 22)

red	Motor winding 1
black	Motor winding 2
white	Motor winding 3

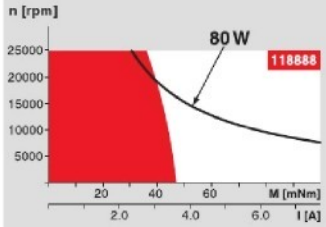
Connection sensors (Cable AWG 26)[1]

green	V_{Hall} 4.5 ... 24 VDC
blue	GND
red / grey	Hall sensor 1
black / grey	Hall sensor 2
white / grey	Hall sensor 3

Wiring diagram for Hall sensors see page 27

1) Not lead through in combination with resolver.

Operating Range Comments



Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

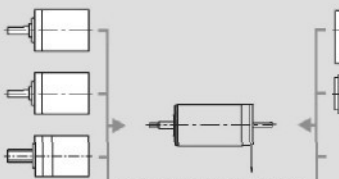
Assigned power rating

maxon Modular System Overview on page 16 - 21

Planetary Gearhead
∅32 mm
0.75 - 4.5 Nm
Page 232

Planetary Gearhead
∅32 mm
0.75 - 6.0 Nm
Page 234 / 236

Spindle Drive
∅32 mm
Page 251 / 252 / 253



Encoder HED_5540
500 CPT,
3 channels
Page 269 / 271

Resolver Res 26
∅26 mm
10 V
Page 278

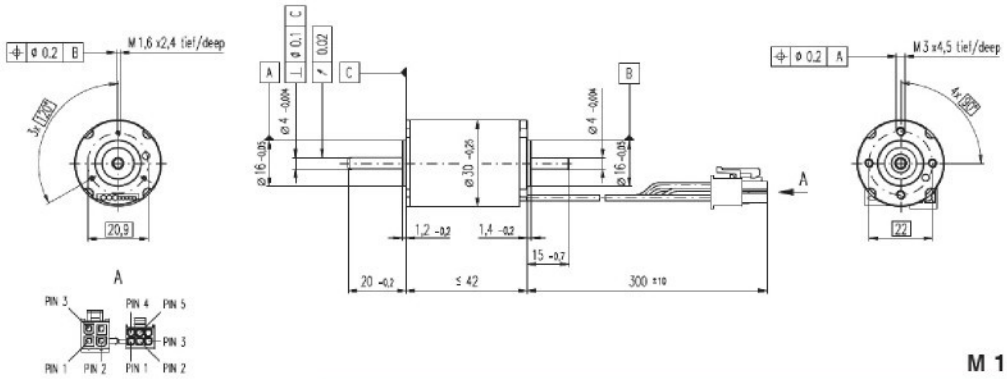
Recommended Electronics:

DECS 50/5	Page 288
DEC 50/5	289
DECV 50/5	295
DEC 70/10	295
DES 50/5	296
EPOS2 24/5	303
EPOS2 50/5	303
EPOS 70/10	303
EPOS P 24/5	306
Notes	**20**

May 2009 edition / subject to change

maxon EC-max

EC-max 30 ∅30 mm, brushless, 40 Watt



M 1:2

Stock program
Standard program
Special program (on request)

			Order Number			
			272766	272768	272769	272770
Motor Data						
	Values at nominal voltage					
1	Nominal voltage	V	12.0	24.0	36.0	48.0
2	No load speed	rpm	8700	9270	9160	9270
3	No load current	mA	202	112	73.2	55.9
4	Nominal speed	rpm	6840	7210	7080	7190
5	Nominal torque (max. continuous torque)	mNm	35.0	34.3	33.9	34.0
6	Nominal current (max. continuous current)	A	2.85	1.50	0.976	0.743
7	Stall torque	mNm	153	160	154	157
8	Starting current	A	11.8	6.57	4.18	3.24
9	Max. efficiency	%	76	76	76	76
	Characteristics					
10	Terminal resistance phase to phase	Ω	1.01	3.65	8.61	14.8
11	Terminal inductance phase to phase	mH	0.088	0.31	0.713	1.24
12	Torque constant	mNm / A	12.9	24.3	36.8	48.6
13	Speed constant	rpm / V	738	393	259	197
14	Speed / torque gradient	rpm / mNm	57.8	59.1	60.6	59.9
15	Mechanical time constant	ms	6.66	6.81	6.98	6.90
16	Rotor inertia	gcm²	11.0	11.0	11.0	11.0

Specifications

Thermal data

17	Thermal resistance housing-ambient	8.6 K / W
18	Thermal resistance winding-housing	1 K / W
19	Thermal time constant winding	3.12 s
20	Thermal time constant motor	777 s
21	Ambient temperature	-20 ... +100°C
22	Max. permissible winding temperature	+155°C

Mechanical data (preloaded ball bearings)

23	Max. permissible speed	15000 rpm
24	Axial play at axial load < 6.0 N	0 mm
	> 6.0 N	0.14 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	5.5 N
27	Max. force for press fits (static)	100 N
	(static, shaft supported)	2000 N
28	Max. radial loading, 5 mm from flange	25 N

Other specifications

29	Number of pole pairs	1
30	Number of phases	3
31	Weight of motor	165 g

Values listed in the table are nominal.

Connection motor (Cable AWG 20)

red	Motor winding 1	Pin 1
black	Motor winding 2	Pin 2
white	Motor winding 3	Pin 3
	N.C.	Pin 4

Connector	Article number
Molex	39-01-2040

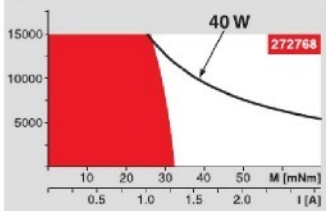
Connection sensors (Cable AWG 26)

yellow	Hall sensor 1	Pin 1
brown	Hall sensor 2	Pin 2
grey	Hall sensor 3	Pin 3
blue	GND	Pin 4
green	V_{Hall} 4.5 ... 24 VDC	Pin 5
	N.C.	Pin 6

Connector	Article number
Molex	430-25-0600

Wiring diagram for Hall sensors see page 27

Operating Range



Comments

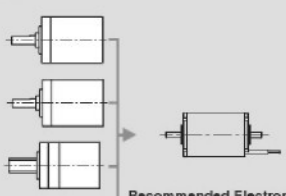
Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
– Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 16 - 21



Planetary Gearhead
∅32 mm
1.0 - 8.0 Nm
Page 236 / 237

Koaxdrive
∅32 mm
1.0 - 4.5 Nm
Page 238

Spindle Drive
∅32 mm
Page 251 / 252 / 253

Encoder MR
500 / 1000 CPT,
3 channels
Page 264

Encoder HEDL 5540
500 CPT,
3 channels
Page 271

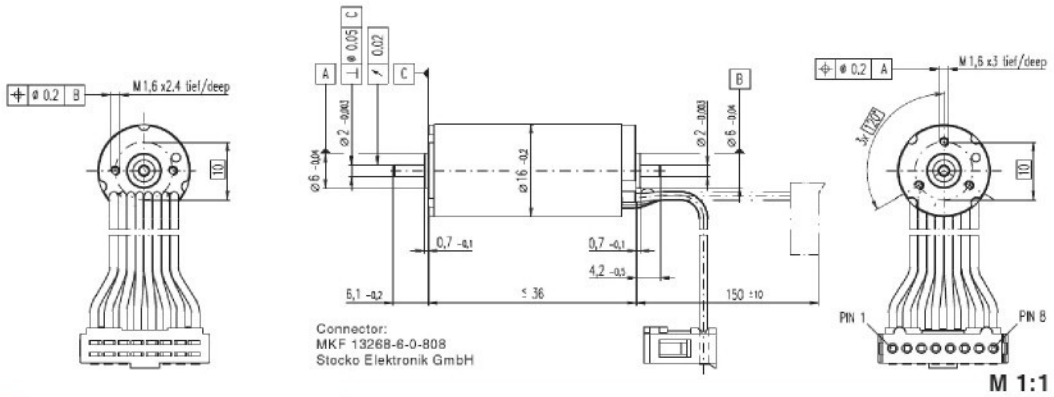
Brake AB 20
∅20 mm
24 VDC, 0.1 Nm
Page 314

Recommended Electronics:

DECS 50/5	Page 288
DEC 50/5	289
DECV 50/5	295
DES 50/5	296
EPOS2 24/5	303
EPOS2 50/5	303
EPOS P 24/5	306
Notes	**20**

May 2009 edition / subject to change

EC-max 16 ∅16 mm, brushless, 8 Watt



M 1:1

Stock program
Standard program
Special program (on request)

			Order Number							
			283831	283832	283833	283834	283835			
Motor Data										
	Values at nominal voltage									
1	Nominal voltage	V	6.0	9.0	12.0	18.0	24.0			
2	No load speed	rpm	12100	11900	12000	11900	11900			
3	No load current	mA	119	77.5	58.5	38.8	29.0			
4	Nominal speed	rpm	7170	7110	7310	7180	7350			
5	Nominal torque (max. continuous torque)	mNm	7.62	7.80	8.04	7.90	8.23			
6	Nominal current (max. continuous current)	A	1.74	1.17	0.907	0.593	0.461			
7	Stall torque	mNm	19.2	19.8	21.1	20.3	22.0			
8	Starting current	A	4.17	2.82	2.27	1.45	1.17			
9	Max. efficiency	%	70	70	71	71	72			
	Characteristics									
10	Terminal resistance phase to phase	Ω	1.44	3.19	5.30	12.4	20.5			
11	Terminal inductance phase to phase	mH	0.0343	0.0793	0.140	0.317	0.566			
12	Torque constant	mNm / A	4.61	7.02	9.32	14.0	18.7			
13	Speed constant	rpm / V	2070	1360	1020	681	510			
14	Speed / torque gradient	rpm / mNm	646	619	582	602	556			
15	Mechanical time constant	ms	5.75	5.51	5.18	5.36	4.95			
16	Rotor inertia	gcm²	0.850	0.850	0.850	0.850	0.850			

Specifications

Thermal data

17	Thermal resistance housing-ambient	17.7 K / W
18	Thermal resistance winding-housing	1.41 K / W
19	Thermal time constant winding	0.983 s
20	Thermal time constant motor	427 s
21	Ambient temperature	-20 ... +100°C
22	Max. permissible winding temperature	+155°C

Mechanical data (preloaded ball bearings)

23	Max. permissible speed	20000 rpm
24	Axial play at axial load < 2.0 N	0 mm
	> 2.0 N	0.14 mm
25	Radial play	preloaded
26	Max. axial load (dynamic)	1.5 N
27	Max. force for press fits (static)	40 N
	(static, shaft supported)	400 N
28	Max. radial loading, 5 mm from flange	6 N

Other specifications

29	Number of pole pairs	1
30	Number of phases	3
31	Weight of motor	45 g

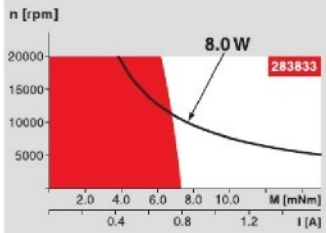
Values listed in the table are nominal.

Connection (Cable AWG 24)

brown	Motor winding 1	Pin 1
red	Motor winding 2	Pin 2
orange	Motor winding 3	Pin 3
yellow	V_{Hall} 4.5 ... 24 VDC	Pin 4
green	GND	Pin 5
blue	Hall sensor 1	Pin 6
violet	Hall sensor 2	Pin 7
grey	Hall sensor 3	Pin 8

Wiring diagram for Hall sensors see page 27

Operating Range



Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
– Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System

Overview on page 16 - 21

Planetary Gearhead
∅22 mm
0.5 - 2.0 Nm
Page 226

Spindle Drive
∅22 mm
Page 249 / 250

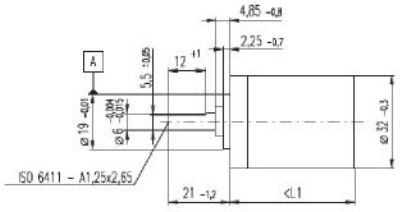
Encoder MR
128 / 256 / 512 CPT,
2 / 3 channels
Page 263

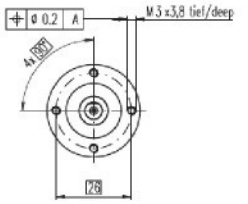
Recommended Electronics:

DECS 50/5	Page 288
DEC 24/1	288
DEC 24/3	289
DEC Module 24/2	289
DECV 50/5	295
DES 50/5	296
EPOS2 Module 36/2	302
EPOS 24/1	302
Notes	**20**

May 2009 edition / subject to change

Planetary Gearhead GP 32 A ∅32 mm, 0.75 - 4.5 Nm





M 1:2

Technical Data

Planetary Gearhead	straight teeth
Output shaft	stainless steel
Shaft diameter as option	8 mm
Bearing at output	ball bearing
Radial play, 5 mm from flange	max. 0.14 mm
Axial play	max. 0.4 mm
Max. radial load, 10 mm from flange	140 N
Max. permissible axial load	120 N
Max. permissible force for press fits	120 N
Sense of rotation, drive to output	=
Recommended input speed	< 6000 rpm
Recommended temperature range	-20 ... +100°C
Extended area as option	-35 ... +100°C

maxon gear

Stock program
Standard program
Special program (on request)

Option: Low-noise version

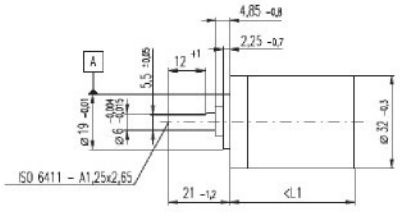
Order Number												
	166155	166158	166163	166164	166169	166174	166179	166184	166187	166192	166197	166202
Gearhead Data												
1 Reduction	3.7 : 1	14 : 1	33 : 1	51 : 1	111 : 1	246 : 1	492 : 1	762 : 1	1181 : 1	1972 : 1	2829 : 1	4380 : 1
2 Reduction absolute	$^{26}/_{7}$	$^{676}/_{49}$	$^{529}/_{16}$	$^{17576}/_{343}$	$^{13824}/_{125}$	$^{421824}/_{1715}$	$^{86112}/_{175}$	$^{19044}/_{25}$	$^{18123776}/_{6575}$	$^{8626176}/_{4375}$	$^{495144}/_{175}$	$^{109503}/_{25}$
3 Max. motor shaft diameter mm	6	6	3	6	4	4	3	3	4	4	3	3
Order Number	166156	166159		166165	166170	166175	166180	166185	166188	166193	166198	166203
1 Reduction	4.8 : 1	18 : 1		66 : 1	123 : 1	295 : 1	531 : 1	913 : 1	1414 : 1	2189 : 1	3052 : 1	5247 : 1
2 Reduction absolute	$^{24}/_{5}$	$^{624}/_{35}$		$^{16224}/_{245}$	$^{6877}/_{56}$	$^{101062}/_{343}$	$^{331776}/_{625}$	$^{36501}/_{40}$	$^{2424488}/_{1715}$	$^{538406}/_{245}$	$^{1907712}/_{625}$	$^{839523}/_{160}$
3 Max. motor shaft diameter mm	4	4		4	3	3	4	3	3	3	3	3
Order Number	166157	166160		166166	166171	166176	166181	166186	166189	166194	166199	166204
1 Reduction	5.8 : 1	21 : 1		79 : 1	132 : 1	318 : 1	589 : 1	1093 : 1	1526 : 1	2362 : 1	3389 : 1	6285 : 1
2 Reduction absolute	$^{23}/_{4}$	$^{299}/_{14}$		$^{3887}/_{49}$	$^{3312}/_{25}$	$^{389376}/_{1225}$	$^{20631}/_{35}$	$^{279841}/_{256}$	$^{9345824}/_{6125}$	$^{2059699}/_{875}$	$^{474513}/_{140}$	$^{6426943}/_{1024}$
3 Max. motor shaft diameter mm	3	3		3	3	4	3	3	4	3	3	3
Order Number		166161		166167	166172	166177	166182		166190	166195	166200	
1 Reduction		23 : 1		86 : 1	159 : 1	411 : 1	636 : 1		1694 : 1	2548 : 1	3656 : 1	
2 Reduction absolute		$^{575}/_{25}$		$^{14975}/_{175}$	$^{1587}/_{10}$	$^{359424}/_{875}$	$^{79488}/_{125}$		$^{1162213}/_{686}$	$^{7962624}/_{3125}$	$^{457056}/_{125}$	
3 Max. motor shaft diameter mm		4		4	3	4	3		3	4	3	
Order Number		166162		166168	166173	166178	166183		166191	166196	166201	
1 Reduction		28 : 1		103 : 1	190 : 1	456 : 1	706 : 1		1828 : 1	2623 : 1	4060 : 1	
2 Reduction absolute		$^{196}/_{5}$		$^{3388}/_{35}$	$^{12167}/_{64}$	$^{89401}/_{196}$	$^{158171}/_{224}$		$^{2238917}/_{1225}$	$^{2058223}/_{784}$	$^{3637633}/_{896}$	
3 Max. motor shaft diameter mm		3		3	3	3	3		3	3	3	
4 Number of stages	1	2	2	3	3	4	4	4	5	5	5	5
5 Max. continuous torque Nm	0.75	2.25	2.25	4.50	4.50	4.50	4.50	4.50	4.50	4.50	4.50	4.50
6 Intermittently permissible torque at gear output Nm	1.1	3.4	3.4	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5
7 Max. efficiency %	80	75	75	70	70	60	60	60	50	50	50	50
8 Weight g	118	162	162	194	194	226	226	226	258	258	258	258
9 Average backlash no load °	0.7	0.8	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10 Mass inertia gcm²	1.5	0.8	0.8	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
11 Gearhead length L1 mm	26.4	36.3	36.3	43.0	43.0	49.7	49.7	49.7	56.4	56.4	56.4	56.4

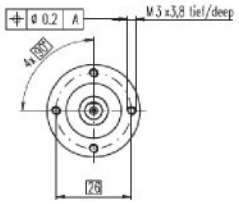
maxon Modular System

+ Motor	Page	+ Sensor / Brake	Page	Overall length [mm] = Motor length + gearhead length + (sensor / brake) + assembly parts											
RE 25	77/79			81.0	90.9	90.9	97.6	97.6	104.3	104.3	104.3	111.0	111.0	111.0	111.0
RE 25	77/79	MR	264	92.0	101.9	101.9	108.6	108.6	115.3	115.3	115.3	122.0	122.0	122.0	122.0
RE 25	77/79	Enc 22	266	95.1	105.0	105.0	111.7	111.7	118.4	118.4	118.4	125.1	125.1	125.1	125.1
RE 25	77/79	HED_5540	268/270	101.8	111.7	111.7	118.4	118.4	125.1	125.1	125.1	131.8	131.8	131.8	131.8
RE 25	77/79	DCT 22	277	103.3	113.2	113.2	119.9	119.9	126.6	126.6	126.6	133.3	133.3	133.3	133.3
RE 25, 20 W	78			69.5	79.4	79.4	86.1	86.1	92.8	92.8	92.8	99.5	99.5	99.5	99.5
RE 25, 20 W	78	MR	264	80.5	90.4	90.4	97.1	97.1	103.8	103.8	103.8	110.5	110.5	110.5	110.5
RE 25, 20 W	78	HED_5540	269/272	90.3	100.2	100.2	106.9	106.9	113.6	113.6	113.6	120.3	120.3	120.3	120.3
RE 25, 20 W	78	DCT22	277	91.8	101.7	101.7	108.4	108.4	115.1	115.1	115.1	121.8	121.8	121.8	121.8
RE 25, 20 W	78	AB 28	316	103.6	113.5	113.5	120.2	120.2	126.9	126.9	126.9	133.6	133.6	133.6	133.6
RE 25, 20 W	78	HED_5540 / AB 28	269/316	120.8	130.7	130.7	137.4	137.4	144.1	144.1	144.1	150.8	150.8	150.8	150.8
RE 25, 20 W	79	AB 28	316	115.1	125.0	125.0	131.7	131.7	138.4	138.4	138.4	145.1	145.1	145.1	145.1
RE 25, 20 W	79	HED_5540/AB 28	268/316	132.2	142.1	142.1	148.8	148.8	155.5	155.5	155.5	162.2	162.2	162.2	162.2
A-max 26	105-112			71.2	81.1	81.1	87.8	87.8	94.5	94.5	94.5	101.2	101.2	101.2	101.2
A-max 26	106-112	MEnc 13	276	78.3	88.2	88.2	94.9	94.9	101.6	101.6	101.6	108.3	108.3	108.3	108.3
A-max 26	106-112	MR	264	80.0	89.9	89.9	96.6	96.6	103.3	103.3	103.3	110.0	110.0	110.0	110.0
A-max 26	106-112	Enc 22	267	85.6	95.5	95.5	102.2	102.2	108.9	108.9	108.9	115.6	115.6	115.6	115.6
A-max 26	106-112	HED_5540	269/271	90.0	99.9	99.9	106.6	106.6	113.3	113.3	113.3	120.0	120.0	120.0	120.0
RE-max 29	135-138			71.2	81.1	81.1	87.8	87.8	94.5	94.5	94.5	101.2	101.2	101.2	101.2
RE-max 29	136/138	MR	264	80.0	89.9	89.9	96.6	96.6	103.3	103.3	103.3	110.0	110.0	110.0	110.0

Planetary Gearhead GP 32 C ∅32 mm, 1.0 - 6.0 Nm

Ceramic Version





M 1:2

Technical Data

Planetary Gearhead	straight teeth
Output shaft	stainless steel
Shaft diameter as option	8 mm
Bearing at output	ball bearing
Radial play, 5 mm from flange	max. 0.14 mm
Axial play	max. 0.4 mm
Max. radial load, 10 mm from flange	140 N
Max. permissible axial load	120 N
Max. permissible force for press fits	120 N
Sense of rotation, drive to output	=
Recommended input speed	< 8000 rpm
Recommended temperature range	-20 ... +100°C
Extended area as option	-35 ... +100°C

Stock program
Standard program
Special program (on request)

Option: Low-noise version

Order Number												
	166930	166933	166938	166939	166944	166949	166954	166959	166962	166967	166972	166977
Gearhead Data												
1 Reduction	3.7 : 1	14 : 1	33 : 1	51 : 1	111 : 1	246 : 1	492 : 1	762 : 1	1181 : 1	1972 : 1	2829 : 1	4380 : 1
2 Reduction absolute	$^{26}/_{7}$	$^{676}/_{49}$	$^{529}/_{16}$	$^{17576}/_{343}$	$^{13824}/_{125}$	$^{421824}/_{1715}$	$^{86112}/_{175}$	$^{19044}/_{25}$	$^{10123776}/_{8575}$	$^{6826176}/_{3375}$	$^{495144}/_{175}$	$^{109503}/_{25}$
3 Max. motor shaft diameter mm	6	6	3	6	4	4	3	3	4	4	3	3
Order Number	166931	166934		166940	166945	166950	166955	166960	166963	166968	166973	166978
1 Reduction	4.8 : 1	18 : 1		66 : 1	123 : 1	295 : 1	531 : 1	913 : 1	1414 : 1	2189 : 1	3052 : 1	5247 : 1
2 Reduction absolute	$^{24}/_{5}$	$^{624}/_{35}$		$^{16224}/_{245}$	$^{6877}/_{56}$	$^{101062}/_{343}$	$^{331776}/_{625}$	$^{36501}/_{40}$	$^{2425488}/_{1715}$	$^{536406}/_{245}$	$^{1907712}/_{625}$	$^{839523}/_{160}$
3 Max. motor shaft diameter mm	4	4		4	3	3	4	3	3	3	3	3
Order Number	166932	166935		166941	166946	166951	166956	166961	166964	166969	166974	166979
1 Reduction	5.8 : 1	21 : 1		79 : 1	132 : 1	318 : 1	589 : 1	1093 : 1	1526 : 1	2362 : 1	3389 : 1	6285 : 1
2 Reduction absolute	$^{23}/_{4}$	$^{299}/_{14}$		$^{3887}/_{49}$	$^{3312}/_{25}$	$^{389376}/_{1225}$	$^{20631}/_{35}$	$^{279841}/_{256}$	$^{9345824}/_{6125}$	$^{2056688}/_{875}$	$^{474513}/_{140}$	$^{6436343}/_{1024}$
3 Max. motor shaft diameter mm	3	3		3	3	4	3	3	4	3	3	3
Order Number		166936		166942	166947	166952	166957		166965	166970	166975	
1 Reduction		23 : 1		86 : 1	159 : 1	411 : 1	636 : 1		1694 : 1	2548 : 1	3656 : 1	
2 Reduction absolute		$^{575}/_{25}$		$^{14976}/_{175}$	$^{1587}/_{10}$	$^{359424}/_{875}$	$^{79488}/_{125}$		$^{1162213}/_{686}$	$^{7952624}/_{3125}$	$^{457056}/_{125}$	
3 Max. motor shaft diameter mm		4		4	3	4	3		3	4	3	
Order Number		166937		166943	166948	166953	166958		166966	166971	166976	
1 Reduction		28 : 1		103 : 1	190 : 1	456 : 1	706 : 1		1828 : 1	2623 : 1	4060 : 1	
2 Reduction absolute		$^{196}/_{5}$		$^{3598}/_{35}$	$^{12167}/_{64}$	$^{89401}/_{196}$	$^{158171}/_{224}$		$^{2238912}/_{1225}$	$^{2058223}/_{784}$	$^{3637933}/_{896}$	
3 Max. motor shaft diameter mm		3		3	3	3	3		3	3	3	
4 Number of stages	1	2	2	3	3	4	4	4	5	5	5	5
5 Max. continuous torque Nm	1	3	3	6	6	6	6	6	6	6	6	6
6 Intermittently permissible torque at gear output Nm	1.25	3.75	3.75	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5
7 Max. efficiency %	80	75	75	70	70	60	60	60	50	50	50	50
8 Weight g	118	162	162	194	194	226	226	226	258	258	258	258
9 Average backlash no load °	0.7	0.8	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10 Mass inertia gcm²	1.5	0.8	0.8	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
11 Gearhead length L1 mm	26.4	36.3	36.3	43.0	43.0	49.7	49.7	49.7	56.4	56.4	56.4	56.4

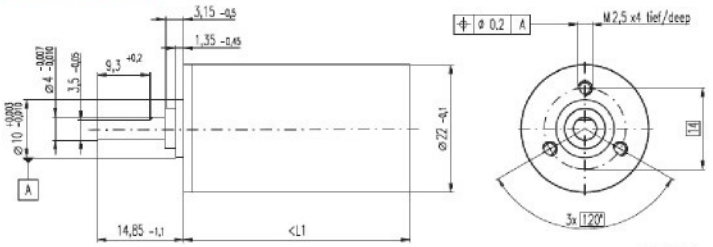
maxon Modular System

+ Motor	Page	+ Sensor / Brake	Page	Overall length [mm] = Motor length + gearhead length + (sensor / brake) + assembly parts											
RE 25, 10 W	77/79			81.0	90.9	90.9	97.6	97.6	104.3	104.3	104.3	111.0	111.0	111.0	111.0
RE 25, 10 W	77/79	MR	264	92.0	101.9	101.9	108.6	108.6	115.3	115.3	115.3	122.0	122.0	122.0	122.0
RE 25, 10 W	77/79	Enc 22	266	95.1	105.0	105.0	111.7	111.7	118.4	118.4	118.4	125.1	125.1	125.1	125.1
RE 25, 10 W	77/79	HED_ 5540	268/270	101.8	111.7	111.7	118.4	118.4	125.1	125.1	125.1	131.8	131.8	131.8	131.8
RE 25, 10 W	77/79	DCT 22	277	103.3	113.2	113.2	119.9	119.9	126.6	126.6	126.6	133.3	133.3	133.3	133.3
RE 25, 20 W	78			69.5	79.4	79.4	86.1	86.1	92.8	92.8	92.8	99.5	99.5	99.5	99.5
RE 25, 20 W	78	MR	264	80.5	90.4	90.4	97.1	97.1	103.8	103.8	103.8	110.5	110.5	110.5	110.5
RE 25, 20 W	78	HED_ 5540	269/272	90.3	100.2	100.2	106.9	106.9	113.6	113.6	113.6	120.3	120.3	120.3	120.3
RE 25, 20 W	78	DCT22	277	91.8	101.7	101.7	108.4	108.4	115.1	115.1	115.1	121.8	121.8	121.8	121.8
RE 25, 20 W	78	AB 28	316	103.6	113.6	113.6	120.2	120.2	126.9	126.9	126.9	133.6	133.6	133.6	133.6
RE 25, 20 W	78	HED_ 5540 / AB 28	269/316	120.8	130.7	130.7	137.4	137.4	144.1	144.1	144.1	150.8	150.8	150.8	150.8
RE 25, 20 W	79	AB 28	316	115.1	125.0	125.0	131.7	131.7	138.4	138.4	138.4	145.1	145.1	145.1	145.1
RE 25, 20 W	79	HED_5540 / AB 28	316	132.2	142.1	142.1	148.8	148.8	155.5	155.5	155.5	162.2	162.2	162.2	162.2
RE 30, 60 W	80			94.5	104.4	104.4	111.1	111.1	117.8	117.8	117.8	124.5	124.5	124.5	124.5
RE 30, 60 W	80	MR	265	105.9	115.8	115.8	122.5	122.5	129.2	129.2	129.2	135.9	135.9	135.9	135.9
RE 35, 90 W	81			97.4	107.3	107.3	114.0	114.0	120.7	120.7	120.7	127.4	127.4	127.4	127.4
RE 35, 90 W	81	MR	265	108.8	118.7	118.7	125.4	125.4	132.1	132.1	132.1	138.8	138.8	138.8	138.8
RE 35, 90 W	81	HED_ 5540	268/270	118.4	128.3	128.3	135.0	135.0	141.7	141.7	141.7	148.4	148.4	148.4	148.4
RE 35, 90 W	81	DCT 22	277	115.5	125.4	125.4	132.1	132.1	138.8	138.8	138.8	145.5	145.5	145.5	145.5
RE 35, 90 W	81	AB 28	316	133.5	143.4	143.4	150.1	150.1	156.8	156.8	156.8	163.5	163.5	163.5	163.5
RE 35, 90 W	81	HEDS 5540 / AB 28	268/316	150.6	160.5	160.5	167.2	167.2	173.9	173.9	173.9	180.6	180.6	180.6	180.6
A-max 26	105-112			71.2	81.1	81.1	87.8	87.8	94.5	94.5	94.5	101.2	101.2	101.2	101.2
A-max 26	106-112	MEnc 13	275	78.3	88.2	88.2	94.9	94.9	101.6	101.6	101.6	108.3	108.3	108.3	108.3
A-max 26	106-112	MR	264	80.0	89.9	89.9	96.6	96.6	103.3	103.3	103.3	110.0	110.0	110.0	110.0
A-max 26	106-112	Enc 22	267	85.6	95.5	95.5	102.2	102.2	108.9	108.9	108.9	115.6	115.6	115.6	115.6
A-max 26	106-112	HED_ 5540	169/170	90.0	99.9	99.9	106.6	106.6	113.3	113.3	113.3	120.0	120.0	120.0	120.0
A-max 32	113/115			89.4	99.3	99.3	106.0	106.0	112.7	112.7	112.7	119.4	119.4	119.4	119.4
A-max 32	114/116			88.0	97.9	97.9	104.6	104.6	111.3	111.3	111.3	118.0	118.0	118.0	118.0
A-max 32	114/116	MR	265	99.2	109.1	109.1	115.8	115.8	122.5	122.5	122.5	129.2	129.2	129.2	129.2
A-max 32	114/116	HED_ 5540	268/270	108.8	118.7	118.7	125.4	125.4	132.1	132.1	132.1	138.8	138.8	138.8	138.8

