ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDICÍ TECHNIKY

SYSTÉM PRO ŘÍZENÍ MANIPULÁTORŮ A ROBOTŮ

Jakub ŠTOLA

BAKALÁŘSKÁ PRÁCE
2006

Vedoucí práce: Ing. Pavel PÍŠA

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 26.5.2006 …….

podpis

Poděkování

Na tomto místě bych chtěl poděkovat především vedoucímu mé práce Ing. Pavlu Píšovi, jehož věcná rada a pomoc mě vždy posunula o krok dále v mé práci a rozšířila mé poznání dané problematiky. Dále chci poděkovat své rodině, která mi vytváří pohodlné a klidné zázemí po celou dobu studia.

Katedra řídicí techniky Školní rok: 2005/2006

Zadání bakalářské práce

Student: Jakub Štola

Obor: Kybernetika a měření

Název tématu: Systém pro řízení manipulátorů a robotů

Zásady pro vypracování

1. Seznamte se s procesorovými deskami MO_CPU2.
2. Seznamte se se staršími pracemi zabývající se portováním systému uClinux s jádrem 2.0 a systému RTEMS na tyto desky. Seznamte se s řízením robotů BlueBot.
3. Proveďte zprovoznění novějších verzí alespoň jednoho ze systémů na těchto deskách a na systém portujte subsystém víceosého řízení vycházejícího ze systému PXMC pro roboty BlueBot.
4. Pokuste se doplnit tento systém o propojení na komunikaci CAN/CANopen s využitím výsledků projektu OCERA.

Seznam odborné literatury:

Zdrojové kódy systému PXMC, PiKRON s.r.o., 2001-2005
Dokumentace k projektu OCERA, CVUT, 2004-2005

Vedoucí bakalářské práce: Ing. Pavel Píša

Datum zadání bakalářské práce: zimní semestr 2005/2006

Termín odevzdání bakalářské práce: 26.5.2006

Anotace

Cílem této bakalářské práce je portace systému víceosého řízení pod OS reálného času RTEMS. V první části práce seznamuje s možnostmi portace na příkladech jednoduchých programů. Práce dále seznamuje obecně se systémem víceosého řízení založeném na systému PXMC pro školní roboty BlueBot. V další části je popsána samotná portace systému na operační systém RTEMS. Na závěr byla pro otestování funkčnosti systému napsána přímá a inverzní kinematická úloha pro roboty BlueBot.

Annotation

The aim of this bachelor thesis is a portation of multi-axes control system under the real-time operating system RTEMS. At the beginning of my thesis there are some possibilities of the portation on examples of simply programmes. Next there is a general description of multi-axes control system that is based on PXMC system designed for school robots BlueBot. The thesis continues with a description of portation on RTEMS operating system itself. At the end of thesis a direct and inverse kinematics tasks for BlueBot robots were designed for purposes of testing of the functionality of the system.

Obsah

Anotace ……...i

1 Úvod 2

2 Operační systém RTEMS 3

2.1 Popis systému 3
2.2 Možnosti použití 4
2.3 Význam termínů..... 5

3 Portace 6

3.1 Možnosti portace 6
3.2 Příklady portace na jednoduchých programech 6

4 Systém víceosého řízení PXMC 9

4.1 Princip systému 9
4.1.1 Struktura pxmc_state_t 11
4.1.2 PXMC proces 12
4.1.3 Knihovní funkce 12
4.2 Použitý hardware 13
4.2.1 Procesorová deska MO_CPU1 13
4.2.2 Jednotka MARS 8 a robot BlueBot 15

5 Portace sytému PXMC na RTEMS 17

5.1 Popis portace základu systému 17
5.2 Úprava dalších částí systému 18
5.2.1 Přidání kontrolních funkcí 18
5.2.2 Přidání řídicích funkcí 19

6 Přímá a inverzní kinematická úloha robotu BlueBot 21

6.1 Popis robota BlueBot 21
6.2 Řešení přímé kinematické úlohy 22
6.3 Řešení inverzní kinematické úlohy 24
6.3.1 Matematické řešení IKU 24
6.3.2 Ošetření chybových stavů 27

7 Otestování funkce systému 28

8 Závěr 29

Literatura 30

Seznam příloh 31

Kapitola 1

Úvod

Tato bakalářská práce se zabývá možnostmi portace programů a systému pod volně šiřitelný operační systém reálného času RTEMS. Popisem tohoto systému se zabývám v samostatné kapitole. Po seznámení se s tímto systémem a s možnostmi portace na příkladech jednoduchých školních programů, bylo hlavní těžiště mé práce v portování systému pro víceosé řízení založeném na systému PXMC pro školní roboty BlueBot. Popisem systému PXMC se zabývám v samostatné kapitole. Hardware na kterém jsem testoval svou práci je procesorová deska MO_CPU1 vyvinutá firmou PIKRON s.r.o. Tato deska je zakomponována do řídicí jednotky MARS 8 umožňující ovládat manipulátor až s osmi osami. Na tuto jednotku byl v mém případě připojen výše zmiňovaný výukový robot BlueBot. Použitý hardware je popsán v samostatné kapitole. Stěžejní část mé práce - postup operace portování tohoto systému na operační systém - je popsána v následující kapitole. Po provedení této operace jsem k otestování funkčnosti systému vyřešil přímou a inverzní kinematickou úlohu pro robot BlueBot. Na závěr jsem se pokusil připravit demo ukázku pohybu robotu s využitím mých úprav popsaného systému.

Kapitola 2

Operační systém RTEMS

V této kapitole se pokusím nastínit základní popis tohoto operačního systému a všeobecné možnosti použití. Dále se zaměřím na možnosti využití v mé práci.

2.1 Popis systému

RTEMS je zkratka pro **R**eal-**T**ime **E**xecutive for **M**ultiprocessor **S**ystems (operační systém reálného času pro multiprocesorové systémy). Výraz "exekutíva" se používá pro operační systémy, které se na koncové zařízení přenášejí slinkovné se zdrojovými kódy aplikace. Systém, požadované knihovny a aplikace jsou spojeny do jednoho binárního obrazu. Systém RTEMS je open-source(volně šiřitelný) distribuovaný z velké části pod GNU General Public License (GNU GPL). Systém začal být vyvíjen koncem osmdesátých let na požadavek americké armády na systém reálného času pro řízení střel bez licenčních poplatků a založený na standardech. Písmeno „M" tehdy ve zkratce znamenalo Military.

RTEMS má implementováno rozhranní(API) podle standardů POSIX 1003.1b včetně threadů. podporuje přibližně 80% tohoto standardu (301 z 362 funkcí), dále ITRON a vlastní RTEMS Classic rozhraní. Každá skupina systémových funkcí (subsystémů) je implementována odpovídajícím manažerem umístěném v nezávislém knihovním modulu.

Podporované jazyky jsou ISO/ANSI C, ISO/ANSI C++ a Ada95. Mezi základní rysy jádra (kernel) systému patří: podpora multitaskingu, podpora homogenních a heterogenních multiprocesorových systémů (úlohy však nemigrují mezi procesory a jsou vždy svázány s jedním předefinovaným), preemptivní plánování řízené událostmi a založené na prioritách, meziúlohová komunikace a synchronizace, dědění priorit, rozsah priorit 1-255 pro RTEMS Classic API (1 – nejvyšší priorita) a 1-254 pro POSIX API (254 – nejvyšší priorita), aktivní řízení přerušení, dynamická alokace paměti a vysoká úroveň konfigurovatelnosti. Podpora sítí: vysoce výkonný port FreeBSD TCP/IP zásobník, protokoly: UDP, TCP + ICMP, DHCP, RARP, BOOTP, PPPD. Servery: FTP server (FTPD), Web Server (HTTPD), Telnet Server (Telnetd). Systém dále podporuje ladění pomocí GNU a DDD debugerů, stejně tak jako ladění přes ethernet a sériový port.

Podporován je také skriptovací jazyky Python. Vývojové prostředí: GCC a další GNU nástroje.

RTEMS podporuje celou řadu procesorů založených na různých architekturách, samozřejmě včetně procesoru Motorola MC68376 použitého v této práci. Veškeré informace včetně kompletní dokumentace uvádí [1] a [2]

2.2 Možnosti použití

Operační RTEMS nalézá své největší uplatnění v oblasti embedded aplikací a zároveň tam, kde jsou potřeba vlastnosti operačního systému reálného času s jeho možnostmi plánování atd. Embedded systémem lze rozumět jakýkoliv počítačový sytém, který je zabudován do většího systému skládajícího se z více technologií jako např. digitální a analogová elektronika, mechanické zařízení a senzory. Jak se stávají hardwarové platformy více výkonné, vyžaduje většina embedded systémů operační systému reálného času zabudovaný přímo do nich samotných. Výkon takového zabudovaného operačního systému určuje úspěšnost systému, bez ohledu na to jak efektivně pracuje hardware. Stejně tak jak roste složitost embedded hardwaru, roste velikost a složitost embedded operačních systémů. Systém musí rutině zvládat činnosti, které byly před nedávnem pro svou náročnost nepředstavitelné. Dnešní embedded aplikace osahují milion i více řádek kódu.

