

# **Knihovna pro ovládání úchopné hlavice UG20**

Library for Control of Grasping End-effector UG20

Stanislav Vláčil

---

Bakalářská práce  
2013



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2012/2013

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Stanislav VLÁČIL**  
Osobní číslo: **A10068**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **prezenční**

Téma práce: **Knihovna pro ovládání úchopné hlavice UG20**

Zásady pro vypracování:

1. Seznamte se programovacím jazykem VAL3 pro robot Stäubli UNIMATION TX40.
2. Seznamte se s ovládáním úchopné hlavice přes řídicí jednotku.
3. Vytvořte knihovnu v jazyce VAL3 pro ovládání úchopné hlavice UG20.
4. Na vhodně zvolených příkladech demonstруйте možnosti vytvořené knihovny.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. CS8C Controller : Instruction manual. Faverage: Stäubli, 2007, 186 s.
2. VAL3 Reference Manual. Faverage: Stäubli, 2007, 186 s.
3. Arm ? TX series 40 family. Faverage: Stäubli, 2007, 84 s.
4. Základní popis, funkce a ovládání: Stäubli roboty řady RS, TX, RX. Zlín, 2010, 70 s.
5. ZMEŠKAL, Marek. Průmyslový robot Stäubli UNIMATION TX40 ? uživatelský manuál. Zlín, 2009. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně.
6. JAZAR, Reza N. Theory of Applied Robotics: Kinematics, Dynamics, and Control. New York: Springer, 2007. ISBN 9780387689647.
7. CRAIG, John J. Introduction To Robotics: Mechanics And Control, 3/E. New Jersey: Pearson Education, 2008. ISBN 9788131718360.
8. Assembly and Operating Manual 2-Finger Parallel Gripper Type EGN. Bahnhofstr, 2008.
9. Library serialPortEn Documentation: Description and requirements. Stäubli, 2008.

Vedoucí bakalářské práce:

**Ing. Petr Navrátil, Ph.D.**

Ústav řízení procesů

Datum zadání bakalářské práce:

**24. února 2013**

Termín odevzdání bakalářské práce:

**14. června 2013**

Ve Zlíně dne 24. února 2013

prof. Ing. Vladimír Vašek, CSc.

*děkan*



prof. Ing. Vladimír Vašek, CSc.

*ředitel ústavu*

## **ABSTRAKT**

Bakalářská práce se zabývá tvorbou knihovny k úchopné hlavici UG20, která je připojena jako nástroj k průmyslovému robotu Stäubli UNIMATION TX40. Teoretická část obsahuje základní popis robota, úchopné hlavice, programovacího jazyka VAL3 a vývojového prostředí Stäubli Robotics Studio. Popisuje způsob komunikace hlavice přes rozhraní RS232. Praktická část se skládá z popisu vytvořené knihovny, ukázkou některých funkcí a příkladem jejího využití.

Klíčová slova: knihovna, úchopná hlavice, průmyslový robot, VAL3, RS232, Stäubli

## **ABSTRACT**

The bachelor thesis deals with the creation of library for control of grasping end-effector UG20, which is attached as an tool of industrial robot Stäubli UNIMATION TX40. A theoretical part of thesis presents a basic description of grasping end-effector, VAL3 programming language and integrated development environment Stäubli Robotics Studio. Describes the communication interface RS232. A practical part consists of a description of the created library, samples of some library functions and examples of its use.

Keywords: library, grasping end-effector, industrial robot, VAL3, RS232, Stäubli

Rád bych poděkoval Ing. Petru Navrátilovi Ph.D., vedoucímu mé bakalářské práce, za jeho pomoc a poskytnuté rady a připomínky při tvorbě bakalářské práce. Rád bych poděkoval také svým rodičům za podporu ve studiu.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 30.5.2013