May 2009 edition / subject to change

Planetary Gearhead GP 22 C ∅22 mm, 0.5 - 2.0 Nm

Ceramic Version



M 1:1

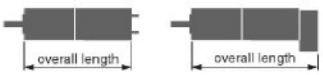
Technical Data

Planetary Gearhead	straight teeth
Output shaft	stainless steel, hardened
Bearing at output	ball bearing
Radial play, 10 mm from flange	max. 0.2 mm
Axial play	max. 0.2 mm
Max. radial load, 10 mm from flange	70 N
Max. permissible axial load	100 N
Max. permissible force for press fits	100 N
Sense of rotation, drive to output	–
Recommended input speed	< 8000 rpm
Recommended temperature range	-20 ... +100°C
Extended area as option	-35 ... +100°C

Stock program
Standard program
Special program (on request)

Order Number											
	143971	143974	143980	143986	143990	143996	144002	144004	144011	144017	144023
Gearhead Data											
1 Reduction	3.8 : 1	14 : 1	53 : 1	104 : 1	198 : 1	370 : 1	690 : 1	742 : 1	1386 : 1	1996 : 1	3189 : 1
2 Reduction absolute	15/4	225/16	3375/64	87723/845	50625/256	1658001/4561	69049/100	759375/1024	15234038/10985	28812027/14440	15643323/500
3 Max. motor shaft diameter mm	4	4	4	3.2	4	3.2	4	4	3.2	3.2	4
Order Number	143972	143975	143981	143987	143991	143997	144003	144006	144012	144018	144024
1 Reduction	4.4 :1	16 : 1	62 : 1	109 : 1	231 : 1	389 : 1	690 : 1	867 : 1	1460 : 1	2102 : 1	3728 : 1
2 Reduction absolute	57/13	855/52	12825/208	2187/20	192375/832	263169/676	1121931/1625	2888625/3328	3847535/2704	7105583/3380	30292137/8125
3 Max. motor shaft diameter mm	3.2	3.2	3.2	4	3.2	3.2	3.2	3.2	3.2	3.2	3.2
Order Number	143973	143976	143982	143988	143992	143998	144005	144007	144013	144019	144025
1 Reduction	5.4 : 1	19 : 1	72 : 1	128 : 1	270 : 1	410 : 1	850 : 1	1014 : 1	1538 : 1	2214 : 1	4592 : 1
2 Reduction absolute	27/5	3249/169	48735/676	41553/325	731025/2704	6561/16	531441/625	10965375/10816	98415/64	177147/80	14348907/3125
3 Max. motor shaft diameter mm	2.5	3.2	3.2	3.2	3.2	4	2.5	3.2	4	4	2.5
Order Number		143977	143983	143989	143993	143999		144008	144014	144020	
1 Reduction		20 : 1	76 : 1	157 : 1	285 : 1	455 : 1		1068 : 1	1621 : 1	2458 : 1	
2 Reduction absolute		81/4	1215/16	19683/125	18225/64	5003211/10985		273375/256	60962057/37259	13005697/5293	
3 Max. motor shaft diameter mm		4	4	2.5	4	3.2		4	3.2	3.2	
Order Number		143978	143984		143994	144000		144009	144015	144021	
1 Reduction		24 : 1	84 : 1		316 : 1	479 : 1		1185 : 1	1707 : 1	2589 : 1	
2 Reduction absolute		1539/65	185193/2197		2777895/8788	124659/260		41568425/35152	15006939/8788	9363793/1300	
3 Max. motor shaft diameter mm		3.2	3.2		3.2	3.2		3.2	3.2	3.2	
Order Number		143979	143985		143995	144001		144010	144016	144022	
1 Reduction		29 : 1	89 : 1		333 : 1	561 : 1		1249 : 1	1798 : 1	3027 : 1	
2 Reduction absolute		729/25	4617/52		69255/208	2368521/4225		10388625/832	373977/208	63950067/21125	
3 Max. motor shaft diameter mm		2.5	3.2		3.2	3.2		3.2	3.2	3.2	
4 Number of stages	1	2	3	3	4	4	4	5	5	5	5
5 Max. continuous torque Nm	0.5	0.6	1.2	1.2	1.8	1.8	1.8	2.0	2.0	2.0	2.0
6 Intermittently permissible torque at gear output Nm	0.8	0.9	1.9	1.9	2.7	2.7	2.7	3.0	3.0	3.0	3.0
7 Max. efficiency %	84	70	59	59	49	49	49	42	42	42	42
8 Weight g	42	55	68	68	81	81	81	94	94	94	94
9 Average backlash no load °	1.0	1.2	1.6	1.6	2.0	2.0	2.0	2.0	2.0	2.0	2.0
10 Mass inertia gcm²	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
11 Gearhead length L1* mm	25.4	32.2	39.0	39.0	45.8	45.8	45.8	52.6	52.6	52.6	52.6

*L1 is - 2.6 mm for calculating the overall length



maxon Modular System

+ Motor	Page	+ Sensor / Brake	Page	Overall length [mm] = Motor length + gearhead length + (sensor / brake) + assembly parts									
A-max 19	97/98			51.6	58.4	65.2	65.2	72.0	72.0	72.0	78.8	78.8	78.8	78.8
A-max 19, 1.5 W	98	MR	260/262	56.7	63.5	70.3	70.3	77.1	77.1	77.1	83.9	83.9	83.9	83.9
A-max 19, 1.5 W	98	Enc 22	267	66.0	72.8	79.6	79.6	86.4	86.4	86.4	93.2	93.2	93.2	93.2
A-max 19, 1.5 W	98	MEnc 13	275	59.1	65.9	72.7	72.7	79.5	79.5	79.5	86.3	86.3	86.3	86.3
A-max 19, 2.5 W	99/100			54.2	61.0	67.8	67.8	74.6	74.6	74.6	81.4	81.4	81.4	81.4
A-max 19, 2.5 W	100	MR	260/262	58.5	65.3	72.1	72.1	78.9	78.9	78.9	85.7	85.7	85.7	85.7
A-max 19, 2.5 W	100	Enc 22	267	68.6	75.4	82.2	82.2	89.0	89.0	89.0	95.8	95.8	95.8	95.8
A-max 19, 2.5 W	100	MEnc 13	275	61.7	68.5	75.3	75.3	82.1	82.1	82.1	88.9	88.9	88.9	88.9
A-max 22	101-104			54.6	61.4	68.2	68.2	75.0	75.0	75.0	81.8	81.8	81.8	81.8
A-max 22	102/104	MR	260/262	59.6	66.4	73.2	73.2	80.0	80.0	80.0	86.8	86.8	86.8	86.8
A-max 22	102/104	Enc 22	267	69.0	75.8	82.6	82.6	89.4	89.4	89.4	96.2	96.2	96.2	96.2
A-max 22	102/104	MEnc 13	275	61.7	68.5	75.3	75.3	82.1	82.1	82.1	88.9	88.9	88.9	88.9
RE-max 21	127/128			51.6	58.4	65.2	65.2	72.0	72.0	72.0	78.8	78.8	78.8	78.8
RE-max 21, 3.5 W	128	MR	261/263	56.7	63.5	70.3	70.3	77.1	77.1	77.1	83.9	83.9	83.9	83.9
RE-max 21	129/130			54.2	61.0	67.8	67.8	74.6	74.6	74.6	81.4	81.4	81.4	81.4
RE-max 21, 6 W	130	MR	261/263	58.5	65.3	72.1	72.1	78.9	78.9	78.9	85.7	85.7	85.7	85.7
RE-max 24	131-134			54.6	61.4	68.2	68.2	75.0	75.0	75.0	81.8	81.8	81.8	81.8
RE-max 24	132/134	MR	261/263	59.6	66.4	73.2	73.2	80.0	80.0	80.0	86.8	86.8	86.8	86.8

maxon accessories

Brake AB 20, 24 VDC, 0.1 Nm

Important Information

- Permanent magnet - single-face brake for DC (dry operation). Braking in unpowered condition.
- Holding brake, prevents rotation of the shaft at standstill or with turned off motor power.
- Not recommended for braking rotating motor shift.
- It is recommended to lower the voltage applied to the brake after it has been energized, for the purpose of reducing heat loss.

Stock program
Standard program
Special program (on request)

Order Number	
301212	301213

Type

Combination

+ Motor	Page	+ Gearhead	Page	+ Tacho	Page	Overall length [mm] / ● see: + Gearhead
EC-max 22, 12 W	174					67.0
EC-max 22, 12 W	174	GP 22, 0.5 - 2.0 Nm	221			●
EC-max 22, 25 W	175					93.5
EC-max 22, 25 W	175	GP 32, 1 - 6 Nm	230			●
EC-max 30, 40 W	176					74.1
EC-max 30, 40 W	176	GP 32, 1 - 6 Nm	230			●
EC-max 30, 40 W	176			HEDL 5540	254	94.7
EC-max 30, 40 W	176	GP 32, 1 - 6 Nm	230	HEDL 5540	254	●
EC-max 30, 60 W	177					86.1
EC-max 30, 60 W	177	GP 42, 3 - 15 Nm	233			●
EC-max 30, 60 W	177			HEDL 5540	254	116.7
EC-max 30, 60 W	177	GP 42, 3 - 15 Nm	233	HEDL 5540	254	●
EC-power 30	184					79.1
EC-power 30	184	GP 42, 3 - 15 Nm	233			●
EC-power 30	184			HEDL 5540	254	99.7
EC-power 30	184	GP 42, 3 - 15 Nm	233	HEDL 5540	254	●
EC-power 30	185					86.1
EC-power 30	185	GP 42, 3 - 15 Nm	233			●
EC-power 30	185			HEDL 5540	254	116.7
EC-power 30	185	GP 42, 3 - 15 Nm	233	HEDL 5540	254	●

Technical Data

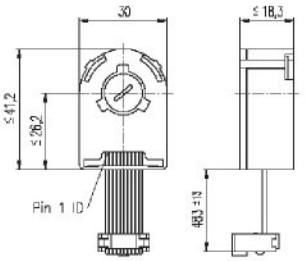
Static braking moment at 20°C	> 0.1 Nm	Nominal voltage, smoothed		24 VDC ± 10 %
Mass inertia	1.6 gcm²	Resistance		R_{20} = 227 Ω ± 6 %
Max. permissible speed	49 000 rpm	Duty cycle		100 %
Weight	29 g	Reaction time	– Rise time	12 ms
Ambient temperature range	-5 ... +80°C		– Fall time	6 ms

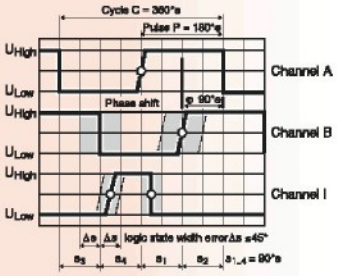
Pin Allocation

Cable (AWG 26)	Designation
red	U_{Brake} + 24 VDC
blue	U_{Brake} GND

maxon tacho

Encoder HEDL 5540, 500 CPT, 3 Channels, with Line Driver RS 422





Stock program
Standard program
Special program (on request)

Order Number	110512	110514	110516
Type			
Counts per turn	500	500	500
Number of channels	3	3	3
Max. operating frequency (kHz)	100	100	100
Shaft diameter [mm]	3	4	6

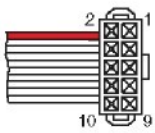
Combination

+ Motor	Page	+ Gearhead	Page	+ Brake	Page	Overall length [mm] / ● see: + Gearhead		
EC-max 30, 40 W	176					62.6		
EC-max 30, 40 W	176	GP 32, 1 - 6 Nm	230			●		
EC-max 30, 40 W	176			AB 20	286	94.7		
EC-max 30, 40 W	176	GP 32, 1 - 6 Nm	230	AB 20	286	●		
EC-max 30, 60 W	177					84.6		
EC-max 30, 60 W	177	GP 42, 3 - 15 Nm	233			●		
EC-max 30, 60 W	177			AB 20	286	116.7		
EC-max 30, 60 W	177	GP 42, 3 - 15 Nm	233	AB 20	286	●		
EC-max 40, 70 W	178						81.4	
EC-max 40, 70 W	178	GP 42, 3 - 15 Nm	233				●	
EC-max 40, 70 W	178			AB 28	287		121.4	
EC-max 40, 70 W	178	GP 42, 3 - 15 Nm	233	AB 28	287		●	
EC-max 40, 120 W	179						111.4	
EC-max 40, 120 W	179	GP 52, 4 - 30 Nm	236				●	
EC-max 40, 120 W	179			AB 28	287		151.4	
EC-max 40, 120 W	179	GP 52, 4 - 30 Nm	236	AB 28	287		●	
EC-powermax 30	184					87.6		
EC-powermax 30	184	GP 42, 3 - 15 Nm	233			●		
EC-powermax 30	184			AB 20	286	79.1		
EC-powermax 30	184	GP 42, 3 - 15 Nm	233	AB 20	286	●		
EC-powermax 30	185					94.6		
EC-powermax 30	185	GP 42, 3 - 15 Nm	233			●		
EC-powermax 30	185			AB 20	286	96.1		
EC-powermax 30	185	GP 42, 3 - 15 Nm	233	AB 20	286	●		

Technical Data

Supply voltage	5 V ± 10 %
Output signal	EIA Standard RS 422
drivers used:	DS26LS31
Phase shift Φ (nominal)	90°e
Logic state width s	min. 45°e
Signal rise time (typical at C_L = 25 pF, R_L = 2.7 kΩ, 25°C)	180 ns
Signal fall time (typical at C_L = 25 pF, R_L = 2.7 kΩ, 25°C)	40 ns
Index pulse width (nominal)	90°e
Operating temperature range	0 ... +70°C
Moment of inertia of code wheel	≤ 0.6 gcm²
Max. angular acceleration	250 000 rad s⁻²
Output current per channel	min. -20 mA, max. 20 mA
Option	1000 counts per turn, 2 channels

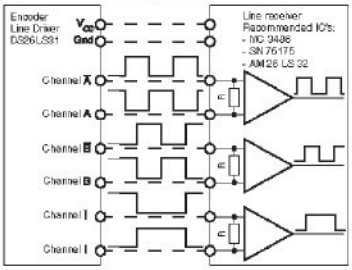
Pin Allocation



1 N.C
2 V_{CC}
3 GND
4 N.C
5 Channel $\overline{A}$
6 Channel A
7 Channel $\overline{B}$
8 Channel B
9 Channel $\overline{I}$ (Index)
10 Channel I (Index)

Pin type Berg 24677-0
flat band cable AWG 28

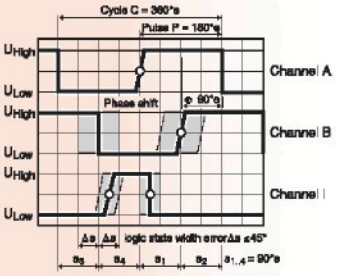
Connection example



Terminal resistance R = typical 100 Ω

April 2006 edition / subject to change

Encoder MR, Type M, 128 - 512 CPT, 2 / 3 Channels, with Line Driver



Stock program
Standard program
Special program (on request)

Order Number					
228179	228177	228181	228182	201937	201940

Type						
Counts per turn	128	128	256	256	512	512
Number of channels	2	3	2	3	2	3
Max. operating frequency (kHz)	80	80	160	160	320	320

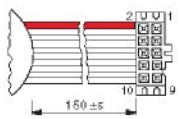
Combination						Overall length [mm] / ● see: + Gearhead					
+ Motor	Page	+ Gearhead	Page	+ Brake	Page						
EC 16, 15 W	154					50.9	50.9	50.9	50.9	50.9	50.9
EC 16, 15 W	154	GP 16, 0.1 - 0.3 Nm	213			●	●	●	●	●	●
EC 16, 40 W	155					66.9	66.9	66.9	66.9	66.9	66.9
EC 16, 40 W	155	GP 22, 0.5 - 2.0 Nm	221			●	●	●	●	●	●
EC 22, 20 W	157					50.5	50.5	50.5	50.5	50.5	50.5
EC 22, 20 W	157	GP 22, 0.5 - 2.0 Nm	221			●	●	●	●	●	●
EC 22, 50 W	159					68.7	68.7	68.7	68.7	68.7	68.7
EC 22, 50 W	159	GP 22, 0.5 - 2.0 Nm	221			●	●	●	●	●	●
EC-max 16, 5 W	171					31.3	31.3	31.3	31.3	31.3	31.3
EC-max 16, 5 W	171	GP 16, 0.1 - 0.3 Nm	213			●	●	●	●	●	●
EC-max 16, 8 W	173					43.3	43.3	43.3	43.3	43.3	43.3
EC-max 16, 8 W	173	GP 22, 0.5 - 2.0 Nm	221			●	●	●	●	●	●
EC-max 22, 12 W	174					41.7	41.7	41.7	41.7	41.7	41.7
EC-max 22, 12 W	174	GP 22, 0.5 - 2.0 Nm	221			●	●	●	●	●	●
EC-max 22, 25 W	175					58.2	58.2	58.2	58.2	58.2	58.2
EC-max 22, 25 W	175	GP 32, 1 - 6 Nm	230			●	●	●	●	●	●

Technical Data

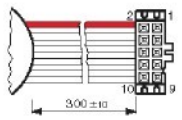
Supply voltage V_{CC}	5 V ± 5 %
Output signal	TTL compatible
Index pulse width (nominal)	90°e
Operating temperature range	-25 ... +85°C
Moment of inertia of code wheel	≤ 0.09 gcm²
Output current per channel	max. 5 mA

Attention: The index signal I is synchronised with channel A or B.

Pin Allocation



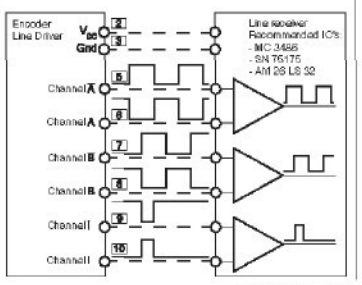
MR Encoder EC-max 16 / EC-max 22



MR Encoder EC 16 / EC 22

1 N.C.
2 VCC
3 GND
4 N.C.
5 Channel A
6 Channel Ā
7 Channel B
8 Channel B̄
9* Channel I (Index)
10* Channel Ī (Index)
DIN Connector 41651
flat band cable AWG 28
*version with 3 channels

Connection example



joystick.cpp

keyboard.cpp

kvadr_ss.cpp

kvadr_pos.cpp

Sou_sys.cpp

te_pivotace.cpp

kolo.cpp

phi_Cz.cpp

matice.cpp

jizda.cpp

zuzeni.cpp

text.cpp

disabling.cpp

extern.cpp

7 Závěr

V této práci je představeno kompletní konstrukční řešení kombinovaného robotizovaného podvozku, který umožňuje jak jízdu, tak kráčení. Toto řešení je poměrně unikátní, protože robotické podvozky obvykle umožňují pouze jeden ze způsobů pohybu a pokud umožňují oba způsoby, tak v dost omezené podobě.

Princip podvozku bude použit u vozíků pro osoby se sníženou pohyblivostí. Každá ze čtyř noh podvozku má pět stupňů volnosti, z nichž čtyři jsou ovládány servopohony. Celkem 20 stupňů volnosti ukazuje na poměrně velkou složitost konstrukčního řešení. Koncept umožňuje v režimu jízdy plynulou změnu rozvoru a rozchodu (změna rozvoru a rozchodu není na sobě nezávislá) a také nastavení světlé výšky podvozku.

Výroba samotného fyzického prototypu je vzhledem ke složitosti součástí drahou záležitostí. Z tohoto důvodů bylo využito možností moderní výpočetní techniky a byl nejprve sestaven simulační model.

Na simulačním modelu byla ověřena koncepce podvozku a funkčnost jeho odpružení využívajícím odvalováním hřebenu po ozubeném kole. Dále byly z modelu získány síly působící na jednotlivé nohy během jízdy, což pomůže při návrhu optimalizované konstrukce, neboť při prvotním návrhu tyto síly nebyly známé a byly odhadnuté z předpokládané hmotnosti podvozku.

Dalším z problémů, se kterým bylo nutné se vypořádat, je návrh regulačního algoritmu, který bude udržovat při jízdě po nerovném terénu rám podvozku ve vodorovné poloze (nebo v poloze, která se tomu bude blížit). Tento princip byl nejdříve vyzkoušen na jednoduchém 2D modelu bez odpružení, který byl pro tyto potřeby značně zjednodušen. Po zjištění vlivů regulačních parametrů na tento zjednodušený model byl obdobný princip aplikován na simulační model samotného konstrukčního řešení.

Model řízený kinematicky (ideálními motory) byl otestován jak na syntetickém testu, v němž robot stojí na pístech, které se pohybují podle navzájem různých harmonických funkcí, tak při simulaci jízdy na „náhodně“ vytvořeném terénu. Řadou simulací na těchto dvou případech byl zjištěn vliv parametrů regulace nejen na kvalitu regulovaného procesu (udržení vodorovné polohy podvozku), ale i vliv těchto parametrů na požadavky, které jsou kladeny na servopohony. Bylo zjištěno, že s rostoucí kvalitou výsledné regulované veličiny (co nejmenší náklon rámu od vodorovné polohy) rostou i žádané hodnoty momentů a otáček. Z těchto dvou protichůdných požadavků bylo zvoleno kompromisní řešení, které při přijatelném náklonu rámu podvozku od vodorovné polohy neklade na servopohon nereálné požadavky.

Dále byl sestaven model, v němž byly nahrazeny kinematické vazby momenty, které se řídí skutečnými momentovými charakteristikami servopohonů. Z tohoto modelu byl zjištěn vliv regulačních konstant řídících jednotek servopohonů.

Byly provedeny také simulace některých vylepšení regulovaného procesu (například rychlosti jízdy v závislosti na aktuálním naklopení rámu) a také regulace přímého směru jízdy. Na závěr byla vyzkoušena simulace jízdy po schodech dolů, což je jeden z požadavků na schopnosti, který byl na podvozek kladen.

Výsledkem práce je nejen detailní konstrukční model celého podvozku, ale také algoritmus regulace, a částečně parametrický model, u kterého lze měnit hmotové charakteristiky, v omezené míře také geometrii. Na tomto modelu lze samozřejmě testovat i další manévry než byly v této práci ukázány.

Všechny cíle vytyčené pro tuto práci byly splněny.

Simulační, případně výpočetní modely, je vždy vhodné porovnat se skutečností. Proto by dalším krokem měla být výroba podvozku a porovnání naměřených výsledků s výsledky ze simulací, což podá věrohodnější informace než samotný simulační model.

8 Seznam použité literatury

[1] Svatopluk Černoch: Strojně technická příručka, SNTL, Praha 1968

[2] Lubomír Pešík: Části strojů, stručný přehled 1,2 ISBN 80-7083-939-6, Liberec 2005

[3] Jan Leinveber, Pavel Vávra.: Strojnické tabulky, ISBN 80-7361-033-7, Úvaly 2006

[4] Doc. Ing. Ludvík Prášil, Csc. a kol: Části a mechanismy strojů, Liberec 1988

[5] Prof. Ing. František Boháček, DrSc. a kol: Části a mechanismy strojů II, hřídele, tribologie a ložiska, Brno 1987

[6] SKF Hlavní katalog, Praha 1998

[7] Milan Bezdíček, Pavel Čoupek, Robert Grepl, Jakub Hrabec, Jiří Konvičný, Martin Krejčiřík, Jiří Krejsa, Jiří Radoš, Jan Rajlich, František Šolc, Stanislav Věchet: Mechatronika, vybrané problémy, Brno 2008

[8] Dan Negrut, Andrew Dyer: ADAMS/Solver Primer, Ann Arbor, 2004

[9] Korf Jaroslav: Konstrukce podvozkové nohy robotizovaného podvozku, Liberec, 2007

[10] Denk Miroslav: Matematický model kinematiky robotizovaného podvozku se šestnácti stupni volnosti, Liberec, 2007

[11] Košťál Miroslav: Experimentální náprava robotizovaného podvozku, Liberec, 2010

[12] Vlček Jiří: Automatizace, 2008

[13] Klán Petr: Moderní metody nastavení PID regulátorů, část I. Procesy s přechodovou charakteristikou typu „S“. *Automa,* 2000, číslo 9, str. 54 – 57

[14] Klán Petr: Moderní metody nastavení PID regulátorů, část II. Integrační procesy. *Automa,* 2001, číslo 1, str. 52 – 55

[15] Pivoňka Petr: Fyzikální pohled na nastavování parametrů PID regulátorů metodou Zieglera a Nicholse. 0 *Automa*, 2003, číslo 3, str. 70 – 75

[16] Hlava Jaroslav: Číslicové PID regulatory. <www.fm.tul.cz/~krtsub/fm/par/digitalPID.pdf >

[17] Nastavování PID regulátorů. <www.fm.vslib.cz/~krtsub/fm/par/Nastavovani_PID.pdf>

[18] Shlegel Miloš: Průmyslové PID regulatory: tutorial. <www.rexcontrols.cz/downloads/clanky/PIDTutor_CZ.pdf>

[19] Hlava, Jaroslav: Prostředky automatického řízení II: Analogové a číslicové regulátory, elektrické pohony, průmyslové komunikační systémy. Praha 2000, Vydavatelství ČVUT

[20] Modrlák Osvald: Syntéza regulačních obvodů <www.fm.vslib.cz/~krtsub/fm/tr1/tar1_syn.pdf >

[21] Modrlák Osvald: Úvod do identifikace < www.fm.vslib.cz/~krtsub/fm/tr1/tar1_zid.pdf >

[22] Modrlák Osvald: Úvod do diskrétní parametrické identifikace < www.fm.vslib.cz/~krtsub/fm/modrlak/pdf/tar2_did.pdf >

[23] Modrlák Osvald: Základy číslicového řízení <www.fm.vslib.cz/~krtsub/fm/modrlak/pdf/tar2_zcr.pdf >

[24] Exploring Mount Erebus by walking robot: David Wttergreen, Chuck Thorpe, Red Whittaker, Robotics and Auntonomous Systems 11 (1993), str. 171 - 185

[25] <http://www.ottobock.com>

[26] <http://www.tankchair.com/gallery.htm>

[27] <http://www.kemcare.co.nz>

[28] <http://www.radicalmobility.com/products.htm>

[29] < http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=1185>

[30] <http://www.gel.usherbrooke.ca/laborius/projects/AZIMUT/index.html>

[31] <http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=1141>

[32] <http://biorobots.cwru.edu/projects/whegs/miniwhegs.html>

[33] <http://www.botjunkie.com/2007/09/04/hybrid-robot-can-walk-n-roll/>

[34] < http://www.cim.mcgill.ca/~jasmith/

[35] <http://www.robots-dreams.com/2006/02/robot_builder_p.html>

9 Publikace

Vzhledem k patentovému řízení a nebezpečí předzvěřejnění byla práce publikována pouze minimálně.

