ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

# Průmyslový robot RSP 01 – nadřazené řízení

Šiška Matěj

**BAKALÁŘSKÁ PRÁCE**
2007

**Vedoucí práce: Doc. Ing. Jiří Bayer, CSc.**

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídicí techniky

Školní rok: 2006/2007

# Zadání bakalářské práce

Student: Matěj Š i š k a

Obor: Kybernetika a měření

Název tématu: Průmyslový robot RSP 01 – nadřazené řízení

## Zásady pro vypracování:

1. Seznamte se s mechanickou koncepcí průmyslového robota RSP 01 a s požadavky na základní řízení jeho pohybových os.
2. V návaznosti na paralelní BP V.Sedláčka (základní řízení robota) realizujte řízení pohybových os robota s využitím "motion controlleru" PMAC fy. Delta Tau z nadřazeného PC prostřednictvím sériového kanálu RS232.
3. Na základě kinematiky robota navrhněte jeho řízení v kartézských souřadnicích.
4. Vypracujte podrobnou dokumentaci připojení robota k nadřízenému řídicímu systému a dokumentaci programového vybavení.

***Seznam odborné literatury:*** Dodá vedoucí práce

**Vedoucí bakalářské práce:** Doc. Ing. Jiří Bayer, CSc.

**Datum zadání bakalářské práce:** zimní semestr 2006/07

**Termín odevzdání bakalářské práce:** 15. 8. 2007

Prof. Ing. Michael Šebek, DrSc.
**vedoucí katedry**

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L.S. -4-

v zast.

Prof. Ing. Zbyněk Škvor, CSc.
**děkan**

V Praze, dne 6. 3. 2007

## *Prohlášení*

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady ( literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 20.8.2007 …….

Podpis

## *Poděkování*

Zde bych rád vyjádřil poděkování lidem, kteří přispěli ke zdárnému dokončení této bakalářské práce. V prvé řadě určitě rodině, že se mnou měla trpělivost a ušetřila mě obvyklých domácích činností, abych se mohl maximálně věnovat této práci a díky níž mohu i studovat tuto školu. Dále lidem působícím přímo na škole. Vedoucímu této práce, váženému panu Doc. Ing. Jiřímu Bayerovi, Csc., který mne vedl správným směrem ke zdárnému vyřešení problému. Panu Ing. Pavlu Píšovi, za pomoc při prosazení myšlenky řídící jednotky MARS 8b a její následné nasazení v laboratoři. Rovněž bych chtěl poděkovat panu Ing. Petru Porazilovi, pracovnímu kolegovi pana Ing. Píši, který taktéž pomáhal při nasazení řešení řízení pomoci MARS 8b. Rád bych i poděkoval lidem z průmyslu, panu Ing. Tiborovi Vilhamovi a panu Rapčanovi z firmy Rapčan Servis Detva s.r.o, kteří mi umožnili nahlédnout do původní dokumentace a byli se mnou i ochotni na dané téma diskutovat.

## *Abstrakt*

V této práci je popsán současný stav robotu OJ-10RS. K čemu sloužil a čeho byl součástí. Následuje popis konstrukce a parametry robotu a rozbor bývalého řízení včetně návrhnu obecného principu nového moderního řešení. Na základě principu moderního řízení navrhuji 4 konkrétní varianty řízení. Ty jsou postupně rozebrány včetně popisu použitých komponent. Nejvíce je popsána výsledná implementovaná varianta i s detailnějším popisem řídící jednotky MARS 8b. Řízení robotu v prostoru je řešeno přes DKT, hlavně IKT robotu. IKT je implementována i v MATLABu, z kterého robot řídím. K obsluze jsou využity zdrojové kódy od CMP, které jsou upraveny pro mou potřebu. Rovněž jsem jich několik musel i přímo vytvořit. Ty všechny jsou zde stručně popsané včetně grafického znázornění vzájemné provázanosti funkcí. Na závěr popisuji způsob monitorování pomocí karty HUMUSOFT MF614 v prostředí MATLAB, včetně obrázků vytvořeného 3D modelu robotu. Přílohy tvoří poslední část dokumentu.

## *Abstract*

This work describes the present state of the robot OJ-10RS. What it has served to or what it has been part of. Description of the construction, parameters of the robot and analysis of the former control including the design of general principle of the modern solution follow. I design 4 specific control variations on the basis of modern control principle, these are successively analysed including the description of the components used. The final implemented variation is described in detail together with the detailed description of the control unit MARS 8b. The control of the robot in space is solved through DKT, mainly IKT robot. IKT is also implemented in MATLAB, where I control the robot. For the operation are used source codes by CMP that are edited for my need. I also had to create several new ones. All of the source codes are briefly described here including the graphic demonstration of the reciprocally related functions. In conclusion I describe the way of monitoring with the assistance of the PC card HUMUSOFT MF614 in MATLAB including the image of created 3D model of the robot. The appendix is the last part of the document.

# Obsah

# 1 Historie robotiky a její definice

## *1.1 Historie robotiky*

Člověk je od přírody tvor lenivý a tak snaha lidstva o usnadnění si každodenních, většinou rutinních úkolů, ať už doma či v zaměstnání vedla k úvahám o jejich automatizaci.

Zaměřme se nyní spíše na úkoly pracovního rázu a přesuňme se do doby 18. století, kdy probíhala 1. průmyslová revoluce. Jedním z hlavním znaků této revoluce bylo hromadné zavádění strojů do výroby – počátky automatizace, tzn. běžné lidské opakující se pracovní úkony byly nahrazeny stroji. Výhody strojů jsou jednak v jejich „neúnavnosti“, přesnosti, rychlosti, bezchybnosti - jsou-li dobře navrženy, apod. Sice je pravda, že většinou pořizovací náklady nejsou zrovna nízké, ty se však časem vrátí a opět vychází nasazení strojů i ekonomicky výhodnější než lidská práce. Nyní poskočme v čase o něco blíže současnosti, tj. do 2. průmyslové na přelom 19. a 20. století. V tomto případě se jednalo spíše o vědecko-technickou revoluci, tzn. vznikaly nové výrobní postupy, nové materiály, nové zdroje energie (elektřina, výbušné motory…) a rovněž se vědní obory více rozčlenili. Rovněž i v této době se pokračovalo v automatizaci výroby.

Významných rokem pro robotiku byl letopočet 1920, kdy český spisovatel Karel Čapek poprvé ve své divadelní hře R.U.R. (Rossum's Universal Robots překládáno jako Rossumovi univerzální roboti) použil slovo „robot“. Robotem pojmenoval stroj, který se vzhledově a funkčně podobal člověku. Do té doby byly „roboty“ spíše jen atrakcí na výstavách pro pobavení, avšak po této zlomové události se již objevují první praktické aplikace. Např. tzv. teleoperátoři, kteří byli využívání pro manipulaci s radioaktivními materiály a jinými nebezpečnými materiály (1940 - 47). V tomto období rovněž definoval Isaac Asimov tři zákony robotiky:

1) Robot nesmí škodit životu a zdraví lidí svou činností nebo nečinností
2) Robot se musí podřídit každému příkazu člověka s výjimkou těch, které by vedly k porušení 1. zákona
3) Robot se musí bránit každému ohrožení své existence, jestliže obrana nevede k porušení 2. zákona

V r. 1949 byl zahájen výzkum numericky řízených obráběcích strojů. Roku 1961 je dán do provozu první průmyslový robot UNIMATE pro lití pod tlakem u firmy General

Motors (USA). Vývoj tohoto robotu je spojen se jmény George Devol (patent na řízení programem s děrnými štítky), Joseph Engelberger (strojní část robotu) a universitou Columbia University USA. Během 60.let minulého století rovněž začaly vznikat výrobní linky – tj. zařízení, které do sebe implementují více dílčích činností. V roce 1964 jsou otevřeny laboratoře umělé inteligence (UI) na Massachutess Institute of Technology (M.I.T.), Stanford Research Institute (S.R.I.) a dalších institucích v USA. Mají se zabývat mimo jiné využitím UI v robotice. Roku 1968 je postaven na S.R.I. mobilní robot Shakey vybavený viděním. Dále roku 1977 dává do prodeje své velmi zdařilé roboty evropská firma ASEA. Následně roku 1979 jsou uvedeny na trh roboty koncepce Selective Compliant Articulated Robot Arm (SCARA). Průmyslové roboty se stávají běžným prostředkem automatizace manipulačních operací především v automobilovém průmyslu. Rovněž jsou masivně používány pro svařování plamenem, elektrickým obloukem, bodovým svařováním. Jsou používány pro nanášení barev a všude tam, kde jsou manipulační operace pro člověka nebezpečné a zdraví škodlivé. Počáteční předstih USA ve výzkumu, ale hlavně ve využití robotů, přebírá Japonsko. Po roce 1980 začínají být první průmyslový roboty vybavováni počítačovým viděním, čidly hmatu a dalšími prvky, které zatím spadaly do oblasti výzkumu UI. Po roku 1995 se objevuje první chirurgický robotický systém pro tzv. minimálně invazivní chirurgii. Na Marsu je roku 1997 vysazen robot Sojourner. Zhruba ve stejném období jsou položeny základy mezinárodním organizacím Federation of International Robot-soccer Association (FIRA) a RoboCup, které organizují soutěže robotů ve fotbale. Cílem těchto organizací je především urychlení výzkumu v robotice. RoboCup má dokonce ve své preambuli za cíl, aby robotický tým porazil lidský tým – mistra světa v r. 2050 v regulérním fotbalovém zápase. V r. 2000 předvádí firma Honda svého humanoidního robota ASIMO a SONY předvádí své zoocidy (robot inspirovaný zvířetem) AIBO.

## Definice robotu

Stejně jako se vyvíjel robot sám, vyvíjela se i jeho definice. Cílem bylo zkoncipovat takový popis, který by jednoznačně vyloučil automaty, manipulátory a další samočinná zařízení. Poslední mně známá definice a uznávaná i standardem ISO je 1990 ISO TR 8376:

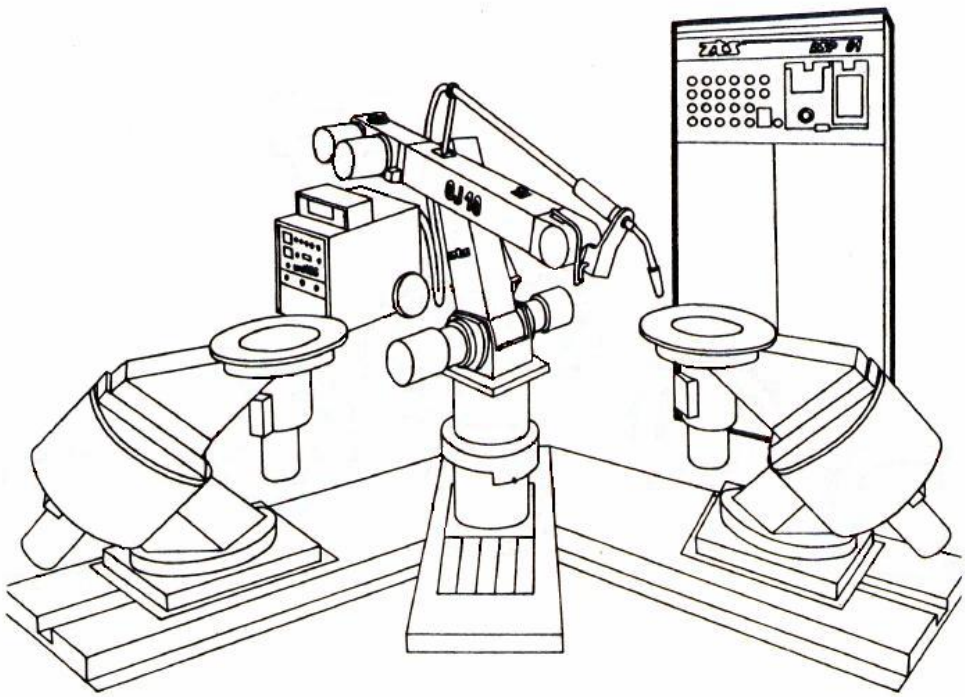
*„ Průmyslový robot je automaticky řízený reprogramovatelný víceúčelový manipulační stroj, stacionární nebo umístěný na pojezdu, přičemž je určen na použití v průmyslové automatizaci.“*

# 2 Identifikace mechaniky průmyslového robotu OJ-10RS

Tak určitě prvním krokem při zahájení práce na této bakalářské práci je zjistit jaký je současný stav zařízení, podívat se k čemu sloužilo či čeho bylo součástí.

## *2.1 Robotizované svářecí pracoviště*

**Robotizované svářecí pracoviště**



*Obr. č. 1 – kompletní pohled na robotizované pracoviště*

Robot OJ-10RS, jímž se zabývám, tvořil nedílnou součást robotizovaného svářecího pracoviště (*viz.* O*br. č. 1*) vytvořené firmou ZTS Detva, ve své době sídlící na Slovensku.

Tato firma se však po revoluci přetransformovala, čímž získání jakékoliv dodatečné dokumentace je téměř nemožné. Rovněž je politováníhodné, že na katedře žádné originální podklady nejsou.

Zbytek pracoviště tvořila ještě původní řídící jednotka RSP 01 a jeden pár polohovadel OJ-10P. Celé pracoviště, jak již z názvu plyne, sloužilo ke svařování. Konkrétně ke svařování obloukovému. Pracoviště je datováno k roku 1987, alespoň podle výrobního štítku umístěného přímo na těle robotu. Ve škole je nyní dispozici pouze funkční robot OJ-10RS, polohovadla OJ-10P byla vyhozena, což nyní zjišťuji, že nebylo zrovna nejlepší

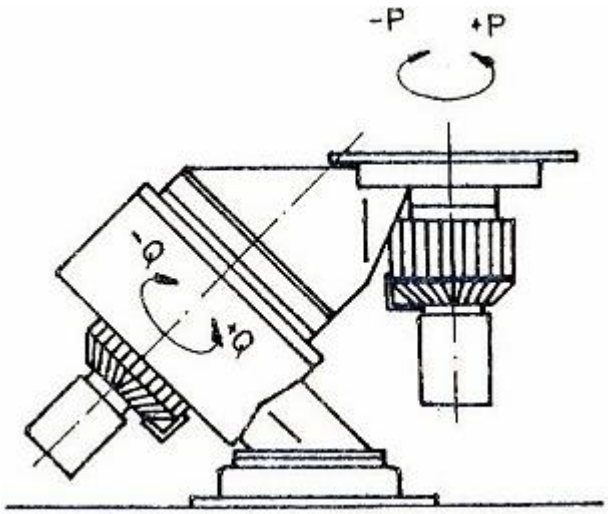
rozhodnutí. Dále je k dispozici ovládací jednotka RSP 01, která však několik let byla skladována venku a nechána napospas rozmarům počasí.

Do pátrání po dokumentaci jsem se dal sám. Pomocí internetu jsem nalezl několik institucí, kde s tímto robotem měli zkušenosti. Nakonec jsem navázal spolupráci s firmou Rapčan Servis Detva s.r.o ze Slovenska. Pan Rapčan pracoval v původním podniku ZTS Detva (výrobce robotu OJ-10RS) a měl k dispozici podrobnou dokumentaci, kterou mi přes svého spolupracovníka pana Ing. Tibora Vilhana poskytl.

**Robot OJ-10RS**

*Viz samostatná kapitola: 2.2.1 Popis robotu OJ-10RS*

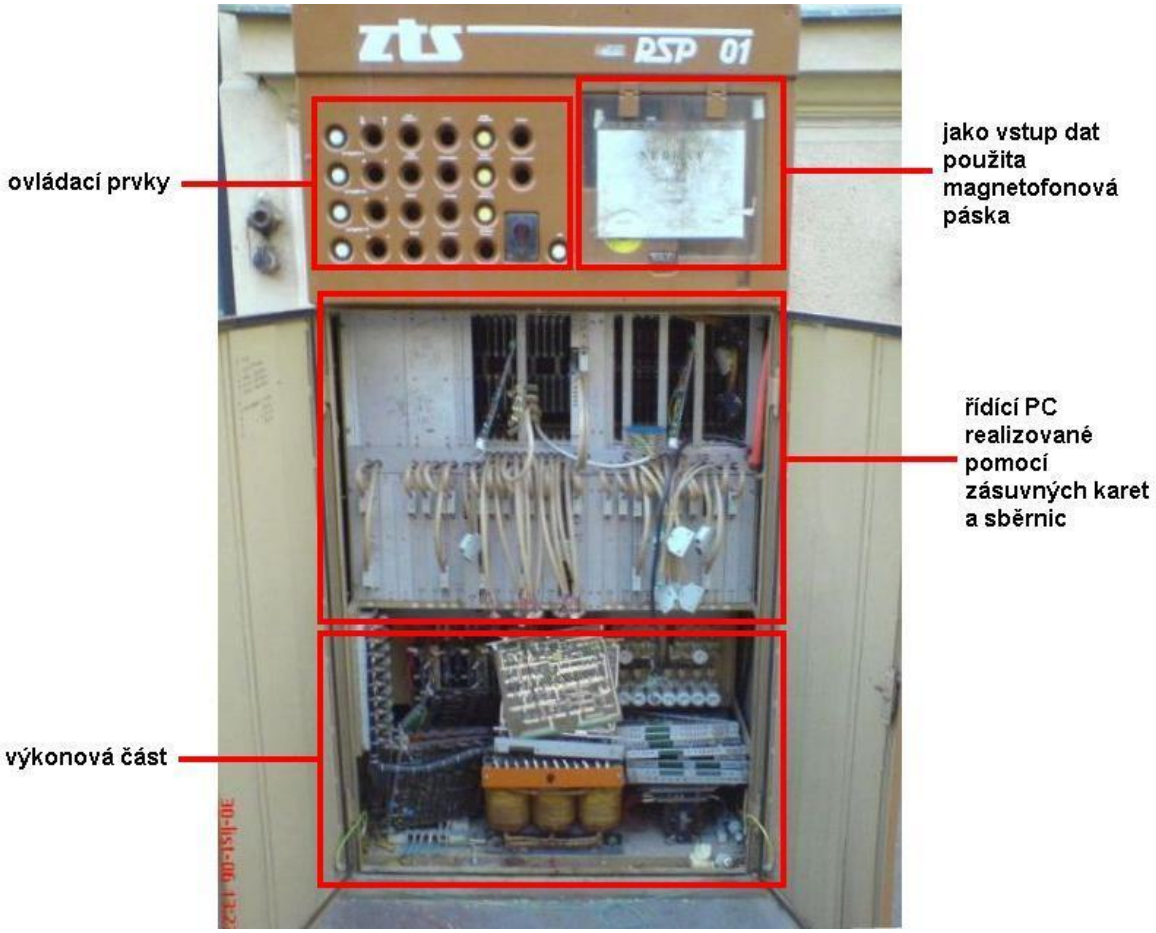
**Polohovadlo OJ-10P**




*Obr. č. 2 – polohovadlo včetně zobrazení os otáčení*

Toto je velmi důmyslné zařízení (*viz. Obr. č. 2*), pomocí kterého lze poměrně jednoduše zvýšit počet stupňů volnosti robotu z původních 5ti na 7. Jedná se o tzv. svářecí stoly, které mají 2 stupně volnosti. V jednom robotizovaném pracovišti byla tyto zařízení právě dvě. Tzn. že např. na prvním stole robot právě pracoval, kdežto na druhém stole mohl v klidu pracovník připravovat další část na sváření. Když robot dokončil práci na současném stole, přesunul se na ten druhý, kde mu již pracovník připravil novou část na sváření a pracovník šel zpět na první stůl, kde již odebral hotový svár a připravil další část na sváření. Tímto se minimalizovaly prostoje svářecí linky a zvýšila se produktivita.

**Řídící jednotka RSP 01**




*Obr. č. 3 – aktuální stav původní řídící jednotky včetně základního popisu její částí*

Řídící skříň (*viz. Obr. č. 3*) dosahuje nemalých rozměrů, což na svou dobu bylo jistě typické provedení. Obsahovala tehdejší kompletní řídící počítač realizovaný pomocí zásuvných karet propojených sběrnicí včetně vlastního aktivního chladícího systému. Jako vstupní zařízení sloužila magnetofonová páska, na které byl nahrán program. Pomocí ovládacího panelu se dalo pracovat s programem na magnetofonové pásce či přímo pohybovat robotem podle potřeby. Pro získání hodnot aktuální polohy robotu a jeho rychlosti se využívala čidla selsyny a tachodynama. Selsyn slouží pro zjištění aktuální polohy a tachodynamo měří aktuální rychlost otáčení. Na dně řídící jednotky byla umístěna výkonová část. Skládá se z jedno hlavního a několika pomocných transformátorů. Hlavní transformátor obstarává zdroj výkonu pro jednotlivé motory. Výkon je přiváděn na výkonové desky, které obsahují výkonové můstky. Na základě řídících signálů, které generuje jiná část řídícího počítače z aktuální polohy robotu a potřebného akčního zásahu, desky generují požadovaný

PWM signál, který pak posílají na jednotlivé motory na robotu. Pomocné transformátory obsluhují napájení pomocné elektroniky – řídící počítač, chlazení apod.

Z obrázku řídící skříně je jasně patrné, že kolegové ze školy již na řídící skříni pracovali. Robot řídili pomocí řídící skříně a pohybového kontroléru PMAC. Zároveň nahradili původní čidla (selsyny a tachodynama) modernějšími IRC. Ty v sobě integrují schopnosti obou původních čidel, tudíž v jejich využití budu i pokračovat.

## Obloukové svařovaní

Jedná se o způsob tavného svařování využívající tepla elektrického oblouku k roztavení přídavného materiálu (elektrody) i materiálu základního. Obloukové svařování je například pod tavidlem, v ochranné atmosféře (argonu) apod.

## Obloukové svařování za použití robotů

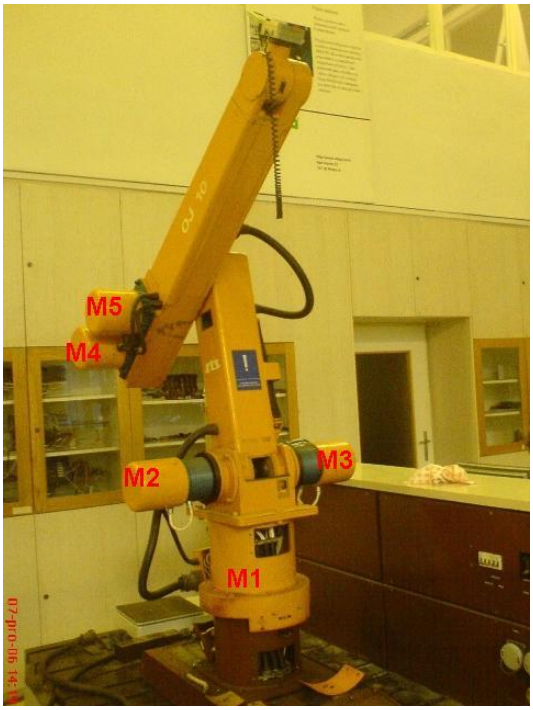
Důležitou vlastností robotů pro obloukové svařování je jejich adaptabilita. Rozeznáváme její čtyři stupně: hlídání, stabilizace, řízení, optimalizace .