Operační systém reálného času má ucelený soubor charakteristik, kterým se liší od běžných operačních systémů. OS reálného času je řízen a musí reagovat na události skutečného světa za dodržení přesných požadavků, které na něj klade prostředí se kterým interaguje. Bezchybnost systému nezávisí jen na výsledku výpočtu, nýbrž také na čase ve kterém je výsledek spočítán. Nejdůležitější z vlastností OS reálného času je schopnost přijímat a reagovat na vnější podněty za přesně daných časových podmínek.

RTEMS, vzhledem k faktu, že je volně k dispozici široké veřejnosti k úpravám a použití jako open-source, nabízí alternativu ke komerčním operačním systémům. Nespornou výhodou je jeho podpora širokého spektra platforem, neustálý vývoj systému, publikování nových verzí a kvalitní dokumentace.

2.3 Význam termínů

V této podkapitole vysvětlím význam některých nejzákladnějších pojmů vyskytujících se v souvislosti s operačními systémy reálného času.

- *Semafory (Semaphores):*
  Nejmenší synchronizační prostředek poskytující základ pro složitější synchronizační mechanismy definované v aplikaci.
- *Zamknutí paměti procesu (Process Memory Locking):*
  Zařízení pro zlepšení výkonu svázáním programu aplikace s operační pamětí RAM počítače. To zabrání možné latenci zavedené operačním systémem do úložných částí programu, které nejsou referencovány v druhotných paměťových zařízeních.
- *Paměťově mapované soubory (Memory Mapped Files):*
  Zařízení dovolující aplikaci přistupovat k souborům, jako k částem adresového prostoru
- *Sdílené paměťové objekty (Shared Memory Objects):*
  Objekty představující paměť, která může být mapována současně do adresového prostoru více než jednoho procesu.
- *Prioritní plánování (Priority Scheduling):*
  Výkon zlepšující zařízení dovolující aplikacím rozhodnout v jakém pořadí bude připraveným vláknům přidělován přístup k procesoru.
- *Rozšíření o real-timové signály (Realtime Signal Extension):*
  Zařízení k zlepšení determinismu povolující asynchronním signálům aplikace zařadit se do fronty bez narušení kompatibility s již existujícími signálovými funkcemi.
- *Časovače (Timers):*
  Mechanismus umožňující upozornit vlákno na dosažení určitého času, nebo uplynutí definovaného časového úseku.
- *Meziprocesní komunikace (Interprocess Communication):*
  Funkční vylepšení komunikace pro lokální procesní komunikaci.
- *Synchronizované vstupy a výstupy (Synchronized Input and Output):*
  Robustní vylepšení datových vstupů a výstupů, tak že aplikace může zajistit, že právě manipulovaná data jsou fyzicky přítomna i v sekundárních paměťových zařízeních.

- *Asynchronní vstupy a výstupy (Asynchronous Input and Output):*
  Funkční vylepšení dovolující aplikaci zpracovávat frontu vstupu a výstupů s asynchronním oznamováním dokončení.

Kapitola 3

Portace

V této kapitole vysvětlím pojem „portace“ a popíší postup, kterým jsem se seznamoval s operačním systémem RTEMS a s ostatními nástroji potřebnými pro následující práci.

3.1 Možnosti portace

Pojem portace znamená přenesení daného systému pod jinou platformu či jiný systém. V mém případě jsem portoval programy pod operační systém RTEMS. Nespornou výhodou této operace je, že k základnímu programu (systému) získáme výhody plynoucí z prostředí tohoto operačního systému, mezi než se řadí především příkazová řádka, možnosti plánování úloh, sledování spuštěných úloh (vláken) systému a mnohé další podpory, které tento OS nabízí a které lze využít v dalším rozvoji aplikace.

3.2 Příklady portace na jednoduchých programech

Pro seznámení se s operačním systémem RTEMS jsem použil vzorové programy pro výuku předmětu Počítačové systémy z webové stránky předmětu [3]. Tyto programy jsem postupně upravoval tak, aby byly spustitelné v prostředí operačního systému RTEMS. Testoval jsem programy z následujících skupin:

- Procesy a signály
- Vlákna

Ostatní skupiny pro mě vzhledem k mé další práci nebyly v danou chvíli aktuální.
Programy jsem testoval na desce MO_CPU2 s procesorem Motorola MC68376. K přenosu programů na desku jsem použil sériovou linku a k ladění programů rozhranní GDB

Procesy a signály:

Vzhledem k faktu, že OS RTEMS nepodporuje UNIX Process API a jeho standardní funkce fork(), exec(), execl(), execv(), pthread_atfork(), wait() atd., jsem musel tyto příklady simulovat pomocí vláken. První jednoduchý příklad na standardní čtení příkazové řádky zůstal beze změn, avšak příklad na procesy, kdy jeden proces vytvoří jiný a pak jej signálem zabije, nešel pod RTEMS přenést. Vytvořil jsem proto dvě vlákna a pokusil jsem se nasimulovat výstup tak, aby vypadal jako u původního programu, přičemž příkazy, které program vykonávají jsou samozřejmě jiné a využívají příkazy, kterými se v RTEMSu řídí vlákna. Oba programy(původní s procesy i nový jsou příloze této práce). V dalším přikladu jsem naopak použil RTEMSem podporované POSIXové operace s vlákny a simuloval jsem tak příklad, kdy jeden proces vytvoří potomka a pak čeká na jeho ukončení.

Vlákna:

Jak jsem již zmínil v kapitole 1, RTEMS podporuje standard POSIX včetně vláken. Proto bylo přenesení příkladů na základní operace s vlákny velmi snadné, neboť šlo použít veškeré známé standardní příkazy na práci s vlákny (create(), join() atd.).

Výčet funkcí pro vytvoření a smazaní vlákna (POSIX API/RTEMS Classic API):

- *pthread_create()/rtems_task_create()* - vytvoří nové výkonné vlákno
- *pthread_exit()/rtems_task_delete()* – smaže vlákno
- *pthread_cancel()* – zruší vlákno na nadcházejícím rušícím bodu
- *pthread_detach*() – oddělí vlákno, takže nemusí být ukončováno funkcí join
- *pthread_join()* – spojení vlákna pro jeho ukončení
- *pthread_self(), _equal(), _once(), _setschedparam(),_getschedparam(), _attr_getxxx(), _attr_setxxx()* – další POSIX funkce pro kontrolu vláken

Výčet funkcí pro práci se semafory:

POSIXové binární, čítací, s/bez timeoutů, pojmenované/nepojmenované

- *sem_init() and sem_destroy()* – vytvoření a zrušení nepojmenovaného semaforu
- *sem_open() and sem_close()* – otevření/vytvoření a zavření semaforu
- *sem_unlink()* – smazání nepoužívaného semaforu
- *sem_wait(), sem_trywait(), sem_timedwait()* – pokus o zamknutí semaforu, pokud je nedostupné (hodnota je nula) čeká na uvolnění. Neblokující verze s definovanými timeouty
- *sem_post()* – odemkne semafor
- *sem_getvalue()* – získá aktuální hodnotu semaforu

Pod RTEMS jsem přenesl následující programy:

- vícevláknový Hello World
- příklad chyby souběhu
- opravená chyba souběhu
- práce se semafory
- čtení ze souboru

Většina programů zůstala beze změn a šlo jen o to, jak správně nadefinovat priority vláknen, semafory a v případě čtení ze souboru, jak přenést na desku soubor. Vzhledem k tomu, že RTEMS podporuje souborový systém (In-Memory Filesystem IMFS), šlo ke zdrojovým kódům přilinkovat adresář s požadovaným souborem.

Zajímavostí byl příklad na demonstraci chyby souběhu, kdy se dvě vlákna se stejnou prioritou pokouší současně přistupovat k jedné proměnné. Tento příklad za normálních okolností vrací špatný výsledek a je tedy na něm demonstrována nutnost použití zámku na sdílenou proměnnou(mutexu). Na systému RTEMS se mi ovšem tento jev nepodařilo nasimulovat a program, i bez zamknutí sdílené proměnné, dával správný výsledek. Přesnou příčinu tohoto jevu jsem nezjistil, avšak domnívám se, že důvodem je dědění priorit a plánovací systém RTEMSu.