Veřejné publikace:

M. Denk, J. Korf. M. Šír: ROBOTIZED CHASSIS, Modeling of the Mechanical and Mechatronics Systems MMaMS'2009, Zemplínska Šírava, Slovakia, September 22. -24. 2009, page: 86, ISBN 978-80-553-0288-1, anglicky (33%)

M. Denk, J. Korf. M. Šír: ROBOTIZED CHASSIS, Metalurgija - Journal for Theory and Practice in Metellurgy, page: 168, ISSN 0543-5846, anglicky (40%)

J. Korf: SIMULAČNÍ MODELY ROBOTIZOVANÉHO PODVOZKU, Workshop pro doktorandy FS a FT TUL, Rokytnice nad Jizerou, 20. -23. září 2010, str: 190, ISBN 978-80-7372-642-3, česky

Neveřejné publikace (výzkumné zprávy):

J. Korf : Optimalizace niťové páky za účelem snížení setrvačných sil (pro firmu Minerva Boskovice)

J. Korf : Vedení řetězu pomocí vačky

J. Korf : Deformační a napěťová analýza konzole brzdového pedálu

J. Korf, O. Medůna : Optimalizace upínací desky elektroerozivní hloubičky CM 1265 C (pro firmu Modelárna LIAZ, s.r.o.)

J. Korf : Výpočet laserového obráběcího stroje LS64 (pro firmu SITEC Industrietechnologie GmbH)

J. Korf : Deformační analýza testovací stolice (pro firmu Linet)

J. Korf : Výpočet návrhu laserového obráběcího stroje LS108 (pro firmu SITEC Industrietechnologie GmbH)

J. Korf : Napěťová analýza rámu zadních dveří a konzoly podvozku Mi – 171 (pro firmu Modelárna LIAZ, s.r.o.)

J. Korf : Výpočet rámové konstrukce (pro firmu TPCA)

J. Korf, O. Medůna : Studie modelu pasivního ramene SPR 09 (pro firmu MZ Liberec, a.s.)

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#include "extern.h"

/*
*-----------------------------------------------------------------------------
*****************Definice seznamu operací a proměnných******************
*-----------------------------------------------------------------------------
*/
enum myenum operation;

bool start_pokus   = false;
bool start_zuzeni   = false;
bool start_krok     = false;
bool start_jizda    = false;
bool start_disabling = false;
bool start_ensabling = false;
bool zapis          = false;

int i_plan_rejd = 21;
int i_snek_rejd = 22;
int rozl_enco_rejd = 4*500;

int i_plan_vyro = 86;
int i_snek_vyro = 20;
int rozl_enco_vyro = 4*500;

int i_plan_pivo = 29;
int i_snek_pivo = 22;
int rozl_enco_pivo = 4*512;

int zobr = 0;

int rejd_zakl_pozice = 20;
int rejd_faze_zuzeni = 30;
int vyro_zakl_pozice = 20;

WORD rejd_1 = 1;
WORD vyro_1 = 2;
WORD pivo_1 = 3;
WORD rota_1 = 4;

WORD rejd_2 = 5;
WORD vyro_2 = 6;
WORD pivo_2 = 7;
WORD rota_2 = 8;

WORD rejd_3 = 9;
WORD vyro_3 = 10;
WORD pivo_3 = 11;
WORD rota_3 = 12;

WORD rejd_4 = 13;
WORD vyro_4 = 14;
WORD pivo_4 = 15;
WORD rota_4 = 16;

BOOL vyro_1_done ,pivo_1_done ,rejd_1_done;
BOOL vyro_2_done ,pivo_2_done ,rejd_2_done;
BOOL vyro_3_done ,pivo_3_done ,rejd_3_done;
BOOL vyro_4_done ,pivo_4_done ,rejd_4_done;

int   windowWidth;
int   windowHeight;
/*
*-----------------------------------------------------------------------------
***Funkce pro inicializaci okna a pro ošetření změny velikosti okna*******
*-----------------------------------------------------------------------------
*/
void onResize(int w, int h) {
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glViewport(0, 0, w, h);
     gluPerspective(45,(double)w/(double)h,1,100);
     glMatrixMode(GL_MODELVIEW);
     windowWidth=w;
     windowHeight=h;
}


```

```
 90: void init() {
 91:     glPointSize(3.0f);                          // body zobrazovat s rûmûrem 3 pixely
 92:     glEnable(GL_POINT_SMOOTH);                  // povolení antialiasingu bodu
 93:     glLineWidth(0.5f);                          // úsečky zobrazovat široké 1 pixel
 94:     glEnable(GL_LINE_SMOOTH);                   // povolení antialiasingu úseček
 95:     glDisable(GL_CULL_FACE);
 96:      glEnable(GL_DEPTH_TEST);
 97:      glEnable(GL_COLOR_MATERIAL);
 98:      glClearColor(0.0,0.0,0.0,0.0);
 99:      glEnable(GL_LIGHTING);
100:      glEnable(GL_LIGHT0);
101: }
102:
103: int fps,timeSinceStart,frames,lastTime;
104: /*
105: *-----------------------------------------------------------------------
106: *******Funkce pro měření počtu snímků za vteřinu**********
107: *-----------------------------------------------------------------------
108: */
109: void timer() {
110:      frames++;
111:      timeSinceStart=glutGet(GLUT_ELAPSED_TIME);  // čas od startu aplikace
112:      if ((timeSinceStart-lastTime)>1000) { // pokud uplynula vteřina od předchozího zápisu
113:         lastTime=timeSinceStart;
114:         fps=frames; // zapíše se, kolik proběhlo snímků za poslední vteřinu
115:         frames=0;
116:      }
117: }
118:
119:
120: float uhel;
121:
122: /*
123: *-----------------------------------------------------------------------
124: *******Hlavní zobrazovací funkce**********
125: *-----------------------------------------------------------------------
126: */
127: void onDisplay(void){
128:
129:
130:     timer();
131:     glClear( GL_DEPTH_BUFFER_BIT);
132:     glClearColor(1.0, 1.0, 1.0, 0.0);
133:     glClear(GL_COLOR_BUFFER_BIT);               // vymazání bitovych rovin barvového bufferu
134:
135:     setOrthogonalProjection();
136:     drawInfoText();                             // vypsat textové informace - viz text.cpp
137:     setPerspectiveProjection();
138:
139: /*
140: *-----------------------------------------------------------------------
141: ****Funkce pro výpočet natočení kol a pro generaci tranfor. matic*******
142: *-----------------------------------------------------------------------
143: */
144: nastaveni();
145: vytvorMatice();
146: /*
147: *-----------------------------------------------------------------------
148: ************Vykreslení středu zatáčky a základny podvozku**************
149: *-----------------------------------------------------------------------
150: */
151: if (zobr == 0 || zobr == 2 || zobr == 3) {
152:
153:   glLoadIdentity();
154:   glPushMatrix();
155:
156:     glMultMatrixf(MaticeGlobal);
157:     glColor3d(0.0,0.0,1.0);
158:      sou_sys();
159:      glTranslatef(0.0,-r,0.0);
160:      glColor3d(0.0,0.0,0.0);
161:      glutSolidSphere(0.1,  50, 50);
162:
163:      glPopMatrix();
164:
165:
166:
167:     glPushMatrix();
168:     glMultMatrixf(MaticePodvozek);
169:     glColor3d(1.0,0.0,0.0);
170:      sou_sys();
171:     kvadr_ss_v_T(2.0,1.0,0.05);
172:     glPopMatrix();
173:  /*
174: *-----------------------------------------------------------------------
175: *****************První noha*******************
176: *-----------------------------------------------------------------------
177: */
178:     glPushMatrix();
```

```
179:     glMultMatrixf(M_U1);
180:     glColor3d(0.0,0.0,0.0);
181:     glutSolidSphere(0.07,20,20);
182:     sou_sys();
183:     glPopMatrix();
184:
185:     glPushMatrix();
186:     glMultMatrixf(M_A1);
187:     glColor3d(0.0,1.0,0.0);
188:     sou_sys();
189:     kvadr_pos(0.45, 0.05, 0.05);
190:     glPopMatrix();
191:
192:     glPushMatrix();
193:     glMultMatrixf(M_B1);
194:     glColor3d(0.0,1.0,1.0);
195:     glutSolidSphere(0.05,20,20);
196:     sou_sys();
197:     kvadr_pos(1.8, 0.05, 0.05);
198:     glPopMatrix();
199:
200:     glPushMatrix();
201:     glMultMatrixf(M_C1);
202:     glColor3d(1.0,0.0,1.0);
203:     sou_sys();
204:     glutSolidSphere(0.05,20,20);
205:     te_pivotace();
206:     glPopMatrix();
207:
208:     glPushMatrix();
209:     glMultMatrixf(M_D1);
210:
211:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
212:
213:     kolo();
214:     glPopMatrix();
215: /*
216: *-------------------------------------------------------------------------
217: *****************Druhá noha******************
218: *-------------------------------------------------------------------------
219: */
220:     glPushMatrix();
221:     glMultMatrixf(M_U2);
222:     glColor3d(0.0,0.0,0.0);
223:     glutSolidSphere(0.07,20,20);
224:     sou_sys();
225:     glPopMatrix();
226:
227:     glPushMatrix();
228:     glMultMatrixf(M_A2);
229:     glColor3d(0.0,1.0,0.0);
230:     sou_sys();
231:     kvadr_pos(0.45, 0.05, 0.05);
232:     glPopMatrix();
233:
234:     glPushMatrix();
235:     glMultMatrixf(M_B2);
236:     glColor3d(0.0,1.0,1.0);
237:     glutSolidSphere(0.05,20,20);
238:     sou_sys();
239:     kvadr_pos(1.8, 0.05, 0.05);
240:     glPopMatrix();
241:
242:     glPushMatrix();
243:     glMultMatrixf(M_C2);
244:     glColor3d(1.0,0.0,1.0);
245:     sou_sys();
246:     glutSolidSphere(0.05,20,20);
247:     te_pivotace();
248:     glPopMatrix();
249:
250:     glPushMatrix();
251:     glMultMatrixf(M_D2);
252:
253:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
254:
255:     kolo();
256:     glPopMatrix();
257:
258: /*
259: *-------------------------------------------------------------------------
260: *****************Třetí noha******************
261: *-------------------------------------------------------------------------
262: */
263:     glPushMatrix();
264:     glMultMatrixf(M_U3);
265:     glColor3d(0.0,0.0,0.0);
266:     glutSolidSphere(0.07,20,20);
267:     sou_sys();
```

```
268:     glPopMatrix();
269:
270:     glPushMatrix();
271:     glMultMatrixf(M_A3);
272:     glColor3d(0.0,1.0,0.0);
273:     sou_sys();
274:     kvadr_pos(0.45, 0.05, 0.05);
275:     glPopMatrix();
276:
277:     glPushMatrix();
278:     glMultMatrixf(M_B3);
279:     glColor3d(0.0,1.0,1.0);
280:     glutSolidSphere(0.05,20,20);
281:     sou_sys();
282:     kvadr_pos(1.8, 0.05, 0.05);
283:     glPopMatrix();
284:
285:     glPushMatrix();
286:     glMultMatrixf(M_C3);
287:     glColor3d(1.0,0.0,1.0);
288:     sou_sys();
289:     glutSolidSphere(0.05,20,20);
290:     te_pivotace();
291:     glPopMatrix();
292:
293:     glPushMatrix();
294:     glMultMatrixf(M_D3);
295:
296:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
297:
298:     kolo();
299:     glPopMatrix();
300:
301: /*
302: *-----------------------------------------------------------------------------
303: *****************Čtvrtá noha*******************
304: *-----------------------------------------------------------------------------
305: */
306:     glPushMatrix();
307:     glMultMatrixf(M_U4);
308:     glColor3d(0.0,0.0,0.0);
309:     glutSolidSphere(0.07,20,20);
310:     sou_sys();
311:     glPopMatrix();
312:
313:     glPushMatrix();
314:     glMultMatrixf(M_A4);
315:     glColor3d(0.0,1.0,0.0);
316:     sou_sys();
317:     kvadr_pos(0.45, 0.05, 0.05);
318:     glPopMatrix();
319:
320:     glPushMatrix();
321:     glMultMatrixf(M_B4);
322:     glColor3d(0.0,1.0,1.0);
323:     glutSolidSphere(0.05,20,20);
324:     sou_sys();
325:     kvadr_pos(1.8, 0.05, 0.05);
326:     glPopMatrix();
327:
328:     glPushMatrix();
329:     glMultMatrixf(M_C4);
330:     glColor3d(1.0,0.0,1.0);
331:     sou_sys();
332:     glutSolidSphere(0.05,20,20);
333:     te_pivotace();
334:     glPopMatrix();
335:
336:     glPushMatrix();
337:     glMultMatrixf(M_D4);
338:
339:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
340:
341:     kolo();
342:     glPopMatrix();
343:
344: }
345: /*
346: *-----------------------------------------------------------------------------
347: *****************Zjištění informací o dokončení pohybů*******************
348: *-----------------------------------------------------------------------------
349: */
350:     VCS_GetMovementState(COM1, pivo_1,&pivo_1_done,&pErrorCode);
351:     VCS_GetMovementState(COM1, vyro_1,&vyro_1_done,&pErrorCode);
352:     VCS_GetMovementState(COM1, rejd_1,&rejd_1_done,&pErrorCode);
353:
354:     VCS_GetMovementState(COM2, pivo_2,&pivo_2_done,&pErrorCode);
355:     VCS_GetMovementState(COM2, vyro_2,&vyro_2_done,&pErrorCode);
356:     VCS_GetMovementState(COM2, rejd_2,&rejd_2_done,&pErrorCode);
```

```
357:
358:     VCS_GetMovementState(COM3, pivo_3,&pivo_3_done,&pErrorCode);
359:     VCS_GetMovementState(COM3, vyro_3,&vyro_3_done,&pErrorCode);
360:     VCS_GetMovementState(COM3, rejd_3,&rejd_3_done,&pErrorCode);
361:
362:     VCS_GetMovementState(COM4, pivo_4,&pivo_4_done,&pErrorCode);
363:     VCS_GetMovementState(COM4, vyro_4,&vyro_4_done,&pErrorCode);
364:     VCS_GetMovementState(COM4, rejd_4,&rejd_4_done,&pErrorCode);
365: /*
366: *-------------------------------------------------------------------------
367: ****************Rozdělení jednotlivých operací******************
368: *-------------------------------------------------------------------------
369: */
370:
371:     if(start_enabling)
372:      enabling();
373:
374:     if(start_disabling)
375:      disabling();
376:
377:     if(start_jizda)
378:      jizda();
379:
380:     if(start_zuzeni)
381:      zuzeni();
382:
383:     if(start_krok)
384:      krok();
385:
386:     if(zapis)
387:      fprintf(fw,"%d\t\t%d\t%f\t%d\n",glutGet(GLUT_ELAPSED_TIME),cas,
    (float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1),fps);
388:      glutSwapBuffers();
389: }
390: /*
391: *-------------------------------------------------------------------------
392: ****************Funkce main******************
393: *-------------------------------------------------------------------------
394: */
395: int main (int argc, char **argv) {
396:
397:     if(glutDeviceGet(GLUT_HAS_JOYSTICK) != 0) {
398:
399:         return 0;
400:     }
401:
402:         glutInit(&argc,argv);
403:     glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE);
404:     glutInitWindowSize(640,480);
405:
406:     glutInitWindowPosition(500,200);
407:
408:     glutCreateWindow(";-)");
409:
410:
411:     glEnable(GL_COLOR_MATERIAL);
412:     glutDisplayFunc(init);
413:     glutKeyboardFunc(keyboard);           //registrace funkce pro stisk klávesnice
414:     glutSpecialFunc(keyboard_special);    //registrace funkce pro stisk speciálních kláves
415:     glutJoystickFunc(joystick, 10);       //registrace funkce pro práci s joystickem
416:     glutIdleFunc(onDisplay);
417:     glutMouseFunc(onMouseButton);         //registrace funkce pro stisk tlačítka myši
418:     glutMotionFunc(onMouseMotion);        //registrace funkce pro pohyb myši
419:     glutReshapeFunc(onResize);
420:     init();
421:     glutMainLoop();
422:
423:     return 0;
424: }
425:
```

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#include "extern.h"
#define Pi 3.14159265
/*
*-----------------------------------------------------------------------------
*****************Definice proměnných******************
*-----------------------------------------------------------------------------
*/
float sign(float i);
float xo;
float yo;
float zo;
float r;
int xova;

BOOL Absolute = TRUE;
BOOL Immediately = TRUE ;
/*
*-----------------------------------------------------------------------------
*************Definice proměnných k identifikaci portu***************
*-----------------------------------------------------------------------------
*/
char* COM_1 = "COM1";
char* COM_2 = "COM1";
char* COM_3 = "COM1";
char* COM_4 = "COM1";

HANDLE COM1,COM2,COM3,COM4;
int power=0;
int faze = 0;
/*
*-----------------------------------------------------------------------------
*************Ošetření pohybu joystickem***************
*-----------------------------------------------------------------------------
*/
void joystick(unsigned int joystick, int x, int y, int z)
{
    xova = x;
    r = (float)  -2.0f*tan(((float) x)/1000.0f*Pi/2.0f-Pi/2.0f);
    xo = y/100.0f ;
    zo = -5+z/500.0f ;
/*
*-----------------------------------------------------------------------------
*************Ošetření stisku tlačítek joysticku***************
*-----------------------------------------------------------------------------
*/
switch (joystick)
    {
        case GLUT_JOYSTICK_BUTTON_A:
/*
*-----------------------------------------------------------------------------
*************Navázání spojení s řídícími jednotkami***************
*-----------------------------------------------------------------------------
*/
              COM1 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_1,&pErrorCode);
//            COM2 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_2,&pErrorCode);
//            COM3 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_3,&pErrorCode);
//            COM4 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_4,&pErrorCode);
              VCS_SetProtocolStackSettings(COM1,115200,500,&pErrorCode);
//            VCS_SetProtocolStackSettings(COM2,115200,500,&pErrorCode);
//            VCS_SetProtocolStackSettings(COM3,115200,500,&pErrorCode);
//            VCS_SetProtocolStackSettings(COM4,115200,500,&pErrorCode);


            COM2 = COM1;
            COM3 = COM1;
            COM4 = COM1;
/*
*-----------------------------------------------------------------------------
*************Vymazání předchozích chybových hlášení***************
*-----------------------------------------------------------------------------
*/
            VCS_ClearFault(COM1,rejd_1,&pErrorCode);
            VCS_ClearFault(COM1,vyro_1,&pErrorCode);
            VCS_ClearFault(COM1,pivo_1,&pErrorCode);
            VCS_ClearFault(COM1,rota_1,&pErrorCode);

            VCS_ClearFault(COM2,rejd_2,&pErrorCode);
            VCS_ClearFault(COM2,vyro_2,&pErrorCode);
```

```
90:             VCS_ClearFault(COM2,pivo_2,&pErrorCode);
91:             VCS_ClearFault(COM2,rota_2,&pErrorCode);
92:
93:             VCS_ClearFault(COM3,rejd_3,&pErrorCode);
94:             VCS_ClearFault(COM3,vyro_3,&pErrorCode);
95:             VCS_ClearFault(COM3,pivo_3,&pErrorCode);
96:             VCS_ClearFault(COM3,rota_3,&pErrorCode);
97:
98:             VCS_ClearFault(COM4,rejd_4,&pErrorCode);
99:             VCS_ClearFault(COM4,vyro_4,&pErrorCode);
100:            VCS_ClearFault(COM4,pivo_4,&pErrorCode);
101:            VCS_ClearFault(COM4,rota_4,&pErrorCode);
102: /*
103: *-----------------------------------------------------------------------
104: ************Nastavení operačních módů***************
105: *-----------------------------------------------------------------------
106: */
107:            VCS_SetOperationMode(COM1,rejd_1,0x01/*Profile Position Mode*/,&pErrorCode);
108:            VCS_SetOperationMode(COM1,vyro_1,0x01/*Profile Position Mode*/,&pErrorCode);
109:            VCS_SetOperationMode(COM1,pivo_1,0x01/*Profile Position Mode*/,&pErrorCode);
110:            VCS_SetOperationMode(COM1,rota_1,-0x02/*Velocity Mode       */,&pErrorCode);
111:
112:            VCS_SetOperationMode(COM2,rejd_2,0x01/*Profile Position Mode*/,&pErrorCode);
113:            VCS_SetOperationMode(COM2,vyro_2,0x01/*Profile Position Mode*/,&pErrorCode);
114:            VCS_SetOperationMode(COM2,pivo_2,0x01/*Profile Position Mode*/,&pErrorCode);
115:            VCS_SetOperationMode(COM2,rota_2,-0x02/*Velocity Mode       */,&pErrorCode);
116:
117:            VCS_SetOperationMode(COM3,rejd_3,0x01/*Profile Position Mode*/,&pErrorCode);
118:            VCS_SetOperationMode(COM3,vyro_3,0x01/*Profile Position Mode*/,&pErrorCode);
119:            VCS_SetOperationMode(COM3,pivo_3,0x01/*Profile Position Mode*/,&pErrorCode);
120:            VCS_SetOperationMode(COM3,rota_3,-0x02/*Velocity Mode       */,&pErrorCode);
121:
122:            VCS_SetOperationMode(COM4,rejd_4,0x01/*Profile Position Mode*/,&pErrorCode);
123:            VCS_SetOperationMode(COM4,vyro_4,0x01/*Profile Position Mode*/,&pErrorCode);
124:            VCS_SetOperationMode(COM4,pivo_4,0x01/*Profile Position Mode*/,&pErrorCode);
125:            VCS_SetOperationMode(COM4,rota_4,-0x02/*Velocity Mode       */,&pErrorCode);
126: /*
127: *-----------------------------------------------------------------------
128: ************Nastavení parametrů pohybu***************
129: *-----------------------------------------------------------------------
130: */
131:            VCS_SetPositionProfile(COM1,rejd_1,8700,20000,20000,&pErrorCode);
132:            VCS_SetPositionProfile(COM1,vyro_1,15000,20000,20000,&pErrorCode);
133:            VCS_SetPositionProfile(COM1,pivo_1,12000,20000,20000,&pErrorCode);
134:
135:            VCS_SetPositionProfile(COM2,rejd_2,8700,20000,20000,&pErrorCode);
136:            VCS_SetPositionProfile(COM2,vyro_2,15000,20000,20000,&pErrorCode);
137:            VCS_SetPositionProfile(COM2,pivo_2,12000,20000,20000,&pErrorCode);
138:
139:            VCS_SetPositionProfile(COM3,rejd_3,8700,20000,20000,&pErrorCode);
140:            VCS_SetPositionProfile(COM3,vyro_3,15000,20000,20000,&pErrorCode);
141:            VCS_SetPositionProfile(COM3,pivo_3,12000,20000,20000,&pErrorCode);
142:
143:            VCS_SetPositionProfile(COM4,rejd_4,8700,20000,20000,&pErrorCode);
144:            VCS_SetPositionProfile(COM4,vyro_4,15000,20000,20000,&pErrorCode);
145:            VCS_SetPositionProfile(COM4,pivo_4,12000,20000,20000,&pErrorCode);
146:
147:
148:            power=1;
149:            if (!start_krok && !start_jizda && !start_zuzeni){
150:            operation=ENABLING;
151:            start_enabling = true;
152:            pred_manevr=5;
153:            } else {
154:                   if( start_zuzeni) {
155:                      if(faze == 4) faze = 5;;
156:                      if(faze == 8) {faze = 9;}
157:                      operation = ZUZENI;
158:                      start_jizda = false;
159:                      pred_manevr=3;}
160:                   else{
161:                    if( start_krok) {
162:                      if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
163:                      operation = KROK;
164:                      start_jizda = false;
165:                      pred_manevr=3;}
166:                    else{
167:                      //operation = JIZDA;
168:                     // start_jizda = true;
169:                     // pred_manevr=3;
170:                      }
171:                }
172:              }
173:
174:           break;
175:        case GLUT_JOYSTICK_BUTTON_B:
176:             if( start_krok) {
177:                      if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
178:                      operation = KROK;
```

```
179:                        start_jizda = false;
180:                        pred_manevr=4;}
181:                 else{
182:                operation=DISABLING;
183:                start_disabling = true;
184:                pred_manevr=4;
185:                }
186:
187:                //VCS_CloseAllDevices(&pErrorCode);
188:
189:                break;
190:
191:            case GLUT_JOYSTICK_BUTTON_C:
192:                 if( start_krok) {
193:                         if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
194:                         operation = KROK;
195:                         start_jizda = false;
196:                         pred_manevr=1;}
197:                 else {
198:                  operation=ZUZENI;
199:                  start_zuzeni = true;
200:                  if(faze == 4){ faze = 5;}
201:                   if(faze == 8) {faze = 9;}
202:                  pred_manevr=1;
203:                  }
204:                break;
205:
206:            case GLUT_JOYSTICK_BUTTON_D:
207:                if( start_zuzeni) {
208:                  if(faze == 4){ faze = 5;}
209:                  if(faze == 8) {faze = 9;}
210:                  operation = ZUZENI;
211:                  start_krok = false;
212:                  pred_manevr=2;}
213:                else{
214:                  operation=KROK;
215:                  start_krok = true;
216:                  pred_manevr=2;
217:                  if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
218:                  }
219:
220:                break;
221:
222:
223:
224:        }
225:
226:        glutPostRedisplay();
227: }
228:
```

```
1: /*
2: *--------------------------------------------------------------------------
3: *****************Načtení příslušných knihoven******************
4: *--------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "Definitions.h"
12: #include "extern.h"
13:
14: /*
15: *--------------------------------------------------------------------------
16: *****************Informace o předchozím manévru******************
17: *--------------------------------------------------------------------------
18: */
19: int pred_manevr=0;                /* 0   -   nic
20:                                      1   -   zuzeni
21:                                      2   -   krok
22:                                      3   -   jizda
23:                                      4   -   disabling
24:                                      5   -   enabling    */
25:
26:
27: HANDLE m_KeyHandle;
28:
29:     HANDLE KeyHandle;
30:     __int8 Mode;
31:     DWORD pErrorCode;
32:     DWORD pBaudrate;
33:     DWORD pTimeout;
34:     DWORD NodeId;
35:
36:
37: long int StartPosition;
38:
39: /*
40: *--------------------------------------------------------------------------
41: *****************Ošetření stisku klávesy******************
42: *--------------------------------------------------------------------------
43: */
44:
45: void keyboard(unsigned char keyboard, int x, int y)
46: {
47:     switch (keyboard)
48:     {
49:
50:          case 'n':
51:             faze_krok++;
52:             break;
53:          case 'l':
54:             operation=KROK;
55:             start_krok = true;
56:             pred_manevr=2;
57:             break;
58:
59:          case 'j':
60:             operation=JIZDA;
61:             start_jizda = true;
62:             pred_manevr=3;
63:             break;
64:
65:          case 'k':
66:             operation=ZUZENI;
67:             start_zuzeni = true;
68:             if(faze == 4) faze++;
69:             pred_manevr=1;
70:             break;
71:
72:
73: /*
74: *--------------------------------------------------------------------------
75: *****************Možnost kontrolního zápisu do souboru******************
76: *--------------------------------------------------------------------------
77: */
78:         case 'o':
79:             if (zapis){ zapis = false;
80:             fclose(fw);}
81:             else {zapis = true;
82:                   fw = fopen("rozdily.txt","w");};
83:             break;
84:
85:         case 'r':
86:             operation=ROTATE;
87:             break;
88:
89:         case 'y':
```

```
            operation=POSUN;
            break;
/*
*-------------------------------------------------------------------------
*****************Změna způsobu zobrazení******************
*-------------------------------------------------------------------------
*/
        case 'c':
            if (zobr == 5) zobr = 0;
            else if (zobr == 4) zobr++;
            else if (zobr == 3) zobr++;
            else if (zobr == 2) zobr++;
            else if (zobr == 1) zobr++;
            else if (zobr == 0) zobr++;

            break;

            default:
                break;

    }

    glutPostRedisplay();
}
/*
*-------------------------------------------------------------------------
*****************Ošetření stisku speciální klávesy******************
*-------------------------------------------------------------------------
*/
void keyboard_special(int keyboard_special, int x, int y)
{
    switch (keyboard_special)
    {


        case GLUT_KEY_LEFT:
            xnew_T -= 0.2;
            break;
        case GLUT_KEY_RIGHT  :
            xnew_T += 0.2;
            break;
        case GLUT_KEY_UP  :
            ynew_T += 0.2;
            break;
        case GLUT_KEY_DOWN  :
            ynew_T -= 0.2;
            break;
            default:
                break;
    }

    glutPostRedisplay();
}

```

```
/*
*----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
/* Kvadr o rozmerech a,b,c - SS je v tezisti kvadru
*  - a ve smeru osy x
*  - b ve smeru osy y
*  - c ve smeru osy z
*/

void kvadr_ss_v_T(float a, float b, float c) {

     a /= 2;
     b /= 2;
     c /= 2;


     //stěna  kolma na z > 0
     glBegin(GL_QUADS);

     glNormal3f(0.0,0.0,1.0);
     glVertex3f(a,-b,c);
     glVertex3f(-a,-b,c);
     glVertex3f(-a,b,c);
     glVertex3f(a,b,c);
     glEnd();


     //stěna  kolma na z < 0
     glBegin(GL_QUADS);
     glNormal3f(0.0,0.0,-1.0);
     glVertex3f(a,b,-c);
     glVertex3f(-a,b,-c);
     glVertex3f(-a,-b,-c);
     glVertex3f(a,-b,-c);

     glEnd();

     //stěna  kolma na y > 0
     glBegin(GL_QUADS);
     glNormal3f(0.0,1.0,0.0);
     glVertex3f(a,b,c);
     glVertex3f(-a,b,c);
     glVertex3f(-a,b,-c);
     glVertex3f(a,b,-c);
     glEnd();

     //stěna  kolma na y < 0
     glBegin(GL_QUADS);
     glNormal3f(0.0,-1.0,0.0);
     glVertex3f(-a,-b,c);
     glVertex3f(a,-b,c);
     glVertex3f(a,-b,-c);
     glVertex3f(-a,-b,-c);

     glEnd();

     //stěna  kolma na x < 0
     glBegin(GL_QUADS);
     glNormal3f(-1.0,0.0,0.0);

     glVertex3f(-a,b,c);
     glVertex3f(-a,-b,c);
     glVertex3f(-a,-b,-c);
     glVertex3f(-a,b,-c);
     glEnd();

     //stěna  kolma na x > 0
     glBegin(GL_QUADS);
     glNormal3f(1.0,0.0,0.0);
     glVertex3f(a,-b,c);
     glVertex3f(a,b,c);
     glVertex3f(a,b,-c);
     glVertex3f(a,-b,-c);
     glEnd();

}
```

```
 1: /*
 2: *----------------------------------------------------------------------------
 3: *****************Načtení příslušných knihoven*******************
 4: *----------------------------------------------------------------------------
 5: */
 6: #include <gl/glut.h>
 7: #include <math.h>
 8: #include <stdarg.h>
 9: #include <stdio.h>
10: #include <stdlib.h>
11: /* Kvadr o rozmerech a,b,c - SS je v tezisti kvadru
12: *  - a ve smeru osy x
13: *  - b ve smeru osy y
14: *  - c ve smeru osy z
15: */
16:
17: void kvadr_pos(float a, float b, float c) {
18:
19:      a /= 2;
20:      b /= 2;
21:      c /= 2;
22:
23:
24:      //stěna  kolma na z > 0
25:      glBegin(GL_QUADS);
26:
27:      glNormal3f(0.0,0.0,1.0);
28:      glVertex3f(2.0*a,-b,c);
29:      glVertex3f(0.0,-b,c);
30:      glVertex3f(0.0,b,c);
31:      glVertex3f(2.0*a,b,c);
32:      glEnd();
33:
34:
35:      //stěna  kolma na z < 0
36:      glBegin(GL_QUADS);
37:      glNormal3f(0.0,0.0,-1.0);
38:      glVertex3f(2.0*a,b,-c);
39:      glVertex3f(0.0,b,-c);
40:      glVertex3f(0.0,-b,-c);
41:      glVertex3f(2.0*a,-b,-c);
42:
43:      glEnd();
44:
45:      //stěna  kolma na y > 0
46:      glBegin(GL_QUADS);
47:      glNormal3f(0.0,1.0,0.0);
48:      glVertex3f(2.0*a,b,c);
49:      glVertex3f(0.0,b,c);
50:      glVertex3f(0.0,b,-c);
51:      glVertex3f(2.0*a,b,-c);
52:      glEnd();
53:
54:      //stěna  kolma na y < 0
55:      glBegin(GL_QUADS);
56:      glNormal3f(0.0,-1.0,0.0);
57:      glVertex3f(0.0,-b,c);
58:      glVertex3f(2.0*a,-b,c);
59:      glVertex3f(2.0*a,-b,-c);
60:      glVertex3f(0.0,-b,-c);
61:
62:      glEnd();
63:
64:      //stěna  kolma na x < 0
65:      glBegin(GL_QUADS);
66:      glNormal3f(-1.0,0.0,0.0);
67:
68:      glVertex3f(0.0,b,c);
69:      glVertex3f(0.0,-b,c);
70:      glVertex3f(0.0,-b,-c);
71:      glVertex3f(0.0,b,-c);
72:      glEnd();
73:
74:      //stěna  kolma na x > 0
75:      glBegin(GL_QUADS);
76:      glNormal3f(1.0,0.0,0.0);
77:      glVertex3f(2.0*a,-b,c);
78:      glVertex3f(2.0*a,b,c);
79:      glVertex3f(2.0*a,b,-c);
80:      glVertex3f(2.0*a,-b,-c);
81:      glEnd();
82:
83: }
84:
```

```
/*
*----------------------------------------------------------------------------
*****************Načtení příslušných knihoven*******************
*----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
void printString3d(int x, int y, int z, char *text)
{
    glRasterPos3i(x, y, z);                        // pozice prvniho znaku retezce
    for (; *text; text++)                        // pruchod retezcem
        glutBitmapCharacter(GLUT_BITMAP_9_BY_15, *text);// vykresleni jednoho znaku
};
/*
*----------------------------------------------------------------------------
*****************Funkce pro vyznačení symbolu pro SS*******************
*----------------------------------------------------------------------------
*/

void sou_sys(){
    char popis_x[2],popis_y[2],popis_z[2];
    sprintf(popis_x,  "x");
    sprintf(popis_y,  "y");
    sprintf(popis_z,  "z");
     glBegin(GL_LINES);

     glNormal3f(0.0,0.0,1.0);

     glVertex3f(0.0,0.0,0.0);
     glVertex3f(1.0,0.0,0.0);


     glEnd();
     printString3d(1, 0,0, popis_x);

     glBegin(GL_LINES);
     glNormal3f(0.0,0.0,1.0);
     glVertex3f(0.0,0.0,0.0);
     glVertex3f(0.0,1.0,0.0);
     glEnd();
     printString3d(0, 1,0, popis_y);
     glBegin(GL_LINES);
     glNormal3f(0.0,0.0,1.0);
     glVertex3f(0.0,0.0,0.0);
     glVertex3f(0.0,0.0,1.0);

     glEnd();
     printString3d(0, 0,1, popis_z);
     }