Cílem je, aby svařovací hubice vždy důsledně sledovala svařovací dráhu, dokázala najít její počátek, vhodně se polohovala v závislosti na tvaru spáry a aby docházelo k dynamickému měnění svařovacích parametrů (posuv drátu, svařovací proud a napětí, přísun ochranného plynu apod). Zajištění sledování svařovací dráhy se v dřívější době provádělo pomocí taktilních snímačů (např. vodicí tyčinka ve spáře), v současnosti pomocí optických a indukčních snímačů.

Pro optimální využití robotu jsou pracoviště osazena polohovadly, která umožní v kooperaci s robotem svařit i složité svary.

## 2.2 Popis robotu OJ-10RS

### 2.2.1 Popis konstrukce robotu OJ-10RS




*Obr. č. 4 – dvojice pohledů na robot OJ-10RS včetně označení jeho motorů*

Svářecí robot OJ-10RS (*viz. Obr. č. 4)* je poměrně robustní konstrukce. Parametry udávající délky ramen lze vyčíst z originálních podkladů, které jsou uvedeny v *Příloze č. 1 a 2*. Veškerá konstrukce je kovová, na vnějším plášti je jen minimum plastů. A to pouze v případě konektorů či průchodek pro kabelové spojení jednotlivých částí (ramen), kde původní konektory bude potřeba alespoň z části vyměnit za nové. Robot má 5 stupňů volnosti nebo-li 5 os. Každá osa je vybavena samostatným motorem, o motorech viz. další *kapitola 2.2.2 Popis motorů a jejich parametry, převodovky, čidla*. Pojedu-li s popisem od základny (ta část, kterou je robot přichycen k podložce) narazím na první motor značený jako M1, který obstarává rotaci robota, jeho pracovní rozsah je 260º. Postupem vzhůru po těle robotu se nachází po dvojici stejných motorů M2 a M3, které jsou umístěné na stejné smyšlené ose. Motor M2 pohybuje prvním pohyblivým ramenem. To má rozsah ±40º. Motor tvořící s ním dvojici, značený jako M3 pohybuje druhým pohyblivým ramenem. Na druhém pohyblivém ramenu je opět umístněna dvojice stejných, avšak oproti předchozím již rozměrově menších a výkonově slabších , motorů. Tyto motory pohybují tzv. zápěstím, což je pohyblivé zakončení

raene robotu. Motor M4 pohybuje zápěstím o ±115º a motor M5 točí zápěstím kolem jeho osy v rozsahu ±305º.

### 2.2.2 Popis motorů a jejich parametry, převodovky, čidla

Robot disponuje 5ti stupni volnosti, což odpovídá 5ti motorům. Každý motor má samostatné napájení a řízení. Na každou osu motoru je umístěno inkrementální čidlo IRC.

**Motory**

Jedná se o stejnosměrné motory, které jsou řízeny pomocí PWM signálů generovaných řídící jednotkou. Více viz. bakalářská práce Václava Sedláčka a souhrnná *Tabulka č. 1*.

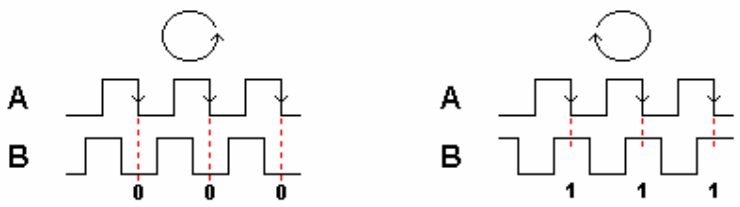
| Č. motoru | Typové označení | $I_n$ / A | $U_n$ / V | P / W | Převodovka |
|---|---|---|---|---|---|
| 1 | SRD 350 | 7,4 | 62 | 350 | HP 120-207-I-3-DK |
| 2 | SRD 350 | 7,4 | 62 | 350 | HP 100-207-I-2 |
| 3 | SRD 350 | 7,4 | 62 | 350 | HP 100-207-I-2 |
| 4 | SRD 80 | 13,6 | 15,5 | 100 | HP 60-124-I/II-2 |
| 5 | SRD 80 | 13,6 | 15,5 | 100 | HP 60-124-I/II-2 |

*Tabulka č. 1 – stručný souhrn parametrů motorů*

**Čidlo IRC**

Optické inkrementální čidlo, které při otáčení osou čidla generuje pulsy. Ty je pak možno v pomocné logice zpracovávat a určovat tak směr otáčení, rychlost otáčení apod.

Principiálně tedy čidlo generuje 2 obdélníkové průběhy *(viz. Obr. č. 5)*, které jsou však fázově posunuty o přibližně 90 stupňů. Při příchodu hrany v jednom průběhu se čte stav druhého signálu. Je-li v log. 1 pak přičítáme, kdež to je-li v log. 0, odečítáme.



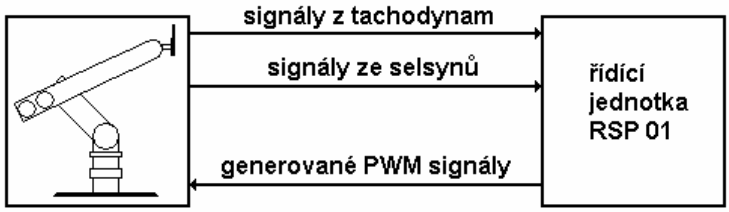

*Obr. č. 5 – princip IRC čidla*

Takto získám zákl. jednoduchou přesnost, tedy že čítám pouze na jednu hranu signálu. Tzv. dvojitá přesnost využívá pro čítání sestupnou i vzestupnou hranu. Čtyřnásobná přesnost, kterou využívám při měření i já, využívá zaměnitelnosti výstupů, čímž získám dvojnásobnou přesnost oproti předchozímu použití.

# 3 Požadavky na řízení

## 3.1 Blokové schéma původního řízení

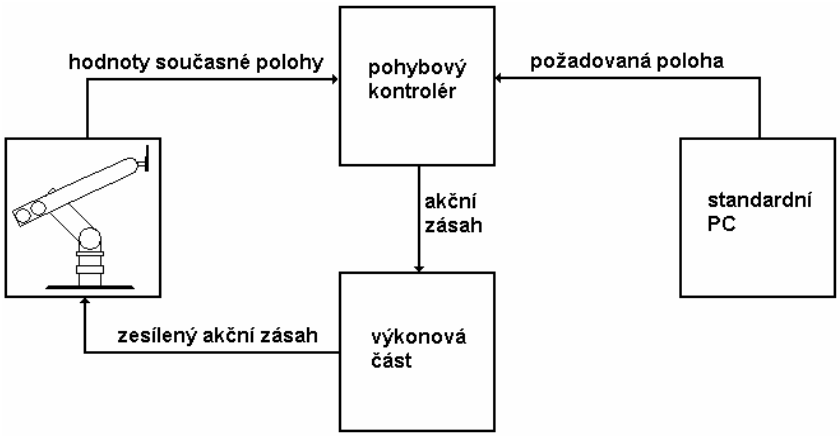
Původně byl robot řízený pomocí firmou dodávané řídící jednotky RSP 01 *viz kapitola: 2.1 Robotizované svářecí pracoviště.* Dále viz. *Obr. č. 6.*




*Obr. č. 6 – blokové schéma původního řízení včetně signálů*

## 3.2 Blokové schéma nového řízení – obecný popis řízení přes PC

Představa moderního řízení (*viz. Obr. č. 7*) počítá s využitím běžného stolního PC. Uživatel si na PC navolí libovolnou trajektorii pohybu (PC je zároveň interfacem pohybového kontroléru). Tato trajektorie je z obslužného SW na PC přes komunikační kanál nahrána do pohybového kontroléru. Ten propočítá jednotlivé body trajektorie, provede interpolace pohybu apod. Rovněž určí akční zásahy do jednotlivých motorů. Pohybový kontrolér přizpůsobuje akční zásahy s ohledem na zpětnou vazbu od čidel na robotu. Tyto akční zásahy se přes výkonovou část zesílí na úroveň potřebnou pro vybuzení jednotlivých motorů na robotu a odešlou se již přímo na jednotlivé motory. Toto se děje v uzavřené smyčce.




*Obr. č. 7 – blokové schéma nového řízení včetně obecných signálů*

# 4 Možnosti řízení

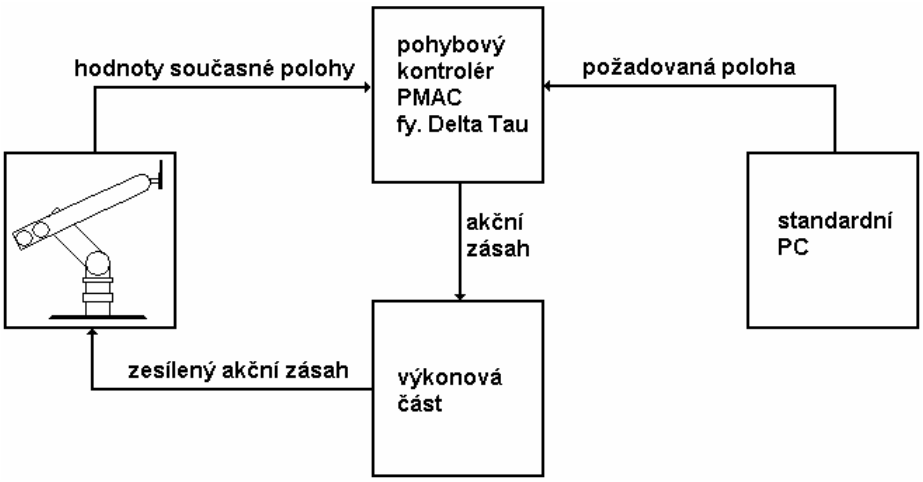
Na základě obecného popisu moderního způsobu řízení bych nyní uvedl čtyři varianty, o kterých by se dalo uvažovat.

1) PC – pohybový kontrolér PMAC - výkonová část
2) PC – karta HUMUSOFT MF614 – SW MATLAB SIMULINK – výkonová část
3) PC – řídící jednotka MARS 8b
4) PC – MARS 8b – SW MATLAB SIMULINK – karta HUMUSOFT MF614

U všech uvádím v následujících kapitolách blokové schéma včetně popisu jednotlivých použitých komponent.

## *4.1 PC – pohybový kontrolér PMAC – výkonová část*

Toto řešení, tedy pomocí standardního stolního PC, pohybového kontroléru PMAC fy. DELTA TAU a výkonové částí je i navrhováno v zadání bakalářské práce. Rozeberu tedy toto řešení (*viz Obr. č. 8*).




*Obr. č. 8 – blokové schéma návrhu nového řízení „PC – pohybový kontrolér PMAC – výkonová část“ včetně obecných signálů*

**Klasické stolní PC**

Základní účel PC je vytvářet interface pro pohybový kontrolér. Dále zprostředkovává uživatelské vstupy a rovněž by ho bylo možné využít ke zpětnému monitoringu skutečných pohybů robotu.

Konfigurace PC:

Operační systém MICROSOFT WINDOWS 2000 - SP4

Procesor INTEL PENTIUM 4 – 3GHz

Operační paměť 512 MB RAM

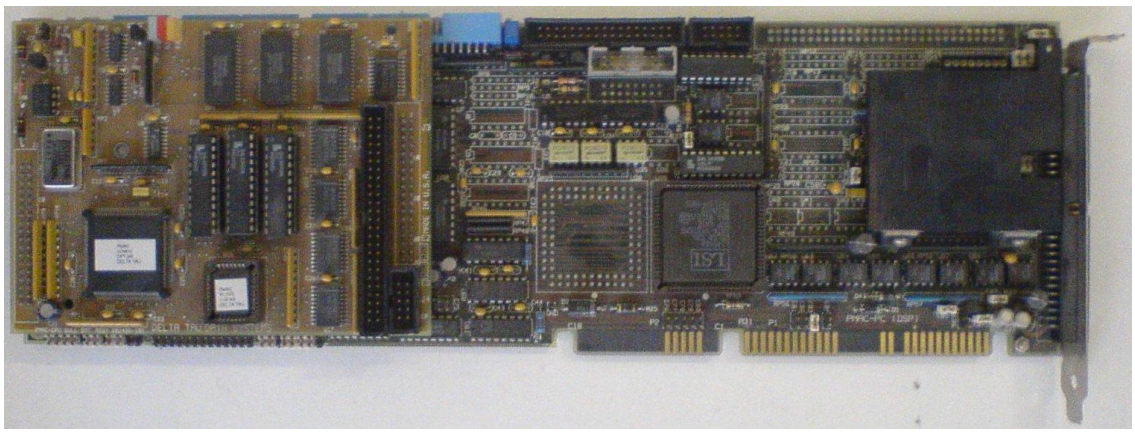
**Pohybový kontrolér**

Pohybový kontrolér je rovněž určitě rozumné řešení. Právě pohybový kontrolér je totiž přímo projektován na výpočty trajektorií, interpolací apod. a dokáže tedy tyto výpočty vypočíst nejrychlejším možným způsobem včetně dalších nejrůznějších potřebných korekcí. Jedná se většinou o kartu či sadu karet obdobných klasickým PC kartám, ke kterým je potřeba dodat napájecí zdroj. S PC je propojen běžnou komunikační linkou čí sběrnici.

### 4.1.1 Pohybový kontrolér PMAC PC

PMAC je zkratkou pro Programovací multi-osý kontrolér (Programmaable Mutli-Axis Controller2). Je optimalizován jako interface pro tradiční servořízení s jednotlivými analogovými vstupy reprezentující rychlost nebo kroutící moment. Software je schopný řídit až 8 os.

PMAC PC (*viz. Obr. č. 9*) je rozšiřující full size ISA deska s menší přídavnou deskou obsahující procesorovou jednotku.



*Obr č. 9 – foto pohybového kontroléru PMAC PC*

Na této přídavné desce běží digitální signálový procesor MOTOROLA DSP56002 s frekvencí 20 MHz. Mimo jiné ještě obsahuje část rozšiřujících konektorů. Celek (obě desky) pracují s dvacetičtyřmi 128k statickými paměťmi tvořící zálohu RAM. Ty jsou zálohovány baterii. Firmware je uložen v osmi 128k PROM pamětěch. Každý kanál pro řízení

osy ještě dále obsahuje 16ti bitový analogový výstup ±10V, 3 kanálový diferenciální dekodér vstupů, čtyři vstupní a dvě výstupní značky. S celkem je možné komunikovat jak přes ISA sběrnici, obsahuje-li PC rozšiřující ISA slot, nebo přes sériové rozhraní RS232/422.

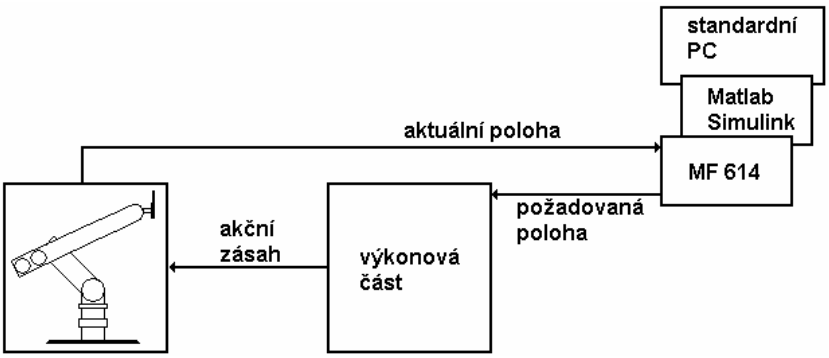
**Výkonová část**

Výkonová část je však úskalím. Vzhledem k nevyhovujícím skladovacím podmínkám původní řídící skříně si troufám i říci, že původní transformátory jsou až životu nebezpečné. Nehledě na to, že by bylo potřeba nově realizovat výkonovou část přímo pro naše potřeby,. Nejrozumnějším řešením by bylo přímo koupit novou kompletní sériově vyráběnou výkonovou část či si jí od profesionální firmy nechat navrhnout a rovněž od nich i dodat. Při pátrání po komerčním řešením jsem byl neúspěšný. Zbývá tedy varianta návrhu specializovanou firmou.

Výhodou tohoto řešení je síla v řízení přes PMAC. Nevýhodou je absence výkonové části. Tuto variantu tedy zamítám.

## 4.2 PC – karta HUMUSOFT MF614 – SW MATLAB SIMULINK – výkonová část

Základní myšlenka je stejná jako v předchozím případě – PC, pohybový kontrolér, silová část (*viz. Obr. č. 10*). Rozdíl v tomto řešení je, že jsem místo pohybového kontroléru PMAC fy. DELTA TAU chtěl použít I/O kartu MF614 fy. HUMUSOFT a pohybový kontrolér softwarově realizovat v MATLAB SIMULINKu. Karta MF614 má za úkol zpracovávat výstupy IRC čidel robotu a převádět tuto informaci o poloze prostřednictvím svého TOOLBOXu do prostředí MATLAB.

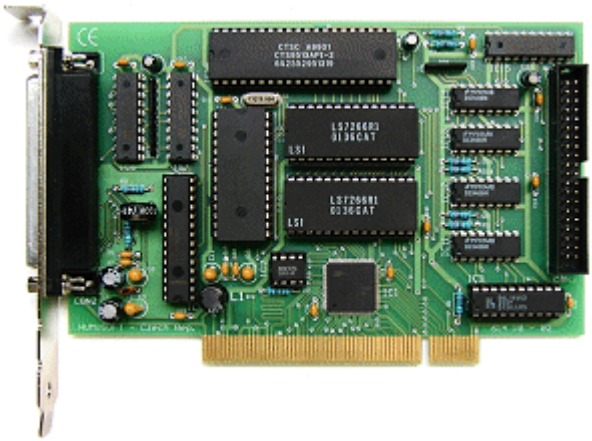



*Obr. č. 10 – blokové schéma návrhu nového řízení „PC – karta HUMUSOFT MF614 – SW MATLAB SIMULINK — výkonová část“ včetně obecných signálů*

Kartu MF614 obsluhuji v prostředí MATLAB SIMULINK. Ke kartě je dodáván REAL TIME TOOLBOX do MATLABu. Po aplikaci tohoto RT TOOLBOXu bych měl získat z MATLAB SIMULINKu, který provozuji na Windows 2000 „systém reálného času – tvz. realtime“.

### 4.2.1 karta HUMUSOFT MF614

Jedná se o multifunkční vstupně/výstupní PCI kartu (*viz. Obr. č. 11)* využitelnou jak v laboratořích, tak i v průmyslových podmínkách. Karta je vyráběna českou firmou HUMUSOFT s.r.o specializující se právě na výrobu měřících karet pro PC.



*Obr. č. 11 - multifunkční vstupně výstupní PCI karta MF614 fy. HUMUSOFT s.r.o*

Obsahuje osm 12ti bitových analogových vstupů, čtyři 12ti bitové analogové výstupy, dosahuje vzorkovací frekvence až do 100kHz. Dále nabízí 8 digitálních vstupů a výstupů, programovatelné vstupní rozsahy A/D převodníku, 4 vstupy inkrementálních snímačů, 5 čítačů/časovačů. Vyniká nízkou spotřebou.

Ke kartě jsou dodávány ovladače pro REAL TIME TOOLBOX, REAL TIME WINDOWS TARGET, xPC TARGET a WINDOWS.

Právě zmiňovaný REAL TIME TOOLBOX, který se zakomponuje do MATLABu, včetně popisované karty je mi již znám a proto bych zde rád tuto znalost využil.

**MATLAB SIMULINK**

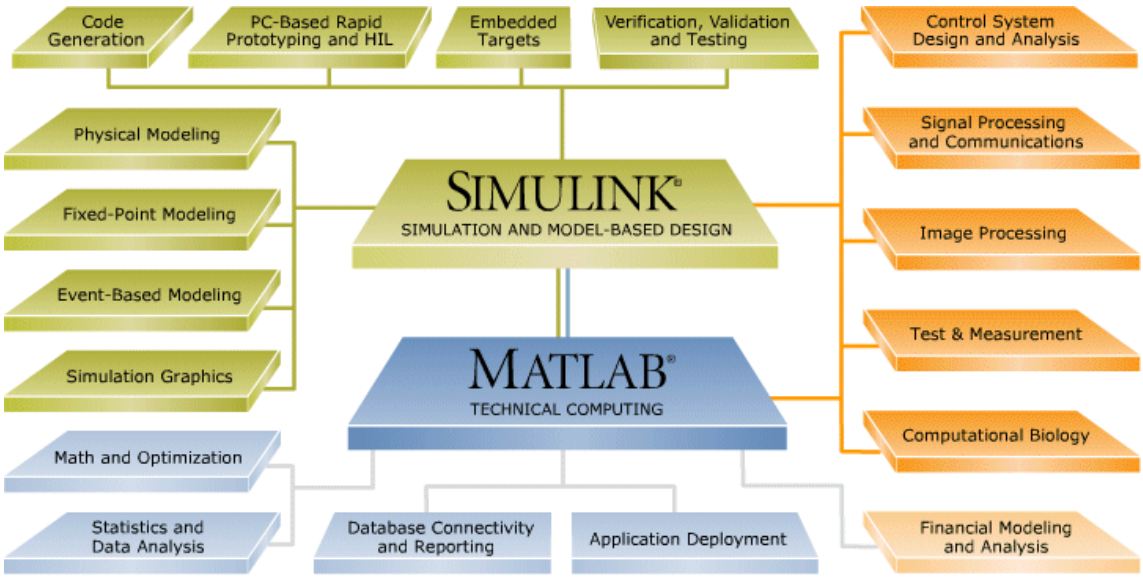
MATLAB je integrované prostředí pro technické výpočty, modelování a simulace, měření a testování, řídící techniku, zpracování signálů, komunikaci, zpracování obrazu a videa, vizualizaci dat apod. Otevřená architektura MATLABu vedla ke vzniku knihoven funkcí, nazývaných TOOLBOXy, které rozšiřují použití programu v příslušných vědních a

technických oborech. Tyto knihovny, navržené a v jazyce MATLABu napsané nejvýznačnějšími světovými odborníky, nabízejí předzpracované specializované funkce, které je možno rozšiřovat, modifikovat, anebo jen čerpat informace z přehledně dokumentovaných algoritmů.

SIMULINK je nadstavba MATLABu pro simulaci a modelování dynamických systémů, který využívá algoritmy MATLABu pro numerické řešení nelineárních diferenciálních rovnic. Poskytuje uživateli možnost rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat a rovnic.

Poslední verze tohoto SW je R2007a.