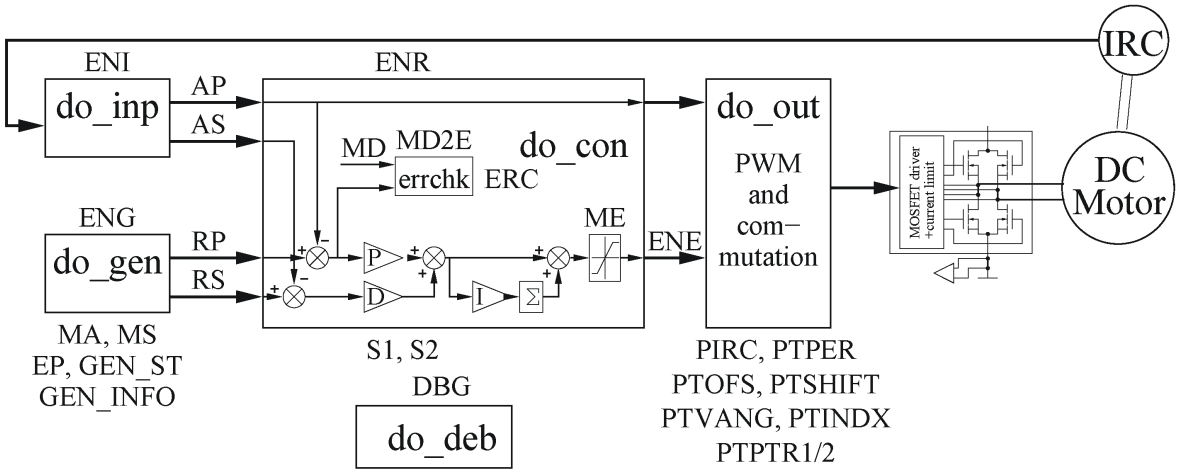
Kapitola 4

Systém víceosého řízení PXMC

V této kapitole se budu zabývat popisem systému víceosého řízení PXMC. V další části stručně popíši procesovou desku MO_CPU1 na které jsem s tímto systémem pracoval.

4.1 Princip systému

Systém PXMC je multiplatformní kód pro řízení stejnosměrných motorů pomocí IRC (inkrementalní čidla) zpětné vazby a pulsně šířkové modulace (PWM). Základní schéma systému je zachyceno na *obrázku 1*.



Obrázek 1: blokové schéma systému PXMC

Význam zkratek:

do_inp ... blok pro získání aktuální hodnoty polohy a rychlosti ze zpětné vazby IRC (blok je specifický pro daný hardware)
AP-okamžitá pozice, AS-okamžitá rychlost

do_gen ... zdroj referenčních signálů
RP-žádaná pozice, RS-žádaná rychlost
MA-maximální zrychlení, MS-maximální rychlost, EP-konečná pozice pohybu, GEN_ST-stav generátoru, GEN_INFO-pole pro výpočet trajektorií

do_con ... PSD regulátor (Proporcionálně-Sumačně-Diferenční)
MD-maximální diference (maximální povolená změna pozice) errch – kontrola chyb,S1,2-zvláštní řídicí konstanty

ENE ... vypočítaná velikost akčního zásahu (napětí, proudu nebo energie)

do_out ... výstup regulátoru, akční člen (pulsně-šířková modulace, komutace)
PIRC, PTPER,PTOFS,PTSHIFT,PTVANG, PTINDX, PTPTRI1/2 - signály pro vícefázové řízení (krokové motory, bezkartáčové motory - BDC)
v jednotce MARS 8 se nepoužívají

do_deb ... podpora ladění

IRC ... inkrementální čidlo

DC Motor ... stejnosměrný motor

Základem systému PXMC je softwarově řešený PSD regulátor, do jehož zpětné vazby je přiveden signál z inkrementálního čidla IRC umístěného na každém regulovaném motoru. Tento signál dávající informaci o aktuální pozici a rychlosti motoru je porovnán s výstupy zdroje referenčních signálu a to sice s žádanou pozicí a rychlostí. Odchylka aktuální pozice je přivedena do proporcionálního regulátoru naopak odchylka aktuální rychlosti je přivedena na vstup diferenčního regulátoru. Součet takto upravených odchylek je poté přiveden na vstup sumačního regulátoru. Blok ME (maximal energy) omezí velikost akčního zásahu a výstup z regulátoru je přiveden do akčního členu tvořeného pro stejnosměrný motor pulsně-šířkovým modulátorem a řízením polarity. Akční zásah je poté přiveden přes MOSFETový můstek a proudové omezení přímo na regulovaný motor.

Software je interně navržen modulárně pro volitelný počet řízených os. Navržený hardware omezuje maximální použitý počet os na osm, což odpovídá i výkonu mikroprocesorového systému, který je při nejnáročnějších fázích operací hledání počáteční polohy a koordinovaných pohybech na současně na všech osách špičkově zatížen na 50%. Polohová regulace a výpočty požadované polohy probíhají se vzorkovací frekvencí 1 kHz. Činnost každé osy je určena informacemi uloženými v její stavové struktuře (stavová struktura pxmc_state_t viz dále). Zde jsou též uloženy ukazatele na funkce, které obstarávají načtení aktuální polohy z hardware, výpočet regulačního zásahu, výstup zásahu na výkonové výstupy, monitorování polohy pro účely ladění a přípravu nové

požadované polohy. Jednotlivé funkce lze pro každou osu za běhu individuálně povolovat, zakazovat, parametrizovat a i měnit, pokud jsou dodrženy určité podmínky.

Pro generování požadované polohy je možné používat množství generátorů trajektorií. Základní jsou:

- generátor pro přejetí nastavenou maximální rychlostí do cílové polohy s lichoběžníkovým profilem rychlosti
- generátor pro nalezení referenční nulové polohy s množstvím různých režimů
- generátor konstantní rychlosti otáčení s plynulým přechodem na požadovanou rychlost a zastavením

Systém dále umožňuje dynamicky definovat skupinu os (v případě potřeby lze rozšířit i na více skupin), které mají provádět pohyby koordinovaně. Skupina umožňuje koordinované přesuny mezi zadanými body v prostoru. Systém je schopen optimalizovat napojení více zadaných úseků tak, aby zpomalení při průjezdu hran bylo co nejmenší. Systém nyní také obsahuje podporu koordinovaných pohybů definovaných spline funkcemi.

4.1.1 Struktura pxmc_state_t

Softwarovou reprezentaci všech signálu fyzického regulátoru představuje v systému PXMC struktura s názvem pxmc_state_t a její implementace se nazývá *mcs* (motor controller state). Ta obsahuje informace pro řízení pohybu jedné osy s jím příslušejícím DC motorem.

Struktura obsahuje mimo jiné:

- příznaky os pro nastavení řízení, generátoru trajektorií a pro stavové informace
- ukazatele na funkce nezbytné při řízení motorů: generátory trajektorií (žádaných hodnot*) (ukazatel pxms_do_gen*), regulátor*(ukazatel pxms_do_con*), výstup *(ukazatel pxms_do_out*), vstup zpětné vazby *(ukazatel pxms_do_inp*). Chod těchto funkcí je synchronizován vzorkovací frekvencí.
- aktuální pozice a rychlost os (*pxms_ap* a *pxms_as*),
- žádaná pozice a rychlost (*pxms_rp*, *pxms_rs*, *pxms_rpfg*, *pxms_rsfg*, fg-pro jemné nastavení),

- parametry os *(pxms_md, pxms_ms, pxms_ma, pxms_inp_info, pxms_out_info*),
- výstup regulátoru(*pxms_ene, pxms_erc*),
- základní parametry pro PID(PSD) regulátory (*pxms_p, pxms_i, pxms_d, pxms_s1, pxms_s2, pxms_me*),
- informace pro bezkartáčové motory
- chybové kódy a pomoc pro ladění

4.1.2 PXMC proces

Jelikož je struktura pxmc_state_t základem systému PXMC, je v prvním kroku nutné tuto strukturu inicializovat. Funkce *pxm_axis_mode(),* která nastaví vybraný druh motoru se v jednotce MARS 8 nepoužívá. Je zde nastaven mód 4 značící DC motor s IRC zpětnou vazbou a PWM regulací.

Dále je nastaven pointer na regulátor *mcs->pxms_do_con* a je zavolána příslušná inicializační funkce *pxmc_dcm_init_fbmode(mcs0)* (pro aplikace s DC motory). Funkce nastaví registry(jednotky časovače a PWM) a nastaví ukazatele na vstupní a výstupní funkce (*mcs->pxms_do_inp, mcs->pxms_do_out).*

Jakmile jsou tyto konfigurace hotovy, odstartuje časovač řídicí vzorkovací periodu. Konec periody aktivuje obsluhu přerušení od čítače. Tato obsluha přerušení *pxmc_sfi_isr(void)* volá funkce odkazované ukazatelem msc:

- **mcs->pxms_do_inp** funkce *pxmc_tpuirc_inp()* načte okamžitou pozici a rychlost,
- **mcs->pxms_do_con** funkce *pxmc_pid_con()* generuje hodnotu pxms_ene podle výpočtu regulátoru,
- **mcs->pxms_do_out** funkce *pxmc_pwm1f_out()* posílá PWM signál na motor,
- **mcs->pxms_do_deb** funkce pomoc při ladění,
- **mcs->pxms_do_gen** funkce generuje trajektorie.

4.1.3 Knihovní funkce

V této podkapitole uvedu několik knihovních funkcí ze systému PXMC, které později použiji ve své práci.

Z knihovních funkcí systému PXMC, které jsem jako první zakomponoval do mé práce, jsem zvolil logicky základní funkce pro řízení motorů.