.....  
Stanislav Vláčil

**OBSAH**

<b>ÚVOD.....</b>	<b>9</b>
<b>I TEORETICKÁ ČÁST .....</b>	<b>10</b>
<b>1 POPIS ROBOTY STÄUBLI TX40.....</b>	<b>11</b>
1.1 POPIS ROBOTICKÉHO RAMENE .....	11
1.1.1 Popis jednotlivých částí ramene .....	12
1.1.2 Technické vlastnosti ramene .....	12
1.1.3 Využití ramene .....	14
1.2 POPIS RUČNÍHO PROGRAMOVACÍHO PANELU .....	14
1.2.1 Podrobný popis ručního programovacího panelu.....	15
1.2.1.1 Volba pracovního módu.....	15
1.2.1.2 Napájení motorů .....	17
1.2.1.3 Nouzový vypínač .....	17
1.2.1.4 Ovládání navigace a pohybu .....	17
1.2.1.5 Múd pohybu .....	17
1.2.1.6 Nastavení rychlosti pohybu.....	19
1.2.1.7 Funkční klávesy .....	20
1.2.1.8 Alfanumerické klávesy .....	20
1.2.1.9 Klávesy pro navigaci a volbu rozhraní .....	20
1.2.1.10 Řízení aplikací.....	21
1.2.1.11 Aktivační spínač.....	22
1.2.1.12 Tlačítka ovládání IO.....	22
1.2.1.13 Generování pohybu ramene v manuálním módu .....	23
1.2.1.14 Osvětlení panelu.....	23
1.3 KONTROLÉR CS8C.....	23
<b>2 POPIS ČELISTÍ.....</b>	<b>25</b>
2.1 POPIS ČELISTÍ ENG80 .....	25
2.2 KONTROLÉR MCS-12.....	27
2.2.1 Technické údaje kontroléru MCS-12 .....	28
2.2.2 Rozložení kontroléru .....	28
2.2.3 Propojení kontroléru MCS-12 s kontrolérem CS8C .....	29
2.3 KOMUNIKAČNÍ PROTOKOL .....	30
2.3.1 Formát dat .....	30
2.3.2 Práce s desetinou čárkou .....	31
2.3.3 Komunikace přes RS-232.....	31
2.3.4 Funkce CRC16 .....	33
2.3.5 Příklad komunikace přes RS-232.....	34
2.3.6 Variace příkazů čelistí .....	34
<b>3 POPIS ROZHRANÍ RS-232 .....</b>	<b>36</b>
<b>4 STÄUBLI ROBOTICS STUDIO.....</b>	<b>38</b>

4.1	VAL3 STUDIO .....	39
4.2	TRANSFER MANAGER.....	40
4.3	PROGRAMOVACÍ JAZYK VAL3 .....	41
4.3.1	Aplikace .....	42
4.3.2	Programy .....	42
4.3.3	Datové typy .....	43
4.3.4	Příklad jednoduchého programu .....	44
4.3.5	Knihovna v jazyce VAL3.....	44
4.3.6	Vybrané instrukce pro práci se sériovou linkou .....	45
<b>II</b>	<b>PRAKTICKÁ ČÁST .....</b>	<b>47</b>
<b>5</b>	<b>POPIS VYTVOŘENÉ KNIHOVNY .....</b>	<b>48</b>
5.1	VYBRANÉ PŘÍKAZY A POPIS JEJICH FUNKCE .....	49
5.1.1	Referenční pohyb .....	49
5.1.2	Přesun pozici .....	50
5.1.3	Úchop .....	50
5.1.4	Nastavení požadované rychlosti .....	51
5.1.5	Nouzové zastavení .....	52
5.1.6	Zjištění konfigurace.....	52
5.2	STRUKTURA VYBRANÝCH PŘÍKAZŮ .....	53
5.2.1	Program pro inicializaci knihovny .....	53
5.2.2	Program pro výpočet CRC16 .....	54
5.2.3	Program pro čtení vstupu .....	55
5.2.4	Program pro odeslání zprávy.....	56
5.2.5	Program vybrané výstupní funkce.....	57
<b>6</b>	<b>UKÁZKOVÝ PROGRAM S VYUŽITÍM KNIHOVNY .....</b>	<b>59</b>
	<b>ZÁVĚR .....</b>	<b>61</b>
	<b>ZÁVĚR V ANGLIČTINĚ .....</b>	<b>62</b>
	<b>SEZNAM POUŽITÉ LITERATURY .....</b>	<b>63</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>64</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>65</b>
	<b>SEZNAM TABULEK.....</b>	<b>67</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>68</b>