Strukturu rozsáhlého SW MATLAB pěkně reprezentuje schéma na *Obr. č. 12*




*Obr. č. 12 – blokové schéma reprezentující strukturu SW MATLAB*

Hlavním nedostatkem tohoto řešení je v nemožnosti použítí karty MF614 jako pohybového kontroléru, jak jsem původně předpokládal. Na základě doporučení pana Ing. Píši, je potřeba aby regulace běžela s periodou cca. 1ms, což je dáno vlastnostmi motorů a požadavky bezpečnosti. Ovšem při testování se nepodařilo vytvořit ani stabilní obdélníkový průběh s periodou 20ms, což by karta podle vyjádření firmy HUMUSOFT měla umět. Další možností by bylo použití WINDOWS-TARGET, což je „tvrdší“ real-time, který by měl již požadavkům na generování signálu vyhovět. Tuto skutečnost jsem však zjistil dosti pozdě, tudíž osvojit si znalost WINDOWS-TARGET již nebyla časově možná. Rovněž absence výkonové částí je dalším úskalím. Tuto variantu zavrhuji.

## 4.3 PC – MARS 8b

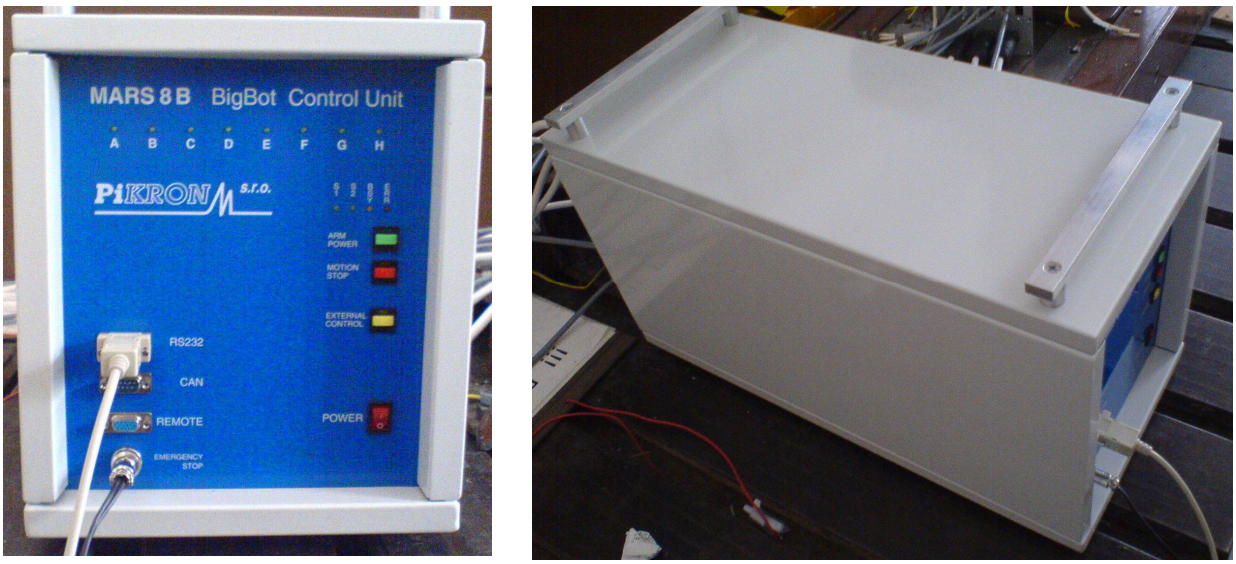
Zcela nejjednodušší řízení je jen s pomocí PC a MARS 8b (*viz Obr. č. 13*).


*Obr. č. 13 – blokové schéma nového návrhu řízení „PC – MARS 8b“ včetně obecných signálů*

Tato varianta řízení využívá plně možnosti řídící jednotky MARS 8b, tedy její výkonovou i řídící část. PC zde představuje jen interface pro MARS 8b, se kterým je spojen přes sériovou linku RS232. Tato linka neoplývá žádnou závratnou rychlostí, ale pro mé potřeby plně postačuje.

### 4.3.1 Řídící jednotka PIKRON MARS 8b

Řídící jednotka MARS 8b (*viz Obr. č. 14*) pražské firmy PIKRON s.r.o. je zakázkově či spíše nízko-sériově vyráběné zařízení.



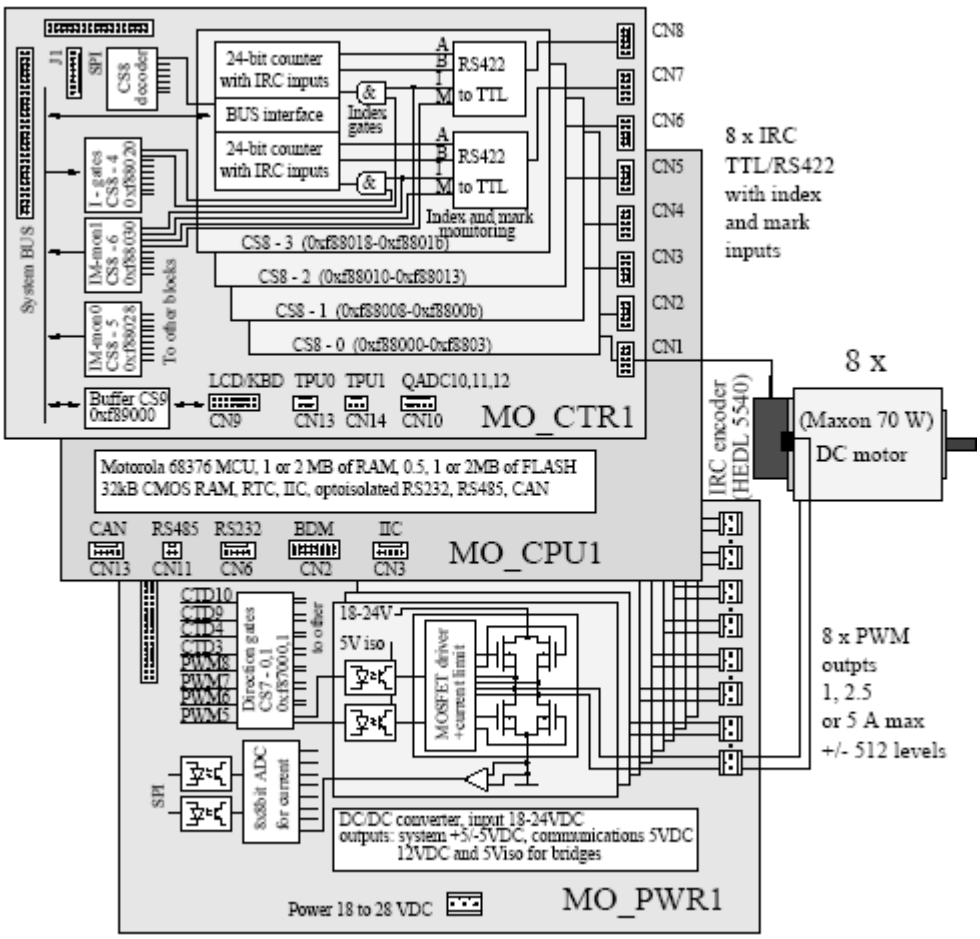

*Obr. č. 14 – pohledy na jednotku MARS 8b firmy PIKRON*

### 4.3.1.1 HW popis jednotky MARS 8b

MARS 8b je moderně řešená elektronická řídící jednotka, integrující v sobě část řídící i výkonovou. Je určena pro regulaci polohy až osmi stejnosměrných motorů. Pro zjišťování jejich aktuální polohy využívá inkrementálních čidel a jedné indexové značky na otáčku motoru. Pomocí elektromechanických koncových přepínačů zapojených do výkonových větví motorů nebo čidel limitní polohy s logickými vstupy se dá snadno vymezit pracovní prostor robotu s ohledem jak na bezpečnost robotu samotného, tak i jeho okolí.

Systém je tvořen několika základními částmi a to MO_CPU2, MO_CTR3 , MO_PWR4, MO_ARM1, MO_BRI5, MO_LED2, MO_ZDR2 a MO_CON1. Číslo za jménem každé části odpovídá jeho verzi. Blokové schéma skládající se z těchto částí je v *Příloha č. 5*.

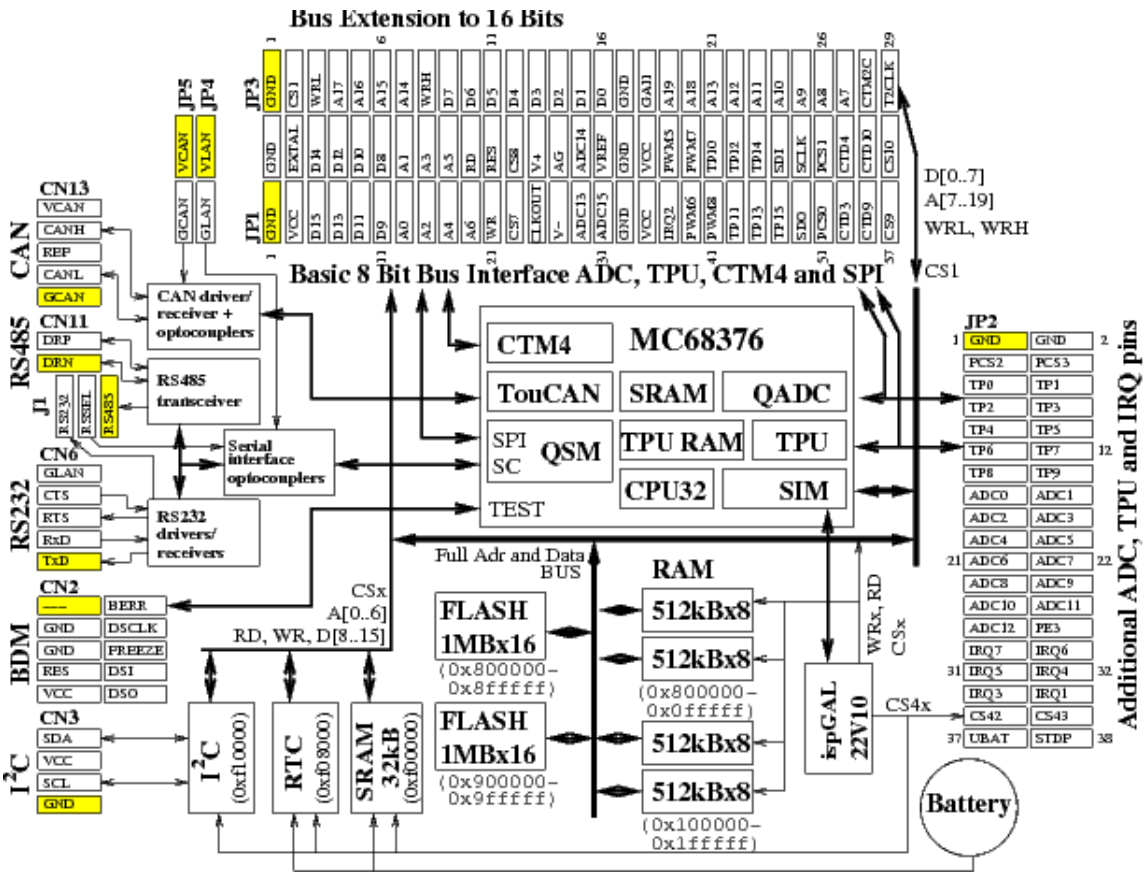
Z hlediska řízení jsou nejdůležitější části MO_CPU, MO_CTR a MO_PWR. Pohled na všechny tři desky je na *Obr. č. 15 (originální velikost obrázku je v Příloze č. 4)* .




*Obr. č. 15 – elektronická část jednotky MARS 8b tvořená třemi deskami*

Mikroprocesorová deska MO_CPU2 (*viz obr. č. 16, originální velikost obrázku je v Příloze č. 3*) je nástupce předchozí verze MO_CPU1. Tato deska je navržena přímo firmou PIKRON pro řízení laboratorních zařízení. Deska je osazena mikrokontrolerem Motorola MC68376 s 32-bitovým jádrem, který disponuje 16ti bitovou datovou a 24 bitovou adresovou sběrnicí. Integruje v sobě např. řadič sběrnice CAN (TouCAN), 16ti kanálový 10ti bitový A/D převodník (QADC), 16ti kanálový časovací koprocesor (TPU). Dále až 8 PWM výstupů, řadič SPI rozhraní s možností až 16 automatických přenosů (QSM) a asynchronní sériové rozhraní UART. Mikrokontroler navíc ještě obsahuje bloky RAM, a to velikosti 4kB a 3,5kB.

Samotná deska pak ještě rozšiřuje možnosti mikrokontroleru o IIC rozhraní, galvanické oddělení pro sběrnici CAN a opticky oddělené budiče RS232/485. 1 nebo 2 MB FLASH paměti v 16ti bitové konfiguraci slouží pro uložení zavaděče. Data a kódy mohou být uloženy ve rychlé SRAM paměti, ta má velikost až 2MB. Do FLASH nebo 32kB CMOS SRAM paměti zálohované baterii je možno dále uložit data, která se uchovají i přes vypnutí systému. Další doplněk systému tvoří obvod reálného času.



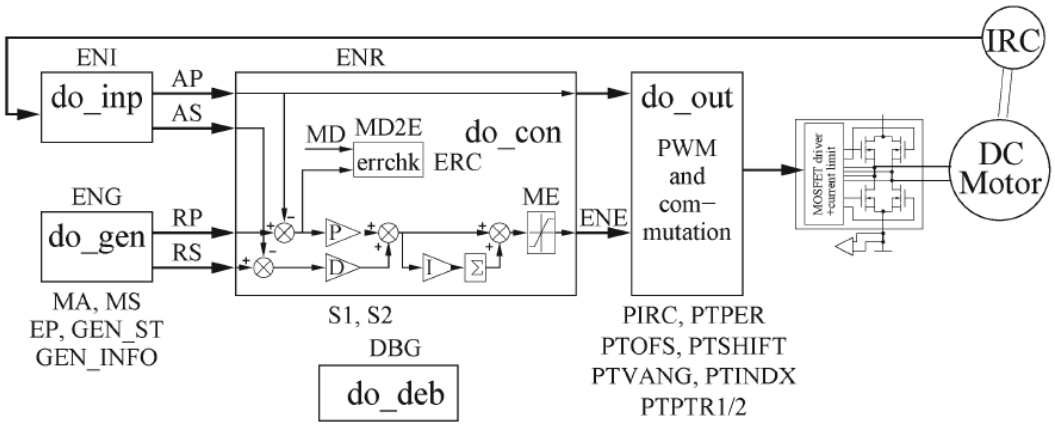

*obr. č. 16 – blokové schéma procesorové desky MO_CPU2*

Deska MO_PWR4 spolu s MO_CTR4 slouží pro řízení manipulátorů a polohovacích systémů. MO_CTR4 disponuje dekodéry pro 8 IRC vstupů, které pak výkonově obsluhuje MO_PWR4. Kromě přijímačů diferenciálních signálů normy RS422S spolu s dekodéry jsou umístěny i samostatné dekodéry s 24 bitovými čítači. Budiče PWM výkonu jsou řízený přímo výstupy z rychlé časovací jednotky CTM4, která je implementována v mikrokontroleru.

### 4.3.1.2 SW popis jednotky MARS 8b

Základem je klasický program v jazyku C, který je zkompilován kompilátorem GNU COMPILER COLLECTION (GCC) přímo na procesor typu MOTOROLA. Tento kompilátor je otevřeně vyvíjen a zároveň je dostupný na mnoho jiných cílových i hostitelských architektur. Program se do jednotky importuje přes servisní vstupy (sériová linka) a ladění je prováděno je prováděno na rozhraní GDB (GNU Debugger – standardní nástroj na hledání chyb v GNU software). K obojímu se dostanu po odejmutí levého bočnímu krytu (při pohledu z předu).

Systém víceosého řízení PXMC má následující blokové schéma (*viz. Obr. č. 17*).




*Obr. č. 17 – blokové schéma víceosého systému řízení PXMC*

Tento systém je multiplatformní, tj. je jednoduše přenositelný na jiný procesor či operační systém. Slouží k řízení stejnosměrných motorů na základě údajů z IRC čidel. Motory řídí pomocí PWM. Následuje popis jednotlivých bloků systému:

do_inp - získává aktuální hodnoty z IRC čidla (rychlost a poloha)
AP – okamžitá poloha, AS – okamžitá rychlost

do_gen - zdroj referenčních signálů
RP – požadovaná poloha, RS – požadovaná rychlost
MA – max. zrychlení, MS – max. rychlost, EP – konečná poloha
GEN_ST – stat. generátoru, GEN_INFO – prostor pro výpočet trajektorie

do_con - PID regulátor
MD – max. diference (max. dovolená změna pozice), errch – správa chyb, S 1, 2 – pomocné řídící konstanty

ENE - hodnota akčního zásahu – velikost napětí, proudu či obecně energie

do_out - výstup regulátoru – akční člen (generování PWM signálu, komutace)
PIRC, PTPER, PTOFS. PTSHIFT, PTVANG, PTINDX, PTPTRII/2 - signály pro vícefázové řízní (např. krokové motory) – nevyužívají se

de_deb - podpora ladění

IRC - IRC čidlo

DC Motor - stejnosměrný motor

Zdrojem informace o poloze a rychlosti robotu je IRC čidlo, které je umístěné na každém jednotlivém řízeném motoru. Blok *do_imp* tyto informace dekóduje a rozdělí na dva samostatné signály. Tyto signály jsou pak porovnány s požadovanou rychlostí *RS* a polohou *RP*. Rozdíly požadovaných a aktuálních hodnot jsou přivedeny do PID regulátoru, který generuje potřebný akční zásah. Akční zásah je pak ještě před výstupem z bloku omezen hodnotou *ME*, která představuje maximální povolený akční zásah. V bloku *do-out* je převeden na odpovídající PWM signál, který je již zaslán přímo na jednotlivé motory.

Polohová regulace a výpočty požadované polohy probíhají s frekvencí 1 kHz. Každá osa se řídí informacemi uloženými v její stavové struktuře. Ta obsahuje mimo jiné ukazatele na funkce, které obstarávají načtení aktuální polohy z HW, výpočet regulačního zásahu, výstup zásahu na výkonové výstupy, monitorování polohy pro účely ladění a přípravu nové požadované polohy.

### 4.3.1.3 Komunikace jednotky s okolím

Komunikace s okolím patří mezi silnou stránku této jednotky. Ať už se bavím o vstupech/výstupech měřících či řídících.

### Měřící a pomocné I/O

8 samostatných měřících vstupů slouží pro připojení IRC čidel, díky nimž jednotka vypočítává a určuje skutečnou polohu natočení motorů a jejich rychlost. Všechny konektory jsou stejného typu CANNON s 15 piny. Jednotka je navíc osazena konektory pro komunikaci s kartou HUMUSOFT MF 614, resp. dvěma kartami. K dispozici jsou vždy dva konektory rovněž typu CANNON se 37 piny pro jednu kartu. Vnitřně to je uspořádáno tak, že vstupy z konektorů CANNON s 15 piny jsou přivedeny rovněž na konektory CANNON se 37 piny.

Dále jsou vyvedený dva pomocné 7 pinové konektory. Na nich je vyvedeno napětí pro obsluhu koncových spínačů (tzv. dorazů) a jedno pomocné napětí 24V pro napájení žárovky na robotu. Toto napětí se sepne spolu se zapnutím výkonů na jednotce a funguje zároveň jako kontrolka přímo na tělu robotu.

### Řídící a komunikační I/O

Jednotku je možné řídit hned z několika zdrojů. Základem je připojení lokální klávesnice. O něco více komfortu přináší řízení přes RS232/485. Po připojení jednotky k PC přes RS232 slouží PC jako ovládací interface pro jednotku. Základní SW pro navázání komunikace a následné posílání příkazů do jednotky je HyperTerminal , jenž je běžně v základní instalaci OS MS Windows (ve Windows 2000 se nachází pod Start/Programy/Příslušenství/Komunikace). Před navázáním komunikace je potřeba nastavit komunikační port na tyto parametry:

Bity za sekundu: 9600 b/s
Datové bity: 8 bitů
Parita: žádná
Počet stop-bitů: 1
Řízení toku: Hardware

Poté je možno již přímo zadávat příkazy pro ovládání jednotky. Ty mají strukturu, skládající se ze jména, operačního znaku a parametru, např. *viz Obr. č. 18*. Jednotlivé části mohou být odděleny mezerami.

| Jméno | Op | Parametry |
|---|---|---|
| Gm | : | xxx.xxx |
| | | -8000,8000 |

*Obr. č. 18 – příklad příkazu – najede motorem „m“ na absolutní polohu „xxx.xxx“*

**Jméno**

Libovolná kombinace písmen a číslic začínající písmenem. V případě příkazů vztahujících se k jednotlivým motorům je jméno na konci doplněno o znak motoru ( A, B, C až H) dále označený 'm'. Příkazy vstupů a výstupů obsahují číslo skupiny 'n', případně číslo trigeru 't' nebo komparátoru 'c'.

**Operační znak**

Definuje jestli se jedná o příkaz ( znak ':' ) nebo dotaz ( znak '?' ). Pro potvrzování může být použit znak ".

**Parametr**

Jeho význam je daný příkazem. Dále bude 'xxx.xxx' označováno desetinné číslo, 'xxx' celé číslo a 'x' číslice. Záporná čísla začínají znakem '-'.

**Základní seznam příkazů**

Příkazy pro najíždění na polohu a referenční značku : Gm, GRm, APm, HHm, SETAPm

Příkazy pro řízení koordinovaných pohybů : COORDGRP, COORDMV, COORDMVT

Nastavení parametrů regulátorů : REGPm, REGIm, REGDm, REGS1m, REGS2m, REGMDm,REGMSm, REGACCm, REGMEm, REGCFGm, REGDBGm, REGSFRQ

Podpora ladění odezvy regulátorů : REGDBGm, REGDBGHIS, REGDBGPRE, REGDBGGNR

Nulování, zastavení a odpojení regulátorů : STOPm, STOP, PURGE, CLEARm, CLEAR, RELEASEm, RELEASE

Potvrzování přijetí a dokončení příkazů : R, Rm,

Status systému : STm, ST, STBSYBITS

Přímé řízení rychlosti : SPDm, SPDTm

Přímé řízení vstupů a výstupů : PWMm, DIGO, DIGOn, DIGI, DIGIn, DIGM, DIGMn, TRIGt, CMPc, CMPREPOc, ADCx

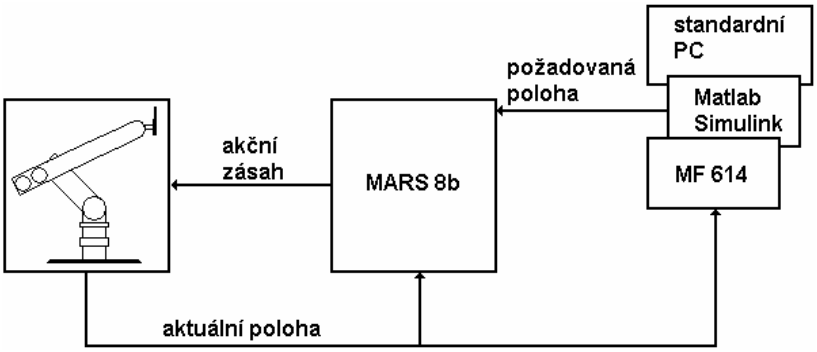
Systémové příkazy :VER, STAMP, ERRSTOP, IDLEREL, REGGOFLG, ECHO, RS323BAUD

Více stránky www.pikron.cz a přiložené CD

Dále je možné s jednotkou komunikovat přes sběrnici CAN a nebo jí řídit přes analogový vstup joystickem.