Funkce:

- *pxmc_go:* posun motoru na zadanou pozici v absolutních souřadnicích
- *pxmc_stop:* zastaví pohyb zadané osy
- *pxmc_ap:* zjistí aktuální pozici daného motoru v absolutních souřadnicích
- *pxmc_hh:* provede Hard Home (nastavení do výchozí pozice) dané osy
- *pxmc_coordmv_grp:* vytvoří ze zadaných motorů skupinu pro koordinované řízení
- *pxmc_coordmv_absmv:* posune skupinu motorů na zadané absolutní polohy

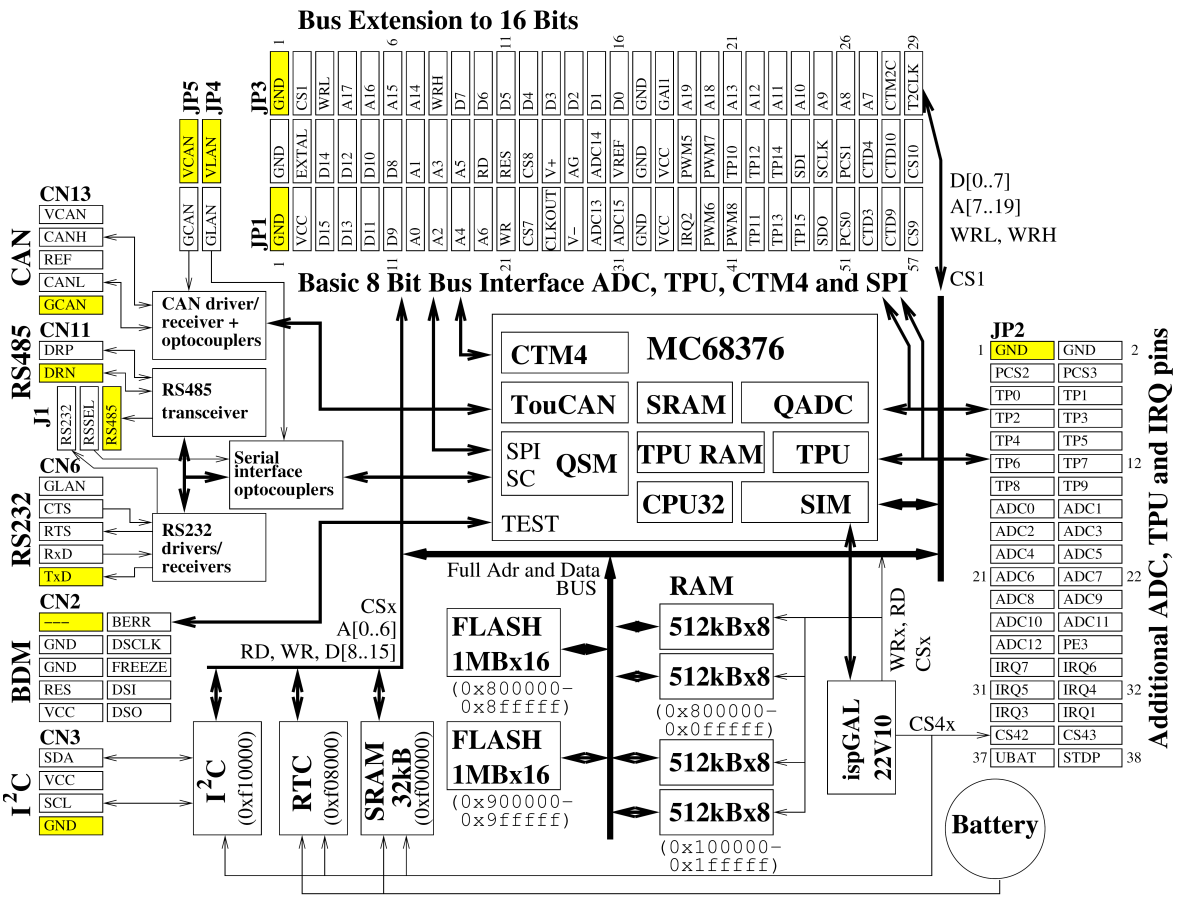
Pro mé potřeby bylo však nutné uvedené funkce mírně upravit, což popisuji v kapitole o portaci systému pod RTEMS.

4.2 Použitý hardware

V této části stručně popíši hardware, na kterém jsem svou práci testoval. Jedná se o procesorovou desku MO_CPU1 firmy PIKRON s.r.o osazenou procesorem Motorola MC68376. Tato deska je zakomponována do řídicí jednotky pro řízení až osmi os MARS 8. Tato jednotka může být použita například pro řízení výukového robota BlueBot.

4.2.1 Procesorová deska MO_CPU1

Na *obrázku 2* je znázorněno blokové schéma procesorové desky MO_CPU1. Deska je osazena mikrokontrolérem Motorola MC68376 s 32-bitovým jádrem. Procesor má vyvedenou 16-ti bitovou datovou a 24-bitovou adresovou sběrnici. Na čipu je integrován řadič sběrnice CAN (TouCAN), 16-ti bitový kanálový 10-ti bitový A/D převodník (QADC), 16 – ti kanálový časovací koprocesor (TPU), dalších až 8 PWM výstupů, ze kterých 4 mohou sloužit jako dvojité jednotky komparátorů nebo záchytů (CTM4), řadič SPI rozhraní s možností až 16 automatických přenosů (QSM) a asynchronní sériové rozhraní UART. Mikrokontrolér navíc obsahuje bloky 4kB a 3,5kB RAM.



Obrázek 2: Blokové schéma desky MO_CPU1 s mikrokontrolérem MC68376

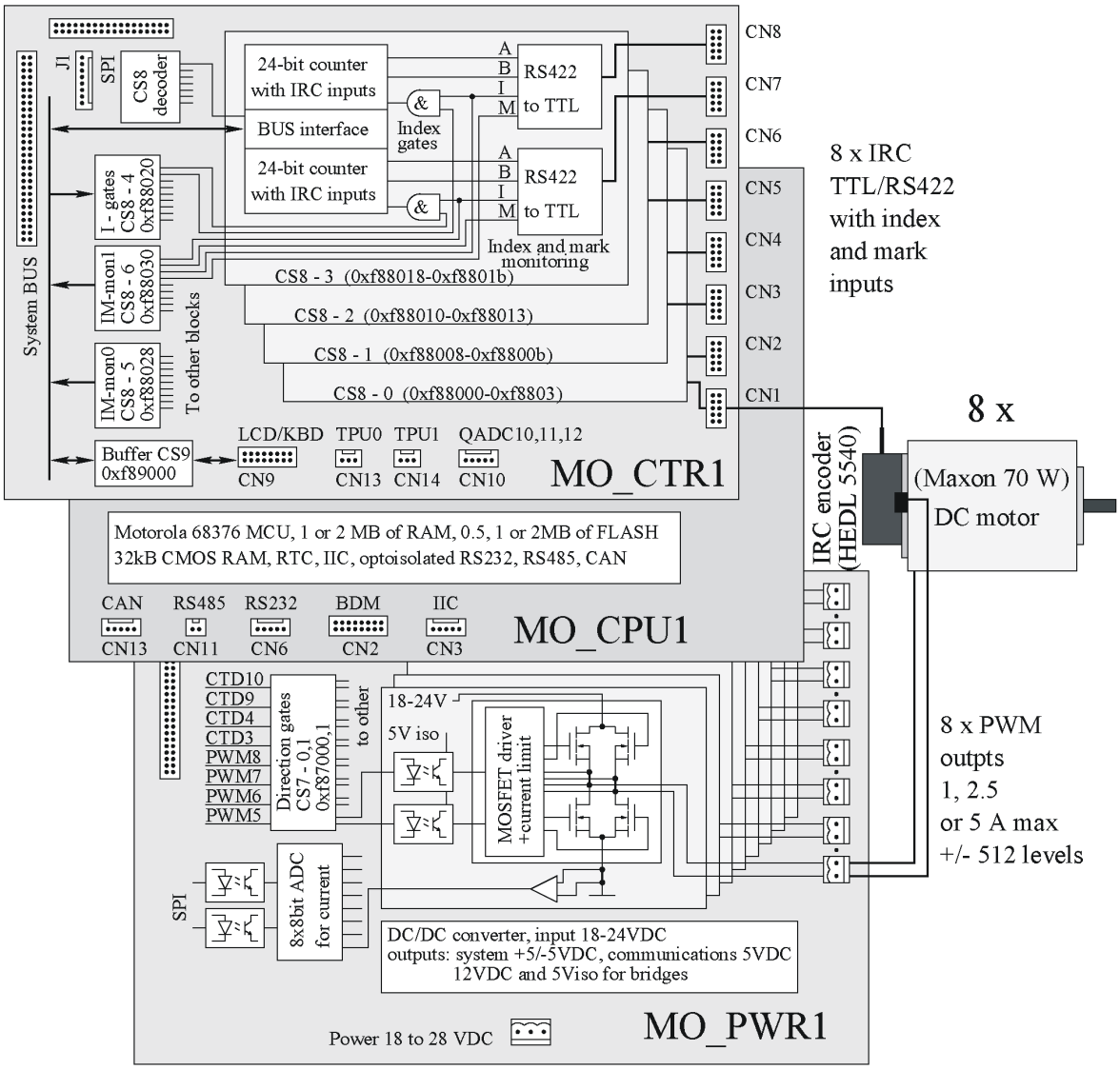
Deska s označením MO_CPU1 obsahuje rozhraní IIC, galvanické oddělení po sběrnici CAN a opticky oddělené budiče sériové linky RS232/485. Pro uložení zavaděče a aplikací slouží 1 nebo 2 MB pamětí FLASH v 16-ti bitové konfiguraci. Paměť FLASH je inicializována a může být kdykoliv přeprogramována i přes rozhraní BDM (Background Debug Mode). Data a další kód může být uložen v rychlé paměti SRAM o velikosti až 2 MB. Data, která mají být uchována i přes vypnutí systému, lze uložit do paměti FLASH nebo do 32 kB baterií zálohované paměti CMOS SRAM. Systém je dále doplněn obvodem reálného času.