## ÚVOD

Už od počátku civilizace lidí byla vždy snaha si práci zjednodušit a také ji zefektivnit. V začátcích k tomu sloužili obyčejné nástroje a stroje obsluhované člověkem. První myšlenky o robotech sahají až do starověku ale roboty, jaké známe dnes, byly zkonstruovány až téměř v druhé polovině dvacátého století. Roboty se používají tam, kde svou práci odvedou lépe díky své produktivitě, přesnosti a odolnosti. Své využití mají také tam, kde lidem mohou ušetřit špinavé, nebezpečné a nudné práce.[9]

Průmyslový robot Stäubli TX40 má své využití v mnoha aplikacích, kde je snaha proces automatizovat. Robotická ramena jsou velice univerzální ve svém nasazení, bez ohledu na průmyslové odvětví nebo typ aplikace, jsou neuvěřitelně přesná, dokážou vykonávat práci bez zastavení a díky šesti stupňům volnosti se přizpůsobí jakékoliv činnosti. Robotická ramena se uplatní v obráběcích aplikacích, při paletizaci, při aplikacích svařování, povrchových úprav materiálů, ve slévárnách při lití kovů a plastů, při manipulaci tvářecích stojů a ve spoustě dalších aplikací. V dnešní době je jedním z největších příkladů nasazení těchto robotů v oblasti výroby automobilů. Automatizované továrny bývají vybaveny stovkami průmyslových robotů, kteří pracují na plně automatizovaných výrobních linkách. Díky obrovské variabilitě robota Stäubli Unimation TX40 patří tento typ k nejrozšířenějším.[1][9]

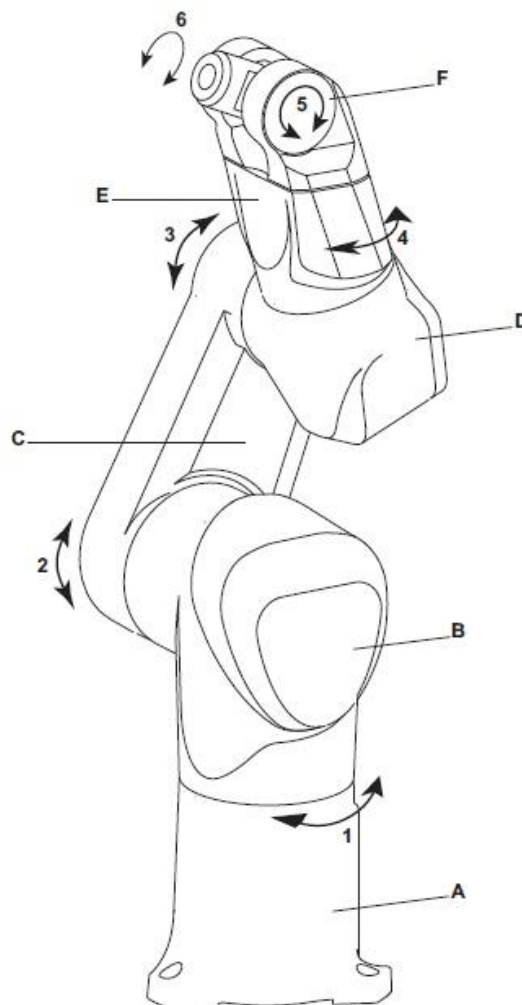
## **I. TEORETICKÁ ČÁST**

## 1 POPIS ROBOTA STÄUBLI TX40

Průmyslový robot se skládá ze tří hlavních částí, robotického ramene, kontroléru a ručního programovacího panelu MCP(Motion Control Panel).[1]

### 1.1 Popis robotického ramene

Rameno se skládá ze částí, které jsou vzájemně propojeny klouby. Umožňuje pohyb v šesti osách, což zajišťuje velký rozsah pohybů. Pohyb kloubů je zajištěn servomotory, které jsou spojeny se snímači polohy. Toto zajistí, že poloha robota je známá v každé situaci. Vysoký stupeň krytí IP65 umožňuje nasazení ramene v náročných podmínkách. Konstrukce umožňuje montáž ramene do různých poloh např. na strop, konstrukci na stěně nebo na podlahu.[2]