## 4.4 PC – SW MATLAB SIMULINK – MARS 8b - karta HUMUSOFT MF614

Toto je řešení, pro které jsem se ve výsledku rozhodl (*viz Obr. č. 19*).




*Obr. č. 19 – blokové schéma realizovaného řízení „PC – SW MATLAB SIMULINK – MARS 8b – karta HUMUSOFT MF614“ včetně obecných signálů*

Opět klasická konvence PC, MATLAB, MARS 8b rozšířená o kartu HUMUSOFT MF614 a SW MATLAB o SIMULINK. Již dříve jsem odůvodnil, že kartu nemohu použít pro řízení. Ale lze s ní jednoduše realizovat monitoring tj. sledování aktuální nebo-li skutečné polohy robotu. Podrobnější rozbor řešení monitoringu rozeberu v kapitole *5.5 SW MATLAB VIRTUAL REALITY TOOLBOX a REAL TIME TOOLBOX pro monitorování reálného pohybu robotu.*

# 5 Řízení robotu v prostoru

Řízení robotu v prostoru je obtížná úloha ať už z hlediska výpočtu či jen pouhé představy. Ne vždy má totiž jednoznačné řešení, tj. může mít i více řešení. Další problém nastává s technickým omezením robotu, kterému ani jedno z najitých řešení nemusí vyhovovat. Základní dva postupy sloužící k vyřešení kinematiky robotu v prostoru jsou

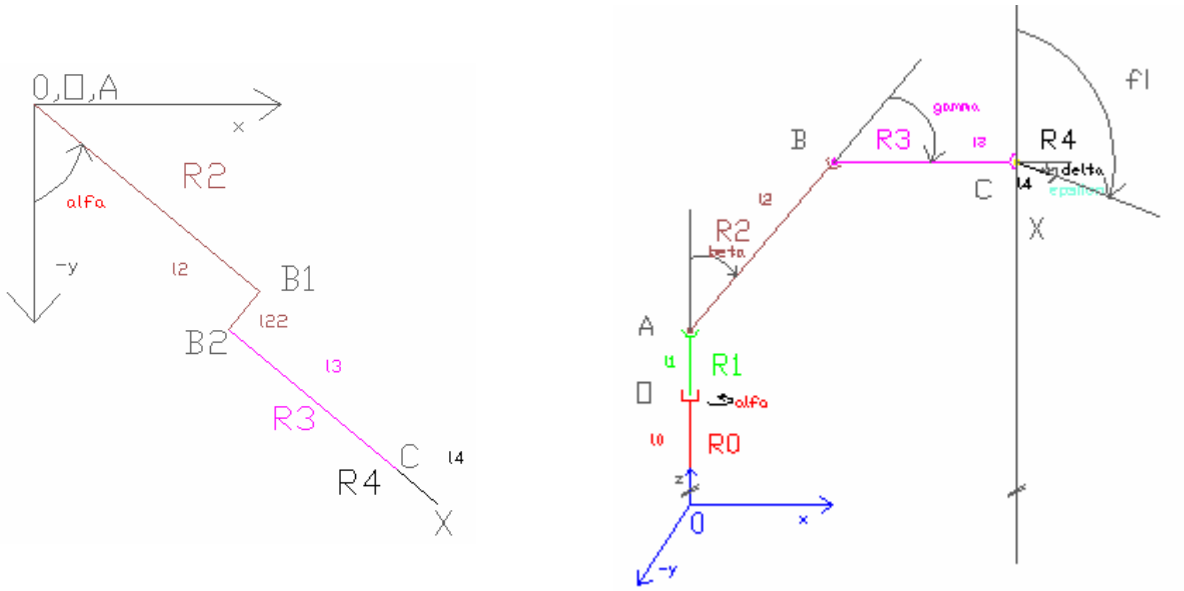
a) Přímá kinematická úloha - DKT
b) Inverzní kinematická úloha - IKT

## *5.1 DKT - Přímá kinematická úloha*

DKT vychází z toho, že znám parametry robotu (délky ramen, typy kloubů) a u každého kloubu jeho natočení. Nevím však koncový bod ramene. Na základě těchto znalostí mohu metodou postupné transformace matematicky popsat celé rameno robotu a získat tak hledaný koncový bod. Tato metoda je pro praxi jen málo využitelná. Jelikož mě právě zajímá, jak se ramena robotu „poskládají“ při znalosti koncového bodu. DKT často používám pouze pro kontrolu správnosti řešení pomocí IKT. V realizaci se jedná o soustavu matic, u kterých se vždy mění pouze velikosti úhlů natočení a případný posun ramene v jednotlivých osách. Výsledný bod je dán součinem těchto dílčích matic. Zde bych rád vyšel z práce kolegy Václava Sedláčka a pouze použil jím odvozené řešení ve stručnější verzi:

| **Rameno** | **Označení délky ramena** | **Délka ramena / m** |
|---|---|---|
| R0 | $l_0$ | 0,26 |
| R1 | $l_1$ | 0,46 |
| R2 | $l_2;l_{22}$ | 0,7;0,187 |
| R3 | $l_3$ | 0,8 |
| R4 | $l_4$ | 0,15 |

*Tab. č. 2 – značení a parametry ramen*




*Obr. č. 20 – dva pohledy na schématický popis kinematiky robotu*

Matice $\mathbf{A_1}$ otáčí celý manipulátor o úhel $\alpha$ rotace je kolem osy z

$$A_1(a) = \begin{pmatrix} \cos a & -\sin a & 0 & 0 \\ \sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_{12}}$ provede posun ve směru osy z o vzdálenost $l_0+l_1$ do kloubu A

$$A_{12} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_0 + l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_2}$ rotuje o úhel $\beta$ kolem osy y tím dojde k otočení ramenem R2

$$A_2(b) = \begin{pmatrix} \cos b & 0 & \sin b & 0 \\ 0 & 1 & 0 & 0 \\ -\sin b & 0 & \cos b & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_{22}}$ provádí posun z bodu A do bodu B1 ve směru osy z o vzdálenost $l_2$.

$$A_{22} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_{23}}$ provede přesun z bodu B1 do B2 ve směru osy y o vzdálenost $l_{22}$.

$$A_{23} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -l_{22} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_3}$ rotuje ramenem R3 o úhel $\gamma$ kolem osy y.

$$A_3(g) = \begin{pmatrix} \cos g & 0 & \sin g & 0 \\ 0 & 1 & 0 & 0 \\ -\sin g & 0 & \cos g & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_{32}}$ provádí posun o vzdálenost $l_3$ z bodu B2 do C.

$$A_{32} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_4}$ rotuje ramenem R4 o úhel $\delta$ kolem osy y.

$$A_4(d) = \begin{pmatrix} \cos d & 0 & \sin d & 0 \\ 0 & 1 & 0 & 0 \\ -\sin d & 0 & \cos d & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matice $\mathbf{A_5}$ provádí posun o vzdálenost $l_4$ z bodu C do X.

$$A_{42} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Celková transformační matice **A**

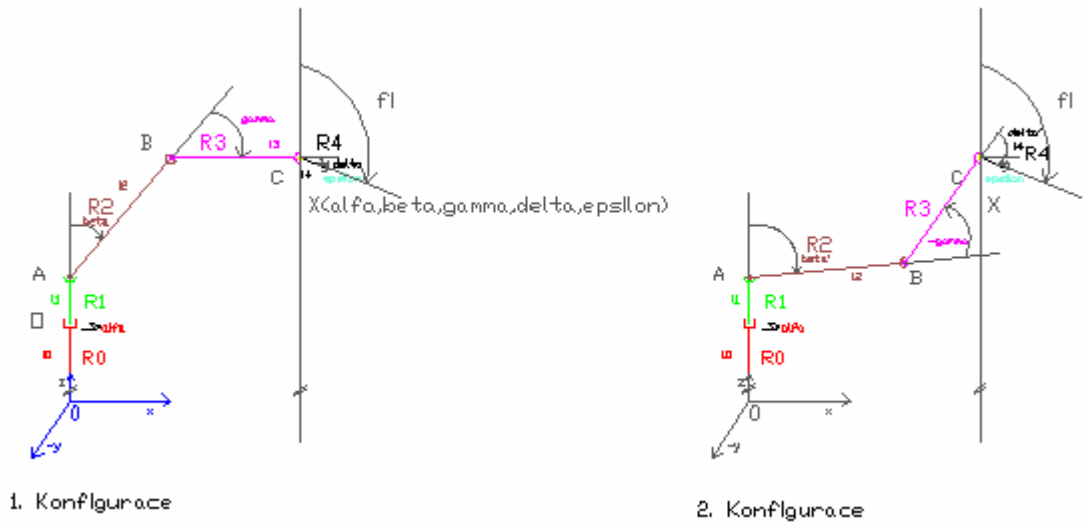
$$A(a,b,g,d) = A_1(a)\cdot A_{12}\cdot A_2(b)\cdot A_{22}\cdot A_{23}\cdot A_3(g)\cdot A_{32}\cdot A_4(d)\cdot A_{42}$$

Mou částí práce bylo vycházet z těchto odvození a realizovat na jejich základě program v MATLABu, jeho zdrojový kód uveden v příloze.

## 5.2 IKT - Inverzní kinematická úloha

Právě pomocí IKT se řeší většina úloh. Tj. znám koncový bod, parametry robotu a jeho ramen a umístění základny (počátku souřadného systému) robotu v prostoru. A zajímá mě, jaké bude vzájemné natočení (poloha) ramen robotu. Pro roboty s vyšším stupněm volnosti neexistuje žádné obecné univerzální řešení. Vždy se jedná a matematicko-geometrické řešení. Tzn., že hledám mezi jednotlivý rameny kružnice jejich dosahu či trojúhelníky a jejich vzájemné průniky. Pomocí nichž pak mohu vypočítat vzájemnou polohu ramen a určit uhly  natočení a posuny jednotlivých ramen . Tímto postupem často získám více kombinací ramen, jak se do mnou požadovaného koncového bodu dostat. Vzniká zde tedy otázka jaké řešení použít. To se většinou řeší právě pomocí DKT či nejrůznějším omezením rozsahů dovolených úhlů. Rád bych se zde opřel o práci kolegy Václava Sedláčka. Pro našeho robota vyřešil IKT pomocí matematicko-geometrické metody.

IKT úloha bude mít 2 konfigurace řešení, které jsou schématicky naznačeny na *Obr. č. 21*. Konfigurace řešení by mohly být 4 a matematicky by to tak vyšlo, ale díky posunu by se jednalo o jinou polohu chapadla (jiný úhel mezi chapadlem a osou y, úhel $\alpha$).




*Obr. č. 21 - 2 konfigurace řešení*

Nejprve provedu výpočet $\alpha$. Provedu kolmý průmět manipulátoru do roviny určenou osami souřadného systému x a *y viz Obr. č. 21*. V této rovině najdu vzdálenost |0X|, kterou potřebuji pro výpočet úhlu $\alpha_a$. Protože tento úhel je vždy ostrý můžu použít funkci arsin. Úhel $\alpha_b$ získám přímo z polohy bodu X užitím zobecněné funkce argtg2. Hledaný úhel $\alpha$ je součet úhlů $\alpha_a$ a $\alpha_b$. Matematicky zapsáno

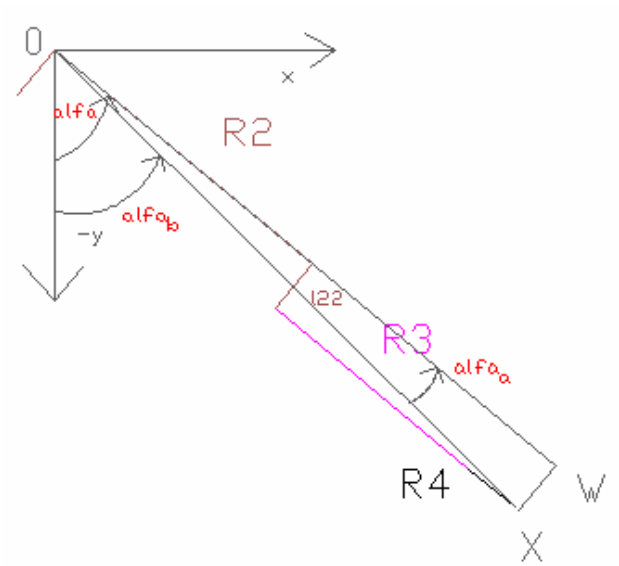
$$|0X| = \sqrt{x_1^2 + x_2^2}$$

$$a_a = ar\sin\left(\frac{l_{22}}{|0X|}\right)$$

$$a_b = \arg tg2(y;x)$$

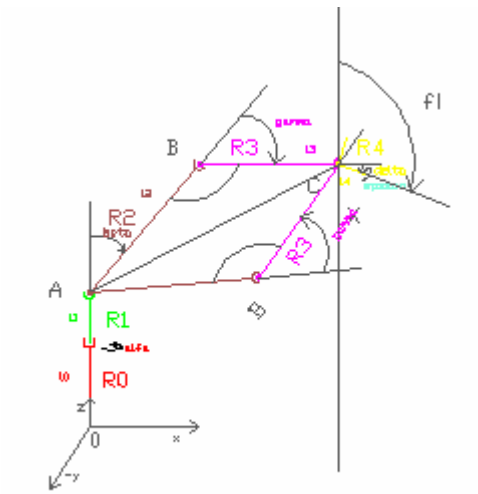
$$a = a_a + a_b$$

kde $x_1$,$x_2$ jsou souřadnice bodu X *viz Obr. č. 22*.




*Obr. č. 22 -* **v***ýpočet* $\alpha$

Další výpočty jsou prováděny v rovině určenou body OAW. V této rovině je manipulátor zobrazen na *Obr. č. 23*.




*Obr. č. 23 - zobrazení pro výpočty některých úhlů*

Nejdříve určím souřadnice X v této rovině a označím je $X_n$. Dále pak mohu určit body C a A. Vypočtu vzdálenost |AC|. Pro tuto vzdálenost platí, že musí být menší jak $l_2+l_3$. Pomocí kosinové věty mohu vypočítat úhel ABC. Úhel ABC využiji pro výpočet $\gamma$. Pro výpočet $\beta$ potřebuji úhly BAC a ACX. Úhel $\delta$ vypočtu jednoduše pomocí úhlů $\gamma$ a $\beta$. Úhel $\varepsilon$ je přímo zadaná hodnota. Formální matematický zápis

$$X_n = \left(\sqrt{|0X|^2 + l_{22}^2}\,;z\right)^T$$

$$C = X_n + l_4(\sin(-p+j);\cos(-p+j))^T$$

$$A = (0;l_0 + l_1)$$

$$|AC| = \sqrt{(c_1 - a_1)^2 + (c_2 - a_2)^2}$$

$$\angle ABC = ar\cos\left(\frac{|AC|^2 - l_3^2 - l_2^2}{-2.l_2 l_3}\right)$$

$$\angle BAC = ar\cos\left(\frac{l_3^2 - l_2^2 - |AC|^2}{-2l_2|AC|}\right)$$

$$\angle ACX = \arg tg2(c_2 - l_1 - l_0;c_1)$$

kde $c_1$, $c_2$, $a_1$, $a_2$ jsou souřadnice bodu C a A.

Výsledná obě řešení obě řešení $X_{inv1,2}$

$$X_{inv1} = (a_a + a_b;p/2 - (\angle BAC + \angle ACX);p - \angle ABC;j - (b_1 + g_1);e)$$

$$X_{inv2} = (a_a + a_b;p/2 + \angle BAC - \angle ACX;-p + \angle ABC;j - (b_2 + g_2);e)$$

kde $\beta_1$, $\gamma_1$ jsou příslušné hodnoty řešení $X_{inv1}$ a $\beta_2$, $\gamma_2$ příslušné hodnoty řešení $X_{inv2}$.

## *5.3 Realizace v řízení v MATLABu*

Ve výsledku jsem tedy pro řízení zvolil SW MATLAB firmy MATHWORKS. Výhoda tohoto SW je jednak školní licence, tudíž nejsem odkázán jen na práci ve škole a rovněž poměrně bohaté znalosti tohoto produktu z předchozích ročníků studia, kde byl využívám při výuce. Dalším přínosem je instalovaná řídící jednotka MARS obdobného typu ve škole, kde je řízena přes funkce napsané v MATLABu. Tudíž jsem měl pro začátek z čeho vycházet. Řídící jednotka spadá pod skupinu Počítačového vidění (CMP), odkud jsem si i jejich zdrojové kódy vypůjčil a použil podle potřeby.

### 5.3.1 Postup řízení přes MATLAB

Mám-li tedy vše správně propojené (PC přes RS232 s MARS 8b, kartu HUMUSOFT MF614 přes 37 žilový vodič s MARS 8b, výkony, pomocné signály a IRC čidla mezi robotem a MARS 8b, připojené bezpečnostní tlačítko na MARS 8b) mohu začít již se samotným řízením. Spustím aktuální verzi MATLABUu (používam MATLAB 2006b nebo-li MATLAB 7.3.0) a zvolím zdrojový adresář s kódy.

Základem je funkce *robotOJ10*. Tato funkce představuje strukturu robotu, tj. jeho parametry ramen, nastavení převodů, nastavení max. a min. rychlostí akcelerací, apod. Jako první spustím funkci *openOJ10(r)*, která *robotOJ10* spouští. To provedu příkazem *r=openOJ10( )*. Tato funkce obstarává navázání komunikace mezi jednotkou MARS 8b a PC, tj. vyresetování portu, navázání komunikace a povolení možnosti zapnout výkony na MARS 8b. Jednotka totiž obsahuje SW ochranu a to takovou, že výkonová část jednotky se sepne, resp. jde sepnout, až po povolení z obslužného SW běžící na řídícím PC. Po výzvě `'Press arm power button...'` tedy sepnu tlačítkem *ARM POWER* výkony na jednotce. V další pořadí spouštění je fce. *OJ10init(r)*. Opět vychází ze struktury *robotOJ10* a na základě ní nejdříve vyresetuje motory, nastaví rychlosti a akcelerace apod. Spustím jí napsáním příkazu *r=OJ10init(r)*. Tímto postupem tedy připravím robota k činnosti.

Samotný pohyb robota mohu řídit hned přes několik funkcí. Jednak mohu řídit každou osu zvlášť pomocí *move* , kde jako parametry jsou jednotlivé úhly natočení, což je vhodné spíše pro případy ladění. Dále můžeme nechat robota najet na mnou zadaný bod *NajedNaBod*, zde je parametrem souřadnice bodu, rychlost a diskontinuita. Jinou možností je nechat si projet přímku mezi dvěma body pomocí fce. *ZAdoBPoUsecce*. Parametrem je počáteční a

koncový bod. Tyto dvě poslední fce. obsahují řešení pohybu pomocí IKT. Její implementace v MATLABu následuje v další kapitole.

### 5.3.2 IKT v MATLABu

Mým úkolem tedy bylo toto řešení implementovat do MATLABUu. Nejjednodušší bude, když tu přímo použiji okomentovaný zdrojový kód.

```
function J=OJ10ikt(robot,X)
%Tato uloha resi inverzni kinematickou ulohu IKT
%Vstupem je X, coz je poloha koncoveho bodu a uhel pod jakym se ma na dany
%bod dostat
%Vystupem je J, coz jsou vzpoctene kloubove souradnice

%Predani parametru z inicializace
l1=robot.l1;
l0=robot.l0;
l2=robot.l2;
l22=robot.l22; %Posun na kloubech
l3=robot.l3;
l4=robot.l4;

%Spocitani ctyr reseni
J(:,:,1)=NaN;
J(:,:,2)=NaN;

for i=1:length(X(1,:)), %for cykl s opakovanim odpovidajici delce X

    %Nejdrive urcime alfa - uhel mezi osou y a spojnici stredu sour.
    %systemu a koncoveho bodu
    %Hledame ji v rovine x-y, tudiz dalsi vzdalenosti a vypocty budou i v
    %teto rovine

    dOX=(X(1,i)^2+X(2,i)^2)^(1/2); %Vypocet delky prepony pocatek souradnic
                                   % - koncovy bod

    %Vzdalenost pocatku a bodu W, ktery lezi na pruniku primky urcene
    %useckou ramene R2 pruniku primky rovnobezne s l22 vedenou bodem X

    dOW=(dOX^2-l22^2)^(1/2); %Vypocet delky odvesny pocatek souradnic –
                             %koncovy bod
    alfaa=asin(l22/dOX); %Urceni uhlu spojnice pocatek sour. sys a
                         %koncoveho bodu X (vnitrni uhel)
    alfab=atan2(X(2,i),X(1,i)); %Funckce v Matlabu vracejici uhel svirajici
                                %s bodem X
    alfa=real(alfaa+alfab); %Vysledny uhel odpovidajici natoceni robota
    J(1,i,1)=alfa; %Predani hodnoty

    %Dalsi vypocty budeme pocitat v rovine OAW
    %Nejdrive urcime souradnice bodu X v teto nove rovine
    newX=[dOW;X(3,i)];
    %Bod C jsme schopni vypocitat
    C=newX+l4*[sin(-(pi-X(4,i)));cos(-(pi-X(4,i)))];
    %Vypocet bodu A v techto novy souradnicich
```

```
    A=[0;(l0+l1)];
    %Vypocet vzdalenosti A a C
    dAC=((C(1,1)-A(1,1))^2+(C(2,1)-A(2,1))^2)^(1/2);

    if(dAC>l2+l3) %Pokud je vzdalenost bodu A a C vetsi jak l2+l3 tak
                  %neexistuje reseni
        J(:,i,:)=NaN; %Robot se na tyto souradnice nemuze dostat
        else %V tomto pripade ma reseni
            ABC=acos((dAC^2-l3^2-l2^2)/(-2*l2*l3)); %Vypocitame uhel mezi
                                                    %ABC pomoci kosinove vety
            %Uhel gamma je pak doplnkem do Pi
            gamma=real(pi-ABC);
            %Zatim tedy budeme uvazovat jedno reseni, az na konci dodelame
            %reseni vsechna, jelikoz se daji z tohoto jednoho vypocitat
            J(3,i,1)=gamma;
            %Dale urcime uhel BAC, potom ho pouzijime jako korekci pro
            %uhel, ktery tvori
            %usecka AC, ten zjistime pomoci fce atan2
            BAC=acos((l3^2-l2^2-dAC^2)/(-2*l2*dAC));
            ACx=atan2(C(2,1)-(l1+l0),C(1,1));
            beta=real(pi/2-(BAC+ACx));
            J(2,i,1)=beta;
            delta=real(X(4,i)-(beta+gamma));
            J(4,i,1)=delta;
            J(5,i,1)=real(X(5,i));
            %Doplneni vsech resenich
            J(1,i,2)=alfa;
            J(2,i,2)=beta+2*BAC;
            J(3,i,2)=-gamma;
            J(4,i,2)=X(4,i)-J(2,i,2)-J(3,i,2);
            J(5,i,2)=X(5,i);

    end
end
end
```

### 5.3.3 Popis jednotlivých funkcí

Jak už jsem se zmínil výše, některé funkce jsem převzal od skupiny CMP, jiné jsem sám vytvořil či upravil. Nyní bych zde udělal výčet funkcí, včetně jejich hrubého popisu. Funkce nebudu popisovat podle abecedy, ale tak, že nejdříve popíši ty základní a poté funkce složitější, které těch základních využívají. Přímo zdrojové kódy jsou pak přiloženy v příloze, kde jsou již řazeny abecedně.