```

```
/*
*----------------------------------------------------------------------------
*****************Načtení příslušných knihoven*******************
*----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "extern.h"

/*
*----------------------------------------------------------------------------
*****************Funkce pro vytvoření tělesa pivotace*******************
*----------------------------------------------------------------------------
*/
void te_pivotace(){
    // glPushMatrix();

     glRotatef(90.0,.0,1.0,0.0);
     glRotatef(-40.0,.0,0.0,1.0);
     kvadr_pos(1.059,0.05,0.05);
     glTranslatef(1.059,0.0,0.0);
     glutSolidCube(0.1);
     glRotatef((180.0-72.0),.0,0.0,1.0);
     kvadr_pos(0.736,0.05,0.05);
    // glPopMatrix();



     }

```

```
/*
*-------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#define PI 3.14159265
GLfloat mat_black_diffuse[] = { 0, 0.0, 0, 0 };
/*
*-------------------------------------------------------------------------
*****************Definice rovin pro ořez******************
*-------------------------------------------------------------------------
*/
GLdouble rov[4] = {0.0,-1.0,0.0,0.45};
GLdouble rov2[4] = {0.0,1.0,0.0,0.3};
int circle_points = 100;
/*
*-------------------------------------------------------------------------
*****************Funkce pro kreslení kola******************
*-------------------------------------------------------------------------
*/
void kolo(){

     glClipPlane(GL_CLIP_PLANE0,rov);
     glClipPlane(GL_CLIP_PLANE1,rov2);
     glEnable(GL_CLIP_PLANE0);
     glEnable(GL_CLIP_PLANE1);
     glRotatef(90,1,0,0);
     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);
     glutWireSphere(0.5, 50, 50);
    glColor3d(1.0,1.0,0.0);
    glutSolidSphere(0.4999,  50, 50);

     glDisable(GL_CLIP_PLANE0);
     glDisable(GL_CLIP_PLANE1);

     glBegin(GL_POLYGON);
/*
*-------------------------------------------------------------------------
*****************Tvorba čelních ploch kol******************
*-------------------------------------------------------------------------
*/

     for(int i = 0; i <circle_points; i++){
     float angle = 2 * PI * i / circle_points;
            float vel = sqrt(pow(0.5,2)-pow(0.45,2));
            glVertex3f(vel*cos(angle),vel*sin(angle),-0.45);

     }
     glEnd();
     glBegin(GL_POLYGON);

     for(int i = 0; i <circle_points; i++){
     float angle = 2 * PI * i / circle_points;
            float vel = sqrt(pow(0.5,2)-pow(0.3,2));
            glVertex3f(vel*cos(angle),vel*sin(angle),0.3);

     }
     glEnd();
     }

```

```
/*
*--------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*--------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "extern.h"
#define Pi 3.14159265
/*
*--------------------------------------------------------------------------
*****************Definice proměnných******************
*--------------------------------------------------------------------------
*/
float r_phi_Cz1,r_phi_By1,r_phi_Az1,r_phi_Cz2,r_phi_By2,r_phi_Az2,r_phi_Cz3,r_phi_By3,r_phi_Az3,r_phi_Cz4,
r_phi_By4,r_phi_Az4;


float pom_phi_Cz1 = -65.0,pom_phi_Cz2,pom_phi_Cz3,pom_phi_Cz4, pomocna;
//float pokus = 1.0;
int kk = 0,pocet = 0;
float pole[3][1000];
float rr_phi_Cz1,rr_phi_Cz2,rr_phi_Cz3,rr_phi_Cz4;
/*
*--------------------------------------------------------------------------
*****************Funkce pro nastavení pivotace******************
*--------------------------------------------------------------------------
*/
void nastaveni(){
     kk++;

/*
*--------------------------------------------------------------------------
*****************Převod jednotek úhlů na radiány******************
*--------------------------------------------------------------------------
*/
     r_phi_Az1 = Pi/180.0 * ((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1))
;
     r_phi_By1 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1));

/*
*--------------------------------------------------------------------------
*****************Rovnice exportovaná z SW Maple******************
*--------------------------------------------------------------------------
*/
     r_phi_Cz1 = -0.1e1 * atan(0.1000000000e0 * cos(r_phi_By1) * (0.7071067813e21 * cos(r_phi_Az1) * r +
0.7071067811e21 * sin(r_phi_Az1) * r - 0.3535533908e21 * sin(r_phi_Az1) + 0.1060660172e22 * cos(r_phi_Az1) -
0.1090000000e22 * sin(r_phi_By1) + 0.4500000000e21 + 0.1800000000e22 * cos(r_phi_By1)) / (0.7071067811e20 *
cos(r_phi_Az1) * r - 0.3535533908e20 * cos(r_phi_Az1) - 0.7071067813e20 * sin(r_phi_Az1) * r - 0.1060660172e21 *
sin(r_phi_Az1) + 0.7071067811e9));
/*
*--------------------------------------------------------------------------
*****************Zpětný převod jednotek******************
*--------------------------------------------------------------------------
*/
     rr_phi_Cz1 = 180.0/Pi*r_phi_Cz1;

/*
*--------------------------------------------------------------------------
*****************Ošetření periody funkce tangens******************
*--------------------------------------------------------------------------
*/
   if (kk > 10){
           if((fabs(pom_phi_Cz1 - rr_phi_Cz1)) > 150.0){
             if(pom_phi_Cz1 < rr_phi_Cz1) phi_Cz1 =rr_phi_Cz1-180.0;
             if(pom_phi_Cz1 > rr_phi_Cz1) phi_Cz1 =rr_phi_Cz1+180.0;

             }else phi_Cz1=rr_phi_Cz1;

           }
 pom_phi_Cz1 = phi_Cz1;



//-------------------------------------------------------------------
/*
*--------------------------------------------------------------------------
*****************Převod jednotek úhlů na radiány******************
*--------------------------------------------------------------------------
*/
     r_phi_Az2 = -Pi/180.0 *
((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az2));
     r_phi_By2 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By2));
/*
*--------------------------------------------------------------------------
*****************Rovnice exportovaná z SW Maple******************
```

```
 83: *------------------------------------------------------------------------
 84: */
 85:      r_phi_Cz2 =atan(0.1000000000e0 * cos(r_phi_By2) * (0.7071067813e21 * cos(r_phi_Az2) * r - 0.1060660172e22 *
     cos(r_phi_Az2) - 0.3535533908e21 * sin(r_phi_Az2) - 0.4500000000e21 - 0.7071067811e21 * sin(r_phi_Az2) * r +
     0.1090000000e22 * sin(r_phi_By2) - 0.1800000000e22 * cos(r_phi_By2)) / (0.7071067811e20 * cos(r_phi_Az2) * r +
     0.3535533908e20 * cos(r_phi_Az2) + 0.7071067813e20 * sin(r_phi_Az2) * r - 0.1060660172e21 * sin(r_phi_Az2) -
     0.707106781e9));
 86: /*
 87: *------------------------------------------------------------------------
 88: *****************Zpětný převod jednotek******************
 89: *------------------------------------------------------------------------
 90: */
 91:      rr_phi_Cz2 = 180.0/Pi*r_phi_Cz2- 180;
 92: /*
 93: *------------------------------------------------------------------------
 94: *****************Ošetření periody funkce tangens******************
 95: *------------------------------------------------------------------------
 96: */
 97:      if (kk > 10){
 98:           if((fabs(pom_phi_Cz2 - rr_phi_Cz2)) > 150.0){
 99:             if(pom_phi_Cz2 < rr_phi_Cz2) phi_Cz2 =rr_phi_Cz2-180.0;
100:             if(pom_phi_Cz2 > rr_phi_Cz2) phi_Cz2 =rr_phi_Cz2+180.0;
101:
102:             }else phi_Cz2=rr_phi_Cz2;
103:            }
104:            pom_phi_Cz2 = phi_Cz2;
105:
106: //---------------------------------------------------------------------
107: /*
108: *------------------------------------------------------------------------
109: *****************Převod jednotek úhlů na radiány******************
110: *------------------------------------------------------------------------
111: */
112:      r_phi_Az3 = Pi/180.0 * ((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az3));
113:      r_phi_By3 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By3));
114:
115: /*
116: *------------------------------------------------------------------------
117: *****************Rovnice exportovaná z SW Maple******************
118: *------------------------------------------------------------------------
119: */
120:      r_phi_Cz3 =-0.1e1 * atan(0.1000000000e0 * cos(r_phi_By3) * (-0.1060660172e22 * cos(r_phi_Az3) -
     0.3535533908e21 * sin(r_phi_Az3) - 0.7071067813e21 * cos(r_phi_Az3) * r + 0.7071067811e21 * sin(r_phi_Az3) * r +
     0.1090000000e22 * sin(r_phi_By3) - 0.4500000000e21 - 0.1800000000e22 * cos(r_phi_By3)) / (0.7071067811e20 *
     cos(r_phi_Az3) * r + 0.7071067813e20 * sin(r_phi_Az3) * r - 0.3535533908e20 * cos(r_phi_Az3) + 0.1060660172e21 *
     sin(r_phi_Az3) + 0.707106781e9));
121:
122: /*
123: *------------------------------------------------------------------------
124: *****************Zpětný převod jednotek******************
125: *------------------------------------------------------------------------
126: */
127:      rr_phi_Cz3 = 180.0/Pi*r_phi_Cz3- 180;
128: /*
129: *------------------------------------------------------------------------
130: *****************Ošetření periody funkce tangens******************
131: *------------------------------------------------------------------------
132: */
133:      if (kk > 10){
134:           if((fabs(pom_phi_Cz3 - rr_phi_Cz3)) > 150.0){
135:             if(pom_phi_Cz3 < rr_phi_Cz3) phi_Cz3 =rr_phi_Cz3-180.0;
136:             if(pom_phi_Cz3 > rr_phi_Cz3) phi_Cz3 =rr_phi_Cz3+180.0;
137:
138:             }else phi_Cz3=rr_phi_Cz3;
139:           }
140:            pom_phi_Cz3 = phi_Cz3;
141:
142: //---------------------------------------------------------------------
143: /*
144: *------------------------------------------------------------------------
145: *****************Převod jednotek úhlů na radiány******************
146: *------------------------------------------------------------------------
147: */
148:      r_phi_Az4 = Pi/180.0 * ((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az4));
149:      r_phi_By4 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By4));
150:
151: /*
152: *------------------------------------------------------------------------
153: *****************Rovnice exportovaná z SW Maple******************
154: *------------------------------------------------------------------------
155: */
156:      r_phi_Cz4 =-0.1e1 * atan(0.1000000000e0 * cos(r_phi_By4) * (0.7071067813e21 * cos(r_phi_Az4) * r +
     0.7071067811e21 * sin(r_phi_Az4) * r + 0.3535533908e21 * sin(r_phi_Az4) - 0.1060660172e22 * cos(r_phi_Az4) -
     0.4500000000e21 + 0.1090000000e22 * sin(r_phi_By4) - 0.1800000000e22 * cos(r_phi_By4)) / (0.1060660172e21 *
     sin(r_phi_Az4) + 0.7071067811e20 * cos(r_phi_Az4) * r + 0.3535533908e20 * cos(r_phi_Az4) - 0.707106781e9 -
     0.7071067813e20 * sin(r_phi_Az4) * r));
157:
158: /*
159: *------------------------------------------------------------------------
```

```
160: *****************Zpětný převod jednotek*******************
161: *-------------------------------------------------------------------------
162: */
163:
164:      rr_phi_Cz4 = 180.0/Pi*r_phi_Cz4- 180;
165: /*
166: *-------------------------------------------------------------------------
167: *****************Ošetření periody funkce tangens******************
168: *-------------------------------------------------------------------------
169: */
170:      if (kk > 10){
171:           if((fabs(pom_phi_Cz4 - rr_phi_Cz4)) > 50.0){
172:             if(pom_phi_Cz4 < rr_phi_Cz4) phi_Cz4 =rr_phi_Cz4-180.0;
173:             if(pom_phi_Cz4 > rr_phi_Cz4) phi_Cz4 =rr_phi_Cz4+180.0;
174:
175:             }else phi_Cz4=rr_phi_Cz4;
176:           }
177:            pom_phi_Cz4 = phi_Cz4;
178:      }
179:
```

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#include "extern.h"
/*
*-----------------------------------------------------------------------------
*****************Definice proměnných******************
*-----------------------------------------------------------------------------
*/
float MaticeGlobal[16],MaticeTglobal[16], MaticePodvozek[16],
      M_U1[16],M_A1[16],M_B1[16],M_C1[16], M_D1[16],
      M_U2[16],M_A2[16],M_B2[16],M_C2[16], M_D2[16],
      M_U3[16],M_A3[16],M_B3[16],M_C3[16], M_D3[16],
      M_U4[16],M_A4[16],M_B4[16],M_C4[16], M_D4[16];



float phi_Lz = 0.0,Lx = 0.0, Ly = 0.0, Lz = 1.5;
float phi_Az1 = (float) rejd_zakl_pozice,phi_Az2 = (float) rejd_zakl_pozice;
float phi_Az3 = (float) -rejd_zakl_pozice,phi_Az4 = (float) rejd_zakl_pozice;

float phi_By1 = (float) vyro_zakl_pozice,phi_By2 = (float) vyro_zakl_pozice;
float phi_By3 = (float) vyro_zakl_pozice,phi_By4 = (float) vyro_zakl_pozice;

float phi_Cz1, phi_Cz2,phi_Cz3, phi_Cz4;//= -phi_Az1-45.0,phi_Cz2 = phi_Az2-90.0;




long maxon_phi_Cz1,maxon_phi_By1,maxon_phi_Az1;
long maxon_phi_Cz2,maxon_phi_By2,maxon_phi_Az2;
long maxon_phi_Cz3,maxon_phi_By3,maxon_phi_Az3;
long maxon_phi_Cz4,maxon_phi_By4,maxon_phi_Az4;


/*
*-----------------------------------------------------------------------------
*************Funkce pro vytvoření tranformačních matic***************
*-----------------------------------------------------------------------------
*/

void vytvorMatice() {
/*
*-----------------------------------------------------------------------------
*****************Zjištění aktuálních poloh motorů,
       na základě kterých se vytvoří tranformační matice******************
*-----------------------------------------------------------------------------
*/

     VCS_GetPositionIs(COM1, rejd_1, &maxon_phi_Az1,&pErrorCode);
     VCS_GetPositionIs(COM1, vyro_1, &maxon_phi_By1,&pErrorCode);
     VCS_GetPositionIs(COM1, pivo_1, &maxon_phi_Cz1,&pErrorCode);

     VCS_GetPositionIs(COM2, rejd_2, &maxon_phi_Az2,&pErrorCode);
     VCS_GetPositionIs(COM2, vyro_2, &maxon_phi_By2,&pErrorCode);
     VCS_GetPositionIs(COM2, pivo_2, &maxon_phi_Cz2,&pErrorCode);

     VCS_GetPositionIs(COM3, rejd_3, &maxon_phi_Az3,&pErrorCode);
     VCS_GetPositionIs(COM3, vyro_3, &maxon_phi_By3,&pErrorCode);
     VCS_GetPositionIs(COM3, pivo_3, &maxon_phi_Cz3,&pErrorCode);

     VCS_GetPositionIs(COM4, rejd_4, &maxon_phi_Az4,&pErrorCode);
     VCS_GetPositionIs(COM4, vyro_4, &maxon_phi_By4,&pErrorCode);
     VCS_GetPositionIs(COM4, pivo_4, &maxon_phi_Cz4,&pErrorCode);

     glPushMatrix();

//Total global
    glLoadIdentity();
    glRotatef(90, 0.0, 0.0 ,1.0);
    glTranslatef(-0.5, -1.5,-10.0);

    glGetFloatv(GL_MODELVIEW_MATRIX, MaticeTglobal);
    glLoadIdentity();
//global
    glMultMatrixf(MaticeTglobal);

    glTranslatef(ynew_T, -xnew_T, znew);
    glRotatef(xnew, 1.0, 0.0 ,0.0);
    glRotatef(-ynew, 0.0, 1.0 ,0.0);          // rotace objektu podle pohybu kurzoru myši
    glGetFloatv(GL_MODELVIEW_MATRIX, MaticeGlobal);
```

```
 90:
 91:     glLoadIdentity();
 92:
 93: //Podvozek
 94:   glMultMatrixf(MaticeGlobal);
 95:
 96:       glTranslatef(Lx, Ly, Lz);
 97:       glRotatef(phi_Lz,0.0,0.0,1.0);
 98:       glGetFloatv(GL_MODELVIEW_MATRIX, MaticePodvozek);
 99:
100:       glLoadIdentity(); // vynuluje se
101: /*
102: *---------------------------------------------------------------------------
103: ****************První noha******************
104: *---------------------------------------------------------------------------
105: */
106: //Uchycení nohy
107:
108:       glMultMatrixf(MaticePodvozek);
109:
110:       glTranslatef(1.0, 0.5,0.0);
111:       glRotatef(45.0,0.0,0.0,1.0);
112:       glRotatef(-0.0,0.0,1.0,0.0);
113:       glGetFloatv(GL_MODELVIEW_MATRIX, M_U1);
114:
115:       glLoadIdentity(); // vynuluje se
116:
117: //První část
118:
119:       glMultMatrixf(M_U1);
120:
121:       glTranslatef(0.0, 0.0,0.0);
122:       glRotatef(((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1)),0.0,0.0,
    1.0);
123:       glGetFloatv(GL_MODELVIEW_MATRIX, M_A1);
124:       glLoadIdentity();
125:
126: //Druhá část
127:
128:       glMultMatrixf(M_A1);
129:
130:       glTranslatef(0.45, 0.0,0.0);
131:       glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1)),0.0,1.0,
    0.0);
132:       glGetFloatv(GL_MODELVIEW_MATRIX, M_B1);
133:       glLoadIdentity();
134:
135: //Třetí část
136:
137:       glMultMatrixf(M_B1);
138:
139:       glTranslatef(1.8, 0.0,0.0);
140:       glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz1)),0.0,0.0,
    1.0);
141:       glGetFloatv(GL_MODELVIEW_MATRIX, M_C1);
142:       glLoadIdentity();
143:
144: //Kolo
145:       glMultMatrixf(M_C1);
146:
147:       glTranslatef(0.0, 0.0,-1.09);
148:       glRotatef(-20.0,1.0,0.0,0.0);
149:       glRotatef(uhel,0.0,1.0,0.0);
150:
151:       glGetFloatv(GL_MODELVIEW_MATRIX, M_D1);
152:       glLoadIdentity();
153:
154: /*
155: *---------------------------------------------------------------------------
156: ****************Druhá noha******************
157: *---------------------------------------------------------------------------
158: */
159: //Uchycení nohy
160:
161:       glMultMatrixf(MaticePodvozek);
162:
163:       glTranslatef(1.0, -0.5,0.0);
164:       glRotatef(-45.0,0.0,0.0,1.0);
165:       glRotatef(-0.0,0.0,1.0,0.0);
166:       glGetFloatv(GL_MODELVIEW_MATRIX, M_U2);
167:
168:       glLoadIdentity(); // vynuluje se
169:
170: //První část
171:
172:       glMultMatrixf(M_U2);
173:
174:       glTranslatef(0.0, 0.0,0.0);
175:       glRotatef(((float)(-360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az2)),0.0,
    0.0,1.0);
```

```
176:      glGetFloatv(GL_MODELVIEW_MATRIX, M_A2);
177:      glLoadIdentity();
178:
179: //Druhá část
180:
181:      glMultMatrixf(M_A2);
182:
183:      glTranslatef(0.45, 0.0,0.0);
184:      glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By2)),0.0,1.0,
     0.0);
185:      glGetFloatv(GL_MODELVIEW_MATRIX, M_B2);
186:      glLoadIdentity();
187:
188: //Třetí část
189:
190:      glMultMatrixf(M_B2);
191:
192:      glTranslatef(1.8, 0.0,0.0);
193:      glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz2)),0.0,0.0,
     1.0);
194:      glGetFloatv(GL_MODELVIEW_MATRIX, M_C2);
195:      glLoadIdentity();
196:
197: //Kolo
198:      glMultMatrixf(M_C2);
199:
200:      glTranslatef(0.0, 0.0,-1.09);
201:      glRotatef(-20.0,1.0,0.0,0.0);
202:      glRotatef(-uhel,0.0,1.0,0.0);
203:
204:      glGetFloatv(GL_MODELVIEW_MATRIX, M_D2);
205:      glLoadIdentity();
206: /*
207: *-----------------------------------------------------------------------------
208: ****************Třetí noha*****************
209: *-----------------------------------------------------------------------------
210: */
211: //Uchycení nohy
212:
213:      glMultMatrixf(MaticePodvozek);
214:
215:      glTranslatef(-1.0, 0.5,0.0);
216:      glRotatef(135.0,0.0,0.0,1.0);
217:      glRotatef(-0.0,0.0,1.0,0.0);
218:      glGetFloatv(GL_MODELVIEW_MATRIX, M_U3);
219:
220:      glLoadIdentity(); // vynuluje se
221:
222: //První část
223:
224:      glMultMatrixf(M_U3);
225:
226:      glTranslatef(0.0, 0.0,0.0);
227:      glRotatef(((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az3)),0.0,0.0,
     1.0);
228:      glGetFloatv(GL_MODELVIEW_MATRIX, M_A3);
229:      glLoadIdentity();
230:
231: //Druhá část
232:
233:      glMultMatrixf(M_A3);
234:
235:      glTranslatef(0.45, 0.0,0.0);
236:      glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By3)),0.0,1.0,
     0.0);
237:      glGetFloatv(GL_MODELVIEW_MATRIX, M_B3);
238:      glLoadIdentity();
239:
240: //Třetí část
241:
242:      glMultMatrixf(M_B3);
243:
244:      glTranslatef(1.8, 0.0,0.0);
245:      glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz3)),0.0,0.0,
     1.0);
246:      glGetFloatv(GL_MODELVIEW_MATRIX, M_C3);
247:      glLoadIdentity();
248:
249: //Kolo
250:      glMultMatrixf(M_C3);
251:
252:      glTranslatef(0.0, 0.0,-1.09);
253:      glRotatef(-20.0,1.0,0.0,0.0);
254:      glRotatef(uhel,0.0,1.0,0.0);
255:
256:      glGetFloatv(GL_MODELVIEW_MATRIX, M_D3);
257:      glLoadIdentity();
258:
259:
```

```
260: /*
261: *-----------------------------------------------------------------------------
262: *****************Čtvrtá noha******************
263: *-----------------------------------------------------------------------------
264: */
265: //Uchycení nohy
266:
267:      glMultMatrixf(MaticePodvozek);
268:
269:      glTranslatef(-1.0, -0.5,0.0);
270:      glRotatef(-135.0,0.0,0.0,1.0);
271:      glRotatef(-0.0,0.0,1.0,0.0);
272:      glGetFloatv(GL_MODELVIEW_MATRIX, M_U4);
273:
274:      glLoadIdentity(); // vynuluje se
275:
276: //První část
277:
278:      glMultMatrixf(M_U4);
279:
280:      glTranslatef(0.0, 0.0,0.0);
281:      glRotatef(((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az4)),0.0,0.0,
     1.0);
282:      glGetFloatv(GL_MODELVIEW_MATRIX, M_A4);
283:      glLoadIdentity();
284:
285: //Druhá část
286:
287:      glMultMatrixf(M_A4);
288:
289:      glTranslatef(0.45, 0.0,0.0);
290:      glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By4)),0.0,1.0,
     0.0);
291:      glGetFloatv(GL_MODELVIEW_MATRIX, M_B4);
292:      glLoadIdentity();
293:
294: //Třetí část
295:
296:      glMultMatrixf(M_B4);
297:
298:      glTranslatef(1.8, 0.0,0.0);
299:      glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz4)),0.0,0.0,
     1.0);
300:      glGetFloatv(GL_MODELVIEW_MATRIX, M_C4);
301:      glLoadIdentity();
302:
303: //Kolo
304:      glMultMatrixf(M_C4);
305:
306:      glTranslatef(0.0, 0.0,-1.09);
307:      glRotatef(-20.0,1.0,0.0,0.0);
308:      glRotatef(-uhel,0.0,1.0,0.0);
309:
310:      glGetFloatv(GL_MODELVIEW_MATRIX, M_D4);
311:      glLoadIdentity();
312:
313:      glPopMatrix();
314: }
315:
```

```
1: /*
2: *------------------------------------------------------------------------
3: *****************Načtení příslušných knihoven******************
4: *------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "Definitions.h"
12: #define Pi 3.14159265
13: #include "extern.h"
14:
15:
16: bool start_enabling = false;
17: int faze_en = 1;
18:
19: void enabling(void){
20:
21: /*
22: *------------------------------------------------------------------------
23: *****Po dokončení pohybů manévr jízda*****
24: *------------------------------------------------------------------------
25: */
26:     if(faze_en == 4 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
27:           operation = JIZDA;
28:           start_enabling = false;
29:           start_jizda = true;
30:     }
31: /*
32: *------------------------------------------------------------------------
33: *****Pivotace na Enable*****
34: *------------------------------------------------------------------------
35: */
36:     if    (faze_en == 3 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done
    && pivo_4_done && rejd_4_done){
37:           VCS_SetEnableState(COM1,rota_1,&pErrorCode);
38:
39:           VCS_SetEnableState(COM2,rota_2,&pErrorCode);
40:
41:           VCS_SetEnableState(COM3,rota_3,&pErrorCode);
42:
43:           VCS_SetEnableState(COM4,rota_4,&pErrorCode);
44:           faze_en++;
45:     }
46: /*
47: *------------------------------------------------------------------------
48: *****Nastavení polohy rejdu, pivotace na Disable*****
49: *------------------------------------------------------------------------
50: */
51:     if    (faze_en == 2 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done
    && pivo_4_done && rejd_4_done){
52:            VCS_SetDisableState(COM1,rota_1,&pErrorCode);
53:            VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
54:
55:            VCS_SetDisableState(COM2,rota_2,&pErrorCode);
56:            VCS_MoveToPosition(COM2,rejd_2, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
57:
58:            VCS_SetDisableState(COM3,rota_3,&pErrorCode);
59:            VCS_MoveToPosition(COM3,rejd_3, (-i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,
    Absolute,Immediately,&pErrorCode);
60:
61:            VCS_SetDisableState(COM4,rota_4,&pErrorCode);
62:            VCS_MoveToPosition(COM4,rejd_4, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
63:
64:           faze_en++;
65:     }
66: /*
67: *------------------------------------------------------------------------
68: Nastavení všech pohonů na Enable a nastavení polohy pivotace a vyrovnávání
69: *------------------------------------------------------------------------
70: */
71:     if    (faze_en == 1 && pivo_1_done  && pivo_2_done &&  pivo_3_done && pivo_4_done){
72:           VCS_SetEnableState(COM1,rejd_1,&pErrorCode);
73:             VCS_SetEnableState(COM1,vyro_1,&pErrorCode);
74:             VCS_SetEnableState(COM1,pivo_1,&pErrorCode);
75:             VCS_SetEnableState(COM1,rota_1,&pErrorCode);
76:           VCS_MoveToPosition(COM1,vyro_1, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
77:           VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
78:
79:            VCS_SetEnableState(COM2,rejd_2,&pErrorCode);
80:             VCS_SetEnableState(COM2,vyro_2,&pErrorCode);
```

```
81:              VCS_SetEnableState(COM2,pivo_2,&pErrorCode);
82:              VCS_SetEnableState(COM2,rota_2,&pErrorCode);
83:            VCS_MoveToPosition(COM2,vyro_2, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
84:            VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
85:
86:             VCS_SetEnableState(COM3,rejd_3,&pErrorCode);
87:              VCS_SetEnableState(COM3,vyro_3,&pErrorCode);
88:              VCS_SetEnableState(COM3,pivo_3,&pErrorCode);
89:              VCS_SetEnableState(COM3,rota_3,&pErrorCode);
90:            VCS_MoveToPosition(COM3,vyro_3, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
91:            VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
92:
93:             VCS_SetEnableState(COM4,rejd_4,&pErrorCode);
94:              VCS_SetEnableState(COM4,vyro_4,&pErrorCode);
95:              VCS_SetEnableState(COM4,pivo_4,&pErrorCode);
96:              VCS_SetEnableState(COM4,rota_4,&pErrorCode);
97:            VCS_MoveToPosition(COM4,vyro_4, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
98:            VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
99:
100:           faze_en++;
101:
102:    }
103:
104:
105:
106:    }
107:
```

```
/*
*------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#define Pi 3.14159265
#include "extern.h"
/*
*------------------------------------------------------------------------
*****************Manévr jízda, podvozek základní konfigurace,
                   pivotace a rychlost řízena joystickem******************
*------------------------------------------------------------------------
*/
void jizda(void){

 if(start_jizda){
      VCS_MoveToPosition(COM1,rejd_1, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az1),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM1,vyro_1, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By1),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM1,pivo_1, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz1),Absolute,
Immediately,&pErrorCode);
     // VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_1, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);


      VCS_MoveToPosition(COM2,rejd_2, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az2),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM2,vyro_2, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By2),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM2,pivo_2, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz2),Absolute,
Immediately,&pErrorCode);
    //  VCS_SetVelocityMust((HANDLE)COM2, (WORD)rota_2, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);