Deska byla realizována s oboustranným osazením technologií SMD na čtyřvrstvém plošném spoji. Mapování paměti do paměťového prostoru je plně konfigurovatelné, protože výběrové signály jsou odvozeny od programovatelných chip-select generátorů mikrokontroléru. Zvolené mapování pamětí i periférií je v obrázku vyznačeno v závorkách. Pro ušetření počtu nutných signálů chip-select pro dekódování i způsobu přístupu k pamětem, byla logika výběru čtení i zápisu horního a dolního byte a výběru jednotlivých

periferií na desce realizována v aplikaci programovatelným logickým obvodem ispGAL22V10.

4.2.2 Jednotka MARS 8 a robot BlueBot

Výše popsaná deska MO_CPU1 je součástí řídicí jednotky MARS 8, která umožňuje řízení až osmi os motorů. Základní blokové schéma servosystému jednotky je na *obrázku 3*.



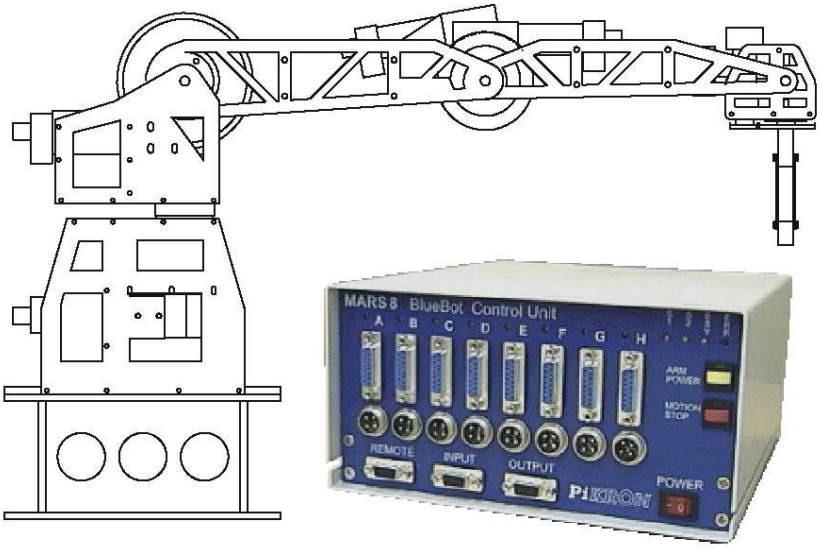
Obrázek 3: Blokové schéma servosystému jednotky MARS 8

V této jednotce je pro řízení manipulátorů a polohovacích zařízení doplněna deska MO_CPU1 deskou MO_CTR1 s dekodéry pro osm IRC vstupů a deskou MO_PWR1

s výkonovými výstupy. Na desce s IRC vstupy byly kromě přijímačů diferenciálních signálů normy RS422 přidány samostatné dekodéry s 24-bitovými čítači. Osm dekodérů a 16-ti bitových čítačů lze sice realizovat jednotkou TPU mikrokontroléru, ale v časová analýza činnosti mikrořadiče TPU ukazovala, že by při vyšších rychlostech otáčení motorů mohlo docházet ke ztrátám polohy. Rozšíření čítačů na 24 nebo 32 bitů je pro reálné rychlosti snadno realizovatelné v obslužném software.

Výkonové budiče PWM jsou řízeny přímo výstupy z rychlé časovací jednotky CTM4 mikrokontroléru. Aby nebylo nutné řídit 16 půlmůstků šestnácti signály, prochází osm šířkově modulovaných logických signálů hradlovací logikou, která řídí polaritu (směr otáčení motoru) každého výstupu. Informace o požadovaných polaritách jsou mikrokontrolérem zapisovány do dvou osmibitových registrů. Výstupní budiče v můstkovém zapojení obsahují nadproudovou ochranu na horních tranzistorech půlmůstků. Celkový proud můstky je snímán, zesílen a převeden osmibitovým AD převodníkem připojeným přes optoizolátory k sběrnici SPI mikrokontroléru.

Jednotku MARS 8 je možné použít například pro řízení výukového robotu BlueBot. Robot BlueBot je robot určený pro výuku, má 6 nezávislých os a 5 stupňů volnosti. Jeho koncový stupeň lze modifikovat (chapadlo, svářecí hrot atd.).



Obrázek 4: výukový robot BlueBot a řídicí jednotka MARS 8

Kapitola 5

Portace sytému PXMC na RTEMS

V této kapitole popíši postup při portaci systému PXMC pod operační systém RTEMS. Uvedu postup zprovoznění základních řídicích částí systému a dále převod jeho knihovních funkcí.

5.1 Popis portace základu systému

Základním úkolem při portaci systému PXMC pod operační systém bylo nahrazení obsluh přerušení periodicky se spouštějící úlohou (vláknem) s vysokou prioritou. Tímto se umožní využití vysoce efektivního přístupu, který RTEMS při práci s vlákny a jejich plánování nabízí. RTEMS například definuje pro každé vytvořené vlákno datovou strukturu TCB (Task Control Block), která obsahuje informace týkající se vykonávání vlákna. TCB struktura je alokována při vytvoření vlákna a uvolněna pro další použití při jeho smazání. TCB prvky jsou modifikovány systémovým voláním způsobeným reakcí aplikace na vnější podněty. TCB je jediná vnitřní datová struktura RTEMSu přístupná z aplikace. Obsahuje jméno vlákna, současnou prioritu, současný a počáteční stav, mód vlákna, TCB uživatelský ukazatel a data požadovaná blokovanými vlákny.

RTEMS podporuje 255 úrovní priorit vláken od 1 do 255 přičemž s rostoucím číslem priorita vlákna klesá. Systém nemá žádné omezení na počet vláken se stejnou prioritou. Počet vláken je omezen pouze velikostí dostupné paměti a definuje se pro každou aplikaci. Priorita vlákna může být následně změněna. Priority používá plánovač k rozhodování které připravené vlákno se vykoná.

Přerušení nejsou převedena na signály nebo jiné asynchronní události. Přerušení jsou v RTEMSu obsluhována C rutinou spojenou s daným vektorem přerušení. Obsluha přerušení může být založena jako jakékoliv běžné vlákno. Obsluze přerušení je dostupná pouze podmnožina API funkcí. Funkce k řízení obsluhy přerušení jsou specifikována systémem a z toho důvodu jsou dostupné pouze RTEMS Classic API funkce.

Výčet funkcí pro práci s přerušením:

- *rtems_interrupt_catch()* – založí nově specifikovanou službu přerušení (ISR - interrupt service routine) pro určený vektor přerušení. Vrací předešlý ISR vektor.
- *rtems_interrupt_disable()* – zruší všechny maskovatelná přerušení a vrátí předchozí úroveň
- *rtems_interrupt_enable()* – povolí maskovatelným přerušením úroveň vrácenou předešlým voláním *rtems_interrupt_disable()*
- *rtems_interrupt_flash()* – dočasně povolí maskovatelným přerušením úroveň vrácenou předešlým voláním *rtems_interrupt_disable()*
- *rtems_interrupt_is_in_progress()* - TRUE pokud procesor právě obsluhuje přerušení. Hodnota TRUE označuje, že volající funkce je obsluha přerušení.

Základní nahrazovaná obsluha přerušení byla obsluha od čítače/časovače, kterým je definovaná vzorkovací perioda řídicího systému. Jednou za tuto periodu je nutné provést jeden regulační cyklus popsaný v kapitole o systému PXMC. Tuto obsluhu jsem tedy nahradil vláknem s prioritou hodnoty 5 (dostatečná hodnota pro tento druh operace). zároveň bylo nutné nadefinovat metodu, která bude zachytávat přerušení od čítače. pokud dojde k přerušení tato metoda vyvolá událost, kterou řídicí vlákno odchytává a pokud nastane, povolí vykonání řídicích příkazů.