**<u>Převzaté a upravené zdrojové kódy od CMP</u>**

**robotOJ10( )**

**robot=robotOJ10(robot)**

Představuje strukturu, která obsahuje základní inicializace robotu:

robot.portname - jméno komunikačního portu

robot.comlib - verze použitého SW

robot.timeout - čas pro kontrolu navázání komunikace

robot.l0-l4 - rozměry jednotlivých ramen dle rozměru z IKT v AutoCadu

robot.irc - počet IRC pulsu na otáčku jednotlivých motorů

robot.gearing - převody jednotlivých převodovek

robot.degtoirc - převod natočení ramen ze stupňů na IRC pulzy

robot.activemotors - seznam aktivních motoru

robot.bound - mezní hodnoty prac. rozsahu v uhlech (min./max.) - SW ochrana

robot.(min/max)speed - min. a max. rychlosti motoru (IRC/256/msec)

robot.idle - za jak dlouho se mají odstavit regulátory (s)

robot.(min/max)acceleration - min. a max. zrychlení

robot.defaultspeed - nastavení výchozí rychlosti

robot.defaultacceleration - nastavení výchozího zrychlení

robot.REGME - maximální výkon PWM (viz. Pikron)

robot.REGPWRON - povolení sepnutí výkonu (viz. Pikron)

robot.REGPWRFLG - povoluje řízení výkonových výstupů z generátorů PWM (viz. Pikron)

robot.REGCFG - průběh rychlostí při pohybu z jedné polohy do druhé a způsob nalezení nulové polohy (viz. Pikron)

robot.povolSWOchranu - obsluha SW ochrany, 1 - povolí, 0 - zakáže

robot.maxY - souřadnice Y

robot.minY1 - souřadnice Y

robot.zlom - souřadnice X kde přechází minY1 do minY2

robot.minY2 - souřadnice Y

robot.REGP - nastaveni P složky regulátoru (viz. Pikron)

robot.REGI - nastaveni I složky regulátoru(viz. Pikron)

robot.REGD - nastaveni D složky regulátoru(viz. Pikron)

**portclose( )**

**[error]=portclose(robot)**

Uzavře komunikační port a vypíše případné chyby

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **portclose( )** | robotOJ10( ) |

**portflush( )**

**[error]=portflush(robot)**

Ošetřeni chyb při zápisu na sériový port

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **portflush( )** | robotOJ10( ) |

**portopen( )**

**[r]=portopen(r)**

Otevření komunikačního portu a nastavení jeho parametrů

r - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **portopen( )** | robotOJ10( ) |

**portreadline( )**

**[line,error]=portreadline(robot)**

Čtení dat z komunikační linky jdoucí z řídící jednotky.

robot - struktura definující parametry robotu

line - přečtená data z komunikační linky

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **portreadline( )** | robotOJ10( ) |

**ports_reset( )**

**ports_reset**

Zresetuje komunikační porty

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **port_reset( )** | - |

**portsettimeout( )**

**[r]=portsettimeout(r,timeout)**

Nastaví max. čas pro čtení komunikačního portu

r - struktura definující parametry robotu

timout – max. doba pro čtení z komunikačního portu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **portsettimeout( )** | robotOJ10( ) |

**portwriteline( )**

**portwriteline(robot,line)**

Zapíše příkazy do řídící jednotky

line – požadovaný příkaz ve formátu STRING, např. 'CLEAR:'

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **portwriteline( )** | robotOJ10( ) |

**OJ10checkirc( )**

**[r]=OJ10checkirc(robot,uhly)**

Testuje zda-li jsou hodnoty pulzů IRC v nastaveném rozsahu.

Je-li v rozsahu r=1, jinak r=0

robot - struktura definující parametry robotu

uhly - vstupní uhly v IRC pulzech k ověření

r - je-li v rozsahu r=1, jinak r=0

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10checkirc( )** | robotOJ10( ) |

**OJ10irctodeg( )**

**b=OJ10irctodeg(robot, a)**

Převede hodnotu IRC pulzů na odpovídající hodnotu ve stupních

robot - struktura definující parametry robotu

a - vstupní matice s hodnotami IRC čidel

b - převedené hodnoty ve stupních

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10irctodeg( )** | robotOJ10( ) |

**degtoirc( )**

**b=degtoirc(robot,a)**

Převede úhly ve stupních na hodnotu pulsů IRC

robot - struktura definující parametry robotu

a - vstupní hodnoty velikosti uhlů ve stupních

b - výstupní hodnoty převedené na hodnotu pulzů IRC

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **degtoirc( )** | robotOJ10( ) |

**radtodeg( )**

**a=radtodeg(b)**

Převede radiány na stupně

b - hodnota úhlu v radiánech

a - vypočtená hodnota úhlu ve stupních

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **radtodeg( )** | - |

**degtorad( )**

**a=degtorad(b)**

Převede hodnoty uhlů ve stupních na hodnotu v radiánech

b - vstupní hodnoty uhlů ve stupních

a - výstupní hodnoty uhlů v radiánech

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **degtorad( )** | - |

**OJ10resetmotors( )**

**OJ10resetmotors(robot)**

Zastaví regulace motorů s chybou a chyby

vynuluje, ostatní motory pokračují v započaté činnosti

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10resetmotors( )** | robotOJ10( )<br>portwriteline( ) |

**OJ10setacceleration( )**

**OJ10setacceleration(robot, accels)**

Nastaví zrychlení pro používané motory

Hodnota zrychlení se použije pro příkazy najeti na polohu při lichoběžníkovém profilu rychlosti

Hodnota odpovídá přírůstku rychlosti za periodu vzorkováni

robot - struktura definující parametry robotu

accels - hodnoty akcelerace pro každý motor

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10setacceleration( )** | robotOJ10( )<br>portwriteline( ) |

**OJ10getirc( )**

**[angles]=OJ10getirc(robot)**

Zjistí všechny používané motory a u nich i hodnoty pulzů IRC čidel

robot - struktura definující parametry robotu

angles - výstupní matice s aktuálními hodnotami IRC čidel

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10getirc( )** | robotOJ10( )<br>portwriteline( )<br>portreadline( ) |

**OJ10clear( )**

**[irc]=OJ10clear(robot)**

Vypnutí řízení motorů a vynulování odečtu polohy a koncové požadované polohy pohybu

Nastaví aktuální hodnoty IRC

robot - struktura definující parametry robotu

irc - předání parametru ze struktury robot

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10clear( )** | robotOJ10( )<br>portwriteline( )<br>OJ10getirc( ) |

**OJ10isready( )**

**[stat]=OJ10isready(robot)**

Ověří, že už se robot dále nebude hýbat a je připravený pro přijetí nových příkazů

robot - struktura definující parametry robotu

stat - 1 .. robot je připraven, 0 .. poslední příkaz se ještě vykonává

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10isready( )** | robotOJ10( )<br>portwriteline( )<br>portreadline( ) |

**OJ10setspeed( )**

**OJ10setspeed(robot, speeds)**

Nastaví rychlost pro každý motor

robot - struktura definující parametry robotu

speeds - nastavení rychlosti pro každou osu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10setspeed( )** | robotOJ10( )<br>portwriteline( ) |

**OJ10close( )**

**OJ10close(robot)**

Provede softhome

Zresetuje motory

Vypne výkony

Zavře port

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10close( )** | robotOJ10( )<br>OJ10softhome( )<br>OJ10resetmotors( )<br>portwriteline( )<br>portclose( ) |

**OJ10init( )**

**[robot]=OJ10init(robot)**

Inicializace robotu s hodnotami ze struktury 'robot'

Zresetuje motory a čeká na dokončení příkazů

Nastaví rychlost, zrychleni, povolí zapnutí výkonů

Omezuje maximální plnění výstupu PWM do motoru

Nastaví hodnoty PID regulátoru

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10init( )** | robotOJ10( )<br>OJ10resetmotors( )<br>OJ10isready( )<br>OJ10waitforready( )<br>OJ10setspeed( )<br>OJ10setaceleration( )<br>portwriteline( ) |

**OJ10moveirc( )**

**OJ10moveirc(robot,irc)**

Ověří zda-li je hodnota IRC pulzů v povoleném rozsahu hodnot IRC pulzu

Najede motory na absolutní hodnoty IRC pulzu

robot - struktura definující parametry robotu

irc - absolutní hodnota v IRC pulzech

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10moveirc( )** | robotOJ10( )<br>portwriteline( ) |

**OJ10moveircs( )**

**OJ10moveircs(r,a,t,disc)**

Pohyb robota do všech pozic 'a' je pomoci koordinovaných pohybu

r - struktura definující parametry robotu

a - pozice ve stupních

t - minimální časový interval mezi dvěma následujícími pozicemi v milisekundách

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10moveircs( )** | robotOJ10( )<br>OJ10waitforready( )<br>portsettimeout() |

**OJ10waitforready( )**

**OJ10waitforready(robot)**

Čeká dokud robot není připraven na další příkazy

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10waitforready( )** | robotOJ10( )<br>portwriteline( )<br>portsettimeout( )<br>portreadline( )<br>OJ10isready( ) |

**openOJ10( )**

**[robot]=openOJ10**

Nastaví porty a naváže komunikaci s řídící jednotkou a povolí zapnutí výkonů

robot - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **openOJ10( )** | robotOJ10( )<br>portopen( )<br>portsettimeout( )<br>portwriteline( )<br>portflush( ) |

## Vytvořené zdrojové kódy

**OJ10ikt( )**

**J=OJ10ikt(robot,X)**

Řeší inverzní kinematickou úlohu IKT

robot - struktura definující parametry robotu

X - poloha koncového bodu a úhel pod jakým se má na daný bod dostat

J - vypočtené kloubové souřadnice

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10ikt( )** | robotOJ10( ) |

**swOmezeni( )**

**r=swOmezeni(robot,X)**

Pro zadaný bod X rozhodne jestli jr možné na něj najet či ne

robot - struktura definující parametry robotu

X - souřadnice bodu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **swOmezeni( )** | robotOJ10( ) |

**zakazDoraz( )**

**zakazDoraz(r)**

Povolí řízení výkonových výstupů z generátorů PWM

Ruší vypnutí výstupu způsobené pohybem robotu mimo vymezenou pracovní oblast

Používat jen po nezbytně nutnou dobu při automatickém návratu do pracovní oblasti

r - struktura definující parametry robota

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **zakazDoraz( )** | robotOJ10( )<br>portwriteline( ) |

**povolDoraz( )**

**povolDoraz(r)**

Povoluje řízení výkonových výstupů z generátorů PWN a reaguje na dorazová čidla

r - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **povolDoraz( )** | robotOJ10( )<br>portwriteline( ) |

**move( )**

**move(robot,uhly)**

Slouží pro pohyb po jednotlivých motorech

robot - struktura definující parametry robotu

uhly - postupně pro každý motor ve stupních

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **move( )** | robotOJ10( )<br>OJ10checkirc( )<br>degtoirc( )<br>OJ10moveirc |

**moves( )**

**moves(robot,uhly,t,disc)**

Koordinovaným pohybem projede všechny body v maximální dávce 200 bodů

robot - struktura definující parametry robotu

a - pozice ve stupních

t - minimální časový interval mezi dvěmi následujícími pozicemi v milisekundach

disc - hodnota diskontinuity

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **moves( )** | robotOJ10( )<br>OJ10checkirc( )<br>degtoirc( )<br>OJ10moveircs |

**NajedNaBod( )**

**NajedNaBod(r,souradnice, rychlost, disc)**

Najede na bod

r - struktura definující parametry robotu

souradnice - souřadnice bodu kam má najet

rychlost - rychlost pohybu

disc - parametr diskontinuity

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **NajedNaBod( )** | robotOJ10( )<br>swOmezeni( )<br>OJ10ikt( )<br>move( )<br>moves( ) |

**Kruznice( )**

**Kruznice(r,polomer,stred,uhly,n,t)**

Vykresli kružnici v prostoru

n - počet bodu

r - struktura definující parametry robotu

stred - střed kružnice

uhly - [rotacex,rotacey,delta,epsilon]

polomer - poloměr kružnice

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **Kruznice( )** | robotOJ10( )<br>OJ10ikt( )<br>swOmezeni( )<br>moves( ) |

**ZAdoBPoUsecce( )**

**ZAdoBPoUsecce(r,bodA,bodB,n,t)**

Projede usecku mezi body A a B v koordinovanych souradnicich

Souradnice jsou v metrech

r - struktura definujici parametry robotu

bodA - souradnice bodu A

bodB - souradnice bodu B

n - pocet bodu na usecku

t - parametr pro rychlost prujezdu mezi body

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **ZAdoBPoUsecce( )** | robotOJ10( )<br>OJ10ikt( )<br>swOmezeni( )<br>moves( ) |

**hhhome( )**

**hhome(r)**

Podle čidel na robotu se provede hardhome, tj. robot se nastaví do základní polohy v pořadí motorů 3,2 a 1

r - struktura definující parametry robotu

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **hhhome( )** | robotOJ10( )<br>portwriteline( )<br>OJ10waitforready( )<br>move( )<br>OJ10clear( ) |

**OJ10softhome( )**

**OJ10softhome(r)**

Pohybuje robotem do nastavené pozice, počká na dokončení pohybu a zresetuje motory

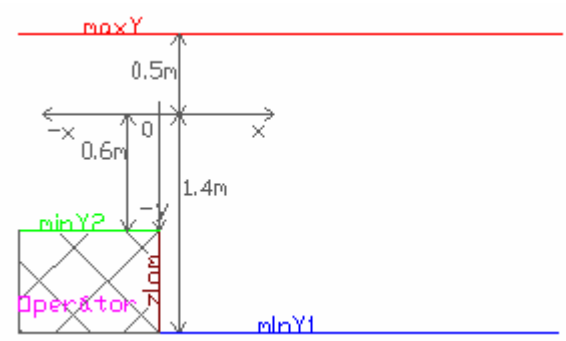
r - struktura definující parametry robotu

to samé jako 'robot'

| Popisovaná funkce | Funkce které obsahuje |
|---|---|
| **OJ10softhome( )** | robotOJ10( )<br>move( )<br>OJ10waitforready( )<br>OJ10resetmotors( ) |

## 5.4 SW ošetření pracovního prostoru

Zde bych se opět odkázal na kolegu Sedláčka. On popsal pracovní prostor robotu (*viz. Obr. č. 24*) podle možností laboratoře kde se nachází. Jednak s ohledem na bezpečnost jakožto robotu samotného a na bezpečnost okolního personálu. Nynější umístění není pro robot zrovna nejvhodnější, jedná se spíše o umístění provizorní. Po plánované rekonstrukci laboratoře K09 by se měl přesunout na jiné, lepší místo. Zde by měl mít kolem sebe více prostoru a nemusel by být jeho pracovní prostor omezen na minimum jako v našem nynějším případě. Omezení pracovního prostoru se tedy detekováno jak limitními HW spínači na těle robotu, tzv. dorazy, tak i na úrovni SW, kde se dá lehce tento prostor modifikovat podle aktuální potřeby. Nyní se tedy zabývejme touto variantou, *viz. Obr. č. 24* převzatý právě od Sedláčka.




*Obr. č. 24 – omezený pracovní prostor*

Operátor je člověk, který robot řídí. Bloček „operátor“ na obrázku představuje tedy jeho osobu, stůl a ovládací PC. Dalšími omezeními je okolní vybavení laboratoře, tj. stoly po obou stranách (*minY1* a *maxY1*). V osách ±*x* žádné omezení není potřeba, tudíž zde může plně využít dosahových rozsahových možností.

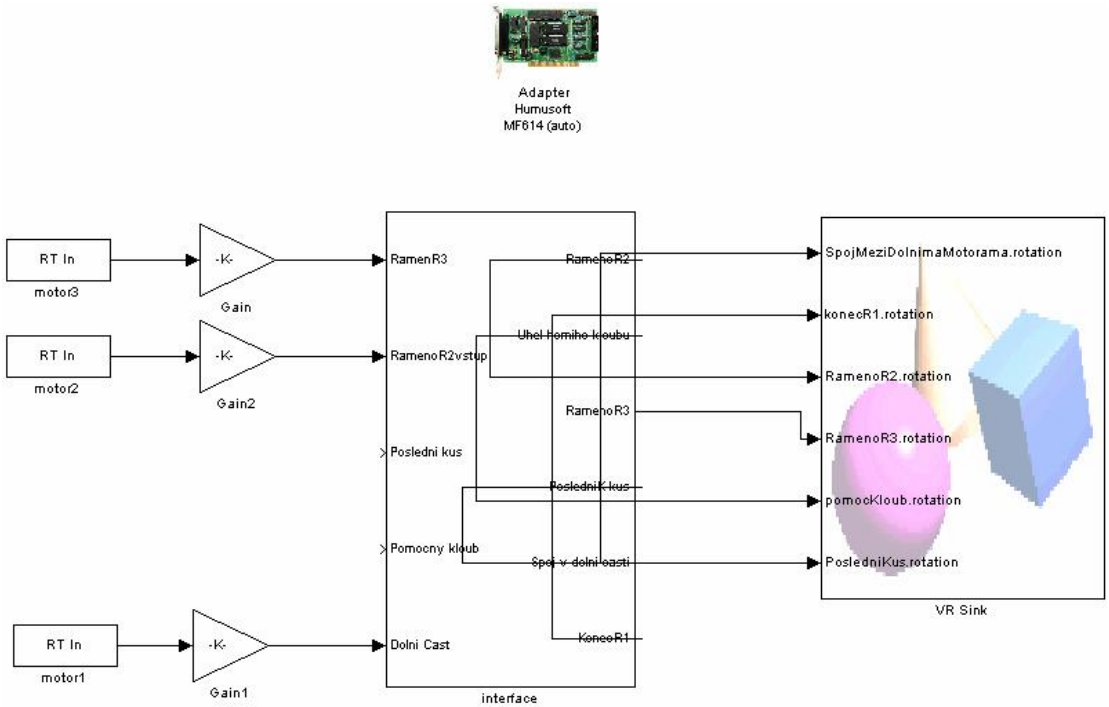
## 5.5 SW MATLAB VIRTUAL REALITY TOOLBOX a REAL TIME TOOLBOX pro monitorování reálného pohybu robota

Monitoring, nebo-li sledování, slouží k pozorování pohybu reálného předmětu, v mém případě robotu, na dálku. Představa je taková, že v jedné místnosti mám robot, kterého řídím a ve druhé místnosti, která je oddělená od předchozí, mám řídící počítač. Z řídícího PC

zadávám příkazy pro pohyby robotu, avšak jestli se skutečně a vůbec jak hýbe, již nevidím. Proto využívám monitoring, který zprostředkuje přenos reálného pohybu pomocí senzorů na libovolně vzdálené řídící středisko. Jako signály pro sledování využívám hodnoty z IRC čidel, které jsou svedeny do MARS 8b. Odtud jsou dvojicí 37 žilových vodičů přivedeny do PC karty HUMUSOFT MF614. Kartu obsluhuji přes SW MATLAB a SIMULINK.

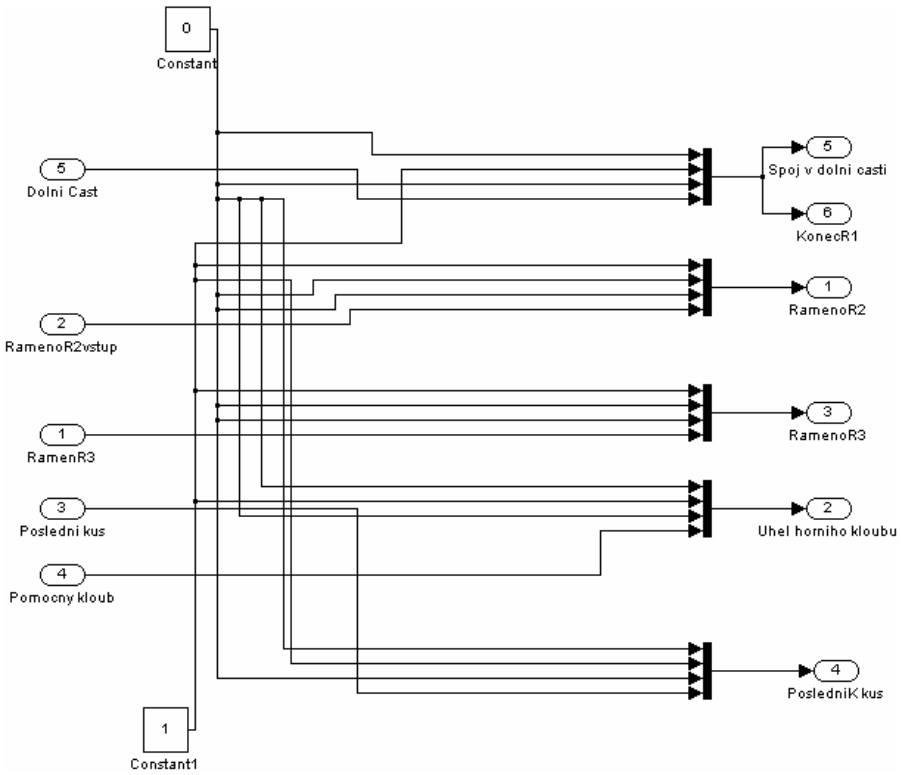
Dále budu pracovat v SIMULINKu (*viz Obr. č. 25*). Zde se mi karta jeví jako jeden bloček, přetažením na pracovní plochu jí inicializuji a mohu jí nastavit řadu parametrů.

Bloky *motor* představují vstupy do karty a odpovídají hodnotě IRC pulzů z čidel na robotu. Přes bloky *Gain* provedu přepočet na stupně a na danou převodovku.