      VCS_MoveToPosition(COM3,rejd_3, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az3),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM3,vyro_3, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By3),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM3,pivo_3, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz3),Absolute,
Immediately,&pErrorCode);
     // VCS_SetVelocityMust((HANDLE)COM3, (WORD)rota_3, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);

      VCS_MoveToPosition(COM4,rejd_4, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az4),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM4,vyro_4, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By4),Absolute,
Immediately,&pErrorCode);
      VCS_MoveToPosition(COM4,pivo_4, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz4),Absolute,
Immediately,&pErrorCode);
      VCS_SetVelocityMust((HANDLE)COM4, (WORD)rota_4, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);}
}

```

```
1: /*
2: *------------------------------------------------------------------------
3: *****************Načtení příslušných knihoven*******************
4: *------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "Definitions.h"
12: #define Pi 3.14159265
13: #include "extern.h"
14:
15:
16:
17:
18:
19: void zuzeni(void){
20:    if(faze == 0) faze = 1;
21:    if(start_jizda ) start_jizda = false;
22:
23: /*
24: *------------------------------------------------------------------------
25: ************Zpět na předchozí manévr**************
26: *------------------------------------------------------------------------
27: */
28:    if    (faze == 11 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
29:
30:           VCS_SetEnableState(COM1,rota_1,&pErrorCode);
31:
32:           VCS_SetEnableState(COM2,rota_2,&pErrorCode);
33:
34:           VCS_SetEnableState(COM3,rota_3,&pErrorCode);
35:
36:           VCS_SetEnableState(COM4,rota_4,&pErrorCode);
37:
38:           faze = 0;
39:
40:           if(pred_manevr == 1 || pred_manevr == 3){
41:             operation=JIZDA;
42:             start_jizda = true;}
43:
44:           if(pred_manevr == 2 ) {
45:             operation=KROK;
46:             start_krok = true;
47:             }
48:           if(pred_manevr == 4 ) {
49:             operation=DISABLING;
50:             start_disabling = true;
51:             }
52:
53:
54:           start_zuzeni = false;
55:     }
56: /*
57: *------------------------------------------------------------------------
58: ************Návrat do základní konfigurace**************
59: *------------------------------------------------------------------------
60: */
61:    if    (faze == 10 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
62:
63:
64:           VCS_MoveToPosition(COM3,rejd_3, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
65:
66:           VCS_MoveToPosition(COM4,rejd_4, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
67:
68:           faze++;
69:
70:     }
71: /*
72: *------------------------------------------------------------------------
73: ************Natočení kol do pozice vhodné pro pohyb rejdu**************
74: *------------------------------------------------------------------------
75: */
76:    if    (faze == 9 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
77:
78:
79:           VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
80:           VCS_SetDisableState(COM3,rota_3,&pErrorCode);
81:
82:           VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
```

```
 83:           VCS_SetDisableState(COM4,rota_4,&pErrorCode);
 84:
 85:           faze++;
 86:
 87:     }
 88:
 89: /*
 90: *-----------------------------------------------------------------------
 91: ***************Nastavení pohonu kol do enable*****************
 92: *-----------------------------------------------------------------------
 93: */
 94:    if    (faze == 7 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
 95:
 96:
 97:           VCS_SetEnableState(COM3,rota_3,&pErrorCode);
 98:
 99:           VCS_SetEnableState(COM4,rota_4,&pErrorCode);
100:
101:           faze++; }
102:
103: /*
104: *-----------------------------------------------------------------------
105: ***************Změna rejdu*****************
106: *-----------------------------------------------------------------------
107: */
108:    if    (faze == 6 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
109:
110:           VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
     Immediately,&pErrorCode);
111:
112:           VCS_MoveToPosition(COM2,rejd_2, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
     Immediately,&pErrorCode);
113:
114:           VCS_MoveToPosition(COM3,rejd_3, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
115:
116:           VCS_MoveToPosition(COM4,rejd_4, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
117:           faze++;
118:     }
119:
120: /*
121: *-----------------------------------------------------------------------
122: ************Natočení kol do pozice vhodné pro pohyb rejdu**************
123: *-----------------------------------------------------------------------
124: */
125:     if    (faze == 5 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
126:
127:           VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
128:           VCS_SetDisableState(COM1,rota_1,&pErrorCode);
129:
130:           VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
131:           VCS_SetDisableState(COM2,rota_2,&pErrorCode);
132:
133:           VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
134:           VCS_SetDisableState(COM3,rota_3,&pErrorCode);
135:
136:           VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
137:           VCS_SetDisableState(COM4,rota_4,&pErrorCode);
138:
139:           faze++;
140:
141:     }
142: /*
143: *-----------------------------------------------------------------------
144: ***************Nastavení pohonu kol do enable*****************
145: *-----------------------------------------------------------------------
146: */
147:     if    (faze == 3 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
148:
149:           VCS_SetEnableState(COM1,rota_1,&pErrorCode);
150:
151:           VCS_SetEnableState(COM2,rota_2,&pErrorCode);
152:
153:
154:           faze++;
155:     }
156: /*
157: *-----------------------------------------------------------------------
158: ***************Změna rejdu*****************
159: *-----------------------------------------------------------------------
```

```
160: */
161:     if    (faze == 2 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
162:           VCS_MoveToPosition(COM1,rejd_1, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
163:
164:           VCS_MoveToPosition(COM2,rejd_2, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
165:
166:
167:           faze++;
168:     }
169: /*
170: *-------------------------------------------------------------------------
171: ************Natočení kol do pozice vhodné pro pohyb rejdu**************
172: *-------------------------------------------------------------------------
173: */
174:     if    (faze == 1 && pivo_1_done && pivo_2_done && pivo_3_done && pivo_4_done){
175:
176:            VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
177:            VCS_SetDisableState(COM1,rota_1,&pErrorCode);
178:
179:            VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
180:            VCS_SetDisableState(COM2,rota_2,&pErrorCode);
181:
182:
183:           faze++;
184:
185:     }
186:
187:
188: /*
189: *-------------------------------------------------------------------------
190: ***************Při fázi 4 a 8 ovládání pohybu joystickem*****************
191: *-------------------------------------------------------------------------
192: */
193:
194:
195:     if(operation == ZUZENI && ((faze ==4) || (faze ==8)))
196:     {
197:
198:
199:       VCS_MoveToPosition(COM1,pivo_1, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz1),Absolute,
     Immediately,&pErrorCode);
200:       VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_1, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
201:
202:       VCS_MoveToPosition(COM2,pivo_2, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz2),Absolute,
     Immediately,&pErrorCode);
203:       VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_2, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
204:
205:       VCS_MoveToPosition(COM3,pivo_3, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz3),Absolute,
     Immediately,&pErrorCode);
206:       VCS_SetVelocityMust((HANDLE)COM3, (WORD)rota_3, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
207:
208:       VCS_MoveToPosition(COM4,pivo_4, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz4),Absolute,
     Immediately,&pErrorCode);
209:       VCS_SetVelocityMust((HANDLE)COM3, (WORD)rota_4, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
210:
211:     }
212:     }
213:
```

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#define Pi 3.14159265
#include "extern.h"


/*
*-----------------------------------------------------------------------------
*****************Definice proměnných******************
*-----------------------------------------------------------------------------
*/
float pomoc = 0.0;
int faze_krok = 0;
int cas,start_cas;

int v2 = 10360/2;
int v1 = v2 / 2;
int v3 = 0*2;
int pocetek_rejd_krok = 10;
int v_n = 1900/20*86/2;
FILE *fw;
/*
*-----------------------------------------------------------------------------
*****************Funkce zajišťující krok,
                                    prozatím jedna noha******************
*-----------------------------------------------------------------------------
*/

void krok(void){
 if(start_jizda) start_jizda = false;


 if(faze_krok == 0) faze_krok = 1;

/*
*-----------------------------------------------------------------------------
*********Ošetření předchozího manévru***********
*-----------------------------------------------------------------------------
*/
 if(faze_krok == 11 && rejd_1_done){
    if(pred_manevr == 1) {
            operation=ZUZENI;
            faze_krok = 0;
            start_zuzeni = true;
            start_krok = false;

    }
    if(pred_manevr == 4){
                    operation=DISABLING;
                    faze_krok = 0;
                    start_disabling = true;
                    start_krok = false;}

    if(pred_manevr == 2 || pred_manevr == 3){
                    operation=JIZDA;
                    faze_krok = 0;
                    start_jizda = true;
                    start_krok = false;}

 }
/*
*-----------------------------------------------------------------------------
*********Návrat do základní konfigurace***********
*-----------------------------------------------------------------------------
*/
  if(faze_krok == 10 && pivo_1_done && rejd_1_done){

    VCS_SetEnableState(COM1,rota_1,&pErrorCode);
    faze_krok++;
 }



 if(faze_krok == 9 && vyro_1_done ){
   VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
Immediately,&pErrorCode);
     VCS_SetDisableState(COM1,rota_1,&pErrorCode);
    faze_krok++;
 }
```

```
 89:
 90:
 91:
 92:  if(faze_krok == 8){
 93:      VCS_SetOperationMode(COM1,rejd_1,0x01/*Profile Position Mode*/,&pErrorCode);
 94:      VCS_SetOperationMode(COM1,vyro_1,0x01/*Profile Position Mode*/,&pErrorCode);
 95:      VCS_MoveToPosition(COM1,vyro_1, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
 96:      faze_krok++;
 97:  }
 98:
 99:
100:
101:
102:  if(faze_krok == 7 && vyro_1_done && rejd_1_done) {
103:  VCS_SetOperationMode(COM1,rejd_1,-0x02/*Profile Position Mode*/,&pErrorCode);
104:  VCS_SetOperationMode(COM1,vyro_1,-0x02/*Profile Position Mode*/,&pErrorCode);
105:  faze_krok = 4;}
106:
107:  if(faze_krok == 6 ) {
108:  if( fabs((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1) -
    pocetek_rejd_krok) > 7.0 ||   //3.0
109:      fabs((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1) - vyro_zakl_pozice)
    > 5.0 ){     //     2.0
110:  //pomoc = fabs((float)(360.0/(float)(i_plan*i_snek*rozl_enco)*(float)maxon_phi_Az1)) -
    ((i_plan*i_snek*rozl_enco)/360*pocetek_rejd_krok);
111:
112:  VCS_SetOperationMode(COM1,rejd_1,0x01/*Profile Position Mode*/,&pErrorCode);
113:  VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*pocetek_rejd_krok,Absolute,
    Immediately,&pErrorCode);
114:
115:
116:  VCS_SetOperationMode(COM1,vyro_1,0x01/*Profile Position Mode*/,&pErrorCode);
117:  VCS_MoveToPosition(COM1,vyro_1, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
118:
119:
120:  faze_krok++;}
121:  else   faze_krok = 4;}
122:
123:
124: /*
125: *-----------------------------------------------------------------------------
126: *********Vlastní fáze kroku***********
127: *-----------------------------------------------------------------------------
128: */
129:  if(faze_krok == 5 ){
130:      cas = (glutGet(GLUT_ELAPSED_TIME) - start_cas);
131:      if(cas < 1000){
132:        VCS_SetVelocityMust(COM1,rejd_1, v1,&pErrorCode);
133:        VCS_SetVelocityMust(COM1,vyro_1, 0,&pErrorCode);}
134:      if((cas > 1000) && (cas < 1500)){
135:        VCS_SetVelocityMust(COM1,rejd_1, -v2,&pErrorCode);
136:        VCS_SetVelocityMust(COM1,vyro_1, -v_n,&pErrorCode);}
137:      if((cas > 1500) && (cas < 2000)){
138:        VCS_SetVelocityMust(COM1,rejd_1, -v3,&pErrorCode);
139:        VCS_SetVelocityMust(COM1,vyro_1, v_n,&pErrorCode);}
140:      if(cas > 2000){
141:        VCS_SetVelocityMust(COM1,rejd_1, 0,&pErrorCode);
142:        VCS_SetVelocityMust(COM1,vyro_1, 0,&pErrorCode);
143:        faze_krok++;    }
144:
145:
146:  }
147:
148: /*
149: *-----------------------------------------------------------------------------
150: *********Začátek měření času***********
151: *-----------------------------------------------------------------------------
152: */
153:  if(faze_krok == 4 && rejd_1_done ){
154:      start_cas = glutGet(GLUT_ELAPSED_TIME);
155:
156:     faze_krok++;
157:  }
158: /*
159: *-----------------------------------------------------------------------------
160: *********Změna pracovních módů pro manévr chůze***********
161: *-----------------------------------------------------------------------------
162: */
163:  if(faze_krok == 3 && rejd_1_done  ){
164:
165:
166:    VCS_SetOperationMode(COM1,rejd_1,-0x02/*Profile Position Mode*/,&pErrorCode);
167:    VCS_SetOperationMode(COM1,vyro_1,-0x02/*Profile Position Mode*/,&pErrorCode);
168:     VCS_SetEnableState(COM1,rota_1,&pErrorCode);
169:     faze_krok++;
170:  }
171: /*
```

```
172: *-----------------------------------------------------------------------
173: *********Přesun do konfigurace vhodné pro počátek manévru***********
174: *-----------------------------------------------------------------------
175: */
176:  if(faze_krok == 2 && pivo_1_done && rejd_1_done ){
177:    VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*pocetek_rejd_krok,Absolute,
     Immediately,&pErrorCode);
178:     faze_krok++;
179:  }
180:
181:  if     (faze_krok == 1 && pivo_1_done){
182:
183:           VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
184:            VCS_SetDisableState(COM1,rota_1,&pErrorCode);
185:           faze_krok++;
186:
187:     }
188: /*
189: *-----------------------------------------------------------------------
190: ****************Kolo je při tomto manévru zabrzděno******************
191: *-----------------------------------------------------------------------
192: */
193:  if(operation == KROK)
194:     {
195:       VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_1, 0,(DWORD*) &pErrorCode);
196:     }
197:  }
198:
```

```
1: /*
2: *----------------------------------------------------------------------------
3: *****************Načtení příslušných knihoven******************
4: *----------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "extern.h"
12: /*
13: *----------------------------------------------------------------------------
14: *****************Definice proměnných******************
15: *----------------------------------------------------------------------------
16: */
17: char *nazvy[] = {"Rotate", "Translate","Rejd","Jizda","Zuzeni","Krok","Disabling","Enabling","Pokus",
    "Vyrovnavani","Posun"};
18: char *en[] = {"Disable", "Enable"};
19: char *bo[] = {"false","true"};
20:
21: /*
22: *----------------------------------------------------------------------------
23: *****************Nastavení perspektivní projekce******************
24: *----------------------------------------------------------------------------
25: */
26:
27: void setPerspectiveProjection(void)
28: {
29:     glMatrixMode(GL_PROJECTION);                // změna aktuální modifikované matice
30:     glLoadIdentity();
31:     gluPerspective(45,(double)windowWidth/(double)windowHeight,1,100);// nastavení perspektivní kamery
32:     glMatrixMode(GL_MODELVIEW);                 // změna aktuální modifikované matice
33:     glLoadIdentity();
34: }
35:
36: /*
37: *----------------------------------------------------------------------------
38: *****************Nastavení ortogonalní projekce******************
39: *----------------------------------------------------------------------------
40: */
41:
42: void setOrthogonalProjection(void)
43: {
44:     glMatrixMode(GL_PROJECTION);                // změna aktuální modifikované matice
45:     glLoadIdentity();
46:     glOrtho(0, windowWidth, 0, windowHeight, -100, 100);
47:     glMatrixMode(GL_MODELVIEW);                 // změna aktuální modifikované matice
48:     glLoadIdentity();
49: }
50:
51: void printString(int x, int y, char *text)
52: {
53:     glRasterPos2i(x, y);                        // pozice prvního znaku řetězce
54:     for (; *text; text++)
55:         glutBitmapCharacter(GLUT_BITMAP_9_BY_15, *text);// vykreslení jednoho znaku
56: };
57: extern int lastTime,fps;
58: extern long maxon_phi_Cz1;
59:
60: /*
61: *----------------------------------------------------------------------------
62: *****************Funkce pro výpis informativního textu******************
63: *----------------------------------------------------------------------------
64: */
65: void drawInfoText(void)
66: {
67:     char str[100];
68:
69:     glColor3f(0.0,0.0,0.0);
70:     if (zobr == 0 || zobr == 1) {
71:
72:
73:     sprintf(str, "Operation:   %s", nazvy[operation]);
74:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-20, str);
75:                                                                          if(r < 2000) {
76:                                                                          if( r > -500.0){
77:                                                                            sprintf(str, "Polomer:
    %3.2f", r);
78:
    printString(glutGet(GLUT_WINDOW_WIDTH)-250, glutGet(GLUT_WINDOW_HEIGHT)-20, str);}
79:                                                                          if( r < -500.0 ){
80:                                                                            sprintf(str, "Polomer:
    daleko");
81:
    printString(glutGet(GLUT_WINDOW_WIDTH)-250, glutGet(GLUT_WINDOW_HEIGHT)-20, str);}}
82:
83:
                if(power == 1){
```

```
84:
                glColor3f(0.0,1.0,0.0);
85:
                     sprintf(str,  "%s", en[power]);
86:
                     printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-20, str);
87:
88:
                    }
89:
90:
                   if(power == 0 && frames % 4 == 0){
91:
                     glColor3f(1.0,0.0,0.0);
92:
                     sprintf(str,  "%s", en[power]);
93:
                     printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-20, str);
94:
                   }
95:
                   glColor3f(0.0,0.0,0.0);
96:
97:
                   sprintf(str, "fps: %d",fps);
98:
                   printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-35, str);
99:
100:   sprintf(str, "Rejd_1_done     : %s",bo[rejd_1_done]);
101:       printString(0, glutGet(GLUT_WINDOW_HEIGHT)-50, str);
102:   sprintf(str, "Vyro_1_done     : %s",bo[vyro_1_done]);
103:       printString(0, glutGet(GLUT_WINDOW_HEIGHT)-65, str);
104:   sprintf(str, "Pivo_1_done     : %s",bo[pivo_1_done]);
105:       printString(0, glutGet(GLUT_WINDOW_HEIGHT)-80, str);
106:
107:
108:   sprintf(str,  "maxon_phi_rejd:    %3.2f",
    (float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1));
109:   printString(0, glutGet(GLUT_WINDOW_HEIGHT)-110, str);
110:   sprintf(str,  "maxon_phi_vyro:    %3.2f",
    (float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1));
111:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-125, str);
112:   sprintf(str,  "maxon_phi_pivo:    %3.2f",
    (float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz1));
113:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-140, str);
114:
115:   sprintf(str,  "PC_   phi_pivo:    %3.2f", phi_Cz1);
116:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-155, str);
117:
118:
119:   sprintf(str,  "Zuzeni:  faze = %d",faze);
120:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-185, str);
121:   sprintf(str,  "         bool = %s",bo[start_zuzeni]);
122:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-200, str);
123:
124:   sprintf(str,  "Krok   :  faze = %d",faze_krok);
125:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-230, str);
126:   sprintf(str,  "         bool = %s",bo[start_krok]);
127:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-245, str);
128:
129:   sprintf(str,  "Jizda :  ");
130:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-275, str);
131:   sprintf(str,  "         bool = %s",bo[start_jizda]);
132:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-290, str);
133:
134:
135:   sprintf(str,  "%f",
    fabs((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1) -
    pocetek_rejd_krok));
136:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-305, str);
137:
138: }
139:  if (zobr == 3 ||zobr == 4)    {
140:      sprintf(str, "fps: %d",fps);
141:      printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-35, str);}
142:
143: }
144:
```

```
1: /*
2: *-----------------------------------------------------------------------
3: *****************Načtení příslušných knihoven*******************
4: *-----------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "Definitions.h"
12: #define Pi 3.14159265
13: #include "extern.h"
14:
15:
16:
17: int faze_dis = 0;
18:
19: void disabling(void){
20:
21:    if( start_zuzeni) {
22:    if(faze == 4) faze = 12;
23:    if(faze == 8) {faze = 9;}
24:    operation = ZUZENI;
25:    start_disabling = false;
26:  }else if(faze_dis == 0) faze_dis = 1;
27:
28:    if( start_jizda)
29:          start_jizda = false;
30:
31: /*
32: *-----------------------------------------------------------------------
33: *******Nastavení všech pohonů na Disable************
34: *-----------------------------------------------------------------------
35: */
36:
37:    if(faze_dis == 4 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
38:             VCS_SetDisableState(COM1,rejd_1,&pErrorCode);
39:             VCS_SetDisableState(COM1,vyro_1,&pErrorCode);
40:             VCS_SetDisableState(COM1,pivo_1,&pErrorCode);
41:             VCS_SetDisableState(COM1,rota_1,&pErrorCode);
42:
43:             VCS_SetDisableState(COM2,rejd_2,&pErrorCode);
44:             VCS_SetDisableState(COM2,vyro_2,&pErrorCode);
45:             VCS_SetDisableState(COM2,pivo_2,&pErrorCode);
46:             VCS_SetDisableState(COM2,rota_2,&pErrorCode);
47:
48:             VCS_SetDisableState(COM3,rejd_3,&pErrorCode);
49:             VCS_SetDisableState(COM3,vyro_3,&pErrorCode);
50:             VCS_SetDisableState(COM3,pivo_3,&pErrorCode);
51:             VCS_SetDisableState(COM3,rota_3,&pErrorCode);
52:
53:             VCS_SetDisableState(COM4,rejd_4,&pErrorCode);
54:             VCS_SetDisableState(COM4,vyro_4,&pErrorCode);
55:             VCS_SetDisableState(COM4,pivo_4,&pErrorCode);
56:             VCS_SetDisableState(COM4,rota_4,&pErrorCode);
57:             power=0;
58:             exit(0);
59:    }
60: /*
61: *-----------------------------------------------------------------------
62: *******Nastavení hodnoty vyrovnávaní a pivotace na nula************
63: *-----------------------------------------------------------------------
64: */
65: if    (faze_dis == 3 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
66:           VCS_MoveToPosition(COM1,vyro_1, 0,Absolute,Immediately,&pErrorCode);
67:           VCS_MoveToPosition(COM1,pivo_1, 0,Absolute,Immediately,&pErrorCode);
68:
69:           VCS_MoveToPosition(COM2,vyro_2, 0,Absolute,Immediately,&pErrorCode);
70:           VCS_MoveToPosition(COM2,pivo_2, 0,Absolute,Immediately,&pErrorCode);
71:
72:           VCS_MoveToPosition(COM3,vyro_3, 0,Absolute,Immediately,&pErrorCode);
73:           VCS_MoveToPosition(COM3,pivo_3, 0,Absolute,Immediately,&pErrorCode);
74:
75:           VCS_MoveToPosition(COM4,vyro_4, 0,Absolute,Immediately,&pErrorCode);
76:           VCS_MoveToPosition(COM4,pivo_4, 0,Absolute,Immediately,&pErrorCode);
77:           faze_dis++;
78:     }
79: /*
80: *-----------------------------------------------------------------------
81: *******Nastavení hodnoty rejdu na nula************
82: *-----------------------------------------------------------------------
83: */
84:     if    (faze_dis == 2 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done &&
    rejd_3_done && pivo_4_done && rejd_4_done){
85:           VCS_MoveToPosition(COM1,rejd_1, 0,Absolute,Immediately,&pErrorCode);
86:
```

```
87:           VCS_MoveToPosition(COM2,rejd_2, 0,Absolute,Immediately,&pErrorCode);
88:
89:           VCS_MoveToPosition(COM3,rejd_3, 0,Absolute,Immediately,&pErrorCode);
90:
91:           VCS_MoveToPosition(COM4,rejd_4, 0,Absolute,Immediately,&pErrorCode);
92:           faze_dis++;
93:       }
94: /*
95: *-----------------------------------------------------------------------
96: *******Nastavení hodnoty pivotace a nastavení pohonu na Disable************
97: *-----------------------------------------------------------------------
98: */
99:       if     (faze_dis == 1 && pivo_1_done  && pivo_2_done &&  pivo_3_done && pivo_4_done ){
100:          VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
101:           VCS_SetDisableState(COM1,rota_1,&pErrorCode);
102:
103:           VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
104:           VCS_SetDisableState(COM2,rota_2,&pErrorCode);
105:
106:           VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
107:           VCS_SetDisableState(COM3,rota_3,&pErrorCode);
108:
109:            VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
110:           VCS_SetDisableState(COM4,rota_4,&pErrorCode);
111:          faze_dis++;
112:
113:      }
114:      }
115:
```

```
1: /*****************************************************************************************
2:                                 bool
3: *****************************************************************************************/
4: extern bool start_pokus;
5: extern bool start_zuzeni;
6: extern bool start_krok;
7: extern bool start_jizda;
8: extern bool start_disabling;
9: extern bool start_enabling;
10: extern bool zapis;
11:
12: /*****************************************************************************************
13:                                 enum
14: *****************************************************************************************/
15: extern enum myenum{
16:    ROTATE,
17:     TRANSLATE,
18:     REJD,JIZDA,ZUZENI,KROK,DISABLING,ENABLING,POKUS,
19:     VYROVNAVANI,POSUN,//PIVOTACE
20: } operation;
21:
22: extern enum myenum operation;
23:
24: /*****************************************************************************************
25:                                 funkce pro kresleni
26: *****************************************************************************************/
27:
28: void kvadr_ss_v_T(float a, float b, float c);
29: void kolo();
30: void sou_sys();
31: void drawInfoText(void);
32: void printString(int x, int y, char *text);
33: void setPerspectiveProjection(void);
34: void setOrthogonalProjection(void);
35: void kvadr_pos(float a, float b, float c);
36: void te_pivotace();
37:
38:
39:
40: /*****************************************************************************************
41:                                 funkce pro OpenGL
42: *****************************************************************************************/
43:
44: void joystick(unsigned int joystick, int x, int y, int z);
45: void keyboard(unsigned char keyboard, int x, int y);
46: void onMouseButton(int button, int state, int x, int y);
47: void onMouseMotion(int x, int y);
48: void keyboard_special(int keyboard_special, int x, int y);
49:
50: /*****************************************************************************************
51:                                 vlastni funkce
52: *****************************************************************************************/
53:
54: void vytvorMatice();
55: void nastaveni();
56: void zuzeni();
57: void disabling(void);
58: void enabling(void);
59: void pokus(void);
60: void krok();
61: void jizda(void);
62: float sign(float i);
63: int fce_uhel(int x);
64:
65: /*****************************************************************************************
66:                                 float
67: *****************************************************************************************/
68: extern float MaticeGlobal[16],MaticeTglobal[16], MaticePodvozek[16],
69:       M_U1[16],M_A1[16],M_B1[16],M_C1[16], M_D1[16],
70:       M_U2[16],M_A2[16],M_B2[16],M_C2[16], M_D2[16],
71:       M_U3[16],M_A3[16],M_B3[16],M_C3[16], M_D3[16],
72:       M_U4[16],M_A4[16],M_B4[16],M_C4[16], M_D4[16];
73: extern float xo;
74: extern float yo;
75: extern float zo;
76: extern float phi_Lz,r, phi_Az1, phi_By1,pom_phi_Cz1,phi_Cz1,phi_Az2, phi_By2, phi_Cz2,phi_Cz3,phi_Az3, phi_By3,
    pom_phi_Cz3,phi_Az4, phi_By4, phi_Cz4,pom_phi_Cz4;
77: extern float xnew_T,ynew_T;
78: extern float p_phi_Cz1,p_phi_Cz2,p_phi_Cz3,p_phi_Cz4;
79:
80: extern float r,uhel,vyska;
81: extern float pomoc;
82:
83:
84:
85:
86: /*****************************************************************************************
87:                                 int
88: *****************************************************************************************/
```

```
 89: extern int faze,faze_krok;
 90: extern int   xnew, ynew, znew;
 91: extern int   xold, yold, zold;
 92: extern int   xx, yy, zz,xova;
 93: extern int   windowWidth,windowHeight;
 94: extern int yy,pocet,power,frames;
 95: extern int pred_manevr;
 96: extern int cas,start_cas;
 97:
 98: extern int zobr;
 99: extern int pocetek_rejd_krok;
100: extern int rejd_zakl_pozice,rejd_faze_zuzeni,vyro_zakl_pozice,faze_dis;
101: extern int i_plan_rejd;
102: extern int i_snek_rejd;
103: extern int rozl_enco_rejd;
104:
105: extern int i_plan_vyro;
106: extern int i_snek_vyro;
107: extern int rozl_enco_vyro;
108:
109: extern int i_plan_pivo;
110: extern int i_snek_pivo;
111: extern int rozl_enco_pivo;
112:
113:
114: /*******************************************************************************************
115:                                 long
116: *******************************************************************************************/
117: extern int faze;
118: extern int   xnew, ynew, znew;
119: extern int   xold, yold, zold;
120: extern int   xx, yy, zz;
121: extern long maxon_phi_Cz1,maxon_phi_By1,maxon_phi_Az1,maxon_phi_Cz2,maxon_phi_By2,maxon_phi_Az2,maxon_phi_Cz3,
     maxon_phi_By3,maxon_phi_Az3,maxon_phi_Cz4,maxon_phi_By4,maxon_phi_Az4;
122:
123: /*******************************************************************************************
124:                                 promenne pro MAXON
125: *******************************************************************************************/
126: extern HANDLE m_KeyHandle;
127:
128:
129: extern     HANDLE COM1,COM2,COM3,COM4;
130: extern     __int8 Mode;
131: extern  DWORD pErrorCode;
132: extern BOOL Absolute ;
133: extern BOOL Immediately;
134: extern BOOL rejd_1_done ,pivo_1_done,vyro_1_done,rejd_2_done ,pivo_2_done,vyro_2_done,rejd_3_done ,pivo_3_done,
     vyro_3_done,rejd_4_done ,pivo_4_done,vyro_4_done ;
135: extern WORD vyro_1,rejd_1,pivo_1,rota_1,vyro_2,rejd_2,pivo_2,rota_2,vyro_3,rejd_3,pivo_3,rota_3,vyro_4,rejd_4,
     pivo_4,rota_4;
136:
137:
138:
139:
140: /*******************************************************************************************
141:                                 FILE
142: *******************************************************************************************/
143:
144: extern FILE *fw;
145:
```

```
/*
*--------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*--------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#include "extern.h"

/*
*--------------------------------------------------------------------------
*****************Definice seznamu operací a proměnných******************
*--------------------------------------------------------------------------
*/
enum myenum operation;

bool start_pokus   = false;
bool start_zuzeni   = false;
bool start_krok     = false;
bool start_jizda    = false;
bool start_disabling = false;
bool start_ensabling = false;
bool zapis          = false;

int i_plan_rejd = 21;
int i_snek_rejd = 22;
int rozl_enco_rejd = 4*500;

int i_plan_vyro = 86;
int i_snek_vyro = 20;
int rozl_enco_vyro = 4*500;

int i_plan_pivo = 29;
int i_snek_pivo = 22;
int rozl_enco_pivo = 4*512;

int zobr = 0;

int rejd_zakl_pozice = 20;
int rejd_faze_zuzeni = 30;
int vyro_zakl_pozice = 20;

WORD rejd_1 = 1;
WORD vyro_1 = 2;
WORD pivo_1 = 3;
WORD rota_1 = 4;

WORD rejd_2 = 5;
WORD vyro_2 = 6;
WORD pivo_2 = 7;
WORD rota_2 = 8;

WORD rejd_3 = 9;
WORD vyro_3 = 10;
WORD pivo_3 = 11;
WORD rota_3 = 12;

WORD rejd_4 = 13;
WORD vyro_4 = 14;
WORD pivo_4 = 15;
WORD rota_4 = 16;

BOOL vyro_1_done ,pivo_1_done ,rejd_1_done;
BOOL vyro_2_done ,pivo_2_done ,rejd_2_done;
BOOL vyro_3_done ,pivo_3_done ,rejd_3_done;
BOOL vyro_4_done ,pivo_4_done ,rejd_4_done;

int   windowWidth;
int   windowHeight;
/*
*--------------------------------------------------------------------------
***Funkce pro inicializaci okna a pro ošetření změny velikosti okna*******
*--------------------------------------------------------------------------
*/
void onResize(int w, int h) {
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     glViewport(0, 0, w, h);
     gluPerspective(45,(double)w/(double)h,1,100);
     glMatrixMode(GL_MODELVIEW);
     windowWidth=w;
     windowHeight=h;
}