5.2 Úprava dalších částí systému

V této podkapitole uvedu postup přenesení dalších částí systému do systému běžícího pod operačním systémem RTEMS. Jedná se o základní kontrolní a řídicí funkce.

5.2.1 Přidání kontrolních funkcí

Systém PXMC obsahuje také nezbytné funkce, kterými je nutno periodicky kontrolovat stav celého řízeného objektu. Tyto funkce bylo samozřejmě nutné také převést pod operační systém. Základní kontrolní funkce jsem rozdělil do tří samostatných vláken, přičemž všem jsem přiřadil stejnou prioritu hodnoty 10. Každá s funkcí (vláken) se periodicky spouští po definovaném čase.

První vlákno zodpovědné za kontrolu chybových stavů systému se spouští se jednou za 8ms. Pokud nastane chyba systému kontrolní funkce odpojí a zastaví řídicí proces. Do tohoto vlákna byla také přidána funkce kontrolující nečinnost motorů jednotlivých os. Pokud toto nastane tak po definovaném čase odpojí regulační smyčku řídicí daný nečinný motor.

Další vlákno také kontroluje stav systému a podle něj řídí stav diod na čelním panelu jednotky MARS 8. Tato úloha vyhodnocuje například zda je daná osa v činnosti nebo v chybovém stavu atd. Vlákno vykonávající tuto činnost spouštím každých 250 ms, což je dostatečná doba na to aby diody na panelu neblikaly.

Poslední kontrolní úloha spouštěná každých 10 ms kontroluje stav systému při koordinovaném řízení.

Inicializace těchto kontrolních vláken proběhne po spuštění systému zavoláním metody která vytvoří a spustí tyto vlákna.

5.2.2 Přidání řídicích funkcí

Po přidání všech základních řídicích a kontrolních funkcí systému PXMC jsem do příkazové řádky (shell) RTEMSu začal doplňovat příkazy pro řízení motorů. V první řadě to byly příkazy, které jsem zmiňoval v kapitole o systému PXMC, a to sice příkazy pro absolutní posun motoru na zadanou pozici, zjištění aktuální pozice a zastavení motoru.

Příklad přidání příkazu do shellu:

```
shell_add_cmd("mcgo", "app","go to target position",
shell_do_reg_go);
```

Tento příkaz přidá do příkazové řádky nový příkaz *mcgo m X*, což znamená posun motoru m na absolutní pozici X. Funkce po obdržení příkazu zavolá výkonnou funkci *shell_do_reg_go*, která provede rozkódování příkazu z argumentů příkazové řádky a zavolá příslušnou funkci systému PXMC (v tomto případě *pxmc_go*). Tímto způsobem jsem postupně přidal další potřebné příkazy.

Seznam doposud přidaných příkazů:

- *mcgo m X:* posun motoru m o absolutní souřadnici X
- *mcstop m:* zastaví pohyb motoru m
- *mcap m:* zjistí a vypíše aktuální polohu motoru m v absolutních souřadnicích
- *mchh m:* provede „Hard Home“ motoru m (nastavení do výchozí pozice)
- *mchhall:* provede „Hard Home“ všech řízených os najednou
- *mccoordgrp m1,m2,..:* sdruží uvedené motory do skupiny pro koordinované řízení
- *mccoordmv X1,X2,..:* posune sdružené motory o absolutní souřadnice X1,X2...
- *mcpurge m:* uvolní motor m, pokud se dostane do chybového stavu
- *mcdemo:* provede hard home robotu**,** sdruží všechny motory do skupiny a provede pohyb robotu přes definované pozice

Tento postup přidávání příkazů má svou analogii v původním systému PXMC, která obsahuje příkazový procesor na zpracování příkazů zasílaných po sériové lince. Tento procesor zachytává a rozkódovává příkazy a vykonává příslušné funkce. Tento systém samozřejmě obsahuje mnohem více příkazů a s úpravami by šel použít i se systémem PXMC běžícím pod RTEMSem. Toto je jeden z dalších možných kroků v pokračování portace systému. Pro mou následnou práci na přímé a inverzní kinematice pro robot BlueBot však těchto několik přidaných příkazů stačilo.

Dalším krokem bylo zabezpečení volání jednotlivých řídicích funkci proti dead-locku, čili zatuhnutí systému. Vzhledem k tomu, že součástí volání každého příkazu je nutně struktura *pxmc_state_t* popsána výše, musí se každá operace nad touto strukturou ošetřit zamknutím pomocí mutexu. Mutex neboli binární semafor používám v tomto případě k uzamčení celé funkce. Vytvoří se tak bezpečné místo odkud lze volat řídicí funkce bez obav z kolize nad strukturou *pxmc_state_t*.

Příklad vytvoření mutexu(binárního semaforu):

```
rtems_name pxmc_sem=rtems_build_name('I','D','F','S');//jméno mutexu
rtems_id   pxmc_sem_id;             //id mutexu pro jeho obsluhu
status=rtems_semaphore_create(pxmc_sem,1,RTEMS_PRIORITY|RTEMS_BINARY_SEMA
PHORE|RTEMS_INHERIT_PRIORITY|RTEMS_LOCAL,0,&pxmc_sem_id);
                                    //vytvoření mutexu
```

Příklad uzamčení funkce:

synchronizovaná funkce *pxmc_go*

```
int pxmc_go_sync(pxmc_state_t *mcs, long val, int res, int mode)
{
  int ret;
  pxmc_mcs_lock(mcs);  //uzamčení struktury mcs (pointer na pxmc_state_t)
  ret=pxmc_go(mcs,val,res,mode); //spuštění původní funkce
  pxmc_mcs_unlock(mcs);//odemčení struktury mcs
  return ret;          //návrat
}
```

Tímto způsobem vytvořím soubor synchronizovaných funkcí, které lze bezpečně používat v celém programu.

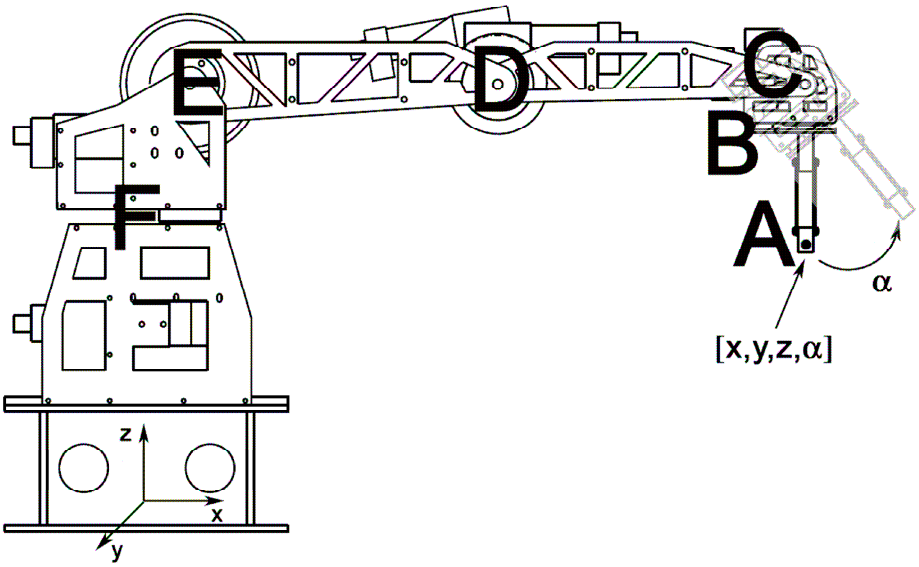
Kapitola 6

Přímá a inverzní kinematická úloha robotu BlueBot

V této kapitole se zabývám řešením přímé a inverzní kinematické úlohy pro školního robota BlueBot. Pomocí těchto řešení ověřím funkčnost řízení mého systému.

6.1 Popis robota BlueBot

Vzhled robota Bluebot1 a volba souřadných os vzhledem ke koncovému bodu ramena jsou dobře patrné z *obrázku 5*. Robot má 5 stupňů volnosti a je složen z rámu, ramena, chapadla a šesti motorů (na obrázku označených velkými písmeny). Všechny motory tvoří rotační vazby. Chapadlo obsahuje mechanismus pro uchopení předmětu (dva prsty, které lze přiblížit nebo oddálit - motorem A). Motor B představuje natočení chapadla o úhel γ, motor C odpovídá úhlu $\alpha 2$, D úhlu ω, E úhlu ψ a F odpovídá úhlu φ. Orientace všech úhlů je patrná z *obrázku 7*.