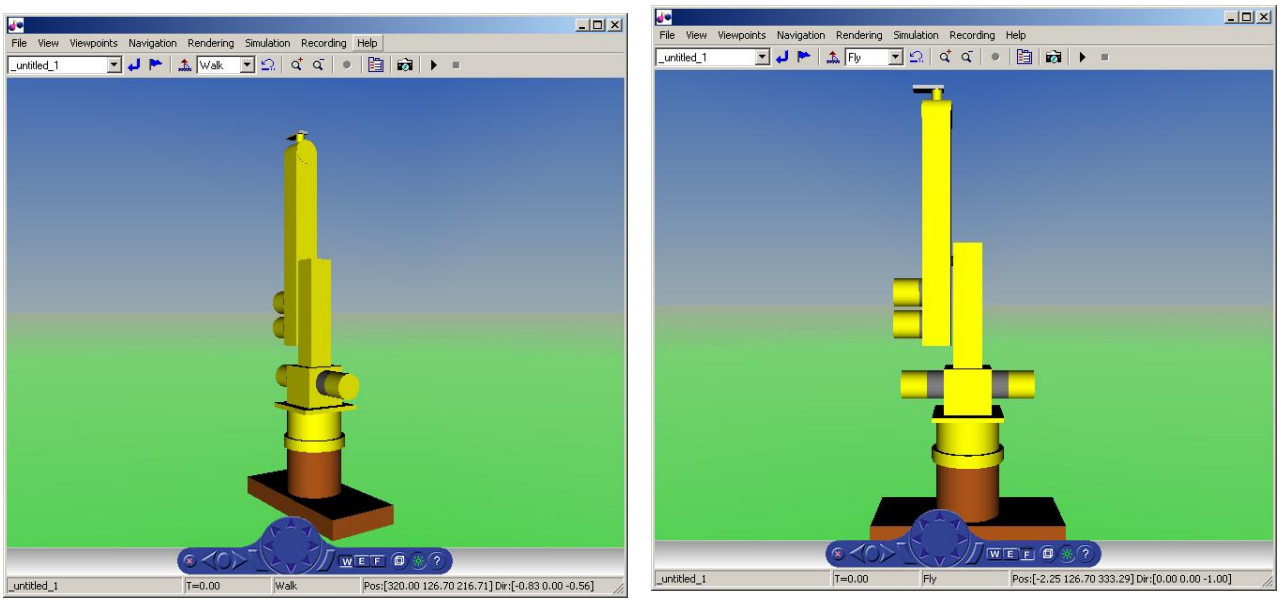

*Obr. č. 25 – schéma pro monitoring ve VIRTUAL REALITY TOOLBOXU pod MATLABem*

Blok *Interface* udává vztah mezi jednotlivými rameny. Na robotu jsou totiž na sobě motory resp. ramena 2,3 a 4,5 závislá. Detail bloku na *Obr. č. 26* představuje vyřešení této vzájemné závislosti.




*Obr. č. 26 – schéma pro monitoring ve VRT pod MATLABem - detail bloku Interface*

Poslední blok *VR sink* představuje tzv. Blok virtuální reality. Spouští 3D modelovací SW VIRTUAL REALITY TOOLBOX (VRT) ve kterém se vytvoří 3D model objektu. Podle připojených signálů do toho bloku v SIMULINKu se pak model pohybuje. VRT umožňuje model pozorovat z různých pohledů (fly, walk..), zobrazit drátěný model, pohled oddalovat či bližovat a rovněž pořizovat záznam ze simulace. Na *Obr. č 27* jsou dva obrázky modelu robotu přímo z prostředí VRT.




*Obr. č. 27 – obrázky modelu robotu v prostředí VRT*

# 6 Závěr

V úvodu jsem provedl seznámení s koncepcí robotu OJ10-RS. Na základě jeho parametrů a parametrů potřebných pro jeho řízení navrhl několik variant jeho moderního řízení. Vybral a realizoval jsem variantu PC – SW MATLAB SIMULINK – MARS 8b - karta HUMUSOFT MF614. S využitím zdrojových kódů od CMP, které jsem podle potřeby upravil a s vytvořením vlastních jsem robot rozpohyboval. Tzn. dokáži s řídící jednotkou přes RS232 z nadřazeného PC navázat spojeni, inicializovat tuto jednotku a nastavit jí pro konkrétní robot. Mohu pohybovat s jednotlivými rameny robotu jednotlivě nebo nechat vykonat koordinovaný pohyb. Koordinovanými pohyby umí robot svým chapadlem projet přímku zadanou počátečním a koncovým bodem nebo kružnici. Ta je zadána středem, poloměrem a rotacemi. Dále umí robot provést softhome, tj. vrátit se na základní polohu zadanou operátorem nebo hardhome, základní poloha je určena pomocí čidel na těle robotu. Tyto čidla jsme zatím umístili pouze tři. Využití nadřazeného PC je rozšířeno o monitorování pohybu robotu. Poloha je snímáná IRC čidly a přes kartu MF614, SW MATLAB a jeho TOOLBOOXy nechávám pohybovat 3D modelem. Z pohledu bezpečnosti se o prostor kolem robotu stará SW ochrana, která definuje rozsah a je lehce modifikovatelná.

Další práce na robotu by spočívala v kompletní realizaci hardhome, tj. osadit i zbylé dvě osy čidly. Celé řízení přenést pod více REALTIMEový systém, např Linux, a přepsat ho např. v jazyku C. Dále pro ovládání vytvořit grafický interface včetně monitorování.

# 7 Literatura


[1] Pikron s.r.o. – výrobce řídící jednotky MARS 8b
- dokumentace k jednotce
www.pikron.com

[2] Rapčan Servis Detva s.r.o. – firma využívající roboty OJ-10RS
- poskytnutí technických výkresů, principů a konzultací
www.rapcan.sk

[3] Humusoft s.r.o – firma vyrábějící kartu MF614
- dokumentace ke kartě
www.humusoft.cz

[4] Mathworks – výrobce SW MATLAB a jeho komponent
www.mathworks.com

[5] Delta Tau – výrobce pohybového kontroléru PMAC PC
- dokumentace ke kontroleru
www.deltatau.com

[6] X33ROB – stránky předmětu Robotika
- převzetí zdrojových kódů
cmp.felk.cvut.cz

[7] Bakalářská práce Jakuba Štoly – Systém pro řízení manipulátorů a robotů
- popis jednotky MARS a principiální popis řízení

[8] Bakalářská práce Václava Sedláčka – Průmyslový robot RSP 01 – základní řízení

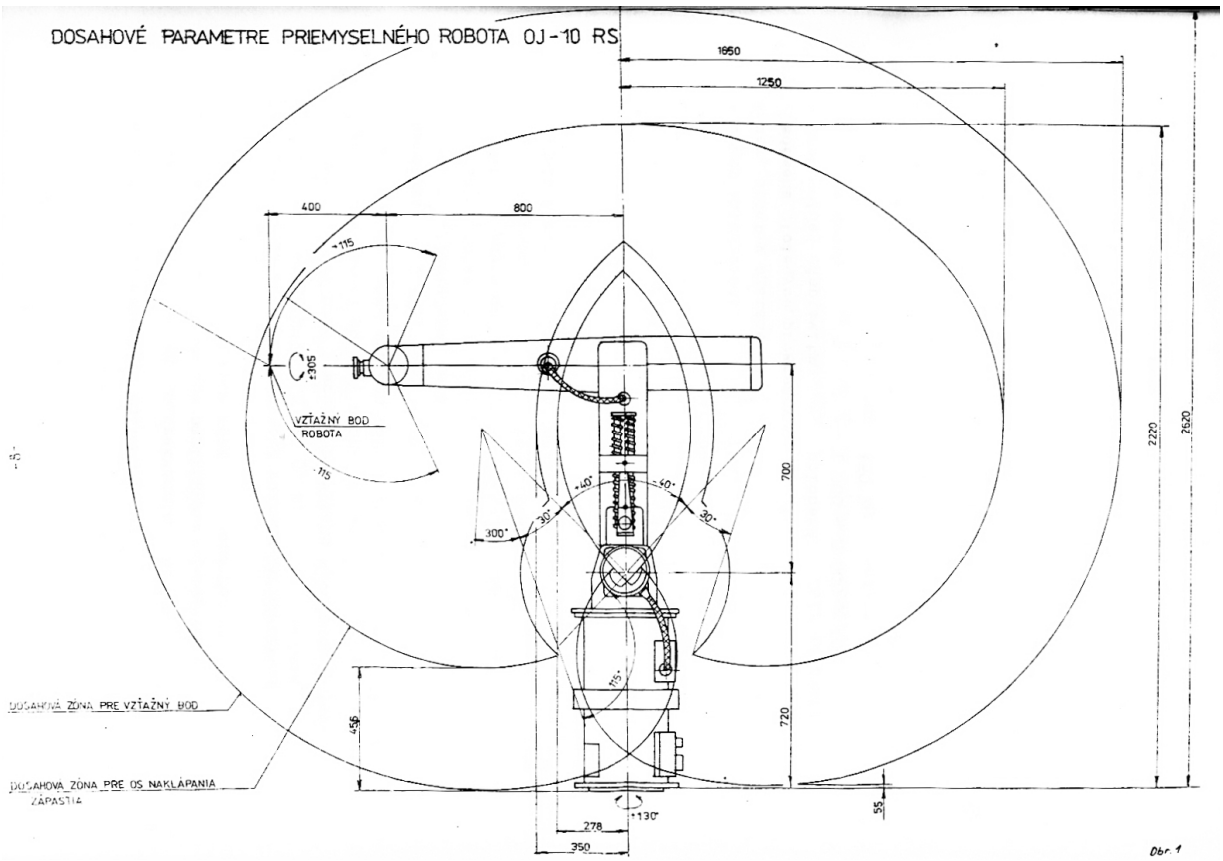
[9] Ing. Pavel Píša – Řídící jednotka s mikrokontrolérem MC68376

# 8 Přílohy

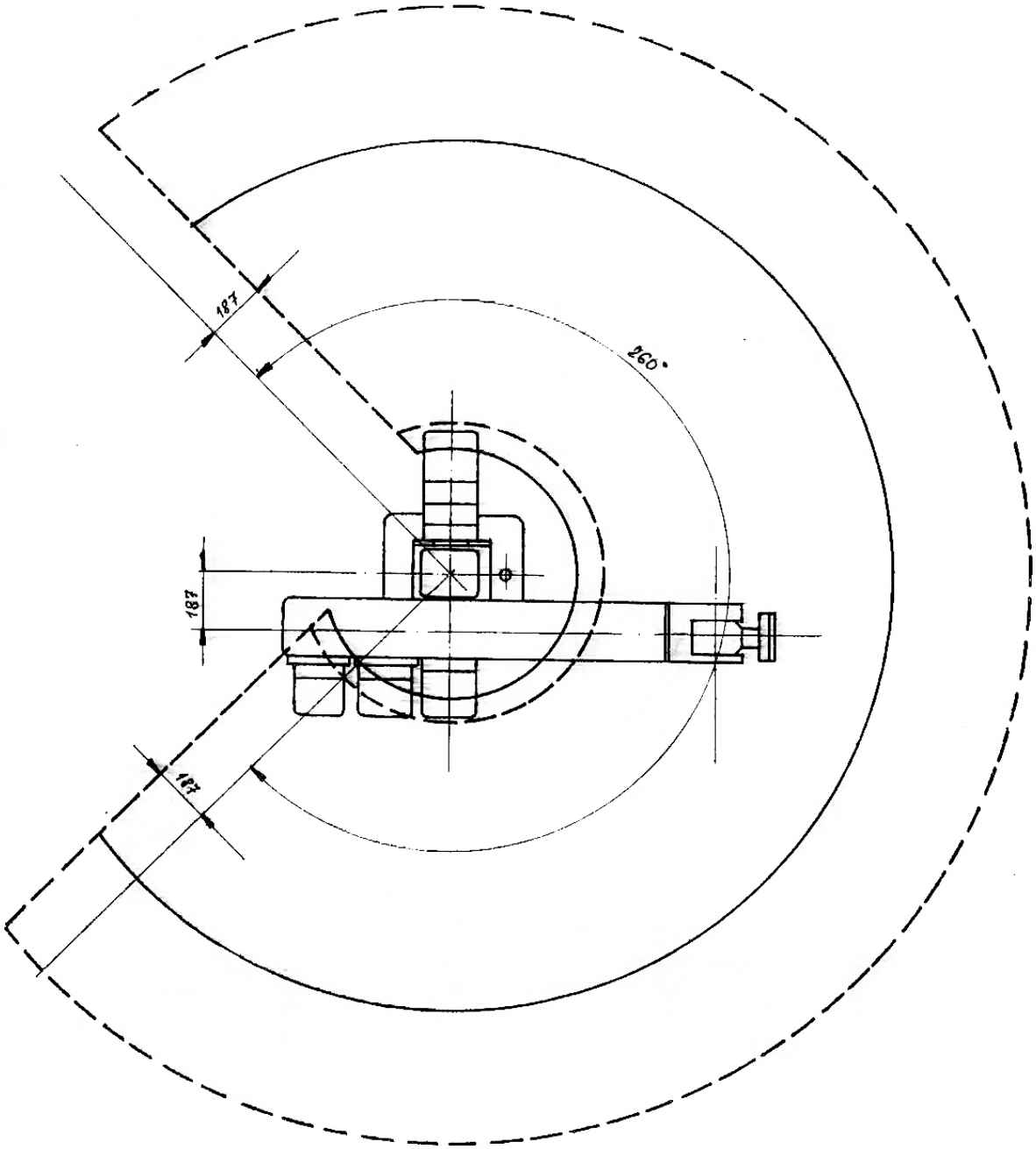
## *8.1 Obsah přiloženého CD*

1) Obsah CD v souboru *Obsah_CD .txt*
2) Elektronická podoba této práce v souboru *BP_2007_Siska_Matej.pdf*
3) Adresář *Pikron*
   a) blokové schéma jednotky
   b) blokové schéma desek
   c) výkresy elektronického zapojení MARS 8b
   d) zapojení konektoru pro Humusoft na MARS 8b
   e) návod na jednotku ve formátu .pdf
4) Adresář *Rapčan*
   a) poskytnutá technická dokumentace
   b) fotky z nasazení robotu ve výrobě
5) Adresář *Humusoft*
   a) dokumentace ke kartě MF614
   b) ovladače ke kartě
6) Adresář *Delta Tau*
   a) dokumentace k PMAC PC
7) Adresář *Bakalářské práce*
   a) Jakub Štola
   b) Václav Sedláček
8) Adresář *Zdrojové kódy*
   a) funkční a popsané zdrojové kódy
9) Adresář *Fotografie*
   a) obsahuje všechny snímky pořízené během práce na robotu
10) Adresář *VRT*
   a) obsahuje soubot Virtual Reality Toolboxu včetne 3D modelu robotu

## *8.2 Rozměrové a dosahové parametry robotu OJ-10RS*

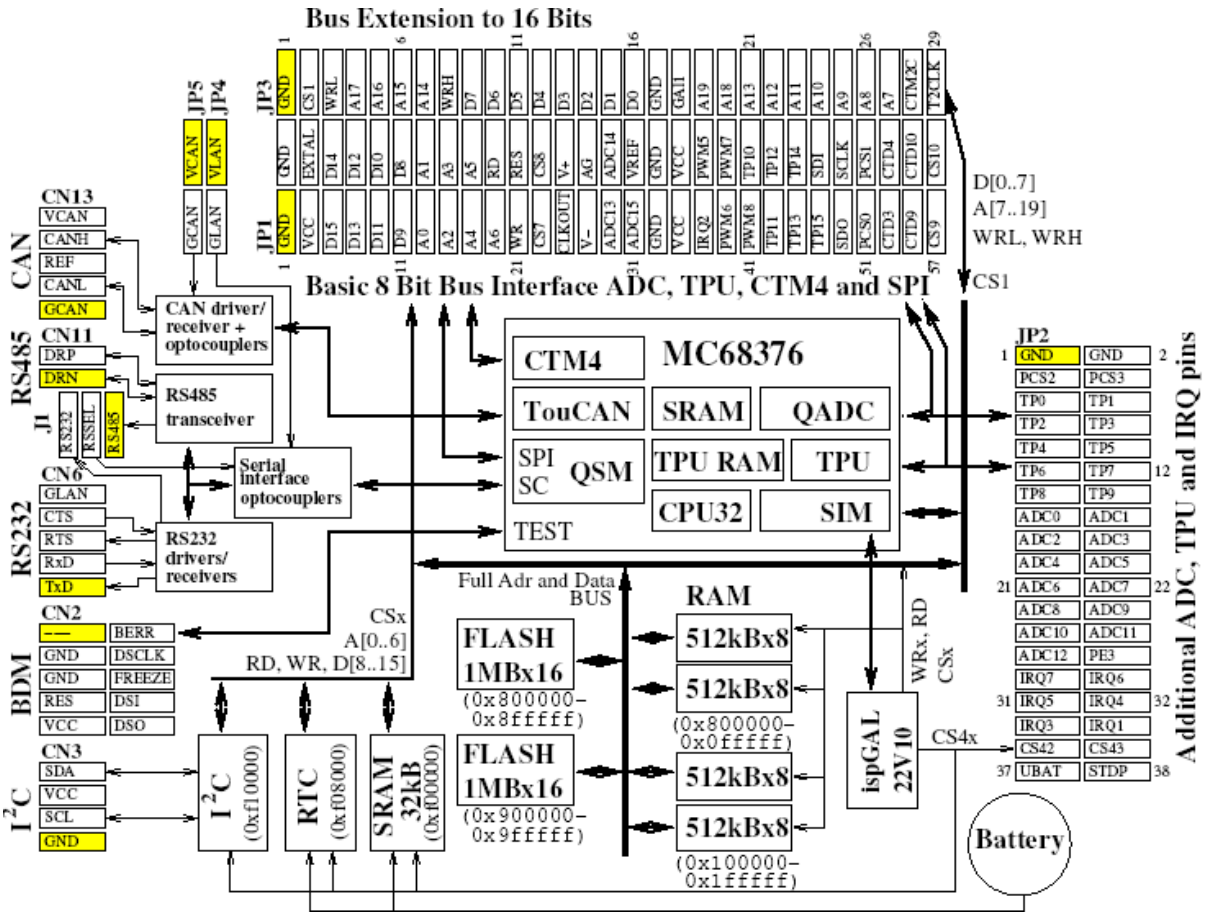


DOSAHOVÉ PARAMETRE PRIEMYSEL. ROBOTA OJ-10 RS

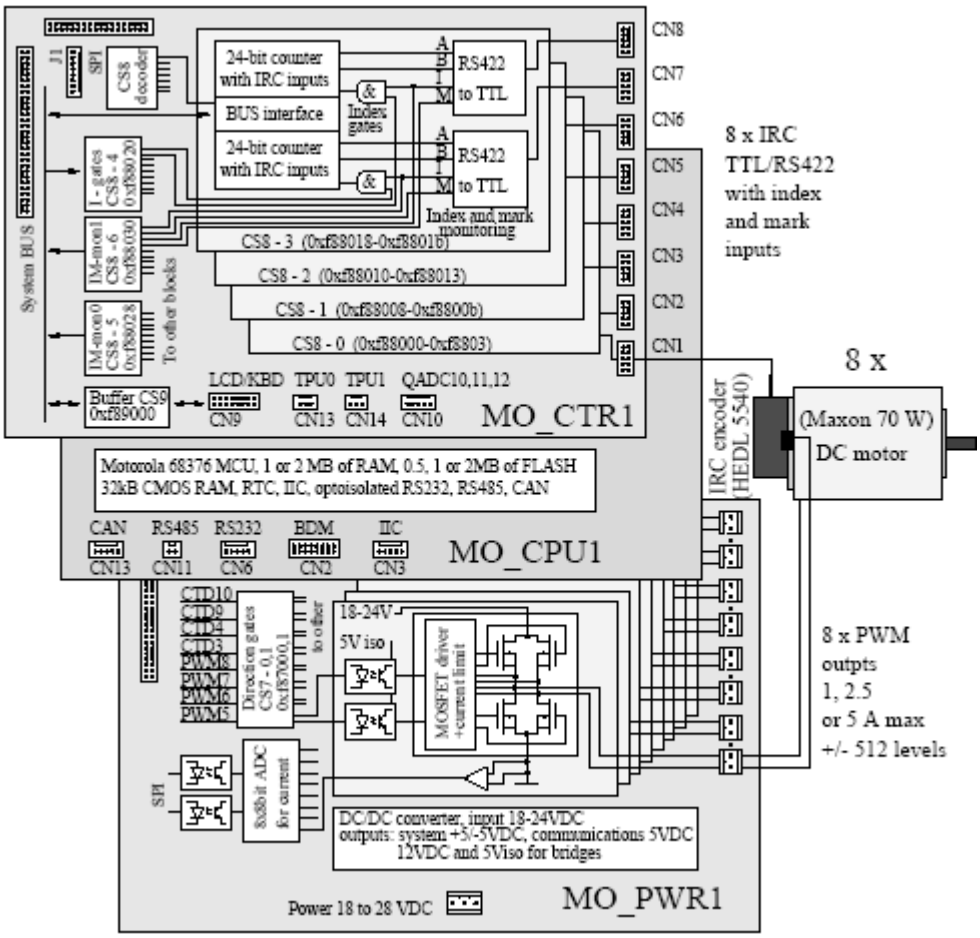



Obr. 2

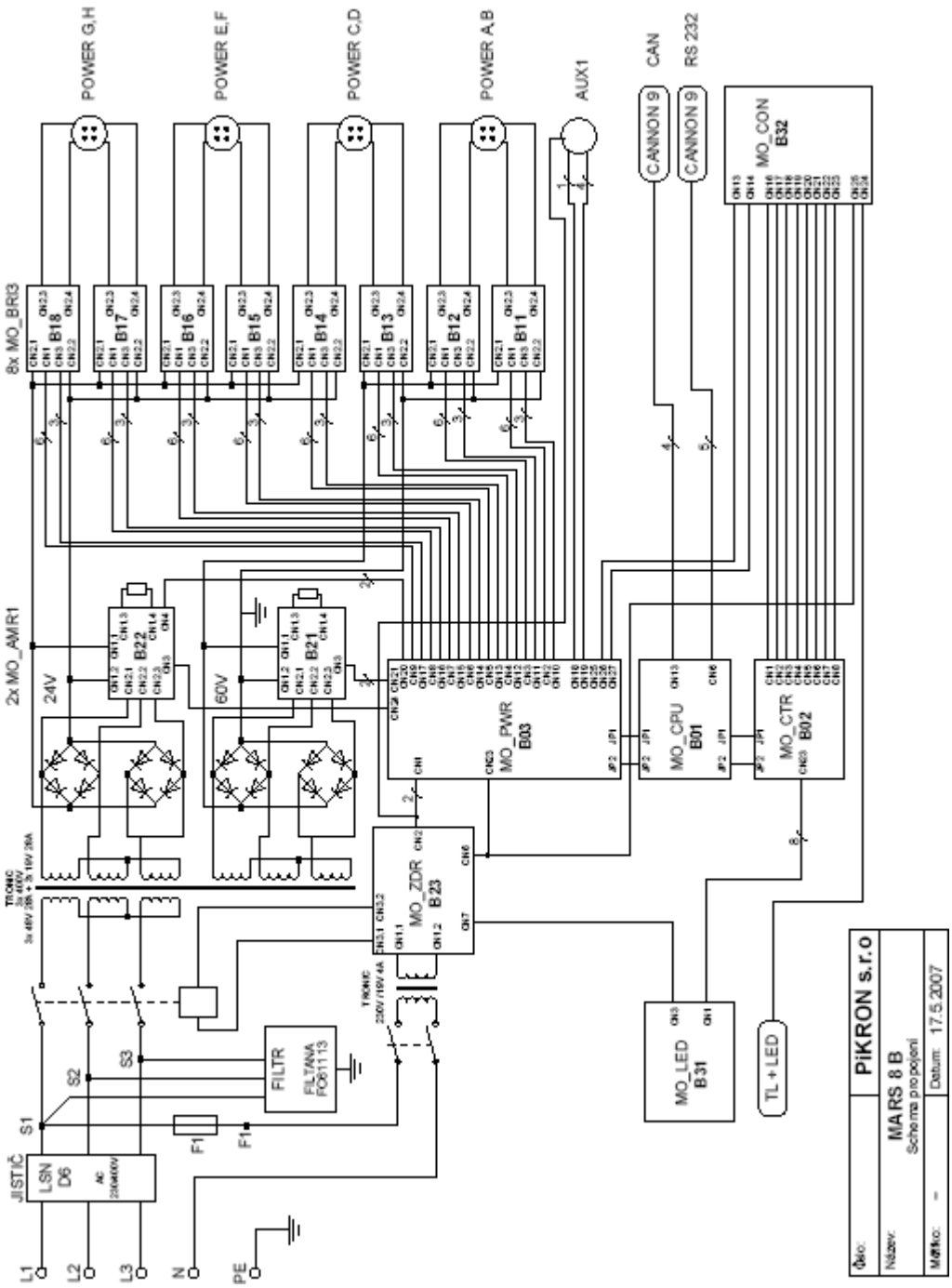
## 8.3 Blokové schéma desky MO_CPU2



## 8.4 Kompletní pohled na všechny tři desky



## 8.5 Kompletní blokové schéma MARS 8b



## 8.6 Značení pinů konektorů pro desku HUMUSOFT MF614

**X1−1**

| | | | |
|---|---|---|---|
| 1 | LEM A | 20 | |
| 2 | LEM B | 21 | |
| 3 | LEM C | 22 | |
| 4 | LEM D | 23 | |
| 5 | | 24 | |
| 6 | | 25 | |
| 7 | | 26 | |
| 8 | | 27 | |
| 9 | AG | 28 | |
| 10 | | 29 | GND |
| 11 | | 30 | DIR A |
| 12 | MARK A | 31 | DIR B |
| 13 | MARK B | 32 | DIR C |
| 14 | MARK C | 33 | DIR D |
| 15 | MARK D | 34 | PWR EN |
| 16 | KSW | 35 | PWR ON |
| 17 | PWR OK | 36 | |
| 18 | IEXT CTR | 37 | |
| 19 | | | |