Obr. 1. Robotické rameno[2]

*Obr. 1. Robotické rameno[2]***1.1.1 Popis jednotlivých částí ramene**

Na obrázku (Obr. 1.) je zobrazeno robotické rameno. Zde je uveden popis jednotlivých částí podle popisků v obrázku. [2]

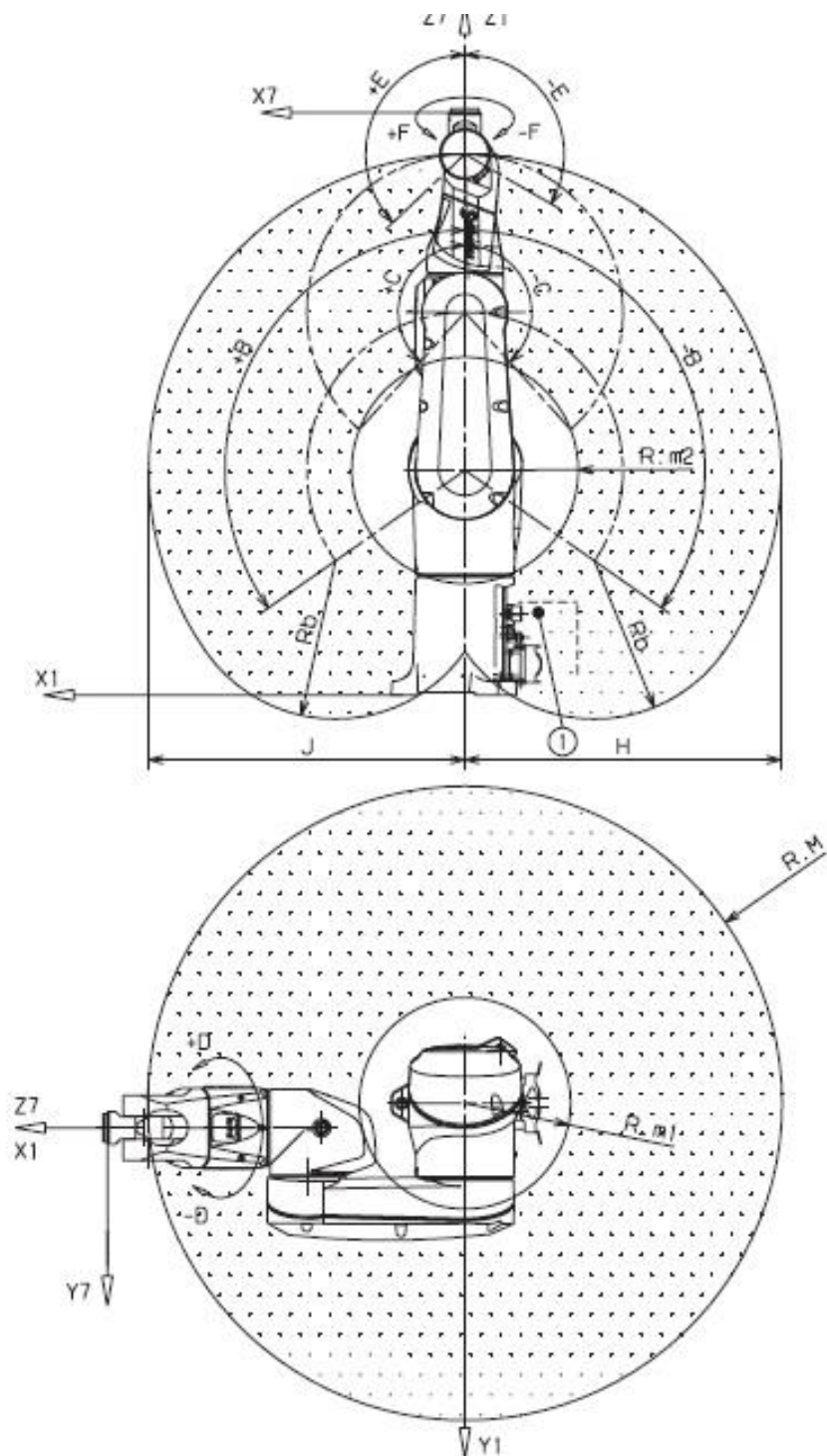
- A- Základna
- B- Rameno
- C- Paže
- D- Loket
- E- Předloktí
- F- Zápěstí