```

```
 90: void init() {
 91:     glPointSize(3.0f);                          // body zobrazovat s rûmûrem 3 pixely
 92:     glEnable(GL_POINT_SMOOTH);                  // povolení antialiasingu bodu
 93:     glLineWidth(0.5f);                          // úsečky zobrazovat široké 1 pixel
 94:     glEnable(GL_LINE_SMOOTH);                   // povolení antialiasingu úseček
 95:     glDisable(GL_CULL_FACE);
 96:      glEnable(GL_DEPTH_TEST);
 97:      glEnable(GL_COLOR_MATERIAL);
 98:      glClearColor(0.0,0.0,0.0,0.0);
 99:      glEnable(GL_LIGHTING);
100:      glEnable(GL_LIGHT0);
101: }
102:
103: int fps,timeSinceStart,frames,lastTime;
104: /*
105: *-----------------------------------------------------------------------------
106: *******Funkce pro měření počtu snímků za vteřinu**********
107: *-----------------------------------------------------------------------------
108: */
109: void timer() {
110:      frames++;
111:      timeSinceStart=glutGet(GLUT_ELAPSED_TIME);  // čas od startu aplikace
112:      if ((timeSinceStart-lastTime)>1000) { // pokud uplynula vteřina od předchozího zápisu
113:         lastTime=timeSinceStart;
114:         fps=frames; // zapíše se, kolik proběhlo snímků za poslední vteřinu
115:         frames=0;
116:      }
117: }
118:
119:
120: float uhel;
121:
122: /*
123: *-----------------------------------------------------------------------------
124: *******Hlavní zobrazovací funkce**********
125: *-----------------------------------------------------------------------------
126: */
127: void onDisplay(void){
128:
129:
130:     timer();
131:     glClear( GL_DEPTH_BUFFER_BIT);
132:     glClearColor(1.0, 1.0, 1.0, 0.0);
133:     glClear(GL_COLOR_BUFFER_BIT);               // vymazání bitovych rovin barvového bufferu
134:
135:     setOrthogonalProjection();
136:     drawInfoText();                             // vypsat textové informace - viz text.cpp
137:     setPerspectiveProjection();
138:
139: /*
140: *-----------------------------------------------------------------------------
141: ****Funkce pro výpočet natočení kol a pro generaci tranfor. matic*******
142: *-----------------------------------------------------------------------------
143: */
144: nastaveni();
145: vytvorMatice();
146: /*
147: *-----------------------------------------------------------------------------
148: ************Vykreslení středu zatáčky a základny podvozku**************
149: *-----------------------------------------------------------------------------
150: */
151: if (zobr == 0 || zobr == 2 || zobr == 3) {
152:
153:   glLoadIdentity();
154:   glPushMatrix();
155:
156:     glMultMatrixf(MaticeGlobal);
157:     glColor3d(0.0,0.0,1.0);
158:      sou_sys();
159:      glTranslatef(0.0,-r,0.0);
160:      glColor3d(0.0,0.0,0.0);
161:      glutSolidSphere(0.1,  50, 50);
162:
163:      glPopMatrix();
164:
165:
166:
167:     glPushMatrix();
168:     glMultMatrixf(MaticePodvozek);
169:     glColor3d(1.0,0.0,0.0);
170:      sou_sys();
171:     kvadr_ss_v_T(2.0,1.0,0.05);
172:     glPopMatrix();
173:  /*
174: *-----------------------------------------------------------------------------
175: ****************První noha******************
176: *-----------------------------------------------------------------------------
177: */
178:     glPushMatrix();
```

```
179:     glMultMatrixf(M_U1);
180:     glColor3d(0.0,0.0,0.0);
181:     glutSolidSphere(0.07,20,20);
182:     sou_sys();
183:     glPopMatrix();
184:
185:     glPushMatrix();
186:     glMultMatrixf(M_A1);
187:     glColor3d(0.0,1.0,0.0);
188:     sou_sys();
189:     kvadr_pos(0.45, 0.05, 0.05);
190:     glPopMatrix();
191:
192:     glPushMatrix();
193:     glMultMatrixf(M_B1);
194:     glColor3d(0.0,1.0,1.0);
195:     glutSolidSphere(0.05,20,20);
196:     sou_sys();
197:     kvadr_pos(1.8, 0.05, 0.05);
198:     glPopMatrix();
199:
200:     glPushMatrix();
201:     glMultMatrixf(M_C1);
202:     glColor3d(1.0,0.0,1.0);
203:     sou_sys();
204:     glutSolidSphere(0.05,20,20);
205:     te_pivotace();
206:     glPopMatrix();
207:
208:     glPushMatrix();
209:     glMultMatrixf(M_D1);
210:
211:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
212:
213:     kolo();
214:     glPopMatrix();
215: /*
216: *---------------------------------------------------------------------------
217: ****************Druhá noha******************
218: *---------------------------------------------------------------------------
219: */
220:     glPushMatrix();
221:     glMultMatrixf(M_U2);
222:     glColor3d(0.0,0.0,0.0);
223:     glutSolidSphere(0.07,20,20);
224:     sou_sys();
225:     glPopMatrix();
226:
227:     glPushMatrix();
228:     glMultMatrixf(M_A2);
229:     glColor3d(0.0,1.0,0.0);
230:     sou_sys();
231:     kvadr_pos(0.45, 0.05, 0.05);
232:     glPopMatrix();
233:
234:     glPushMatrix();
235:     glMultMatrixf(M_B2);
236:     glColor3d(0.0,1.0,1.0);
237:     glutSolidSphere(0.05,20,20);
238:     sou_sys();
239:     kvadr_pos(1.8, 0.05, 0.05);
240:     glPopMatrix();
241:
242:     glPushMatrix();
243:     glMultMatrixf(M_C2);
244:     glColor3d(1.0,0.0,1.0);
245:     sou_sys();
246:     glutSolidSphere(0.05,20,20);
247:     te_pivotace();
248:     glPopMatrix();
249:
250:     glPushMatrix();
251:     glMultMatrixf(M_D2);
252:
253:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
254:
255:     kolo();
256:     glPopMatrix();
257:
258: /*
259: *---------------------------------------------------------------------------
260: ****************Třetí noha******************
261: *---------------------------------------------------------------------------
262: */
263:     glPushMatrix();
264:     glMultMatrixf(M_U3);
265:     glColor3d(0.0,0.0,0.0);
266:     glutSolidSphere(0.07,20,20);
267:     sou_sys();
```

```
268:     glPopMatrix();
269:
270:     glPushMatrix();
271:     glMultMatrixf(M_A3);
272:     glColor3d(0.0,1.0,0.0);
273:     sou_sys();
274:     kvadr_pos(0.45, 0.05, 0.05);
275:     glPopMatrix();
276:
277:     glPushMatrix();
278:     glMultMatrixf(M_B3);
279:     glColor3d(0.0,1.0,1.0);
280:     glutSolidSphere(0.05,20,20);
281:     sou_sys();
282:     kvadr_pos(1.8, 0.05, 0.05);
283:     glPopMatrix();
284:
285:     glPushMatrix();
286:     glMultMatrixf(M_C3);
287:     glColor3d(1.0,0.0,1.0);
288:     sou_sys();
289:     glutSolidSphere(0.05,20,20);
290:     te_pivotace();
291:     glPopMatrix();
292:
293:     glPushMatrix();
294:     glMultMatrixf(M_D3);
295:
296:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
297:
298:     kolo();
299:     glPopMatrix();
300:
301: /*
302: *--------------------------------------------------------------------------
303: ****************Čtvrtá noha******************
304: *---------------------------------_----------------------------------------
305: */
306:     glPushMatrix();
307:     glMultMatrixf(M_U4);
308:     glColor3d(0.0,0.0,0.0);
309:     glutSolidSphere(0.07,20,20);
310:     sou_sys();
311:     glPopMatrix();
312:
313:     glPushMatrix();
314:     glMultMatrixf(M_A4);
315:     glColor3d(0.0,1.0,0.0);
316:     sou_sys();
317:     kvadr_pos(0.45, 0.05, 0.05);
318:     glPopMatrix();
319:
320:     glPushMatrix();
321:     glMultMatrixf(M_B4);
322:     glColor3d(0.0,1.0,1.0);
323:     glutSolidSphere(0.05,20,20);
324:     sou_sys();
325:     kvadr_pos(1.8, 0.05, 0.05);
326:     glPopMatrix();
327:
328:     glPushMatrix();
329:     glMultMatrixf(M_C4);
330:     glColor3d(1.0,0.0,1.0);
331:     sou_sys();
332:     glutSolidSphere(0.05,20,20);
333:     te_pivotace();
334:     glPopMatrix();
335:
336:     glPushMatrix();
337:     glMultMatrixf(M_D4);
338:
339:     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);sou_sys();
340:
341:     kolo();
342:     glPopMatrix();
343:
344: }
345: /*
346: *--------------------------------------------------------------------------
347: ****************Zjištění informací o dokončení pohybů******************
348: *---------------------------------_----------------------------------------
349: */
350:     VCS_GetMovementState(COM1, pivo_1,&pivo_1_done,&pErrorCode);
351:     VCS_GetMovementState(COM1, vyro_1,&vyro_1_done,&pErrorCode);
352:     VCS_GetMovementState(COM1, rejd_1,&rejd_1_done,&pErrorCode);
353:
354:     VCS_GetMovementState(COM2, pivo_2,&pivo_2_done,&pErrorCode);
355:     VCS_GetMovementState(COM2, vyro_2,&vyro_2_done,&pErrorCode);
356:     VCS_GetMovementState(COM2, rejd_2,&rejd_2_done,&pErrorCode);
```

```
357:
358:     VCS_GetMovementState(COM3, pivo_3,&pivo_3_done,&pErrorCode);
359:     VCS_GetMovementState(COM3, vyro_3,&vyro_3_done,&pErrorCode);
360:     VCS_GetMovementState(COM3, rejd_3,&rejd_3_done,&pErrorCode);
361:
362:     VCS_GetMovementState(COM4, pivo_4,&pivo_4_done,&pErrorCode);
363:     VCS_GetMovementState(COM4, vyro_4,&vyro_4_done,&pErrorCode);
364:     VCS_GetMovementState(COM4, rejd_4,&rejd_4_done,&pErrorCode);
365: /*
366: *-------------------------------------------------------------------------
367: ****************Rozdělení jednotlivých operací******************
368: *-------------------------------------------------------------------------
369: */
370:
371:     if(start_enabling)
372:      enabling();
373:
374:     if(start_disabling)
375:      disabling();
376:
377:     if(start_jizda)
378:      jizda();
379:
380:     if(start_zuzeni)
381:      zuzeni();
382:
383:     if(start_krok)
384:      krok();
385:
386:     if(zapis)
387:      fprintf(fw,"%d\t\t%d\t%f\t%d\n",glutGet(GLUT_ELAPSED_TIME),cas,
    (float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1),fps);
388:      glutSwapBuffers();
389: }
390: /*
391: *-------------------------------------------------------------------------
392: ****************Funkce main******************
393: *-------------------------------------------------------------------------
394: */
395: int main (int argc, char **argv) {
396:
397:     if(glutDeviceGet(GLUT_HAS_JOYSTICK) != 0) {
398:
399:         return 0;
400:     }
401:
402:         glutInit(&argc,argv);
403:     glutInitDisplayMode(GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE);
404:     glutInitWindowSize(640,480);
405:
406:     glutInitWindowPosition(500,200);
407:
408:     glutCreateWindow(";-)");
409:
410:
411:     glEnable(GL_COLOR_MATERIAL);
412:     glutDisplayFunc(init);
413:     glutKeyboardFunc(keyboard);           //registrace funkce pro stisk klávesnice
414:     glutSpecialFunc(keyboard_special);    //registrace funkce pro stisk speciálních kláves
415:     glutJoystickFunc(joystick, 10);       //registrace funkce pro práci s joystickem
416:     glutIdleFunc(onDisplay);
417:     glutMouseFunc(onMouseButton);         //registrace funkce pro stisk tlačítka myši
418:     glutMotionFunc(onMouseMotion);        //registrace funkce pro pohyb myši
419:     glutReshapeFunc(onResize);
420:     init();
421:     glutMainLoop();
422:
423:     return 0;
424: }
425:
```

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#include "extern.h"
#define Pi 3.14159265
/*
*-----------------------------------------------------------------------------
*****************Definice proměnných******************
*-----------------------------------------------------------------------------
*/
float sign(float i);
float xo;
float yo;
float zo;
float r;
int xova;

BOOL Absolute = TRUE;
BOOL Immediately = TRUE ;
/*
*-----------------------------------------------------------------------------
*************Definice proměnných k identifikaci portu****************
*-----------------------------------------------------------------------------
*/
char* COM_1 = "COM1";
char* COM_2 = "COM1";
char* COM_3 = "COM1";
char* COM_4 = "COM1";

HANDLE COM1,COM2,COM3,COM4;
int power=0;
int faze = 0;
/*
*-----------------------------------------------------------------------------
*************Ošetření pohybu joystickem****************
*-----------------------------------------------------------------------------
*/
void joystick(unsigned int joystick, int x, int y, int z)
{
    xova = x;
    r = (float)  -2.0f*tan(((float) x)/1000.0f*Pi/2.0f-Pi/2.0f);
    xo = y/100.0f ;
    zo = -5+z/500.0f ;
/*
*-----------------------------------------------------------------------------
*************Ošetření stisku tlačítek joysticku****************
*-----------------------------------------------------------------------------
*/
switch (joystick)
    {
        case GLUT_JOYSTICK_BUTTON_A:
/*
*-----------------------------------------------------------------------------
*************Navázání spojení s řídicími jednotkami****************
*-----------------------------------------------------------------------------
*/
              COM1 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_1,&pErrorCode);
//            COM2 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_2,&pErrorCode);
//            COM3 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_3,&pErrorCode);
//            COM4 = VCS_OpenDevice("EPOS","MAXON_RS232","RS232",COM_4,&pErrorCode);
              VCS_SetProtocolStackSettings(COM1,115200,500,&pErrorCode);
//            VCS_SetProtocolStackSettings(COM2,115200,500,&pErrorCode);
//            VCS_SetProtocolStackSettings(COM3,115200,500,&pErrorCode);
//            VCS_SetProtocolStackSettings(COM4,115200,500,&pErrorCode);


            COM2 = COM1;
            COM3 = COM1;
            COM4 = COM1;
/*
*-----------------------------------------------------------------------------
*************Vymazání předchozích chybových hlášení****************
*-----------------------------------------------------------------------------
*/
            VCS_ClearFault(COM1,rejd_1,&pErrorCode);
            VCS_ClearFault(COM1,vyro_1,&pErrorCode);
            VCS_ClearFault(COM1,pivo_1,&pErrorCode);
            VCS_ClearFault(COM1,rota_1,&pErrorCode);

            VCS_ClearFault(COM2,rejd_2,&pErrorCode);
            VCS_ClearFault(COM2,vyro_2,&pErrorCode);
```

```
            VCS_ClearFault(COM2,pivo_2,&pErrorCode);
            VCS_ClearFault(COM2,rota_2,&pErrorCode);

            VCS_ClearFault(COM3,rejd_3,&pErrorCode);
            VCS_ClearFault(COM3,vyro_3,&pErrorCode);
            VCS_ClearFault(COM3,pivo_3,&pErrorCode);
            VCS_ClearFault(COM3,rota_3,&pErrorCode);

            VCS_ClearFault(COM4,rejd_4,&pErrorCode);
            VCS_ClearFault(COM4,vyro_4,&pErrorCode);
            VCS_ClearFault(COM4,pivo_4,&pErrorCode);
            VCS_ClearFault(COM4,rota_4,&pErrorCode);
/*
*-----------------------------------------------------------------------
************Nastavení operačních módů***************
*-----------------------------------------------------------------------
*/
            VCS_SetOperationMode(COM1,rejd_1,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM1,vyro_1,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM1,pivo_1,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM1,rota_1,-0x02/*Velocity Mode       */,&pErrorCode);

            VCS_SetOperationMode(COM2,rejd_2,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM2,vyro_2,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM2,pivo_2,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM2,rota_2,-0x02/*Velocity Mode       */,&pErrorCode);

            VCS_SetOperationMode(COM3,rejd_3,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM3,vyro_3,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM3,pivo_3,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM3,rota_3,-0x02/*Velocity Mode       */,&pErrorCode);

            VCS_SetOperationMode(COM4,rejd_4,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM4,vyro_4,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM4,pivo_4,0x01/*Profile Position Mode*/,&pErrorCode);
            VCS_SetOperationMode(COM4,rota_4,-0x02/*Velocity Mode       */,&pErrorCode);
/*
*-----------------------------------------------------------------------
************Nastavení parametrů pohybu***************
*-----------------------------------------------------------------------
*/
            VCS_SetPositionProfile(COM1,rejd_1,8700,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM1,vyro_1,15000,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM1,pivo_1,12000,20000,20000,&pErrorCode);

            VCS_SetPositionProfile(COM2,rejd_2,8700,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM2,vyro_2,15000,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM2,pivo_2,12000,20000,20000,&pErrorCode);

            VCS_SetPositionProfile(COM3,rejd_3,8700,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM3,vyro_3,15000,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM3,pivo_3,12000,20000,20000,&pErrorCode);

            VCS_SetPositionProfile(COM4,rejd_4,8700,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM4,vyro_4,15000,20000,20000,&pErrorCode);
            VCS_SetPositionProfile(COM4,pivo_4,12000,20000,20000,&pErrorCode);


            power=1;
            if (!start_krok && !start_jizda && !start_zuzeni){
            operation=ENABLING;
            start_enabling = true;
            pred_manevr=5;
            } else {
                  if( start_zuzeni) {
                     if(faze == 4) faze = 5;;
                     if(faze == 8) {faze = 9;}
                     operation = ZUZENI;
                     start_jizda = false;
                     pred_manevr=3;}
                  else{
                   if( start_krok) {
                     if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
                     operation = KROK;
                     start_jizda = false;
                     pred_manevr=3;}
                   else{
                     //operation = JIZDA;
                    // start_jizda = true;
                    // pred_manevr=3;
                     }
                }
              }

            break;
        case GLUT_JOYSTICK_BUTTON_B:
             if( start_krok) {
                     if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
                     operation = KROK;
```

```
179:                        start_jizda = false;
180:                        pred_manevr=4;}
181:                 else{
182:                operation=DISABLING;
183:                start_disabling = true;
184:                pred_manevr=4;
185:                }
186:
187:                //VCS_CloseAllDevices(&pErrorCode);
188:
189:                break;
190:
191:            case GLUT_JOYSTICK_BUTTON_C:
192:                 if( start_krok) {
193:                         if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
194:                         operation = KROK;
195:                         start_jizda = false;
196:                         pred_manevr=1;}
197:                 else {
198:                  operation=ZUZENI;
199:                  start_zuzeni = true;
200:                  if(faze == 4){ faze = 5;}
201:                   if(faze == 8) {faze = 9;}
202:                  pred_manevr=1;
203:                  }
204:                break;
205:
206:            case GLUT_JOYSTICK_BUTTON_D:
207:                 if( start_zuzeni) {
208:                   if(faze == 4){ faze = 5;}
209:                   if(faze == 8) {faze = 9;}
210:                   operation = ZUZENI;
211:                   start_krok = false;
212:                   pred_manevr=2;}
213:                 else{
214:                   operation=KROK;
215:                   start_krok = true;
216:                   pred_manevr=2;
217:                   if(faze_krok == 7  || faze_krok == 6  || faze_krok == 5 || faze_krok == 4) faze_krok = 8;
218:                   }
219:
220:                break;
221:
222:
223:
224:        }
225:
226:        glutPostRedisplay();
227: }
228:
```

```
/*
*---------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*---------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#include "extern.h"

/*
*---------------------------------------------------------------------------
*****************Informace o předchozím manévru******************
*---------------------------------------------------------------------------
*/
int pred_manevr=0;                /* 0   -   nic
                                     1   -   zuzeni
                                     2   -   krok
                                     3   -   jizda
                                     4   -   disabling
                                     5   -   enabling    */


HANDLE m_KeyHandle;

    HANDLE KeyHandle;
    __int8 Mode;
    DWORD pErrorCode;
    DWORD pBaudrate;
    DWORD pTimeout;
    DWORD NodeId;


long int StartPosition;

/*
*---------------------------------------------------------------------------
*****************Ošetření stisku klávesy******************
*---------------------------------------------------------------------------
*/

void keyboard(unsigned char keyboard, int x, int y)
{
    switch (keyboard)
    {

          case 'n':
             faze_krok++;
             break;
          case 'l':
             operation=KROK;
             start_krok = true;
             pred_manevr=2;
             break;

          case 'j':
             operation=JIZDA;
             start_jizda = true;
             pred_manevr=3;
             break;

          case 'k':
             operation=ZUZENI;
             start_zuzeni = true;
             if(faze == 4) faze++;
             pred_manevr=1;
             break;


/*
*---------------------------------------------------------------------------
*****************Možnost kontrolního zápisu do souboru******************
*---------------------------------------------------------------------------
*/
        case 'o':
            if (zapis){ zapis = false;
            fclose(fw);}
            else {zapis = true;
                  fw = fopen("rozdily.txt","w");};
            break;

        case 'r':
            operation=ROTATE;
            break;

        case 'y':
```

```
            operation=POSUN;
            break;
/*
*--------------------------------------------------------------------------
*****************Změna způsobu zobrazení******************
*--------------------------------------------------------------------------
*/
        case 'c':
            if (zobr == 5) zobr = 0;
            else if (zobr == 4) zobr++;
            else if (zobr == 3) zobr++;
            else if (zobr == 2) zobr++;
            else if (zobr == 1) zobr++;
            else if (zobr == 0) zobr++;

            break;

            default:
                break;

    }

    glutPostRedisplay();
}
/*
*--------------------------------------------------------------------------
****************Ošetření stisku speciální klávesy******************
*--------------------------------------------------------------------------
*/
void keyboard_special(int keyboard_special, int x, int y)
{
    switch (keyboard_special)
    {


        case GLUT_KEY_LEFT:
            xnew_T -= 0.2;
            break;
        case GLUT_KEY_RIGHT  :
            xnew_T += 0.2;
            break;
        case GLUT_KEY_UP  :
            ynew_T += 0.2;
            break;
        case GLUT_KEY_DOWN  :
            ynew_T -= 0.2;
            break;
            default:
                break;
    }

    glutPostRedisplay();
}

```

```
1: /*
2: *--------------------------------------------------------------------------
3: ****************Načtení příslušných knihoven******************
4: *--------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: /* Kvadr o rozmerech a,b,c - SS je v tezisti kvadru
12: *  - a ve smeru osy x
13: *  - b ve smeru osy y
14: *  - c ve smeru osy z
15: */
16:
17: void kvadr_ss_v_T(float a, float b, float c) {
18:
19:      a /= 2;
20:      b /= 2;
21:      c /= 2;
22:
23:
24:      //stěna  kolma na z > 0
25:      glBegin(GL_QUADS);
26:
27:      glNormal3f(0.0,0.0,1.0);
28:      glVertex3f(a,-b,c);
29:      glVertex3f(-a,-b,c);
30:      glVertex3f(-a,b,c);
31:      glVertex3f(a,b,c);
32:      glEnd();
33:
34:
35:      //stěna  kolma na z < 0
36:      glBegin(GL_QUADS);
37:      glNormal3f(0.0,0.0,-1.0);
38:      glVertex3f(a,b,-c);
39:      glVertex3f(-a,b,-c);
40:      glVertex3f(-a,-b,-c);
41:      glVertex3f(a,-b,-c);
42:
43:      glEnd();
44:
45:      //stěna  kolma na y > 0
46:      glBegin(GL_QUADS);
47:      glNormal3f(0.0,1.0,0.0);
48:      glVertex3f(a,b,c);
49:      glVertex3f(-a,b,c);
50:      glVertex3f(-a,b,-c);
51:      glVertex3f(a,b,-c);
52:      glEnd();
53:
54:      //stěna  kolma na y < 0
55:      glBegin(GL_QUADS);
56:      glNormal3f(0.0,-1.0,0.0);
57:      glVertex3f(-a,-b,c);
58:      glVertex3f(a,-b,c);
59:      glVertex3f(a,-b,-c);
60:      glVertex3f(-a,-b,-c);
61:
62:      glEnd();
63:
64:      //stěna  kolma na x < 0
65:      glBegin(GL_QUADS);
66:      glNormal3f(-1.0,0.0,0.0);
67:
68:      glVertex3f(-a,b,c);
69:      glVertex3f(-a,-b,c);
70:      glVertex3f(-a,-b,-c);
71:      glVertex3f(-a,b,-c);
72:      glEnd();
73:
74:      //stěna  kolma na x > 0
75:      glBegin(GL_QUADS);
76:      glNormal3f(1.0,0.0,0.0);
77:      glVertex3f(a,-b,c);
78:      glVertex3f(a,b,c);
79:      glVertex3f(a,b,-c);
80:      glVertex3f(a,-b,-c);
81:      glEnd();
82:
83: }
84:
```

```
/*
*----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
/* Kvadr o rozmerech a,b,c - SS je v tezisti kvadru
*  - a ve smeru osy x
*  - b ve smeru osy y
*  - c ve smeru osy z
*/

void kvadr_pos(float a, float b, float c) {

     a /= 2;
     b /= 2;
     c /= 2;


     //stěna  kolma na z > 0
     glBegin(GL_QUADS);

     glNormal3f(0.0,0.0,1.0);
     glVertex3f(2.0*a,-b,c);
     glVertex3f(0.0,-b,c);
     glVertex3f(0.0,b,c);
     glVertex3f(2.0*a,b,c);
     glEnd();


     //stěna  kolma na z < 0
     glBegin(GL_QUADS);
     glNormal3f(0.0,0.0,-1.0);
     glVertex3f(2.0*a,b,-c);
     glVertex3f(0.0,b,-c);
     glVertex3f(0.0,-b,-c);
     glVertex3f(2.0*a,-b,-c);

     glEnd();

     //stěna  kolma na y > 0
     glBegin(GL_QUADS);
     glNormal3f(0.0,1.0,0.0);
     glVertex3f(2.0*a,b,c);
     glVertex3f(0.0,b,c);
     glVertex3f(0.0,b,-c);
     glVertex3f(2.0*a,b,-c);
     glEnd();

     //stěna  kolma na y < 0
     glBegin(GL_QUADS);
     glNormal3f(0.0,-1.0,0.0);
     glVertex3f(0.0,-b,c);
     glVertex3f(2.0*a,-b,c);
     glVertex3f(2.0*a,-b,-c);
     glVertex3f(0.0,-b,-c);

     glEnd();

     //stěna  kolma na x < 0
     glBegin(GL_QUADS);
     glNormal3f(-1.0,0.0,0.0);

     glVertex3f(0.0,b,c);
     glVertex3f(0.0,-b,c);
     glVertex3f(0.0,-b,-c);
     glVertex3f(0.0,b,-c);
     glEnd();

     //stěna  kolma na x > 0
     glBegin(GL_QUADS);
     glNormal3f(1.0,0.0,0.0);
     glVertex3f(2.0*a,-b,c);
     glVertex3f(2.0*a,b,c);
     glVertex3f(2.0*a,b,-c);
     glVertex3f(2.0*a,-b,-c);
     glEnd();

}

```

```
/*
*----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
void printString3d(int x, int y, int z, char *text)
{
    glRasterPos3i(x, y, z);                        // pozice prvniho znaku retezce
    for (; *text; text++)                         // pruchod retezcem
        glutBitmapCharacter(GLUT_BITMAP_9_BY_15, *text);// vykresleni jednoho znaku
};
/*
*----------------------------------------------------------------------------
*****************Funkce pro vyznačení symbolu pro SS******************
*----------------------------------------------------------------------------
*/

void sou_sys(){
    char popis_x[2],popis_y[2],popis_z[2];
    sprintf(popis_x,  "x");
    sprintf(popis_y,  "y");
    sprintf(popis_z,  "z");
     glBegin(GL_LINES);

     glNormal3f(0.0,0.0,1.0);

     glVertex3f(0.0,0.0,0.0);
     glVertex3f(1.0,0.0,0.0);


     glEnd();
     printString3d(1, 0,0, popis_x);

     glBegin(GL_LINES);
     glNormal3f(0.0,0.0,1.0);
     glVertex3f(0.0,0.0,0.0);
     glVertex3f(0.0,1.0,0.0);
     glEnd();
     printString3d(0, 1,0, popis_y);
     glBegin(GL_LINES);
     glNormal3f(0.0,0.0,1.0);
     glVertex3f(0.0,0.0,0.0);
     glVertex3f(0.0,0.0,1.0);

     glEnd();
     printString3d(0, 0,1, popis_z);
     }

```

```
1: /*
2: *--------------------------------------------------------------------------
3: ****************Načtení příslušných knihoven******************
4: *--------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "extern.h"
12:
13: /*
14: *--------------------------------------------------------------------------
15: ****************Funkce pro vytvoření tělesa pivotace******************
16: *--------------------------------------------------------------------------
17: */
18: void te_pivotace(){
19:     // glPushMatrix();
20:
21:      glRotatef(90.0,.0,1.0,0.0);
22:      glRotatef(-40.0,.0,0.0,1.0);
23:      kvadr_pos(1.059,0.05,0.05);
24:      glTranslatef(1.059,0.0,0.0);
25:      glutSolidCube(0.1);
26:      glRotatef((180.0-72.0),.0,0.0,1.0);
27:      kvadr_pos(0.736,0.05,0.05);
28:     // glPopMatrix();
29:
30:
31:
32:      }
33:
```

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven*******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#define PI 3.14159265
GLfloat mat_black_diffuse[] = { 0, 0.0, 0, 0 };
/*
*-----------------------------------------------------------------------------
*****************Definice rovin pro ořez*******************
*-----------------------------------------------------------------------------
*/
GLdouble rov[4] = {0.0,-1.0,0.0,0.45};
GLdouble rov2[4] = {0.0,1.0,0.0,0.3};
int circle_points = 100;
/*
*-----------------------------------------------------------------------------
*****************Funkce pro kreslení kola*******************
*-----------------------------------------------------------------------------
*/
void kolo(){

     glClipPlane(GL_CLIP_PLANE0,rov);
     glClipPlane(GL_CLIP_PLANE1,rov2);
     glEnable(GL_CLIP_PLANE0);
     glEnable(GL_CLIP_PLANE1);
     glRotatef(90,1,0,0);
     glColor3d(50.0/255.0,249.0/255.0,11.0/255.0);
     glutWireSphere(0.5, 50, 50);
    glColor3d(1.0,1.0,0.0);
    glutSolidSphere(0.4999,  50, 50);

     glDisable(GL_CLIP_PLANE0);
     glDisable(GL_CLIP_PLANE1);

     glBegin(GL_POLYGON);
/*
*-----------------------------------------------------------------------------
*****************Tvorba čelních ploch kol*******************
*-----------------------------------------------------------------------------
*/

     for(int i = 0; i <circle_points; i++){
     float angle = 2 * PI * i / circle_points;
            float vel = sqrt(pow(0.5,2)-pow(0.45,2));
            glVertex3f(vel*cos(angle),vel*sin(angle),-0.45);

     }
     glEnd();
     glBegin(GL_POLYGON);

     for(int i = 0; i <circle_points; i++){
     float angle = 2 * PI * i / circle_points;
            float vel = sqrt(pow(0.5,2)-pow(0.3,2));
            glVertex3f(vel*cos(angle),vel*sin(angle),0.3);

     }
     glEnd();
     }