**X1−2**

| | | | |
|---|---|---|---|
| 1 | LEM E | 20 | |
| 2 | LEM F | 21 | |
| 3 | LEM G | 22 | |
| 4 | LEM H | 23 | |
| 5 | | 24 | |
| 6 | | 25 | |
| 7 | | 26 | |
| 8 | | 27 | |
| 9 | AG | 28 | |
| 10 | | 29 | GND |
| 11 | | 30 | DIR E |
| 12 | MARK E | 31 | DIR F |
| 13 | MARK F | 32 | DIR G |
| 14 | MARK G | 33 | DIR H |
| 15 | MARK H | 34 | |
| 16 | | 35 | |
| 17 | | 36 | |
| 18 | | 37 | |
| 19 | | | |

**X2−1**

| | | | |
|---|---|---|---|
| 1 | IRCA A+ | 20 | IRCD A+ |
| 2 | IRCA A− | 21 | IRCD A− |
| 3 | IRCA B+ | 22 | IRCD B+ |
| 4 | IRCA B− | 23 | IRCD B− |
| 5 | IRCA I+ | 24 | IRCD I+ |
| 6 | IRCA I− | 25 | IRCD I− |
| 7 | IRCB A+ | 26 | |
| 8 | IRCB A− | 27 | |
| 9 | IRCB B+ | 28 | |
| 10 | IRCB B− | 29 | GND |
| 11 | IRCB I+ | 30 | |
| 12 | IRCB I− | 31 | PWMA |
| 13 | IRCC A+ | 32 | |
| 14 | IRCC A− | 33 | PWMB |
| 15 | IRCC B+ | 34 | |
| 16 | IRCC B− | 35 | PWMC |
| 17 | IRCC I+ | 36 | |
| 18 | IRCC I− | 37 | PWMD |
| 19 | GND | | |

**X2−2**

| | | | |
|---|---|---|---|
| 1 | IRCE A+ | 20 | IRCH A+ |
| 2 | IRCE A− | 21 | IRCH A− |
| 3 | IRCE B+ | 22 | IRCH B+ |
| 4 | IRCE B− | 23 | IRCH B− |
| 5 | IRCE I+ | 24 | IRCH I+ |
| 6 | IRCE I− | 25 | IRCH I− |
| 7 | IRCFA+ | 26 | |
| 8 | IRCF A− | 27 | |
| 9 | IRCF B+ | 28 | |
| 10 | IRCF B− | 29 | GND |
| 11 | IRCF I+ | 30 | |
| 12 | IRCF I− | 31 | PWME |
| 13 | IRCG A+ | 32 | |
| 14 | IRCG A− | 33 | PWMF |
| 15 | IRCG B+ | 34 | |
| 16 | IRCG B− | 35 | PWMG |
| 17 | IRCG I+ | 36 | |
| 18 | IRCG I− | 37 | PWMH |
| 19 | GND | | |

**AUX 1**

| | |
|---|---|
| 1 | +24 V |
| 2 | KSW |
| 3 | RSW A |
| 4 | RSW B |
| 5 | RSW C |
| 6 | RSW D |
| ⏚ | GND 24V |

| | | |
|---|---|---|
| Číslo: | | PiKRON s.r.o |
| Název: | MARS 8 B<br>Zapojení konektorů | |
| Měřítko: – | | Datum: 17.5.2007 |

## 8.7 Zdrojové kódy

### 8.7.1 degtoirc( )

```
function b=degtoirc(robot,a)
%b=degtoirc(robot,a) - prevede uhly ve stupnich na hodnotu pulsu IRC
%       robot - struktura definujici parametry robotu
%       a     - vstupni hodnoty velikosti uhlu ve stupnich
%       b     - vystupni hodnoty prevedene na pulzy IRC

b=(a) .* robot.degtoirc;

return
```

### 8.7.2 degtorad( )

```
function a=degtorad(b)
%a=degtorad(b) - Prevede hodnoty uhlu ve stupnich na hodnotu v radianech
%   b - vstupni hodnoty uhlu ve stupnich
%   a - vystupni hodnoty uhlu v radianech

a=b*pi/180;

return
```

### 8.7.3 hhhome( )

```
function hhhome(r)
%hhhome(r) - Podle cidel na robotu se provede hardhome, tj. robot se
%            nastavi do zakladni polohy v poradi motoru 3,2 a 1
%   robot - struktura definujici parametry robotu

portwriteline(r,['HH',r.activemotors(3),':']);
OJ10waitforready(r);
portwriteline(r,['HH',r.activemotors(2),':']);
OJ10waitforready(r);
portwriteline(r,['HH',r.activemotors(1),':']);
OJ10waitforready(r);
move(r,[96 10.5 -97.5 0 0]); %Korekce pohybu od cidla do zakladni polohy
OJ10waitforready(r);
OJ10clear(r);

end
```

### 8.7.4 Kruznice( )

```
function Kruznice(r,polomer,stred,uhly,n,t)
%Kruznice(r,polomer,stred,uhly,n,t) - vykresli kruznici v prostoru
%   n - pocet bodu
%   r - struktura definujici parametry robotu
%   stred - stred kruznice
%   uhly - [rotacex,rotacey,delta,epsilon]
```

```
%   polomer - polomer kruznice

c=1;
K=[];

for i=0:1/n:1 %Odpovida poctu bodu, ktere se budou vykreslovat

    K(:,c)=polomer*[cos(2*pi*i);sin(2*pi*i);0;1/polomer]; %Ziskam
souradnice bodu v jedne rovine
    c=c+1; %Vytvori jednotlive sloupce bodu
end

Rx=[1 0 0 0;0 cos(uhly(1)) -sin(uhly(1)) 0;0 sin(uhly(1)) cos(uhly(1)) 0;0
0 0 1]; %Rotace kolem X
Ry=[cos(uhly(2)) 0 sin(uhly(2)) 0; 0 1 0 0;-sin(uhly(2)) 0 cos(uhly(2)) 0;0
0 0 1]; %Rotace kolem Y
%Rz=[cos(uhly(3)) -sin(uhly(3)) 0 0;sin(uhly(3)) cos(uhly(3)) 0 0; 0 0 1
0;0 0 0 1];
R=Rx*Ry;

Rp=[1 0 0 stred(1);0 1 0 stred(2);0 0 1 stred(3);0 0 0 1]; %Posun do stredu

for i=1:c-1 %Vypoctene body nasobim rotaci
   K(:,i)=R*K(:,i);
   K(:,i)=Rp*K(:,i);
end

K(4,:)=uhly(3); %Natoceni chapadla
K(5,:)=uhly(4);

uhly=OJ10ikt(r,K); %Vypocet IKT
uhly=uhly*180/pi;

j=1;
maxBuffer=150;
buffer=[];
for i=1:length(uhly(1,:,1))
    if(swOmezeni(r,K(:,i)'))

    else
        error('swOmezeni zastavilo pohyb nejspis bod je mimo svuj sw
prostor');
    end

end

for i=1:length(uhly(1,:,1)) %Projede vsechny body urcene k vykresleni
    buffer(:,j,:)=uhly(:,i,:); %Naplnuje postupne buffer body
    j=j+1; %Kazdy bod vlozeny do bufferu inkrementuje ukazatel zaplneni
bufferu
        if(j==maxBuffer || i==length(uhly(1,:,1))) %Pokud je buffer naplnen
nebo uz nejsou dalsi body k vykreslovani
           moves(r,buffer(:,:,1),t,20); %Vykresli body z bufferu
           j=1; %Inicializace ukazatele zaplneni bufferu
           buffer=[]; %Vyprazdneni bufferu
        end
end

end
```

### 8.7.5 move( )

```
function move(robot,uhly)
%move(robot,uhly) - Slouzi pro pohyb po jednotlivych motorech
%   robot - struktura definujici parametry robotu
%   uhly - postupne pro kazdy motor ve stupnich

puvUhly=uhly; %Prohozeni uhlu pro spravne poradi
pomU=uhly(4);
uhly(4)=uhly(5);
uhly(5)=pomU;
uhly(4)=uhly(4)-uhly(5);
uhly(3)=uhly(3)+uhly(2);
if(OJ10checkirc(robot,puvUhly)~=0) %Testovani zda-li jsou hodnoty v
dovolenem rozsahu
irc=degtoirc(robot,uhly); %Prevod uhlu na IRC pulzy
OJ10moveirc(robot,irc); %Pohyb v hodnotach IRC pulzu
else
  disp('pohyb byl zastaven, protoze pro dalsi provedeni by ho dovedlo mimo
svuj pracovni prostor');
end
end
```

### 8.7.6 moves( )

```
function moves(robot,uhly,t,disc)
%moves(robot,uhly,t,disc) Koordinovany pohybprojede vsechny body v
maximalni davce 200 bodu
%   robot - struktura definujici parametry robotu
%   a - pozice ve stupnich
%   t - minimalni casovy interval mezi dvemi nasledujicimi pozicemi v
%       milisekundach
%   disc - hodnota diskontinuity


puvUhly=uhly;
irc=zeros(size(uhly));
if (length(uhly(1,:))<=200) %Kontrola bufferu
for i=1:length(uhly(1,:))
pomU=uhly(4,i);
uhly(4,i)=uhly(5,i);
uhly(5,i)=pomU;
uhly(3,i)=uhly(3,i)+uhly(2,i);
uhly(4,i)=uhly(4,i)-uhly(5,i);
r=OJ10checkirc(robot,puvUhly(:,i)); %Overeni rozsahu IRC pulzu
irc(:,i)=degtoirc(robot,uhly(:,i)'); %Prevod stupnu na IRC pulzy
if(r==0)
error('Moves: Nektery pohyb nemuze byt proveden, protoze robot by se dostal
mimo svuj pracovni prostor');
end
end
OJ10moveircs(robot,int64(irc),t,disc); %Pohyb v koordinovanych pohybech
End

end
```

### 8.7.7 NajedNaBod( )

```
function NajedNaBod(r,souradnice, rychlost, disc)
%NajedNaBod(r,souradnice, rychlost, disc) - Najede na bod
%   r - struktura definujici parametry robotu
%   souradnice - souradnice bodu kam ma najet
%   rychlost - rychlost pohybu
%   disc - parametr diskontinuity

if(swOmezeni(r,souradnice)) %Kontrola rozsahu souradnic
if(nargin==2) %natoci motory na dane uhly
bodyIkt=OJ10ikt(r,souradnice');
uhly=bodyIkt*180/pi;

move(r,uhly(:,:,1)'); %Pohyb po uhlech
end
if(nargin==4) %Najede na polohu pomoci koordinovaneho pohybu
bodyIkt=OJ10ikt(r,souradnice');
uhly=bodyIkt*180/pi;
moves(r,uhly(:,:,1),rychlost,disc); %Pohyb koordinovanymi pohyby
end
else
    disp('swOmezeni zastavilo pohyb nejspis bod je mimo svuj sw prostor');
    return
end
end
```

### 8.7.8 OJ10clear( )

```
function [irc]=OJ10clear(robot)
%[irc]=OJ10clear(robot) - Vypnuti rizeni motoru a vynulovani odectu polohy
%                         a koncove pozadovane polohy pohybu
%                         Nastavi aktualni hodnoty IRC
%   robot - struktura definujici parametry robotu
%   irc - predani parametru ze struktury robot

portwriteline(robot,'CLEAR:'); %Vypne rizeni a vynuluje vsechny motory
irc=OJ10getirc(robot); %Nastavi aktualni hodnoty IRC

end
```

### 8.7.9 OJ10close( )

```
function OJ10close(robot)
%OJ10close(robot) - Provede softhome
%                   Zresetuje motory
%                   Vypne vykony
%                   Povoli ECHO - aby uzivatel videl co pise
%                   Zavre port
%   robot - struktura definujici parametry robotu

if robot.hhflag
  OJ10softhome(robot); %Provede softhome
end

OJ10resetmotors(robot); %Zresetuje motory
```

```
portwriteline(robot,'REGPWRON:0'); %Vypne vykony
portwriteline(robot,'ECHO:1'); %Povoli ECHO - aby uzivatel videl co pise
portclose(robot); %Zavre port

return
```

### 8.7.10 OJ10getirc( )

```
function [angles]=OJ10getirc(robot)
%[angles]=OJ10getirc(robot) - Zjisti vsechny pouzuvane motory a u nich i
%                              hodnoty pulzu IRC cidel
%   robot - struktura definujici parametry robotu
%   angles  - vystupni matice s aktualnimi hodnotama IRC cidel

%Zjisti vsechny pouzivane motory
angles=[];
for i = 1:6
   if robot.activemotors(i) ~= ' '
      portwriteline(robot,['AP',robot.activemotors(i),'?:']);
   end
end

%Vraci hodnoty IRC cidel
for i = 1:6
   if robot.activemotors(i) ~= ' '
      string = portreadline(robot);
      len = length(string)-2;
      if len < 5
         error(['OJ10getirc: unexpected response: "',string,'"']);
%Osetreni vyjimky
      end;
      if ~isequal(string(1:4),['AP',robot.activemotors(i),'='])
         error(['OJ10getirc: unexpected response: "',string,'"']);
%Osetreni vyjimky
      end;
      angles(i) = eval(string(5:len)); %Predani hodnoty IRC cidla
   end
end;

return
```

### 8.7.11 OJ10checkirc( )

```
function [r]=OJ10checkirc(robot,uhly)
%[r]=OJ10checkirc(robot,uhly) - Testuje zda-li jsou hodnoty pulzu IRC v
nastavenem rozsahu
%                              Je-li v rozsahu r=1, jinak r=0
%   robot - struktura definujici parametry robotu
%   uhly - vstupni uhly v IRC pulzech k overeni
%   r - je-li v rozsahu r=1, jinak r=0

%Prochazi uhly, zda-li lezi v pozadovanem intervalu a nastavuje hodnotu r
for i=1:length(uhly(:))
r = (uhly(:)' >= robot.bound(1,:)) & (uhly(:)' <= robot.bound(2,:)); %Lezi-
li v rozsahu, r=1
end;
```

```
for i=1:length(r)
   if r(i)==0
      r=0;
      return;
   end
end
```

### 8.7.12 OJ10ikt( )

```
function J=OJ10ikt(robot,X)
%J=OJ10ikt(robot,X) - resi inverzni kinematickou ulohu IKT
%   X - poloha koncoveho bodu a uhel pod jakym se ma na dany bod dostat
%   J - vypoctene kloubove souradnice jednotlivch ramen
%   robot - struktura definujici parametry robotu

%Predani parametru z inicializace
l1=robot.l1;
l0=robot.l0;
l2=robot.l2;
l22=robot.l22; %Posun na kloubech
l3=robot.l3;
l4=robot.l4;

%Spocitani ctyr reseni
J(:,:,1)=NaN;
J(:,:,2)=NaN;

for i=1:length(X(1,:)), %for cykl s opakovanim odpovidajici delce X

    %Nejdrive urcime alfa - uhel mezi osou y a spojnici stredu sour.
systemu a
    %koncoveho bodu
    %Hledame ji v rovine x-y, tudiz dalsi vzdalenosti a vypocty budou i v
teto
    %rovine

    dOX=(X(1,i)^2+X(2,i)^2)^(1/2); %Vypocet delky prepony pocatek souradnic
- koncovy bod

    %Vzdalenost pocatku a bodu W, ktery lezi na pruniku primky urcene
useckou
    %ramene R2 pruniku primky rovnobezne s l22 vedenou bodem X

    dOW=(dOX^2-l22^2)^(1/2); %Vypocet delky odvesny pocatek souradnic -
koncovy bod
    alfaa=asin(l22/dOX); %Urceni uhlu spojnice pocatek sour. sys a
koncoveho bodu X (vnitrni uhel)
    alfab=atan2(X(2,i),X(1,i)); %Funckce v Matlabu vracejici uhel svirajici
s bodem X
    alfa=real(alfaa+alfab); %Vysledny uhel odpovidajici natoceni robota
    J(1,i,1)=alfa; %Predani hodnoty

    %Dalsi vypocty budeme pocitat v rovine OAW
    %Nejdrive urcime souradnice bodu X v teto nove rovine
    newX=[dOW;X(3,i)];
    %Bod C jsme schopni vypocitat
```

```
    C=newX+l4*[sin(-(pi-X(4,i)));cos(-(pi-X(4,i)))];
    %Vypocet bodu A v techto novy souradnicich
    A=[0;(l0+l1)];
    %Vypocet vzdalenosti A a C
    dAC=((C(1,1)-A(1,1))^2+(C(2,1)-A(2,1))^2)^(1/2);

    if(dAC>l2+l3) %Pokud je vzdalenost bodu A a C vetsi jak l2+l3 tak
neexistuje reseni
        J(:,i,:)=NaN; %Robot se na tyto souradnice nemuze dostat
        else %V tomto pripade ma reseni
            ABC=acos((dAC^2-l3^2-l2^2)/(-2*l2*l3)); %Vypocitame uhel mezi
ABC pomoci kosinove vety
            %Uhel gamma je pak doplnkem do Pi
            gamma=real(pi-ABC);
            %Zatim tedy budeme uvazovat jedno reseni, az na konci dodelame
            %reseni vsechna, jelikoz se daji z tohoto jednoho vypocitat
            J(3,i,1)=gamma;
            %Dale urcime uhel BAC, potom ho pouzijime jako korekci pro
uhel, ktery tvori
            %usecka AC, ten zjistime pomoci fce atan2
            BAC=acos((l3^2-l2^2-dAC^2)/(-2*l2*dAC));
            ACx=atan2(C(2,1)-(l1+l0),C(1,1));
            beta=real(pi/2-(BAC+ACx));
            J(2,i,1)=beta;
            delta=real(X(4,i)-(beta+gamma));
            J(4,i,1)=delta;
            J(5,i,1)=real(X(5,i));
            %Doplneni vsech resenich
            J(1,i,2)=alfa;
            J(2,i,2)=beta+2*BAC;
            J(3,i,2)=-gamma;
            J(4,i,2)=X(4,i)-J(2,i,2)-J(3,i,2);
            J(5,i,2)=X(5,i);

    end
end
end
```

### 8.7.13 OJ10init( )

```
function [robot]=OJ10init(robot)
%[robot]=OJ10init(robot) - Inicializace robota s hodnotami ze sktruktury
'robot'
%                           Zresetuje motory a ceka na dokonceni prikazu
%                           Nastavi rychlost, zrychleni, povoli zapnuti
%                           vykonu
%                           Omezuje maximalni plneni vystupu PWM do motoru
%                           Nastavi hodnoty PID regulatoru
%   robot - struktura definujici parametry robotu

%Zresetuje motory a ceka na dokonceni prikazu
OJ10resetmotors(robot);
OJ10isready(robot);
OJ10waitforready(robot);

%Nastavi rychlost, zrychleni, povoli zapnuti vykonu
OJ10setspeed(robot, robot.defaultspeed);
OJ10setacceleration(robot, robot.defaultacceleration);
```

```
portwriteline(robot,['IDLEREL:',num2str(robot.idle)]);
portwriteline(robot,['REGPWRFLG:',num2str(robot.REGPWRFLG)]);

%Omezuje maximalni plneni vystupu PWM do motoru
for i = 1:6
   if robot.activemotors(i) ~= ' '

portwriteline(robot,['REGME',robot.activemotors(i),':',num2str(robot.REGME(
i))]);
   end
end

% Definuje prubeh rychlosti pri pohybu z jedne polohy do druhe, zpusob
% nalezeni nulove polohy a prepocty logickych a fyzickychsouradnic
if isfield(robot,'REGCFG')
   for i = 1:6
      if robot.activemotors(i) ~= ' '

portwriteline(robot,['REGCFG',robot.activemotors(i),':',num2str(robot.REGCF
G(i))]);
      end
   end
end

%Nastaveni slozek PID regulatoru
if isfield(robot,'REGP')
   for i = 1:6
      if robot.activemotors(i) ~= ' '

portwriteline(robot,['REGP',robot.activemotors(i),':',num2str(robot.REGP(i)
)]);
      end
   end
end
if isfield(robot,'REGI')
   for i = 1:6
      if robot.activemotors(i) ~= ' '

portwriteline(robot,['REGI',robot.activemotors(i),':',num2str(robot.REGI(i)
)]);
      end
   end
end
if isfield(robot,'REGD')
   for i = 1:6
      if robot.activemotors(i) ~= ' '

portwriteline(robot,['REGD',robot.activemotors(i),':',num2str(robot.REGD(i)
)]);
      end
   end
end

% Definuje prubeh rychlosti pri pohybu z jedne polohy do druhe, zpusob
% nalezeni nulove polohy a prepocty logickych a fyzickychsouradnic
if isfield(robot,'REGCFG')
   for i = 1:6
      if robot.activemotors(i) ~= ' '

portwriteline(robot,['REGCFG',robot.activemotors(i),':',num2str(robot.REGCF
G(i))]);
```

```
      end
   end
end

return
```