**1.1.2 Technické vlastnosti ramene**

V následující tabulce jsou uvedeny základní vlastnosti kloubů ramene rychlosti, úhly možného natočení a odchylky.[2]

Kloub	1	2	3	4	5	6
Rozsah pohybu [°]	360	250	276	540	253.5	540
Výchylka z pozice [°]	A ±180	B ±125	C ±138	D ±270	E ±133.5	F ±270
Rychlost [°/s]	287	287	430	410	320	700
Přesnost[°·10 <sup>-3</sup> ]	0.057	0.057	0.122	0.114	0.122	0.172

*Tab. 1. Tabulka vlastností ramene[2]*



Obr. 2. Znáznornění možností pohybů ramene[2]

### 1.1.3 Využití ramene

Rameno je zkonstruováno tak, aby se snadno udržovalo a zároveň jeho nasazení v náročných podmínkách bylo bezproblémové.

Rameno umožňuje montáž libovolného nástroje. Nástroj je možné umístit na zápěstí robota.

Nástroje mohou být mechanické, pneumatické nebo elektrické. Všechny by měly být originální a nainstalovány odborně jinak hrozí ztráta záruky.[1]

## 1.2 Popis ručního programovacího panelu

Robot obsahuje terminál tzv. ruční programovací panel MPC (Motion Control Panel). Skládá se z klávesnice a zobrazovací jednotky LCD, která slouží jako výstupní rozhraní pro komunikaci s uživatelem. Panel lze využít k manuálnímu ovládní robota. Z panelu lze spouštět vytvořené programy, volit módy robota. Obsahuje také tlačítko nouzového vypnutí robota.[3]



Figure. 6.2

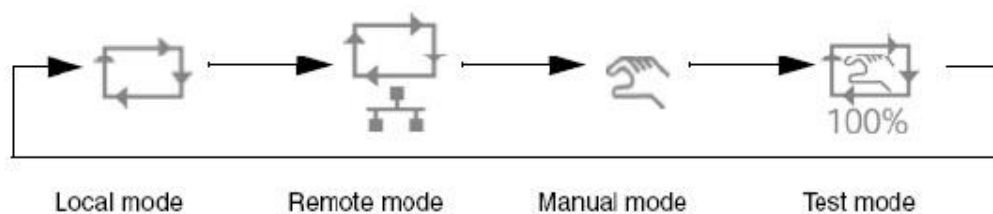
Obr. 3. Manuální programovací panel[3]

- 1 - Volba pracovního módu
- 2 - Napájení motorů
- 3 – Nouzový vypínač
- 4 - Ovládání navigace a pohybu
- 5 - Mód pohybu
- 6 - Nastavení rychlosti pohybu
- 7 - Funkční klávesy
- 8 - Alfanaumerické klávesy
- 9 - Klávesy pro navigaci a volby rozhraní
- 10 - Řízení aplikací
- 11 - Aktivační spínač
- 12 - Tlačítka ovládání IO
- 13 – Generování pohybu ramene v manuálním módu

### 1.2.1 Podrobný popis ručního programovacího panelu

#### 1.2.1.1 Volba pracovního módu

Panel umožňuje zvolit z jednoho z pracovních módů robota: lokální mód, mód vzdálené správy, manuální mód, a testovací mód.



Obr. 4. Výběr módu robota[3]

Po stisku příslušného tlačítka pro volbu módu se nám rozsvítí LED u zvoleného módu.[3]

### **Lokální mód**

Lokální mód umožňuje pohyb robota bez přispění člověka. Robot je řízen naprogramovanou aplikací. V tomto módu musí být zapnuty všechny motory. Při ovládání přes MCP se spuštění aplikace provádí pomocí tlačítka Move/Hold. Operátor může v tomto módu měnit pouze rychlost pohybu.[3]