Obrázek 5: Robot Bluebot1, orientace os, vyznačení koncového bodu (0)

6.2 Řešení přímé kinematické úlohy

Vyřešením přímé kinematické úlohy robota BlueBot získám nástroj pro ověření správnosti následného výpočtů inverzní kinematické úlohy. Pro výpočet PKÚ jsem sestavil transformační matice a v prostředí Matlab jsem napsal m-file, který ze zadaných úhlu natočení motorů spočítá koncový bod chapadla. Transformace provádím v homogenních(projektivních) souřadnicích $\bar{x} = (x; y; z; 1)$, pak platí, že

$$\bar{x} = A\bar{x}_0 ,$$

kde $\bar{x}$ je koncový bod chapadla, $\bar{x}_0$ počátek a **A** je matice 4x4:

$$A = \begin{bmatrix} R & \bar{x}_0 \\ 000 & 1 \end{bmatrix} \text{ a } R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ rotační matice o úhel } \alpha$$

Transformační matice pro robot BlueBot:

(všechny rotace jsou prováděny tak, aby osa Z ležela na rameni robotu, v maticích jsou dosazeny konkrétní hodnoty rozměrů robotu)

A1 otočení o úhel motoru F (úhel φ) kolem osy *z* a posun z bodu (0,0,0) do bodu E:

$$A1 = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 364 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A2 otočení o úhel motoru E (úhel ψ) kolem osy *y* a posun z bodu E do bodu D:

$$A2 = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 250 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A3 otočení o úhel motoru D (úhel ω) kolem osy *y* a posun z bodu D do bodu C:

$$A3 = \begin{bmatrix} \cos\omega & 0 & -\sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ \sin\omega & 0 & \cos\omega & 250 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A4 otočení o úhel motoru D (úhel α2) kolem osy *y* a posun z bodu C na konec chapadla:

$$A4 = \begin{bmatrix} \cos\alpha 2 & 0 & -\sin\alpha 2 & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha 2 & 0 & \cos\alpha 2 & 136{,}5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Výsledná transformační matice **A** tedy bude:

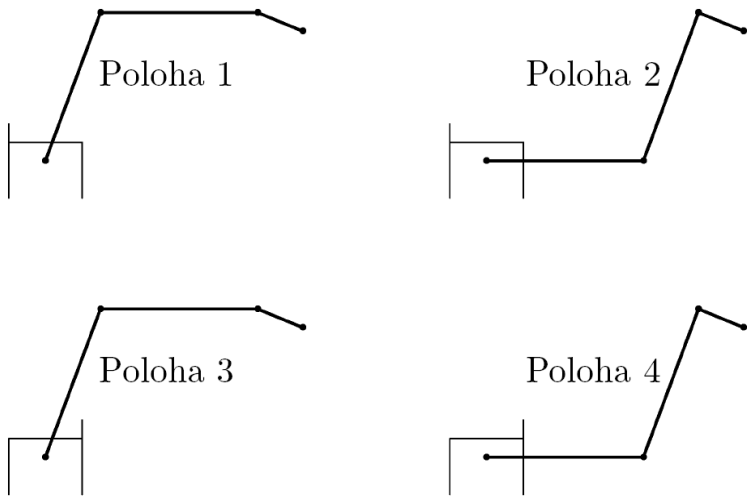
$$A = A1A2A3A4$$

6.3 Řešení inverzní kinematické úlohy

Úkolem úlohy je analyzovat kinematiku robotu, navrhnout postup výpočtu inverzní kinematické úlohy (IKU) a její řešení implementovat v jazyce C. Celé řešení je pak nutné zakomponovat do řídicího systému. Robot je řízen v kloubových souřadnicích (IRC) jednotlivých motorů. Vstupem robota bude zadaná poloha koncového bodu ramene v souřadném systému (*x; y; z*). Robot se na pokyn přesune tak, aby souřadná poloha spojená s chapadlem byla identická se zadanou polohou. Chybové stavy budou ošetřeny.

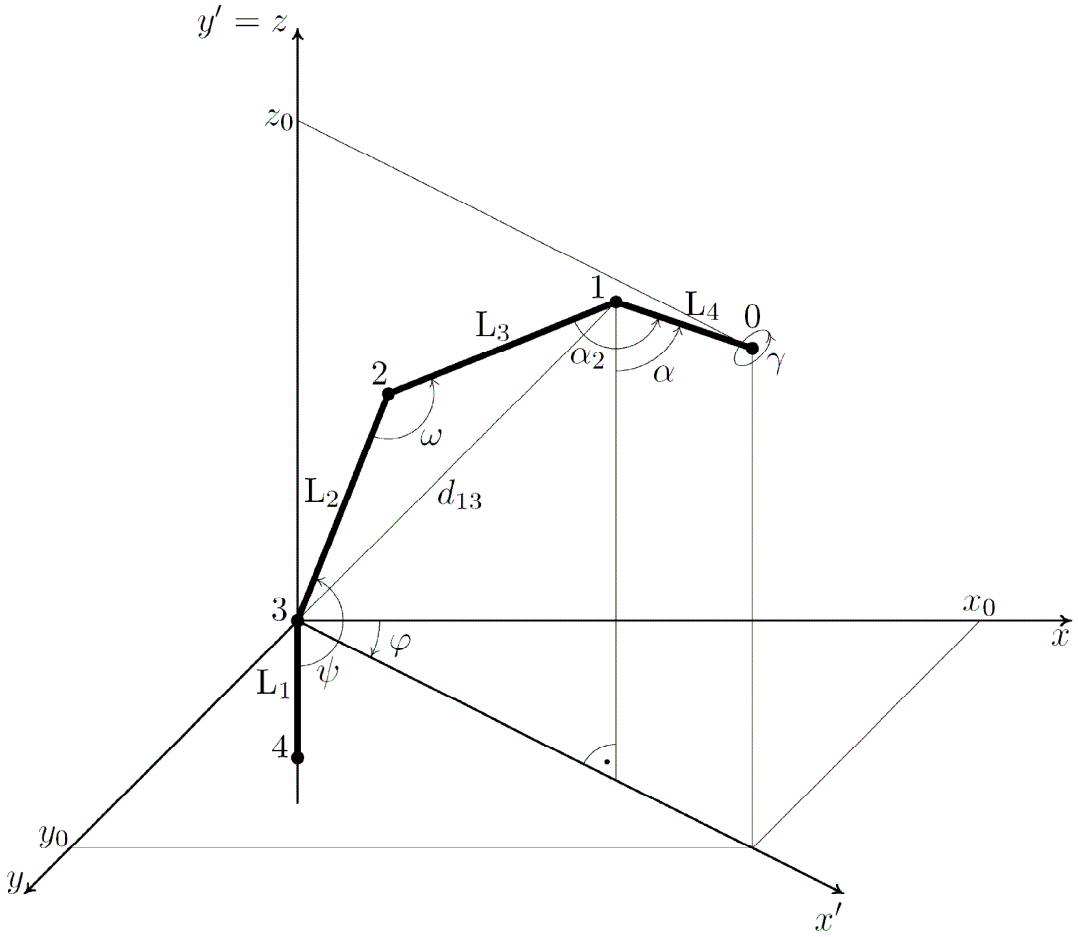
6.3.1 Matematické řešení IKU

Jelikož lze najít pro některé body z operačního prostoru robota, v tomto případě až čtyři možné polohy ramena (viz *obrázek 6*), je nutné také zadat, kterou z možných pozic má rameno robota zaujmout. Pro zjednodušení a zautomatizování celého procesu, výsledný systém obsahuje cyklus, který vybere první realizovatelnou pozici ramene robota a do té nastaví, což postačuje pro potřeby této práce, kterou chci tímto způsobem pouze otestovat. Vstupními proměnnými pro zadání polohy robota tedy jsou souřadnice koncového bodu ramena [*x; y; z*], úhel α (sklon chapadla od svislé osy), nastavení úhlu γ (otočení chapadla) a proměnná určující stisk chapadla. Úkolem algoritmu řešení IKU je tedy vypočítat hodnoty zbylých úhlù, tj. $\alpha 2, \omega, \psi, \varphi$. Následující řešení obsahuje popis výpočtu pro první polohu. Výpočet úhlů pro ostatní polohy je podobný, liší se jen ve znaménkách a smyslech úhlů. Výpočty pro všechny pozice lze nalézt v souboru příkazů systému cmds.c.



Obrázek 6: Možné polohy ramena

Pro jednoduchost a názornost výpočtu položím robota do kartézského souřadného systému *xyz* s počátkem v bodě 3 naznačeném na obrázku 3. Polohu koncového bodu ramene (bod 0) v tomto souřadném systému označme [x_0; y_0; z_0]. V takto navrženém souřadném systému tvoří rameno rovinu *x'y'*.



Obrázek 7: Geometrie robota, položení ramene do roviny *x'y'*.