```

```
/*
*--------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*--------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "extern.h"
#define Pi 3.14159265
/*
*--------------------------------------------------------------------------
*****************Definice proměnných******************
*--------------------------------------------------------------------------
*/
float r_phi_Cz1,r_phi_By1,r_phi_Az1,r_phi_Cz2,r_phi_By2,r_phi_Az2,r_phi_Cz3,r_phi_By3,r_phi_Az3,r_phi_Cz4,
r_phi_By4,r_phi_Az4;


float pom_phi_Cz1 = -65.0,pom_phi_Cz2,pom_phi_Cz3,pom_phi_Cz4, pomocna;
//float pokus = 1.0;
int kk = 0,pocet = 0;
float pole[3][1000];
float rr_phi_Cz1,rr_phi_Cz2,rr_phi_Cz3,rr_phi_Cz4;
/*
*--------------------------------------------------------------------------
*****************Funkce pro nastavení pivotace******************
*--------------------------------------------------------------------------
*/
void nastaveni(){
     kk++;

/*
*--------------------------------------------------------------------------
*****************Převod jednotek úhlů na radiány******************
*--------------------------------------------------------------------------
*/
     r_phi_Az1 = Pi/180.0 * ((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1))
;
     r_phi_By1 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1));

/*
*--------------------------------------------------------------------------
*****************Rovnice exportovaná z SW Maple******************
*--------------------------------------------------------------------------
*/
     r_phi_Cz1 = -0.1e1 * atan(0.1000000000e0 * cos(r_phi_By1) * (0.7071067813e21 * cos(r_phi_Az1) * r +
0.7071067811e21 * sin(r_phi_Az1) * r - 0.3535533908e21 * sin(r_phi_Az1) + 0.1060660172e22 * cos(r_phi_Az1) -
0.1090000000e22 * sin(r_phi_By1) + 0.4500000000e21 + 0.1800000000e22 * cos(r_phi_By1)) / (0.7071067811e20 *
cos(r_phi_Az1) * r - 0.3535533908e20 * cos(r_phi_Az1) - 0.7071067813e20 * sin(r_phi_Az1) * r - 0.1060660172e21 *
sin(r_phi_Az1) + 0.7071067811e9));
/*
*--------------------------------------------------------------------------
*****************Zpětný převod jednotek******************
*--------------------------------------------------------------------------
*/
     rr_phi_Cz1 = 180.0/Pi*r_phi_Cz1;

/*
*--------------------------------------------------------------------------
*****************Ošetření periody funkce tangens******************
*--------------------------------------------------------------------------
*/
   if (kk > 10){
          if((fabs(pom_phi_Cz1 - rr_phi_Cz1)) > 150.0){
            if(pom_phi_Cz1 < rr_phi_Cz1) phi_Cz1 =rr_phi_Cz1-180.0;
            if(pom_phi_Cz1 > rr_phi_Cz1) phi_Cz1 =rr_phi_Cz1+180.0;

            }else phi_Cz1=rr_phi_Cz1;

          }
 pom_phi_Cz1 = phi_Cz1;



//-------------------------------------------------------------------
/*
*--------------------------------------------------------------------------
*****************Převod jednotek úhlů na radiány******************
*--------------------------------------------------------------------------
*/
     r_phi_Az2 = -Pi/180.0 *
((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az2));
     r_phi_By2 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By2));
/*
*--------------------------------------------------------------------------
*****************Rovnice exportovaná z SW Maple******************
```

```
 83: *--------------------------------------------------------------------------
 84: */
 85:      r_phi_Cz2 =atan(0.1000000000e0 * cos(r_phi_By2) * (0.7071067813e21 * cos(r_phi_Az2) * r - 0.1060660172e22 *
     cos(r_phi_Az2) - 0.3535533908e21 * sin(r_phi_Az2) - 0.4500000000e21 - 0.7071067811e21 * sin(r_phi_Az2) * r +
     0.1090000000e22 * sin(r_phi_By2) - 0.1800000000e22 * cos(r_phi_By2)) / (0.7071067811e20 * cos(r_phi_Az2) * r +
     0.3535533908e20 * cos(r_phi_Az2) + 0.7071067813e20 * sin(r_phi_Az2) * r - 0.1060660172e21 * sin(r_phi_Az2) -
     0.707106781e9));
 86: /*
 87: *--------------------------------------------------------------------------
 88: ****************Zpětný převod jednotek******************
 89: *--------------------------------------------------------------------------
 90: */
 91:      rr_phi_Cz2 = 180.0/Pi*r_phi_Cz2- 180;
 92: /*
 93: *--------------------------------------------------------------------------
 94: ****************Ošetření periody funkce tangens******************
 95: *--------------------------------------------------------------------------
 96: */
 97:      if (kk > 10){
 98:           if((fabs(pom_phi_Cz2 - rr_phi_Cz2)) > 150.0){
 99:             if(pom_phi_Cz2 < rr_phi_Cz2) phi_Cz2 =rr_phi_Cz2-180.0;
100:             if(pom_phi_Cz2 > rr_phi_Cz2) phi_Cz2 =rr_phi_Cz2+180.0;
101:
102:             }else phi_Cz2=rr_phi_Cz2;
103:            }
104:            pom_phi_Cz2 = phi_Cz2;
105:
106: //---------------------------------------------------------------------
107: /*
108: *--------------------------------------------------------------------------
109: ****************Převod jednotek úhlů na radiány******************
110: *--------------------------------------------------------------------------
111: */
112:      r_phi_Az3 = Pi/180.0 * ((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az3));
113:      r_phi_By3 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By3));
114:
115: /*
116: *--------------------------------------------------------------------------
117: ****************Rovnice exportovaná z SW Maple******************
118: *--------------------------------------------------------------------------
119: */
120:      r_phi_Cz3 =-0.1e1 * atan(0.1000000000e0 * cos(r_phi_By3) * (-0.1060660172e22 * cos(r_phi_Az3) -
     0.3535533908e21 * sin(r_phi_Az3) - 0.7071067813e21 * cos(r_phi_Az3) * r + 0.7071067811e21 * sin(r_phi_Az3) * r +
     0.1090000000e22 * sin(r_phi_By3) - 0.4500000000e21 - 0.1800000000e22 * cos(r_phi_By3)) / (0.7071067811e20 *
     cos(r_phi_Az3) * r + 0.7071067813e20 * sin(r_phi_Az3) * r - 0.3535533908e20 * cos(r_phi_Az3) + 0.1060660172e21 *
     sin(r_phi_Az3) + 0.707106781e9));
121:
122: /*
123: *--------------------------------------------------------------------------
124: ****************Zpětný převod jednotek******************
125: *--------------------------------------------------------------------------
126: */
127:      rr_phi_Cz3 = 180.0/Pi*r_phi_Cz3- 180;
128: /*
129: *--------------------------------------------------------------------------
130: ****************Ošetření periody funkce tangens******************
131: *--------------------------------------------------------------------------
132: */
133:      if (kk > 10){
134:           if((fabs(pom_phi_Cz3 - rr_phi_Cz3)) > 150.0){
135:             if(pom_phi_Cz3 < rr_phi_Cz3) phi_Cz3 =rr_phi_Cz3-180.0;
136:             if(pom_phi_Cz3 > rr_phi_Cz3) phi_Cz3 =rr_phi_Cz3+180.0;
137:
138:             }else phi_Cz3=rr_phi_Cz3;
139:           }
140:            pom_phi_Cz3 = phi_Cz3;
141:
142: //---------------------------------------------------------------------
143: /*
144: *--------------------------------------------------------------------------
145: ****************Převod jednotek úhlů na radiány******************
146: *--------------------------------------------------------------------------
147: */
148:      r_phi_Az4 = Pi/180.0 * ((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az4));
149:      r_phi_By4 = Pi/180.0 * ((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By4));
150:
151: /*
152: *--------------------------------------------------------------------------
153: ****************Rovnice exportovaná z SW Maple******************
154: *--------------------------------------------------------------------------
155: */
156:      r_phi_Cz4 =-0.1e1 * atan(0.1000000000e0 * cos(r_phi_By4) * (0.7071067813e21 * cos(r_phi_Az4) * r +
     0.7071067811e21 * sin(r_phi_Az4) * r + 0.3535533908e21 * sin(r_phi_Az4) - 0.1060660172e22 * cos(r_phi_Az4) -
     0.4500000000e21 + 0.1090000000e22 * sin(r_phi_By4) - 0.1800000000e22 * cos(r_phi_By4)) / (0.1060660172e21 *
     sin(r_phi_Az4) + 0.7071067811e20 * cos(r_phi_Az4) * r + 0.3535533908e20 * cos(r_phi_Az4) - 0.707106781e9 -
     0.7071067813e20 * sin(r_phi_Az4) * r));
157:
158: /*
159: *--------------------------------------------------------------------------
```

```
160: *****************Zpětný převod jednotek******************
161: *-----------------------------------------------------------------------------
162: */
163:
164:      rr_phi_Cz4 = 180.0/Pi*r_phi_Cz4- 180;
165: /*
166: *-----------------------------------------------------------------------------
167: *****************Ošetření periody funkce tangens******************
168: *-----------------------------------------------------------------------------
169: */
170:      if (kk > 10){
171:           if((fabs(pom_phi_Cz4 - rr_phi_Cz4)) > 50.0){
172:             if(pom_phi_Cz4 < rr_phi_Cz4) phi_Cz4 =rr_phi_Cz4-180.0;
173:             if(pom_phi_Cz4 > rr_phi_Cz4) phi_Cz4 =rr_phi_Cz4+180.0;
174:
175:             }else phi_Cz4=rr_phi_Cz4;
176:           }
177:            pom_phi_Cz4 = phi_Cz4;
178:      }
179:
```

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#include "extern.h"
/*
*-----------------------------------------------------------------------------
*****************Definice proměnných******************
*-----------------------------------------------------------------------------
*/
float MaticeGlobal[16],MaticeTglobal[16], MaticePodvozek[16],
      M_U1[16],M_A1[16],M_B1[16],M_C1[16], M_D1[16],
      M_U2[16],M_A2[16],M_B2[16],M_C2[16], M_D2[16],
      M_U3[16],M_A3[16],M_B3[16],M_C3[16], M_D3[16],
      M_U4[16],M_A4[16],M_B4[16],M_C4[16], M_D4[16];



float phi_Lz = 0.0,Lx = 0.0, Ly = 0.0, Lz = 1.5;
float phi_Az1 = (float) rejd_zakl_pozice,phi_Az2 = (float) rejd_zakl_pozice;
float phi_Az3 = (float) -rejd_zakl_pozice,phi_Az4 = (float) rejd_zakl_pozice;

float phi_By1 = (float) vyro_zakl_pozice,phi_By2 = (float) vyro_zakl_pozice;
float phi_By3 = (float) vyro_zakl_pozice,phi_By4 = (float) vyro_zakl_pozice;

float phi_Cz1, phi_Cz2,phi_Cz3, phi_Cz4;//= -phi_Az1-45.0,phi_Cz2 = phi_Az2-90.0;




long maxon_phi_Cz1,maxon_phi_By1,maxon_phi_Az1;
long maxon_phi_Cz2,maxon_phi_By2,maxon_phi_Az2;
long maxon_phi_Cz3,maxon_phi_By3,maxon_phi_Az3;
long maxon_phi_Cz4,maxon_phi_By4,maxon_phi_Az4;


/*
*-----------------------------------------------------------------------------
*************Funkce pro vytvoření tranformačních matic***************
*-----------------------------------------------------------------------------
*/

void vytvorMatice() {
/*
*-----------------------------------------------------------------------------
*****************Zjištění aktuálních poloh motorů,
       na základě kterých se vytvoří tranformační matice*******************
*-----------------------------------------------------------------------------
*/

     VCS_GetPositionIs(COM1, rejd_1, &maxon_phi_Az1,&pErrorCode);
     VCS_GetPositionIs(COM1, vyro_1, &maxon_phi_By1,&pErrorCode);
     VCS_GetPositionIs(COM1, pivo_1, &maxon_phi_Cz1,&pErrorCode);

     VCS_GetPositionIs(COM2, rejd_2, &maxon_phi_Az2,&pErrorCode);
     VCS_GetPositionIs(COM2, vyro_2, &maxon_phi_By2,&pErrorCode);
     VCS_GetPositionIs(COM2, pivo_2, &maxon_phi_Cz2,&pErrorCode);

     VCS_GetPositionIs(COM3, rejd_3, &maxon_phi_Az3,&pErrorCode);
     VCS_GetPositionIs(COM3, vyro_3, &maxon_phi_By3,&pErrorCode);
     VCS_GetPositionIs(COM3, pivo_3, &maxon_phi_Cz3,&pErrorCode);

     VCS_GetPositionIs(COM4, rejd_4, &maxon_phi_Az4,&pErrorCode);
     VCS_GetPositionIs(COM4, vyro_4, &maxon_phi_By4,&pErrorCode);
     VCS_GetPositionIs(COM4, pivo_4, &maxon_phi_Cz4,&pErrorCode);

     glPushMatrix();

//Total global
    glLoadIdentity();
    glRotatef(90, 0.0, 0.0 ,1.0);
    glTranslatef(-0.5, -1.5,-10.0);

    glGetFloatv(GL_MODELVIEW_MATRIX, MaticeTglobal);
    glLoadIdentity();
//global
    glMultMatrixf(MaticeTglobal);

    glTranslatef(ynew_T, -xnew_T, znew);
    glRotatef(xnew, 1.0, 0.0 ,0.0);
    glRotatef(-ynew, 0.0, 1.0 ,0.0);          // rotace objektu podle pohybu kurzoru myši
    glGetFloatv(GL_MODELVIEW_MATRIX, MaticeGlobal);
```

```
90:
91:     glLoadIdentity();
92:
93: //Podvozek
94:   glMultMatrixf(MaticeGlobal);
95:
96:       glTranslatef(Lx, Ly, Lz);
97:       glRotatef(phi_Lz,0.0,0.0,1.0);
98:       glGetFloatv(GL_MODELVIEW_MATRIX, MaticePodvozek);
99:
100:      glLoadIdentity(); // vynuluje se
101: /*
102: *---------------------------------------------------------------------------
103: *****************První noha*******************
104: *---------------------------------------------------------------------------
105: */
106: //Uchycení nohy
107:
108:      glMultMatrixf(MaticePodvozek);
109:
110:      glTranslatef(1.0, 0.5,0.0);
111:      glRotatef(45.0,0.0,0.0,1.0);
112:      glRotatef(-0.0,0.0,1.0,0.0);
113:      glGetFloatv(GL_MODELVIEW_MATRIX, M_U1);
114:
115:      glLoadIdentity(); // vynuluje se
116:
117: //První část
118:
119:      glMultMatrixf(M_U1);
120:
121:      glTranslatef(0.0, 0.0,0.0);
122:      glRotatef(((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1)),0.0,0.0,
     1.0);
123:      glGetFloatv(GL_MODELVIEW_MATRIX, M_A1);
124:      glLoadIdentity();
125:
126: //Druhá část
127:
128:      glMultMatrixf(M_A1);
129:
130:      glTranslatef(0.45, 0.0,0.0);
131:      glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1)),0.0,1.0,
     0.0);
132:      glGetFloatv(GL_MODELVIEW_MATRIX, M_B1);
133:      glLoadIdentity();
134:
135: //Třetí část
136:
137:      glMultMatrixf(M_B1);
138:
139:      glTranslatef(1.8, 0.0,0.0);
140:      glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz1)),0.0,0.0,
     1.0);
141:      glGetFloatv(GL_MODELVIEW_MATRIX, M_C1);
142:      glLoadIdentity();
143:
144: //Kolo
145:      glMultMatrixf(M_C1);
146:
147:      glTranslatef(0.0, 0.0,-1.09);
148:      glRotatef(-20.0,1.0,0.0,0.0);
149:      glRotatef(uhel,0.0,1.0,0.0);
150:
151:      glGetFloatv(GL_MODELVIEW_MATRIX, M_D1);
152:      glLoadIdentity();
153:
154: /*
155: *---------------------------------------------------------------------------
156: *****************Druhá noha*******************
157: *---------------------------------------------------------------------------
158: */
159: //Uchycení nohy
160:
161:      glMultMatrixf(MaticePodvozek);
162:
163:      glTranslatef(1.0, -0.5,0.0);
164:      glRotatef(-45.0,0.0,0.0,1.0);
165:      glRotatef(-0.0,0.0,1.0,0.0);
166:      glGetFloatv(GL_MODELVIEW_MATRIX, M_U2);
167:
168:      glLoadIdentity(); // vynuluje se
169:
170: //První část
171:
172:      glMultMatrixf(M_U2);
173:
174:      glTranslatef(0.0, 0.0,0.0);
175:      glRotatef(((float)(-360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az2)),0.0,
     0.0,1.0);
```

```
176:      glGetFloatv(GL_MODELVIEW_MATRIX, M_A2);
177:      glLoadIdentity();
178:
179: //Druhá část
180:
181:      glMultMatrixf(M_A2);
182:
183:      glTranslatef(0.45, 0.0,0.0);
184:      glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By2)),0.0,1.0,
     0.0);
185:      glGetFloatv(GL_MODELVIEW_MATRIX, M_B2);
186:      glLoadIdentity();
187:
188: //Třetí část
189:
190:      glMultMatrixf(M_B2);
191:
192:      glTranslatef(1.8, 0.0,0.0);
193:      glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz2)),0.0,0.0,
     1.0);
194:      glGetFloatv(GL_MODELVIEW_MATRIX, M_C2);
195:      glLoadIdentity();
196:
197: //Kolo
198:      glMultMatrixf(M_C2);
199:
200:      glTranslatef(0.0, 0.0,-1.09);
201:      glRotatef(-20.0,1.0,0.0,0.0);
202:      glRotatef(-uhel,0.0,1.0,0.0);
203:
204:      glGetFloatv(GL_MODELVIEW_MATRIX, M_D2);
205:      glLoadIdentity();
206: /*
207: *---------------------------------------------------------------------------
208: ****************Třetí noha******************
209: *---------------------------------------------------------------------------
210: */
211: //Uchycení nohy
212:
213:      glMultMatrixf(MaticePodvozek);
214:
215:      glTranslatef(-1.0, 0.5,0.0);
216:      glRotatef(135.0,0.0,0.0,1.0);
217:      glRotatef(-0.0,0.0,1.0,0.0);
218:      glGetFloatv(GL_MODELVIEW_MATRIX, M_U3);
219:
220:      glLoadIdentity(); // vynuluje se
221:
222: //První část
223:
224:      glMultMatrixf(M_U3);
225:
226:      glTranslatef(0.0, 0.0,0.0);
227:      glRotatef(((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az3)),0.0,0.0,
     1.0);
228:      glGetFloatv(GL_MODELVIEW_MATRIX, M_A3);
229:      glLoadIdentity();
230:
231: //Druhá část
232:
233:      glMultMatrixf(M_A3);
234:
235:      glTranslatef(0.45, 0.0,0.0);
236:      glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By3)),0.0,1.0,
     0.0);
237:      glGetFloatv(GL_MODELVIEW_MATRIX, M_B3);
238:      glLoadIdentity();
239:
240: //Třetí část
241:
242:      glMultMatrixf(M_B3);
243:
244:      glTranslatef(1.8, 0.0,0.0);
245:      glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz3)),0.0,0.0,
     1.0);
246:      glGetFloatv(GL_MODELVIEW_MATRIX, M_C3);
247:      glLoadIdentity();
248:
249: //Kolo
250:      glMultMatrixf(M_C3);
251:
252:      glTranslatef(0.0, 0.0,-1.09);
253:      glRotatef(-20.0,1.0,0.0,0.0);
254:      glRotatef(uhel,0.0,1.0,0.0);
255:
256:      glGetFloatv(GL_MODELVIEW_MATRIX, M_D3);
257:      glLoadIdentity();
258:
259:
```

```
260: /*
261: *---------------------------------_----------------------------------------
262: ****************Čtvrtá noha******************
263: *--------------------------------------------------------------------------
264: */
265: //Uchycení nohy
266:
267:      glMultMatrixf(MaticePodvozek);
268:
269:      glTranslatef(-1.0, -0.5,0.0);
270:      glRotatef(-135.0,0.0,0.0,1.0);
271:      glRotatef(-0.0,0.0,1.0,0.0);
272:      glGetFloatv(GL_MODELVIEW_MATRIX, M_U4);
273:
274:      glLoadIdentity(); // vynuluje se
275:
276: //První část
277:
278:      glMultMatrixf(M_U4);
279:
280:      glTranslatef(0.0, 0.0,0.0);
281:      glRotatef(((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az4)),0.0,0.0,
     1.0);
282:      glGetFloatv(GL_MODELVIEW_MATRIX, M_A4);
283:      glLoadIdentity();
284:
285: //Druhá část
286:
287:      glMultMatrixf(M_A4);
288:
289:      glTranslatef(0.45, 0.0,0.0);
290:      glRotatef(((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By4)),0.0,1.0,
     0.0);
291:      glGetFloatv(GL_MODELVIEW_MATRIX, M_B4);
292:      glLoadIdentity();
293:
294: //Třetí část
295:
296:      glMultMatrixf(M_B4);
297:
298:      glTranslatef(1.8, 0.0,0.0);
299:      glRotatef(((float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz4)),0.0,0.0,
     1.0);
300:      glGetFloatv(GL_MODELVIEW_MATRIX, M_C4);
301:      glLoadIdentity();
302:
303: //Kolo
304:      glMultMatrixf(M_C4);
305:
306:      glTranslatef(0.0, 0.0,-1.09);
307:      glRotatef(-20.0,1.0,0.0,0.0);
308:      glRotatef(-uhel,0.0,1.0,0.0);
309:
310:      glGetFloatv(GL_MODELVIEW_MATRIX, M_D4);
311:      glLoadIdentity();
312:
313:      glPopMatrix();
314: }
315:
```

```
1: /*
2: *---------------------------------------------------------------------------
3: ****************Načtení příslušných knihoven******************
4: *---------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "Definitions.h"
12: #define Pi 3.14159265
13: #include "extern.h"
14:
15:
16: bool start_enabling = false;
17: int faze_en = 1;
18:
19: void enabling(void){
20:
21: /*
22: *---------------------------------------------------------------------------
23: *****Po dokončení pohybů manévr jízda*****
24: *---------------------------------------------------------------------------
25: */
26:     if(faze_en == 4 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
27:           operation = JIZDA;
28:           start_enabling = false;
29:           start_jizda = true;
30:     }
31: /*
32: *---------------------------------------------------------------------------
33: *****Pivotace na Enable*****
34: *---------------------------------------------------------------------------
35: */
36:     if    (faze_en == 3 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done
    && pivo_4_done && rejd_4_done){
37:           VCS_SetEnableState(COM1,rota_1,&pErrorCode);
38:
39:           VCS_SetEnableState(COM2,rota_2,&pErrorCode);
40:
41:           VCS_SetEnableState(COM3,rota_3,&pErrorCode);
42:
43:           VCS_SetEnableState(COM4,rota_4,&pErrorCode);
44:           faze_en++;
45:     }
46: /*
47: *---------------------------------------------------------------------------
48: *****Nastavení polohy rejdu, pivotace na Disable*****
49: *---------------------------------------------------------------------------
50: */
51:     if    (faze_en == 2 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done
    && pivo_4_done && rejd_4_done){
52:            VCS_SetDisableState(COM1,rota_1,&pErrorCode);
53:            VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
54:
55:            VCS_SetDisableState(COM2,rota_2,&pErrorCode);
56:            VCS_MoveToPosition(COM2,rejd_2, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
57:
58:            VCS_SetDisableState(COM3,rota_3,&pErrorCode);
59:            VCS_MoveToPosition(COM3,rejd_3, (-i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,
    Absolute,Immediately,&pErrorCode);
60:
61:            VCS_SetDisableState(COM4,rota_4,&pErrorCode);
62:            VCS_MoveToPosition(COM4,rejd_4, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
63:
64:           faze_en++;
65:     }
66: /*
67: *---------------------------------------------------------------------------
68: Nastavení všech pohonů na Enable a nastavení polohy pivotace a vyrovnávání
69: *---------------------------------------------------------------------------
70: */
71:     if    (faze_en == 1 && pivo_1_done  && pivo_2_done &&  pivo_3_done && pivo_4_done){
72:           VCS_SetEnableState(COM1,rejd_1,&pErrorCode);
73:             VCS_SetEnableState(COM1,vyro_1,&pErrorCode);
74:             VCS_SetEnableState(COM1,pivo_1,&pErrorCode);
75:             VCS_SetEnableState(COM1,rota_1,&pErrorCode);
76:           VCS_MoveToPosition(COM1,vyro_1, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
77:           VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
78:
79:            VCS_SetEnableState(COM2,rejd_2,&pErrorCode);
80:             VCS_SetEnableState(COM2,vyro_2,&pErrorCode);
```

```
 81:             VCS_SetEnableState(COM2,pivo_2,&pErrorCode);
 82:             VCS_SetEnableState(COM2,rota_2,&pErrorCode);
 83:           VCS_MoveToPosition(COM2,vyro_2, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
 84:           VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
 85:
 86:            VCS_SetEnableState(COM3,rejd_3,&pErrorCode);
 87:             VCS_SetEnableState(COM3,vyro_3,&pErrorCode);
 88:             VCS_SetEnableState(COM3,pivo_3,&pErrorCode);
 89:             VCS_SetEnableState(COM3,rota_3,&pErrorCode);
 90:           VCS_MoveToPosition(COM3,vyro_3, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
 91:           VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
 92:
 93:            VCS_SetEnableState(COM4,rejd_4,&pErrorCode);
 94:             VCS_SetEnableState(COM4,vyro_4,&pErrorCode);
 95:             VCS_SetEnableState(COM4,pivo_4,&pErrorCode);
 96:             VCS_SetEnableState(COM4,rota_4,&pErrorCode);
 97:           VCS_MoveToPosition(COM4,vyro_4, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
 98:           VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
 99:
100:           faze_en++;
101:
102:     }
103:
104:
105:
106:     }
107:
```

```
1: /*
2: *------------------------------------------------------------------------
3: ****************Načtení příslušných knihoven******************
4: *------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "Definitions.h"
12: #define Pi 3.14159265
13: #include "extern.h"
14: /*
15: *------------------------------------------------------------------------
16: ****************Manévr jízda, podvozek základní konfigurace,
17:                   pivotace a rychlost řízena joystickem******************
18: *------------------------------------------------------------------------
19: */
20: void jizda(void){
21:
22:  if(start_jizda){
23:       VCS_MoveToPosition(COM1,rejd_1, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az1),Absolute,
    Immediately,&pErrorCode);
24:       VCS_MoveToPosition(COM1,vyro_1, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By1),Absolute,
    Immediately,&pErrorCode);
25:       VCS_MoveToPosition(COM1,pivo_1, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz1),Absolute,
    Immediately,&pErrorCode);
26:      // VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_1, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
27:
28:
29:       VCS_MoveToPosition(COM2,rejd_2, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az2),Absolute,
    Immediately,&pErrorCode);
30:       VCS_MoveToPosition(COM2,vyro_2, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By2),Absolute,
    Immediately,&pErrorCode);
31:       VCS_MoveToPosition(COM2,pivo_2, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz2),Absolute,
    Immediately,&pErrorCode);
32:     //  VCS_SetVelocityMust((HANDLE)COM2, (WORD)rota_2, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
33:
34:       VCS_MoveToPosition(COM3,rejd_3, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az3),Absolute,
    Immediately,&pErrorCode);
35:       VCS_MoveToPosition(COM3,vyro_3, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By3),Absolute,
    Immediately,&pErrorCode);
36:       VCS_MoveToPosition(COM3,pivo_3, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz3),Absolute,
    Immediately,&pErrorCode);
37:      // VCS_SetVelocityMust((HANDLE)COM3, (WORD)rota_3, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
38:
39:       VCS_MoveToPosition(COM4,rejd_4, (long )((i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*phi_Az4),Absolute,
    Immediately,&pErrorCode);
40:       VCS_MoveToPosition(COM4,vyro_4, (long )((i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*phi_By4),Absolute,
    Immediately,&pErrorCode);
41:       VCS_MoveToPosition(COM4,pivo_4, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz4),Absolute,
    Immediately,&pErrorCode);
42:       VCS_SetVelocityMust((HANDLE)COM4, (WORD)rota_4, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);}
43: }
44:
```