### 8.7.14 OJ10irctodeg( )

```
function b=OJ10irctodeg(robot,a)
%b=OJ10irctodeg(robot, a) - Prevede hodnotu IRC pulzu na odpovidajici
hodnotu
%                            ve stupnich
%     robot - struktura definujici parametry robotu
%     a - vstupni matice s hodnotami IRC cidel
%     b - prevedene hodnoty ve stupnich

b=(a)./ robot.degtoirc;

return
```

### 8.7.15 OJ10isready( )

```
function [stat]=OJ10isready(robot)
% [stat]=OJ10isready(robot) - Overi, ze uz se robot dale nebude hybat a je
%                              pripraveny pro prijeti novych prikazu
%       robot - struktura definujici parametry robotu
%       stat - 1 .. robot je pripraven, 0 .. posledni prikaz se jeste
%                                            vykonavá

portwriteline(robot,'ST?'); %Zjisti status vsech motoru jako dekadicke
cislo, ktere je interptetovano jako bitove pole
status = portreadline(robot); %Precte status

%Rozklad dekadickeho cisla na binarni bitove pole
[first, last] = strtok(status,'=');
if isempty(last)
   error(['OJ10isready: unexpected response: "',status,'"']);
end
bin = dec2bin(eval(last(2:size(last,2))),18);
s='';
if bin(15)=='1'
   s = ['error, ']; %V navodu Pikronu Bit 3 - Chyba
end
if bin(2)=='1'
   s=[s,'arm power is off, ']; %V navodu Pikronu Bit 16 - Vypnute vykonove
napajeni
end
if bin(1)=='1'
   s=[s,'motion stop, ']; %V navodu Pikronu Bit 16 - Vypnute vykonove
napajeni
end;
if ~isempty(s)
   error(['OJ10isready: ',s(1:length(s)-2),'.']);
end
stat = ~eval(bin(14));
return
```

### 8.7.16 OJ10moveirc( )

```
function OJ10moveirc(robot,irc)
%OJ10moveirc(robot,irc) - Overi zda-li je hodnota IRC pulzu v povolenem
%                         rozsahu
%                         Najede motory na absolutni hodnoty IRC pulzu
%       robot - struktura definujici parametry robotu
%       irc - absolutni hodnota v IRC pulzech

   for i =1:6
      if robot.activemotors(i) ~= ' '

portwriteline(robot,['G',robot.activemotors(i),':',int2str(irc(i))]);
%Najede motorem na obsolutni polohu
      end
   end
```

### 8.7.17 OJ10moveircs( )

```
function OJ10moveircs(r,a,t,disc)
%OJ10moveircs(r,a,t,disc) - pohyb robota do vsech pozic 'a' je pomoci
%                            koordinovanych pohybu
%   r - struktura definujici parametry robotu
%   a - pozice ve stupnich
%   t - minimalni casovy interval mezi dvema nasledujicimi pozicemi s
%       milisekundach
%disc-rate of discontinuity in speeds (for details see manual for robot
control unit).
%disc can be any nonnegative integer (best performance is achieved with
disc set to about 5)
%example: OJ10moveirc(robot,angles,50,5);

OJ10waitforready(r);
portwriteline(r,['COORDDISCONT:',int2str(disc)]);
portwriteline(r,'COORDGRP:A,B,C,D,E');
a=a';

maxn=200; %Delka bufferu na prikazy
n=1;
while size(a,1)-n >= maxn
   for i=n:(n+maxn-1),


portwriteline(r,['COORDMVT:',int2str(t),',',int2str(a(i,1)),',',int2str(a(i
,2)),',',int2str(a(i,3)),',',int2str(a(i,4)),',',int2str(a(1,5))]);

   end;
   n=n+maxn;
   bin(1)=1;
   while bin(1) == 1
      portwriteline(r,'ST?'); %Zjisti status motoru
      tim=r.timeout;
      r=portsettimeout(r,60);
      status = portreadline(r);
      r=portsettimeout(r,tim);
      [first, last] = strtok(status,'=');
      if isempty(last)
```

```
         error(['OJ10moveircs: unexpected response: "',status,'"']);
      end
      bin = dec2bin(eval(last(2:size(last,2))),8);
   end
end;
for i=n:size(a,1)
      %Koordinovany pohyb s casovanim

portwriteline(r,['COORDMVT:',int2str(t),',',int2str(a(i,1)),',',int2str(a(i
,2)),',',...
            int2str(a(i,3)),',',int2str(a(i,4)),',',int2str(a(i,5))]);

end;
OJ10waitforready(r);
portwriteline(r,'COORDGRP:'); %Vyber motoru pro koordinovany pohyb
```

### 8.7.18 OJ10resetmotors( )

```
function OJ10resetmotors(robot)
%OJ10resetmotors(robot) - Zastavi regulace motoru s chybou a chyby
%                         vynuluje, ostatni motory pokracuji v zapocate
cinnosti
%   robot - struktura definujici parametry robotu

portwriteline(robot,'PURGE:');

return
```

### 8.7.19 OJ10setacceleration( )

```
function OJ10setacceleration(robot, accels)
% OJ10setacceleration(robot, accels)  - Nastavi zryhleni pro pouzivane
%                                       motory
%                                       Hodnota zrychleni se pouzije pro
%                                       prikazy najeti na polohu pri
%                                       lichobeznikovem profilu rychlosti
%                                       Hodnota odpovida prirustku
rychlosti za periodu vzorkovani
%       robot - struktura definujici parametry robotu
%       accels  - hodnoty akcelerace pro kazdy motor

for i=1:6
   if robot.activemotors(i) ~= ' ' %Zjisti pouzivane motory
      if isnan(accels(i))
         %Kontroluje zda-li je akcelerace nad dolni hranici ci pod horni
hranici
      elseif (accels(i)>=robot.minacceleration(i) &&
accels(i)<=robot.maxacceleration(i))
         portwriteline(robot,['REGACC', char(i+64), ':',
int2str(accels(i))]); %Nastavi akceleraci
      else
         error('accelerations out of bound'); %Osetreni chyby
      end
   end
end
return
```

### 8.7.20 OJ10setspeed( )

```
function OJ10setspeed(robot, speeds)
% OJ10setspeed(robot, speeds) - Nastavi rychlost pro kazdy motor
%       robot - struktura definujici parametry robotu
%       speeds - nastaveni rychlosti pro kazdou osu

for i=1:6
   if robot.activemotors(i) ~= ' ' %Zjisti pouzivane motory
      if isnan(speeds(i))
         %Kontroluje zda-li je rychlost nad dolni hranici ci pod horni
hranici
      elseif (speeds(i)>=robot.minspeed(i) && speeds(i)<=robot.maxspeed(i))
         portwriteline(robot,['REGMS', char(i+64), ':',
int2str(speeds(i))]); %Nastavi rychlost
      else
         error('speeds out of bound'); %Osetreni chyby
      end
   end
end

return
```

### 8.7.21 OJ10softhome( )

```
function OJ10softhome(r)
%OJ10softhome(r) - Pohybuje robotem do nastavene pozice,pocka na dokonceni
pohybu
%                  a zresetuje motory
%       r - struktura definujici parametry robotu
%           to same jako 'robot'

move(r,[0 0 0 0 0]); %Pohyb do "pocatku"
OJ10waitforready(r); %Ceka na dokonceni pohybu
OJ10resetmotors(r); %Zresetuje motory

return
```

### 8.7.22 OJ10waitforready( )

```
function OJ10waitforready(robot)
% OJ10waitforready(robot) - Ceka dokud robot neni pripraven na dalsi
%                           prikazy
%       robot - struktura definujici parametry robotu


portwriteline(robot,'R:'); %Dotaz ohledne ukonceni probihajici operace
t=robot.timeout; %Ceka dany cas
robot=portsettimeout(robot,60);
s=portreadline(robot);
robot=portsettimeout(robot,t);
if isequal(s,['R!',13,10])
   return
end
if isequal(s,['FAIL!',13,10])
```

```
   OJ10isready(robot);
   error('OJ10waitforready: command ''R:'' returned ''FAIL!''''); %Osetreni
chyb
end

return
```

### 8.7.23 openOJ10( )

```
function [robot]=openOJ10
%[robot]=openOJ10 - Nastavi porty a navaze komunikaci s ridici jednotkou
%                   Povoli zapnuti vykonu
%   robot - struktura definujici parametry robotu

ports_reset %Zresetuje porty
robot=[];
robot=robotOJ10(robot); %Otevre strukturu 'robot'
robot=portopen(robot); %Otevre port definovany ve stukrute 'robot'

robot=portsettimeout(robot,5); %Nastavi casovy limit pro prenos

robot.robotname = 'OJ10'; %Nastavi jmeno robotu

robot.hhflag = 0; %Nastavi znacku indikujici spojeni s robotem

if ~strcmp(robot.comlib, 'matlab6')
    h=robot.handle;
    if mod(get(h,'ReadStatus'),2)==1
        fclose(robot.handle);
        [result, msg] = fopen(h);  %Otevre seriovy port
        if result     %Kontroluje jestli je port otevren
            error(msg); %Chybova hlaska
        end
    end
end

robot=portsettimeout(robot,5); %Nastavi max. cas pro prenos ve vterinach
portwriteline(robot,'ECHO:0');
pause(0.1); %Ceka na odpoved
portflush(robot);

if isfield(robot,'REGPWRON') %Povoli zapnuti vykonu
   portwriteline(robot,['REGPWRON:',num2str(robot.REGPWRON)]);
   disp('Press arm power button on MARS 8b...');
   disp('Press any button...');
   pause;
   portwriteline(robot,['REGPWRFLG:',num2str(robot.REGPWRFLG)]); %SW
zapnuti vykonu

end
robot.hhflag = 1;
```

### 8.7.24 portclose( )

```
function portclose(robot)
%[error]=portclose(robot) - uzavre komunikacni port a vypise pripadne chyby
%   robot - struktura definujici parametry robotu

switch robot.comlib %Prepne na pouzivany SW
case 'rttool'
   err=fclose(robot.handle);
   switch err
   case 0
   case 2
      error(['fclose error ',num2str(err),': Specified port is not open']);
%Osetreni chyby
   otherwise
      error(['fclose error ',num2str(err),': Unknown error']) %Osetreni
chyby
   end
case 'cport'
   disp('not implemented yet');
case 'matlab6'
    fclose(robot.handle);
end
```

### 8.7.25 portflush( )

```
function portflush(robot)
%[error]=portflush(robot)  Vyprazdnni bufferu komunikačního portu
%   robot - struktura definujici parametry robotu


switch robot.comlib %Prepne na pouzivany SW
case 'rttool'
   rdflush(robot.handle);
   return
   err=wrflush(robot.handle);
   switch err
   case 0
   case 2
      error(['wrflush error ',num2str(err),': Specified port is not
open']); %Osetreni chyby
   case 8
      error(['wrflush error ',num2str(err),': Timeout has occured']);
%Osetreni chyby
   case 64
      error(['wrflush error ',num2str(err),': System failure. System
resources are low to handle this operation.']); %Osetreni chyby
   otherwise
      error(['wrflush error ',num2str(err),': Unknown error']) %Osetreni
chyby
   end
case 'cport'
   disp('not implemented yet');
case 'matlab6'
   while robot.handle.BytesAvailable
       line = fscanf(robot.handle);
   end
end
```

### 8.7.26 portopen( )

```
function [r]=portopen(r)
%[r]=portopen(r) - otevreni komunikacniho portu a nastaveni jeho parametru
%   r - struktura definujici parametry robotu

   h = serial(r.portname,'BaudRate',19200); %Nastaveni rychlosti prenosu
   set(h, 'FlowControl', 'hardware'); %Nastaveni typu rizeni prenosu
   fopen(h);  %Otevre seriovy port
   if strcmp(h.Status, 'closed')
      error('fopen error:  The specified port can not be open'); %Osetreni
chyb
   end
   r.handle = h;
end
```

### 8.7.27 portreadline( )

```
function [line]=portreadline(robot)
%[line,error]=portreadline(robot) - Ctení dat z komunikacni linky jdouci z
%                                   ridici jednotky
%   robot - struktura definujici parametry robotu
%   line - prectena data


switch robot.comlib %Prepne na pouzivany SW
case 'rttool'
   [line, dummy, stop, err] = fscanf(robot.handle,'%s',inf,'\r\n');

   %Osetreni chyb
   switch err
   case 0
   case 1
      error(['fscanf error ',num2str(err),': Receive data error']);
   case 2
      error(['fscanf error ',num2str(err),': Specified port is not open']);
   case 4
   case 8
      error(['fscanf error ',num2str(err),': Timeout has occured']);
   case 32
      error(['fscanf error ',num2str(err),': Input buffer overflow, new
incoming data has been discarded']);
   case 64
      error(['fscanf error ',num2str(err),': System failure. System
resources are low to handle this operation.']);
   otherwise
      error(['fscanf error ',num2str(err),': Unknown error'])
   end
   line = [line,stop];
case 'cport'
   disp('not implemented yet');
case 'matlab6'
   [line, count, msg] = fscanf(robot.handle);
   if msg
      error(['fscanf error:' msg]);
   end
end
```

### 8.7.28 ports_reset( )

```
function ports_reset
%ports_reset - zresetuje komunikacni porty

s = instrfind('Port', 'COM1', 'Status', 'open');
if isobject(s)
    fclose(s);
    clear s;
end
s = instrfind('Port', 'COM2', 'Status', 'open');
if isobject(s)
    fclose(s);
    clear s;
end
```

### 8.7.29 portsettimeout( )

```
function [r]=portsettimeout(r,timeout)
%[r]=portsettimeout(r,timeout) - Nastavi max. cas pro cteni komunikacniho
%                                portu
%   r - struktura definujici parametry robotu
%   timout - max. doba pro cteni z komunikacniho portu

   h = r.handle;
   set(h,'Timeout',timeout);
   r.handle = h;
   r.timeout = timeout;
end
```

### 8.7.30 portwriteline( )

```
function portwriteline(robot,line)
%portwriteline(robot,line) -  Zapise prikazy do ridici jednotky
%   line - prikaz ve formatu STRING
%   robot - struktura definujici parametry robotu

   fprintf(robot.handle, '%s\n', line);
%Slouzi pro kontrolu dat na portu
%    if isfield(robot,'log_handle')
%        if robot.log_handle > 0
%            fprintf(robot.log_handle, '%s\n', line);
%        end
%    end
end
```

### 8.7.31 povolDoraz( )

```
function povolDoraz(r)
%povolDoraz(r) - Povoluje rizeni vykonovych vzstupu z generatoru PWN a
%                reaguje na dorazova cidla
%   r - struktura definujici parametry robotu

portwriteline(r,'REGPWRFLG:1'); %Povoluje rizeni vykonovych z generatoru
PWN a reaguje na dorazova cidla
end
```

### 8.7.32 radtodeg( )

```
function  a=radtodeg(b)
%a=radtodeg(b)  - Prevede radiany na stupne
%   b - hodnota uhlu v radianech
%   a - vypoctena hodnota uhlu ve stupnich

a = b*180/pi;
return
```

### 8.7.33 robotOJ10( )

```
function robot=robotOJ10(robot)
%   robot=robotOJ10(robot) - Zakladni inicializace robotu:
%       robot.portname - jmeno komunikacniho portu
%       robot.comlib - verze pouziteho SW
%       robot.timeout - cas pro kontrolu navazani komunikace
%       robot.l0-l4 - rozmery jednotlivych ramen dle rozmeru z IKT v
AutoCadu
%       robot.irc - pocet IRC pulsu na otacku jednotlivych motoru
%       robot.gearing - prevody jednotlivych prevodovek
%       robot.degtoirc - prevod natoceni ramen ze stupnu na IRC pulzy
%       robot.activemotors - seznam aktivnich motoru
%       robot.bound - mezni hodnoty prac. rozsahu v uhlech (min/max) - SW
ochrana
%       robot.(min/max)speed - min. a max. rychlosti motoru (IRC/256/msec)
%       robot.idle - za jak dlouho se maji odstavit regulatory (s)
%       robot.(min/max)acceleration - min. a max. zrychleni
%       robot.defaultspeed - nastaveni vychozi rychlosti
%       robot.defaultacceleration - nastaveni vychoziho zrychleni
%       robot.REGME - maximalni vykon PWM (viz. Pikron)
%       robot.REGPWRON - povoleni sepnuti vykonu (viz. Pikron)
%       robot.REGPWRFLG - povoluje rizeni vykonovych vystupu z generatoru
%                         PWM (viz. Pikron)
%       robot.REGCFG - prubeh rychlosti pri pohybu z jedne polohy do druhe
%                      a zpusob nalezeni nulove polohy (viz. Pikron)
%       robot.povolSWOchranu - obsluha SW ochrany, 1 -povoli, 0 - zakaze
%       robot.maxY  - souradnice Y
%       robot.minY1 - souradnice Y
%       robot.zlom - souradnice X kde prechazi minY1 do minY2
%       robot.minY2 - souradnice Y
%       robot.REGP - nastaveni P slozky regulatoru (viz. Pikron)
%       robot.REGI - nastaveni I slozky regulatoru(viz. Pikron)
%       robot.REGD - nastaveni D slozky regulatoru(viz. Pikron)


robot.portname='COM1'; %Jmeno komunikacniho portu
robot.comlib='matlab6'; %Verze pouziteho SW
robot.timeout=100; %Cas pro kontrolu navazani komunikace

robot.l0=0.375; %Delky jednotlivych ramen vychazi z vykresu IKT v AutoCADu
robot.l1 = 0.46;
robot.l2 = 0.7;
robot.l22=0.187;
robot.l3=0.8;
robot.l4=0.15;
```

```
%Pocet IRC pulsu na otacku jednotlivych motoru
robot.irc = [-500 -500 500 500 -500];

%Prevody jednotlivych prevodovek
%Hodnoty urcene z podkladu od Rapcan, overeno merenim
robot.gearing = [2.305 2.305 2.305 1.3778 1.3778];

%Prevod natoceni ramen ze stupnu na IRC pulzy
robot.degtoirc = robot.irc .* robot.gearing;

%Motory, ktere pouzivame. Napiseme jen pismena motoru, ktera chceme pouzit
%Napr.: napiseme-li [' BCD F'] pouzivame pouze motory B,C,D a F.
robot.activemotors=['ABCDE  '];

%Mezni hodnoty prac. rozsahu v uhlech (min/max) - SW ochrana
robot.bound=[-120 ,-40,-140 ,-110, -100;...
      120,40,145,100, 100];

%Min. a max. rychlosti motoru (IRC/256/msec)
robot.minspeed = [0 0 0 0 0];
robot.maxspeed = [32000 32000 32000 32000 32000];

%Za jak dlouho se maji odstavit regulatory (s)
robot.idle=20;

%Min. a max. zrychleni
robot.minacceleration = [0 0 0 0 0];
robot.maxacceleration = [32000 32000 32000 32000 32000];

%Nastaveni vychozi rychlosti
robot.defaultspeed = [8000 8000 8000 8000 8000];

%Nastaveni vychoziho zrychleni
robot.defaultacceleration = [5 5 5 5 5];

%Maximalni vykon PWM (viz. Pikron)
robot.REGME = [30000 30000 30000 20000 20000];

%Povoleni sepnuti vykonu (viz. Pikron)
robot.REGPWRON=1;

%Povoluje rizeni vykonovych vystupu z generatoru PWM
robot.REGPWRFLG=1;

%Prubeh rychlosti pri pohybu z jedne polohy do druhe a zpusob nalezeni
%nulove polohy (viz. Pikron)
robot.REGCFG=[39,338,39,39,39];

%Parametry pro SW ochranu;
robot.povolSWOchranu=1;

%Vymezeni prostoru SW ochrany
robot.maxY=0.5; %Souradnice y
robot.minY1=-1.4; %Souradnice y
robot.zlom=0; %Souradnice x kde prechazi minY1 do minY2
robot.minY2=-0.6; %Souradnice y

%Nastaveni parametru PID regulatoru
```

```
robot.REGP=[50,56,50,55,50];
robot.REGI=[10,10,10,10,10];
robot.REGD=[10,20,30,25,30];
```

### 8.7.34 swOmezeni( )

```
function r=swOmezeni(robot,X)
%r=swOmezeni(robot,X) - Pro zadany bod X rozhodne jestli je mozne na nej
%                       najet ci ne
%   robot - struktura definujici parametry robotu
%   X - souradnice bodu

r=1;
if(robot.povolSWOchranu==1) %Povolena SW ochrana
   if(robot.maxY<X(2)) %Omezeni prostoru u lavice
      r=0;
      return
   end
   if(robot.zlom<=X(1) && robot.minY1>X(2))
       r=0;
      return
   end
   if(robot.zlom>X(1)&& robot.minY2>X(2))
      r=0;
      return
   end

else
   r=1;
end

end
```

### 8.7.35 ZAdoBPoUsecce( )

```
function ZAdoBPoUsecce(r,bodA,bodB,n,t)
%ZAdoBPoUsecce(r,bodA,bodB,n,t) - Projede usecku mezi body A a B v
%                                 koordinovanych souradnicich
%                                 Souradnice jsou v metrech
%   r - struktura definujici parametry robotu
%   bodA - souradnice bodu A
%   bodB - souradnice bodu B
%   n - pocet bodu na usecku
%   t - parametr pro rzchlost prujezdu mezi body

A=bodB-bodA;
body=[]; %Body ktere se budou projizdet
 j=1;
 for i=0:1/n:1
 body(:,j)=bodA+A*i;
j=j+1;
 end
uhly=OJ10ikt(r,body);
uhly=uhly*180/pi;
j=1;
maxBuffer=100;
```

```
buffer=[];
%Test sw prostoru
for i=1:length(uhly(1,:,1))
    if(swOmezeni(r,body(:,i)'))

    else
        error('swOmezeni zastavilo pohyb nejspis bod je mimo svuj sw
prostor');
    end

end

for i=1:length(uhly(1,:,1)) %Projede vsechny body urcene k vykresleni
    buffer(:,j,:)=uhly(:,i,:); %Naplnuje postupne buffer body
    j=j+1; %Kazdy bod vlozeny do bufferu inkrementuje ukazatel zaplneni
bufferu
        if(j==maxBuffer || i==length(uhly(1,:,1))) %Pokud je buffer naplnen
nebo uz nejsou dalsi body k vykreslovani
           moves(r,buffer(:,:,1),t,10); %Vykresli body z bufferu
           j=1; %Inicializace ukazatele zaplneni bufferu
           buffer=[]; %Vyprazdneni bufferu
        end

end

end
```

### 8.7.36 zakazDoraz( )

```
function zakazDoraz(r)
%zakazDoraz(r) - Povoli rizeni vykonovych vystupu z generatoru PWM
%                Rusi vypnuti vystupu zpusobene pohybem robotu mimo
%                vymezenou pracovni oblast
%                Pouzivat jen po nezbytne nutnou dobu pri automatickem
%                navratu do pracovni oblasti
%   r - struktura definujici parametry robota

portwriteline(r,'REGPWRFLG:3'); %Povoli rizeni vykonovych vystupu z
generatoru PWM
                                %Rusi vypnuti vystupu zpusobene pohybem
                                %robotu mimo vymezenou pracovni oblast
end
```