### **Mód vzdálené správy**

Mód je podobný lokálnímu módu. Hlavní rozdíl je v zapínání napájení externím systémem nebo příkazem enablePower. Podle nastavení profilu nemusí být funkční některá tlačítka (Tlačítko Move/Hold nebo tlačítko pro napájení motorů). Robot v tomto módu se může začít kdykoliv po zapnutí pohybovat. V jeho blízkosti se tedy nesmějí pohybovat žádné osoby.[3]

### **Manuální mód**

Rameno je ovládáno přes MCP. Mód slouží hlavně pro učení práce s robotem. Také je tento mód vhodný pro ladění aplikací. Pohyb je vykonáván omezenou rychlostí max. 250mm/s. Pro vykonávání aplikace musí být stisknuto tlačítko Move/Hold a také tlačítko pro aktivaci „Dead Men“ nebo musí být MCP umístěno v držáku „Parking“. V tomto režimu nemůže být robot řízen externím zařízením.[3]

### **Test mód**

Tento mód je podobný manuálnímu. Jestliže je obsluha za bezpečnostní bariérou nedovolí pohyb plnou rychlostí.[3]

### 1.2.1.2 Napájení motorů

Toto tlačítko umožňuje zapnutí nebo vypnutí napájení robotického ramene. Pokud tlačítko svítí napájení motorů je zapnuto. V manuálním módu musí být navíc stisknuto aktivační tlačítko „Dead Men“ nebo musí být MCP umístěno v držáku „Parking“.[3]

### 1.2.1.3 Nouzový vypínač

Nouzový vypínač musí být použit, pokud nastane událost, kdy je to naprosto nezbytné a nepředvídané.

Po stisku tlačítka dojde k zastavení a zabrzdění robotického ramene. Další vybavení by mělo být zastaveno a vypnuto.

Pro opětovné spuštění postupujeme tak, že opustíme pracovní prostor ramene a poté otočíme nouzovým vypínačem o ¼ otáčky po směru hodinových ručiček. Vše musí být provedeno v manuálním módu. [3]

### 1.2.1.4 Ovládání navigace a pohybu

Tyto tlačítka jsou aktivní v manuálním módu a umožní generovat pohyb v jednotlivých osách nebo kartézském systému podle zvoleného módu pohybu. [3]

### 1.2.1.5 Mód pohybu

Pokud je rameno v manuálním módu, umožní tyto tlačítka vybrat požadovaný mód pohybu. Tlačítka jsou spřažena se světelnými indikátory, které se po výběru módu rozsvítí. Máme možnost výběru ze čtyř pohybových módů a to kloub, rám, nástroj a bod. Tlačítka znázorňuje následující obrázek. [3]



Obr. 5. Výběr módu pohybu ramene[3]

#### Mód kloub

Při volbě kloub ovládáme rameno pomocí tlačítek pro pohyb a navigaci (3). Pro pohyb jednotlivých kloubů použijeme jednotlivá tlačítka (2). Pro pohyb v kladném směru „+“

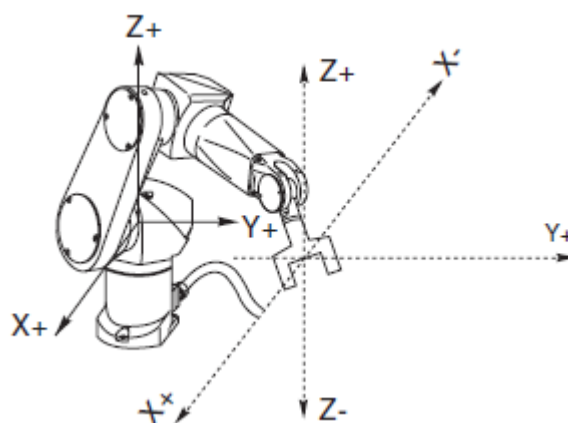
nebo záporném „-“. Tlačítka (4) můžeme volit pomocí „sel“ jednotlivé klouby a druhým tlačítkem opět pohybujeme vybraným kloubem příslušným směrem. Lze ovládat i více kloubů najednou. [3]



Obr. 6. Ovládání pohybu v módu kloub[3]

### Mód rám a mód nástroj

Práce v těchto dvou módech si je velice podobná. Rozdíl je v tom, že v módu rám máme souřadnicový systém vztažený k rámu upevněného robota. Systém má tedy pevný počátek. V módu nástroj, máme systém vztažený ke konci nástroje. Systém má tedy pohyblivý počátek. [3]