Souřadnice koncového bodu ramene v této rovině lze vyjádřit ve tvaru:

$$x_0' = \sqrt{x_0^2 + y_0^2}$$

$$y_0' = z_0$$

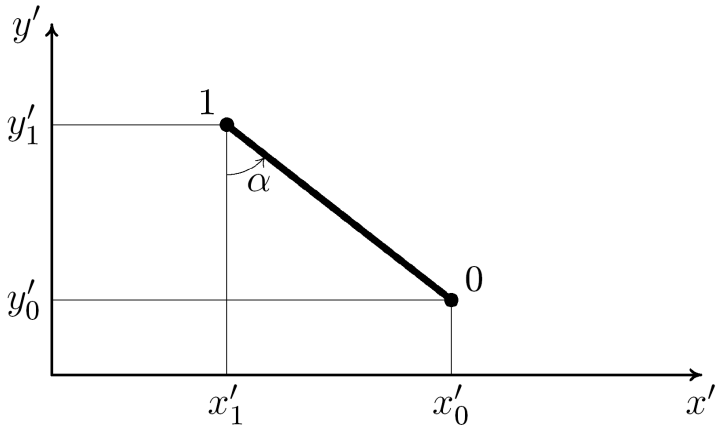
Úhel natočení roviny *x'y'* od roviny *xy* je dán vztahem:

$$\varphi = acrtg \frac{y_0}{x_0}$$

Jelikož již znám polohu bodu 0 v rovině *x'y'* a požadovaný úhel α (ze zadání), mohu podle *obrázku 8* vypočítat polohu bodu 1:

$$x_1' = x_0' - L4 \sin\alpha$$

$$y_1' = y_0' + L4 \cos\alpha$$



Obrázek 8: Výpočet polohy bodu 1 z polohy bodu 0

Nyní spočítám vzdálenost d_{13} bodů 1 a 3. Platí:

$$d_{13} = \sqrt{x_1'^2 + y_1'^2}$$

Protože platí L2 = L3, je trojúhelník s vrcholy v bodech 123 rovnoramenný. Z vlastností rovnoramenného trojúhelníka vyplývá, že:

$$\omega = 2 \arcsin \frac{d_{13}}{2L1}$$

Z polohy bodu 1 a z velikosti úhlu ω lze vypočítat úhel ψ jako:

$$\psi = arctg \frac{y_1'}{x_1'} + \frac{180 - \omega}{2} + 90$$

Zbývá dopočítat jen úhel $\alpha 2$, pro který platí:

$$\alpha 2 = 90 - arctg\frac{y_1'}{x_1'} + \frac{180-\omega}{2} + \alpha$$

Při praktické realizaci, kdy se jako nula v ose *z* bere úroveň podložky, na které robot stojí, je tedy nutné brát v úvahu také vzdálenost L1 a velikost podstavce d_{pod}. Proto místo vztahu $y'_0 = z_0$ počítám se vztahem:

$$y_0' = z_0 - L1 - d_{pod}$$

Ostatní výpočty zůstanou beze změny.

6.3.2 Ošetření chybových stavů

Ne každé řešení, popsané výše uvedeným způsobem, bude fyzikálně realizovatelné. Proto musím při výběru řešení vyřadit ta, která nelze uskutečnit a která vedou ke kolizi robota. Při nastavování polohy ramene robota zejména musíme kontrolovat:

- Jestli nevyžadujeme polohu do které robot nedosáhne. Vzdálenost d_{13} musí být menší nebo maximálně rovna součtu délek L2+L3:

$$d_{13} \leq L2 + L3$$

- Zda lze všechny motory nastavit podle velikosti vypočtených úhlů. Jinými slovy, zda každý úhel leží v daném realizovatelném intervalu.

- Zda požadovaný koncový bod ramene neleží uvnitř konstrukce rámu robota. Požadované body nesmí ležet uvnitř válce o poloměru 10cm a výšky 44cm, který leží na podložce a osa válce splývá s osou y_0.

$$x_0 < 100 \quad \wedge \quad y_0 < 440$$

Kapitola 7

Otestování funkce systému

V této kapitole popíši postup při testování aplikace s využitím výsledků předchozí kapitoly a zároveň naznačím další možnosti, jak výsledky využít.

Výsledky předešlé kapitoly jsou přímá a inverzní kinematická úloha pro školní robot BlueBot. Tyto početní operace jsem do systému zakomponoval jako další příkazové řádky (shell) aplikace. Konkrétně se tedy jde o následující příkazy:

Přímá kinematická úloha:

mcdkt X1,X2,X3,X4,X5 – kde vstupy (Xn) jsou požadované úhly natočení jednotlivých os v pořadí B,C,D,E,F (viz popis robotu), tyto úhly jsou lineárně převedeny na souřadnice motoru v IRC.

Vzorec pro přepočet:

$$IRC = \frac{(Xn - \alpha_{poč}).(IRC_{kon} + IRC_{poč})}{\alpha_{kon} - \alpha_{poč}} + IRC_{poč}$$

kde: $\alpha_{poč}$ | $IRC_{poč}$ – úhel | IRC osy u úhlu Xn v prvním dorazu

α_{kon} | IRC_{kon} – úhel | IRC osy u úhlu Xn v koncovém dorazu

Inverzní kinematická úloha:

mcikt x y z α β *n p* – kde (x,y,z) jsou souřadnice požadovaného bodu, α - je úhel chapadla vůči ose z (při úhlu 0° svírá chapadlo s podložkou úhel 90°), β - je natočení chapadla motorem B, n – nabývá hodnot 0,1 a značí otevření, zavření chapadla a parametrem p lze volitelně nastavit požadovanou pozici robotu.

Další možnost jak vyžít výsledky DKÚ a IKÚ je v pohybu robotu po přímce pomocí splinů. Tento matematický aparát je v původním systému již zakomponován a šlo by tedy s výsledky IKÚ pro jednotlivé body na přímce určit polynomy, které by určovaly, jak řídit motory robotu, aby se pohyboval plynule po přímce.

Závěr

Výsledkem této bakalářské práce je funkční systém víceosého řízení DC motorů PXMC běžící pod operačním systémem RTEMS. Tímto jsem splnil hlavní bod zadání mé práce. Výhody plynoucí z přenesení pod operační systém reálného času jsem popsal v příslušných kapitolách této práce. Tímto vznikl prostor pro další rozvoj aplikace a využití všech dalších možností z této portace vyplývající. V rámci seznámení se s procesorovými deskami MO_CPU1 jsem blíže poznal a pochopil význam hardwarové části stojící za tímto systémem. Práce byla také přínosem v tom, že jsem poznal strukturu a možnosti použití moderního nekomerčního operačního systému reálného času RTEMS a tyto poznatky mohu nadále rozvíjet ve své profesní kariéře.

Dále je namístě naznačit možnosti dalšího pokračování této práce. Dle posledního bodu zadání jsem se měl pokusit o doplnění systému o propojení na komunikaci CAN/CANopen. Tento bod jsem z časových důvodů nesplnil a měl by tedy být prvním krokem v další práci. Vhodné by také bylo do systému zakomponovat původní příkazový procesor systému PXMC, který zpracovával příkazy ze sériové linky. Další možnost je zprovoznění Z-modemu pro tento systém, což by umožnilo za běhu do systému nahrávat soubory například se souřadnicemi pro pohyb robotu. Tento výčet není samozřejmě kompletní a možností je daleko více. Vzhledem k tomu, že mě tato práce bavila a byla velice přínosná i vzhledem ke spolupráci s vedoucím mé práce panem Píšou, rád bych se pokusil tento projekt dále rozvíjet v mé diplomové práci.

Literatura

[1] RTEMS – oficiální stránky projektu RTEMS www.rtems.org
[2] OCERA – Open Components for Embedded Real-time Applications stránky projketu www.ocera.org
[3] X35POS – stránky předmětu Počítačové systémy http://dce.felk.cvut.cz/por/
[4] Pavel Píša – Řídicí jednotka s mikrokontrolérem MC68673
[5] O.Vysoký, P. Píša - CONTROLLERS FOR STANDARD AND PARALLEL ROBOTS
[6] Pikron s.r.o. – stránky výrobce desky MO_CPU1 a jednotky MARS 8 www.pikron.com
[7] Isabelle Rieucros – PXMC Documentations

Seznam příloh

Obsah přiloženého CD:

- elektronická podoba této práce ve formátu .pdf
- zdrojové kódy aplikace:
    - adresář **rtems-muj-test**:
        - kódy testování programů pod RTEMS:
        - bezChybysoubehu.c
        - helloworldMT.c
        - read_file.c
        - s_chybou.c
        - semafor.c
        - sig_kill.c
        - wait.c
    - adresář **rtems-mars8**:
        - podadresář **apfoo**:
            - cmds.c – příkazy pro shell
            - check_task.c – kontrolní vlákna
            - init.c-inicializační program
            - mars_dkt.c-přímá kinematická úloha
            - mars_ikt.c-inverzní kinematická úloha
            - pxmc_coord_sync.c- soubor synchronizovaných funkcí
            - pxmc_sync.c-soubor synchronizovaných funkcí
            - system.h- programové definice
            - app_def.h-aplikační definice
        - podadresář **libs4c/pxmc***:*
            - mo_rtems.c – obsluha přerušení, základ regulace
- pxmc_doc: dokumentace k systému PXMC od Isabelle Rieucros

Použitý software:

- Operační systém: **Debian GNU Linux** jádro 2.6.15
- **Seyon**: 688376 serial interface
- **KDevelop 3.2.2** vývojové prostředí C/C++
- **RTEMS** snapshot 4.6
- **DDD** debugovací prostředí verze 3.3.11