```
1: /*
2: *------------------------------------------------------------------------
3: ****************Načtení příslušných knihoven******************
4: *------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "Definitions.h"
12: #define Pi 3.14159265
13: #include "extern.h"
14:
15:
16:
17:
18:
19: void zuzeni(void){
20:    if(faze == 0) faze = 1;
21:    if(start_jizda ) start_jizda = false;
22:
23: /*
24: *------------------------------------------------------------------------
25: ************Zpět na předchozí manévr**************
26: *------------------------------------------------------------------------
27: */
28:    if    (faze == 11 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
29:
30:           VCS_SetEnableState(COM1,rota_1,&pErrorCode);
31:
32:           VCS_SetEnableState(COM2,rota_2,&pErrorCode);
33:
34:           VCS_SetEnableState(COM3,rota_3,&pErrorCode);
35:
36:           VCS_SetEnableState(COM4,rota_4,&pErrorCode);
37:
38:           faze = 0;
39:
40:           if(pred_manevr == 1 || pred_manevr == 3){
41:             operation=JIZDA;
42:             start_jizda = true;}
43:
44:           if(pred_manevr == 2 ) {
45:             operation=KROK;
46:             start_krok = true;
47:             }
48:           if(pred_manevr == 4 ) {
49:             operation=DISABLING;
50:             start_disabling = true;
51:             }
52:
53:
54:           start_zuzeni = false;
55:     }
56: /*
57: *------------------------------------------------------------------------
58: ************Návrat do základní konfigurace**************
59: *------------------------------------------------------------------------
60: */
61:    if    (faze == 10 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
62:
63:
64:           VCS_MoveToPosition(COM3,rejd_3, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
65:
66:           VCS_MoveToPosition(COM4,rejd_4, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
67:
68:           faze++;
69:
70:     }
71: /*
72: *------------------------------------------------------------------------
73: ************Natočení kol do pozice vhodné pro pohyb rejdu**************
74: *------------------------------------------------------------------------
75: */
76:    if    (faze == 9 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
    pivo_4_done && rejd_4_done){
77:
78:
79:           VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
80:           VCS_SetDisableState(COM3,rota_3,&pErrorCode);
81:
82:           VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
```

```
 83:           VCS_SetDisableState(COM4,rota_4,&pErrorCode);
 84:
 85:           faze++;
 86:
 87:     }
 88:
 89: /*
 90: *-----------------------------------------------------------------------
 91: ***************Nastavení pohonu kol do enable*****************
 92: *-----------------------------------------------------------------------
 93: */
 94:    if    (faze == 7 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
 95:
 96:
 97:           VCS_SetEnableState(COM3,rota_3,&pErrorCode);
 98:
 99:           VCS_SetEnableState(COM4,rota_4,&pErrorCode);
100:
101:           faze++; }
102:
103: /*
104: *-----------------------------------------------------------------------
105: ***************Změna rejdu*****************
106: *-----------------------------------------------------------------------
107: */
108:    if    (faze == 6 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
109:
110:           VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
     Immediately,&pErrorCode);
111:
112:           VCS_MoveToPosition(COM2,rejd_2, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
     Immediately,&pErrorCode);
113:
114:           VCS_MoveToPosition(COM3,rejd_3, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
115:
116:           VCS_MoveToPosition(COM4,rejd_4, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
117:           faze++;
118:     }
119:
120: /*
121: *-----------------------------------------------------------------------
122: ************Natočení kol do pozice vhodné pro pohyb rejdu**************
123: *-----------------------------------------------------------------------
124: */
125:     if    (faze == 5 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
126:
127:           VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
128:           VCS_SetDisableState(COM1,rota_1,&pErrorCode);
129:
130:           VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
131:           VCS_SetDisableState(COM2,rota_2,&pErrorCode);
132:
133:           VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
134:           VCS_SetDisableState(COM3,rota_3,&pErrorCode);
135:
136:           VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
137:           VCS_SetDisableState(COM4,rota_4,&pErrorCode);
138:
139:           faze++;
140:
141:     }
142: /*
143: *-----------------------------------------------------------------------
144: ***************Nastavení pohonu kol do enable*****************
145: *-----------------------------------------------------------------------
146: */
147:     if    (faze == 3 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
148:
149:           VCS_SetEnableState(COM1,rota_1,&pErrorCode);
150:
151:           VCS_SetEnableState(COM2,rota_2,&pErrorCode);
152:
153:
154:           faze++;
155:     }
156: /*
157: *-----------------------------------------------------------------------
158: ***************Změna rejdu*****************
159: *-----------------------------------------------------------------------
```

```
160: */
161:     if    (faze == 2 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
     pivo_4_done && rejd_4_done){
162:           VCS_MoveToPosition(COM1,rejd_1, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
163:
164:           VCS_MoveToPosition(COM2,rejd_2, -(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_faze_zuzeni,Absolute,
     Immediately,&pErrorCode);
165:
166:
167:           faze++;
168:     }
169: /*
170: *-------------------------------------------------------------------------
171: ************Natočení kol do pozice vhodné pro pohyb rejdu*************
172: *-------------------------------------------------------------------------
173: */
174:     if    (faze == 1 && pivo_1_done && pivo_2_done && pivo_3_done && pivo_4_done){
175:
176:            VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
177:            VCS_SetDisableState(COM1,rota_1,&pErrorCode);
178:
179:            VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
180:            VCS_SetDisableState(COM2,rota_2,&pErrorCode);
181:
182:
183:           faze++;
184:
185:     }
186:
187:
188: /*
189: *-------------------------------------------------------------------------
190: ***************Při fázi 4 a 8 ovládání pohybu joystickem*****************
191: *-------------------------------------------------------------------------
192: */
193:
194:
195:     if(operation == ZUZENI && ((faze ==4) || (faze ==8)))
196:     {
197:
198:
199:       VCS_MoveToPosition(COM1,pivo_1, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz1),Absolute,
     Immediately,&pErrorCode);
200:       VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_1, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
201:
202:       VCS_MoveToPosition(COM2,pivo_2, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz2),Absolute,
     Immediately,&pErrorCode);
203:       VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_2, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
204:
205:       VCS_MoveToPosition(COM3,pivo_3, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz3),Absolute,
     Immediately,&pErrorCode);
206:       VCS_SetVelocityMust((HANDLE)COM3, (WORD)rota_3, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
207:
208:       VCS_MoveToPosition(COM4,pivo_4, (long )((i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*phi_Cz4),Absolute,
     Immediately,&pErrorCode);
209:       VCS_SetVelocityMust((HANDLE)COM3, (WORD)rota_4, (long) 87.0*(long)xo,(DWORD*) &pErrorCode);
210:
211:     }
212:     }
213:
```

```
/*
*-----------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*-----------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#define Pi 3.14159265
#include "extern.h"


/*
*-----------------------------------------------------------------------------
*****************Definice proměnných******************
*-----------------------------------------------------------------------------
*/
float pomoc = 0.0;
int faze_krok = 0;
int cas,start_cas;

int v2 = 10360/2;
int v1 = v2 / 2;
int v3 = 0*2;
int pocetek_rejd_krok = 10;
int v_n = 1900/20*86/2;
FILE *fw;
/*
*-----------------------------------------------------------------------------
*****************Funkce zajišťující krok,
                                    prozatím jedna noha******************
*-----------------------------------------------------------------------------
*/

void krok(void){
 if(start_jizda) start_jizda = false;


 if(faze_krok == 0) faze_krok = 1;

/*
*-----------------------------------------------------------------------------
*********Ošetření předchozího manévru***********
*-----------------------------------------------------------------------------
*/
 if(faze_krok == 11 && rejd_1_done){
    if(pred_manevr == 1) {
            operation=ZUZENI;
            faze_krok = 0;
            start_zuzeni = true;
            start_krok = false;

    }
    if(pred_manevr == 4){
                    operation=DISABLING;
                    faze_krok = 0;
                    start_disabling = true;
                    start_krok = false;}

    if(pred_manevr == 2 || pred_manevr == 3){
                    operation=JIZDA;
                    faze_krok = 0;
                    start_jizda = true;
                    start_krok = false;}

 }
/*
*-----------------------------------------------------------------------------
*********Návrat do základní konfigurace***********
*-----------------------------------------------------------------------------
*/
  if(faze_krok == 10 && pivo_1_done && rejd_1_done){

    VCS_SetEnableState(COM1,rota_1,&pErrorCode);
    faze_krok++;
 }



 if(faze_krok == 9 && vyro_1_done ){
   VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*rejd_zakl_pozice,Absolute,
Immediately,&pErrorCode);
     VCS_SetDisableState(COM1,rota_1,&pErrorCode);
    faze_krok++;
 }
```

```
 89:
 90:
 91:
 92:  if(faze_krok == 8){
 93:      VCS_SetOperationMode(COM1,rejd_1,0x01/*Profile Position Mode*/,&pErrorCode);
 94:      VCS_SetOperationMode(COM1,vyro_1,0x01/*Profile Position Mode*/,&pErrorCode);
 95:      VCS_MoveToPosition(COM1,vyro_1, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
 96:      faze_krok++;
 97:  }
 98:
 99:
100:
101:
102:  if(faze_krok == 7 && vyro_1_done && rejd_1_done) {
103:  VCS_SetOperationMode(COM1,rejd_1,-0x02/*Profile Position Mode*/,&pErrorCode);
104:  VCS_SetOperationMode(COM1,vyro_1,-0x02/*Profile Position Mode*/,&pErrorCode);
105:  faze_krok = 4;}
106:
107:  if(faze_krok == 6 ) {
108:  if( fabs((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1) -
    pocetek_rejd_krok) > 7.0 ||   //3.0
109:      fabs((float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1) - vyro_zakl_pozice)
    > 5.0 ){     //     2.0
110:  //pomoc = fabs((float)(360.0/(float)(i_plan*i_snek*rozl_enco)*(float)maxon_phi_Az1)) -
    ((i_plan*i_snek*rozl_enco)/360*pocetek_rejd_krok);
111:
112:  VCS_SetOperationMode(COM1,rejd_1,0x01/*Profile Position Mode*/,&pErrorCode);
113:  VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*pocetek_rejd_krok,Absolute,
    Immediately,&pErrorCode);
114:
115:
116:  VCS_SetOperationMode(COM1,vyro_1,0x01/*Profile Position Mode*/,&pErrorCode);
117:  VCS_MoveToPosition(COM1,vyro_1, (i_plan_vyro*i_snek_vyro*rozl_enco_vyro)/360*vyro_zakl_pozice,Absolute,
    Immediately,&pErrorCode);
118:
119:
120:  faze_krok++;}
121:  else   faze_krok = 4;}
122:
123:
124: /*
125: *-----------------------------------------------------------------------------
126: *********Vlastní fáze kroku***********
127: *-----------------------------------------------------------------------------
128: */
129:  if(faze_krok == 5 ){
130:      cas = (glutGet(GLUT_ELAPSED_TIME) - start_cas);
131:      if(cas < 1000){
132:        VCS_SetVelocityMust(COM1,rejd_1, v1,&pErrorCode);
133:        VCS_SetVelocityMust(COM1,vyro_1, 0,&pErrorCode);}
134:      if((cas > 1000) && (cas < 1500)){
135:        VCS_SetVelocityMust(COM1,rejd_1, -v2,&pErrorCode);
136:        VCS_SetVelocityMust(COM1,vyro_1, -v_n,&pErrorCode);}
137:      if((cas > 1500) && (cas < 2000)){
138:        VCS_SetVelocityMust(COM1,rejd_1, -v3,&pErrorCode);
139:        VCS_SetVelocityMust(COM1,vyro_1, v_n,&pErrorCode);}
140:      if(cas > 2000){
141:        VCS_SetVelocityMust(COM1,rejd_1, 0,&pErrorCode);
142:        VCS_SetVelocityMust(COM1,vyro_1, 0,&pErrorCode);
143:        faze_krok++;   }
144:
145:
146:  }
147:
148: /*
149: *-----------------------------------------------------------------------------
150: *********Začátek měření času***********
151: *-----------------------------------------------------------------------------
152: */
153:  if(faze_krok == 4 && rejd_1_done ){
154:      start_cas = glutGet(GLUT_ELAPSED_TIME);
155:
156:     faze_krok++;
157:  }
158: /*
159: *-----------------------------------------------------------------------------
160: *********Změna pracovních módů pro manévr chůze***********
161: *-----------------------------------------------------------------------------
162: */
163:  if(faze_krok == 3 && rejd_1_done  ){
164:
165:
166:    VCS_SetOperationMode(COM1,rejd_1,-0x02/*Profile Position Mode*/,&pErrorCode);
167:    VCS_SetOperationMode(COM1,vyro_1,-0x02/*Profile Position Mode*/,&pErrorCode);
168:     VCS_SetEnableState(COM1,rota_1,&pErrorCode);
169:     faze_krok++;
170:  }
171: /*
```

```
172: *------------------------------------------------------------------------------
173: *********Přesun do konfigurace vhodné pro počátek manévru***********
174: *------------------------------------------------------------------------------
175: */
176:  if(faze_krok == 2 && pivo_1_done && rejd_1_done ){
177:    VCS_MoveToPosition(COM1,rejd_1, (i_plan_rejd*i_snek_rejd*rozl_enco_rejd)/360*pocetek_rejd_krok,Absolute,
     Immediately,&pErrorCode);
178:     faze_krok++;
179:  }
180:
181:  if    (faze_krok == 1 && pivo_1_done){
182:
183:           VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
     &pErrorCode) ;
184:            VCS_SetDisableState(COM1,rota_1,&pErrorCode);
185:           faze_krok++;
186:
187:     }
188: /*
189: *------------------------------------------------------------------------------
190: ****************Kolo je při tomto manévru zabrzděno******************
191: *------------------------------------------------------------------------------
192: */
193:  if(operation == KROK)
194:     {
195:       VCS_SetVelocityMust((HANDLE)COM1, (WORD)rota_1, 0,(DWORD*) &pErrorCode);
196:     }
197:  }
198:
```

```
1: /*
2: *----------------------------------------------------------------------------
3: *****************Načtení příslušných knihoven******************
4: *----------------------------------------------------------------------------
5: */
6: #include <gl/glut.h>
7: #include <math.h>
8: #include <stdarg.h>
9: #include <stdio.h>
10: #include <stdlib.h>
11: #include "extern.h"
12: /*
13: *----------------------------------------------------------------------------
14: *****************Definice proměnných******************
15: *----------------------------------------------------------------------------
16: */
17: char *nazvy[] = {"Rotate", "Translate","Rejd","Jizda","Zuzeni","Krok","Disabling","Enabling","Pokus",
    "Vyrovnavani","Posun"};
18: char *en[] = {"Disable", "Enable"};
19: char *bo[] = {"false","true"};
20:
21: /*
22: *----------------------------------------------------------------------------
23: *****************Nastavení perspektivní projekce******************
24: *----------------------------------------------------------------------------
25: */
26:
27: void setPerspectiveProjection(void)
28: {
29:     glMatrixMode(GL_PROJECTION);                // změna aktuální modifikované matice
30:     glLoadIdentity();
31:     gluPerspective(45,(double)windowWidth/(double)windowHeight,1,100);// nastavení perspektivní kamery
32:     glMatrixMode(GL_MODELVIEW);                 // změna aktuální modifikované matice
33:     glLoadIdentity();
34: }
35:
36: /*
37: *----------------------------------------------------------------------------
38: *****************Nastavení ortogonální projekce******************
39: *----------------------------------------------------------------------------
40: */
41:
42: void setOrthogonalProjection(void)
43: {
44:     glMatrixMode(GL_PROJECTION);                // změna aktuální modifikované matice
45:     glLoadIdentity();
46:     glOrtho(0, windowWidth, 0, windowHeight, -100, 100);
47:     glMatrixMode(GL_MODELVIEW);                 // změna aktuální modifikované matice
48:     glLoadIdentity();
49: }
50:
51: void printString(int x, int y, char *text)
52: {
53:     glRasterPos2i(x, y);                        // pozice prvního znaku řetězce
54:     for (; *text; text++)
55:         glutBitmapCharacter(GLUT_BITMAP_9_BY_15, *text);// vykreslení jednoho znaku
56: };
57: extern int lastTime,fps;
58: extern long maxon_phi_Cz1;
59:
60: /*
61: *----------------------------------------------------------------------------
62: *****************Funkce pro výpis informativního textu******************
63: *----------------------------------------------------------------------------
64: */
65: void drawInfoText(void)
66: {
67:     char str[100];
68:
69:     glColor3f(0.0,0.0,0.0);
70:     if (zobr == 0 || zobr == 1) {
71:
72:
73:     sprintf(str, "Operation:   %s", nazvy[operation]);
74:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-20, str);
75:                                                                          if(r < 2000) {
76:                                                                          if( r > -500.0){
77:                                                                            sprintf(str, "Polomer:
    %3.2f", r);
78:
    printString(glutGet(GLUT_WINDOW_WIDTH)-250, glutGet(GLUT_WINDOW_HEIGHT)-20, str);}
79:                                                                          if( r < -500.0 ){
80:                                                                            sprintf(str, "Polomer:
    daleko");
81:
    printString(glutGet(GLUT_WINDOW_WIDTH)-250, glutGet(GLUT_WINDOW_HEIGHT)-20, str);}}
82:
83:
                if(power == 1){
```

```
84:
                glColor3f(0.0,1.0,0.0);
85:
                     sprintf(str,  "%s", en[power]);
86:
                     printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-20, str);
87:
88:
                    }
89:
90:
                   if(power == 0 && frames % 4 == 0){
91:
                     glColor3f(1.0,0.0,0.0);
92:
                     sprintf(str,  "%s", en[power]);
93:
                     printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-20, str);
94:
                   }
95:
                   glColor3f(0.0,0.0,0.0);
96:
97:
                   sprintf(str, "fps: %d",fps);
98:
                   printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-35, str);
99:
100:   sprintf(str, "Rejd_1_done    : %s",bo[rejd_1_done]);
101:      printString(0, glutGet(GLUT_WINDOW_HEIGHT)-50, str);
102:   sprintf(str, "Vyro_1_done    : %s",bo[vyro_1_done]);
103:      printString(0, glutGet(GLUT_WINDOW_HEIGHT)-65, str);
104:   sprintf(str, "Pivo_1_done    : %s",bo[pivo_1_done]);
105:      printString(0, glutGet(GLUT_WINDOW_HEIGHT)-80, str);
106:
107:
108:   sprintf(str,  "maxon_phi_rejd:    %3.2f",
    (float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1));
109:   printString(0, glutGet(GLUT_WINDOW_HEIGHT)-110, str);
110:   sprintf(str,  "maxon_phi_vyro:    %3.2f",
    (float)(360.0/(float)(i_plan_vyro*i_snek_vyro*rozl_enco_vyro)*(float)maxon_phi_By1));
111:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-125, str);
112:   sprintf(str,  "maxon_phi_pivo:    %3.2f",
    (float)(360.0/(float)(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)*(float)maxon_phi_Cz1));
113:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-140, str);
114:
115:   sprintf(str,  "PC_   phi_pivo:    %3.2f", phi_Cz1);
116:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-155, str);
117:
118:
119:   sprintf(str,  "Zuzeni:  faze = %d",faze);
120:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-185, str);
121:   sprintf(str,  "         bool = %s",bo[start_zuzeni]);
122:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-200, str);
123:
124:   sprintf(str,  "Krok  :  faze = %d",faze_krok);
125:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-230, str);
126:   sprintf(str,  "         bool = %s",bo[start_krok]);
127:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-245, str);
128:
129:   sprintf(str,  "Jizda :  ");
130:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-275, str);
131:   sprintf(str,  "         bool = %s",bo[start_jizda]);
132:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-290, str);
133:
134:
135:   sprintf(str,  "%f",
    fabs((float)(360.0/(float)(i_plan_rejd*i_snek_rejd*rozl_enco_rejd)*(float)maxon_phi_Az1) -
    pocetek_rejd_krok));
136:     printString(0, glutGet(GLUT_WINDOW_HEIGHT)-305, str);
137:
138: }
139:  if (zobr == 3 ||zobr == 4)   {
140:      sprintf(str, "fps: %d",fps);
141:      printString(glutGet(GLUT_WINDOW_WIDTH)-70, glutGet(GLUT_WINDOW_HEIGHT)-35, str);}
142:
143: }
144:
```

```
/*
*---------------------------------------------------------------------------
*****************Načtení příslušných knihoven******************
*---------------------------------------------------------------------------
*/
#include <gl/glut.h>
#include <math.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "Definitions.h"
#define Pi 3.14159265
#include "extern.h"


int faze_dis = 0;

void disabling(void){

   if( start_zuzeni) {
   if(faze == 4) faze = 12;
   if(faze == 8) {faze = 9;}
   operation = ZUZENI;
   start_disabling = false;
 }else if(faze_dis == 0) faze_dis = 1;

   if( start_jizda)
         start_jizda = false;

/*
*---------------------------------------------------------------------------
*******Nastavení všech pohonů na Disable************
*---------------------------------------------------------------------------
*/

   if(faze_dis == 4 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
pivo_4_done && rejd_4_done){
            VCS_SetDisableState(COM1,rejd_1,&pErrorCode);
            VCS_SetDisableState(COM1,vyro_1,&pErrorCode);
            VCS_SetDisableState(COM1,pivo_1,&pErrorCode);
            VCS_SetDisableState(COM1,rota_1,&pErrorCode);

            VCS_SetDisableState(COM2,rejd_2,&pErrorCode);
            VCS_SetDisableState(COM2,vyro_2,&pErrorCode);
            VCS_SetDisableState(COM2,pivo_2,&pErrorCode);
            VCS_SetDisableState(COM2,rota_2,&pErrorCode);

            VCS_SetDisableState(COM3,rejd_3,&pErrorCode);
            VCS_SetDisableState(COM3,vyro_3,&pErrorCode);
            VCS_SetDisableState(COM3,pivo_3,&pErrorCode);
            VCS_SetDisableState(COM3,rota_3,&pErrorCode);

            VCS_SetDisableState(COM4,rejd_4,&pErrorCode);
            VCS_SetDisableState(COM4,vyro_4,&pErrorCode);
            VCS_SetDisableState(COM4,pivo_4,&pErrorCode);
            VCS_SetDisableState(COM4,rota_4,&pErrorCode);
            power=0;
            exit(0);
   }
/*
*---------------------------------------------------------------------------
*******Nastavení hodnoty vyrovnávaní a pivotace na nula************
*---------------------------------------------------------------------------
*/
if    (faze_dis == 3 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done && rejd_3_done &&
pivo_4_done && rejd_4_done){
          VCS_MoveToPosition(COM1,vyro_1, 0,Absolute,Immediately,&pErrorCode);
          VCS_MoveToPosition(COM1,pivo_1, 0,Absolute,Immediately,&pErrorCode);

          VCS_MoveToPosition(COM2,vyro_2, 0,Absolute,Immediately,&pErrorCode);
          VCS_MoveToPosition(COM2,pivo_2, 0,Absolute,Immediately,&pErrorCode);

          VCS_MoveToPosition(COM3,vyro_3, 0,Absolute,Immediately,&pErrorCode);
          VCS_MoveToPosition(COM3,pivo_3, 0,Absolute,Immediately,&pErrorCode);

          VCS_MoveToPosition(COM4,vyro_4, 0,Absolute,Immediately,&pErrorCode);
          VCS_MoveToPosition(COM4,pivo_4, 0,Absolute,Immediately,&pErrorCode);
          faze_dis++;
    }
/*
*---------------------------------------------------------------------------
*******Nastavení hodnoty rejdu na nula************
*---------------------------------------------------------------------------
*/
    if    (faze_dis == 2 && pivo_1_done && rejd_1_done && pivo_2_done && rejd_2_done && pivo_3_done &&
rejd_3_done && pivo_4_done && rejd_4_done){
          VCS_MoveToPosition(COM1,rejd_1, 0,Absolute,Immediately,&pErrorCode);

```

```
87:           VCS_MoveToPosition(COM2,rejd_2, 0,Absolute,Immediately,&pErrorCode);
88:
89:           VCS_MoveToPosition(COM3,rejd_3, 0,Absolute,Immediately,&pErrorCode);
90:
91:           VCS_MoveToPosition(COM4,rejd_4, 0,Absolute,Immediately,&pErrorCode);
92:           faze_dis++;
93:       }
94: /*
95: *-----------------------------------------------------------------------
96: *******Nastavení hodnoty pivotace a nastavení pohonu na Disable************
97: *-----------------------------------------------------------------------
98: */
99:     if    (faze_dis == 1 && pivo_1_done  && pivo_2_done &&  pivo_3_done && pivo_4_done ){
100:          VCS_MoveToPosition(COM1,pivo_1, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
101:           VCS_SetDisableState(COM1,rota_1,&pErrorCode);
102:
103:           VCS_MoveToPosition(COM2,pivo_2, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
104:           VCS_SetDisableState(COM2,rota_2,&pErrorCode);
105:
106:           VCS_MoveToPosition(COM3,pivo_3, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
107:           VCS_SetDisableState(COM3,rota_3,&pErrorCode);
108:
109:            VCS_MoveToPosition(COM4,pivo_4, -(i_plan_pivo*i_snek_pivo*rozl_enco_pivo)/360*90,Absolute,Immediately,
    &pErrorCode) ;
110:           VCS_SetDisableState(COM4,rota_4,&pErrorCode);
111:          faze_dis++;
112:
113:     }
114:     }
115:
```

```
1: /*****************************************************************************************
2:                                 bool
3: *****************************************************************************************/
4: extern bool start_pokus;
5: extern bool start_zuzeni;
6: extern bool start_krok;
7: extern bool start_jizda;
8: extern bool start_disabling;
9: extern bool start_enabling;
10: extern bool zapis;
11:
12: /*****************************************************************************************
13:                                 enum
14: *****************************************************************************************/
15: extern enum myenum{
16:    ROTATE,
17:     TRANSLATE,
18:     REJD,JIZDA,ZUZENI,KROK,DISABLING,ENABLING,POKUS,
19:     VYROVNAVANI,POSUN,//PIVOTACE
20: } operation;
21:
22: extern enum myenum operation;
23:
24: /*****************************************************************************************
25:                                 funkce pro kresleni
26: *****************************************************************************************/
27:
28: void kvadr_ss_v_T(float a, float b, float c);
29: void kolo();
30: void sou_sys();
31: void drawInfoText(void);
32: void printString(int x, int y, char *text);
33: void setPerspectiveProjection(void);
34: void setOrthogonalProjection(void);
35: void kvadr_pos(float a, float b, float c);
36: void te_pivotace();
37:
38:
39:
40: /*****************************************************************************************
41:                                 funkce pro OpenGL
42: *****************************************************************************************/
43:
44: void joystick(unsigned int joystick, int x, int y, int z);
45: void keyboard(unsigned char keyboard, int x, int y);
46: void onMouseButton(int button, int state, int x, int y);
47: void onMouseMotion(int x, int y);
48: void keyboard_special(int keyboard_special, int x, int y);
49:
50: /*****************************************************************************************
51:                                 vlastni funkce
52: *****************************************************************************************/
53:
54: void vytvorMatice();
55: void nastaveni();
56: void zuzeni();
57: void disabling(void);
58: void enabling(void);
59: void pokus(void);
60: void krok();
61: void jizda(void);
62: float sign(float i);
63: int fce_uhel(int x);
64:
65: /*****************************************************************************************
66:                                 float
67: *****************************************************************************************/
68: extern float MaticeGlobal[16],MaticeTglobal[16], MaticePodvozek[16],
69:       M_U1[16],M_A1[16],M_B1[16],M_C1[16], M_D1[16],
70:       M_U2[16],M_A2[16],M_B2[16],M_C2[16], M_D2[16],
71:       M_U3[16],M_A3[16],M_B3[16],M_C3[16], M_D3[16],
72:       M_U4[16],M_A4[16],M_B4[16],M_C4[16], M_D4[16];
73: extern float xo;
74: extern float yo;
75: extern float zo;
76: extern float phi_Lz,r, phi_Az1, phi_By1,pom_phi_Cz1,phi_Cz1,phi_Az2, phi_By2, phi_Cz2,phi_Cz3,phi_Az3, phi_By3,
    pom_phi_Cz3,phi_Az4, phi_By4, phi_Cz4,pom_phi_Cz4;
77: extern float xnew_T,ynew_T;
78: extern float p_phi_Cz1,p_phi_Cz2,p_phi_Cz3,p_phi_Cz4;
79:
80: extern float r,uhel,vyska;
81: extern float pomoc;
82:
83:
84:
85:
86: /*****************************************************************************************
87:                                 int
88: *****************************************************************************************/
```

```
 89: extern int faze,faze_krok;
 90: extern int   xnew, ynew, znew;
 91: extern int   xold, yold, zold;
 92: extern int   xx, yy, zz,xova;
 93: extern int   windowWidth,windowHeight;
 94: extern int yy,pocet,power,frames;
 95: extern int pred_manevr;
 96: extern int cas,start_cas;
 97:
 98: extern int zobr;
 99: extern int pocetek_rejd_krok;
100: extern int rejd_zakl_pozice,rejd_faze_zuzeni,vyro_zakl_pozice,faze_dis;
101: extern int i_plan_rejd;
102: extern int i_snek_rejd;
103: extern int rozl_enco_rejd;
104:
105: extern int i_plan_vyro;
106: extern int i_snek_vyro;
107: extern int rozl_enco_vyro;
108:
109: extern int i_plan_pivo;
110: extern int i_snek_pivo;
111: extern int rozl_enco_pivo;
112:
113:
114: /*****************************************************************************
115:                                 long
116: *****************************************************************************/
117: extern int faze;
118: extern int   xnew, ynew, znew;
119: extern int   xold, yold, zold;
120: extern int   xx, yy, zz;
121: extern long maxon_phi_Cz1,maxon_phi_By1,maxon_phi_Az1,maxon_phi_Cz2,maxon_phi_By2,maxon_phi_Az2,maxon_phi_Cz3,
     maxon_phi_By3,maxon_phi_Az3,maxon_phi_Cz4,maxon_phi_By4,maxon_phi_Az4;
122:
123: /*****************************************************************************
124:                                 promenne pro MAXON
125: *****************************************************************************/
126: extern HANDLE m_KeyHandle;
127:
128:
129: extern     HANDLE COM1,COM2,COM3,COM4;
130: extern     __int8 Mode;
131: extern  DWORD pErrorCode;
132: extern BOOL Absolute ;
133: extern BOOL Immediately;
134: extern BOOL rejd_1_done ,pivo_1_done,vyro_1_done,rejd_2_done ,pivo_2_done,vyro_2_done,rejd_3_done ,pivo_3_done,
     vyro_3_done,rejd_4_done ,pivo_4_done,vyro_4_done ;
135: extern WORD vyro_1,rejd_1,pivo_1,rota_1,vyro_2,rejd_2,pivo_2,rota_2,vyro_3,rejd_3,pivo_3,rota_3,vyro_4,rejd_4,
     pivo_4,rota_4;
136:
137:
138:
139:
140: /*****************************************************************************
141:                                 FILE
142: *****************************************************************************/
143:
144: extern FILE *fw;
145:
```