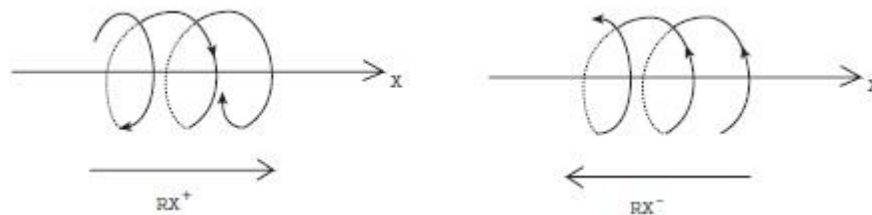
Obr. 7. Souřadnicový systém v módech rám a nástroj[3]

Jako v předchozím módu pro pohyb opět používáme tlačítka (2). Opět lze využít i tlačítka rychlého ovládání (5).



Obr. 8. Ovládání pohybu v módu rám a nástroj[3]

Navíc lze využít rotace kolem jednotlivých os pomocí tlačítek (2), konkrétně tedy tlačítka „RX, RY, RZ“. Důležité je myslet na to, že v módu nástroj se otáčí i souřadnicový systém. [3]



Obr. 9. Znáznornění rotace kolem osy[3]

### Mód bod

Tento mód se využívá pro pohyb do bodu v aplikaci. Pro pohyb na bod v aplikaci budete nejprve vyzváni pro výběr nástroje. [3]

#### 1.2.1.6 Nastavení rychlosti pohybu

Pomocí tohoto tlačítka lze nastavit rychlost pohybu. Nastavení je také závislé na zvoleném pracovním módu. [3]

### 1.2.1.7 Funkční klávesy

Tyto klávesy se využijí pro výběr, který je zobrazen na LCD panelu MCP. Příslušnému tlačítku odpovídá vždy funkce zobrazená nad ním na displeji. [3]

### 1.2.1.8 Alfanumerické klávesy

Tyto klávesy umožňují vkládání dat do aplikací.

### 1.2.1.9 Klávesy pro navigaci a volbu rozhraní

Online nápověda



Pomocí tlačítka „Help“ můžeme kdykoliv vstoupit do online nápovědy. Nápovědu



opouštíme opětovným stiskem tlačítka nebo tlačítkem „Esc“.

Tlačítko „Menu“



Pomocí tlačítka „Menu“ se dostaneme zpět do hlavního menu.

Tlačítko „User“



Pomocí tlačítka „User“ zobrazíme uživatelskou konzolu.

Tlačítka šipek



Tlačítka se využívají pro pohyb a prohlížení v menu.

Tlačítko „End“



Zobrazí nabídku, která je složena. Bývá označena znaménkem „+“.

Tlačítko „Home“



Složí nabídku která je rozbalena. [3]

Tlačítko „Shift“



Umožní přístup k druhotným funkcím tlačítek.

Tlačítko „Esc“



Zruší vstup, vrátí původní hodnoty v poli nebo opustí aktuální stránku.

Tlačítko „Return“



Spustí akci přiřazenou vybrané položce. Umožní modifikaci vybraného pole.

Potvrdí modifikované pole.

Tlačítko „Tab“



Umožní výběr jednotlivých polí.

Tlačítko „Backspace“



Má standardní význam mazání znaků vlevo od kurzoru. [3]

### ***1.2.1.10 Řízení aplikací***

Tlačítko „Stop“



Zastaví provádění aplikace.

Tlačítko „Run“



Umožní spustit aplikaci.

Tlačítko „Move/Hold“



V manuálním módu umožňuje pohyb robota pouze, když jej stiskneme. Po uvolnění

dojde k okamžitému zastavení pohybu.

V lokálním módu a v módu vzdálené správy umožňuje zastavení pohybu ramene. V módu vzdálené správy může být tlačítko neaktivní. [3]

#### 1.2.1.11 Aktivační spínač

Má tři polohy.

1. Rozpojené kontakty pokud není tlačítko stisknuto.
2. Spojené kontakty pokud je stisknuto do střední polohy
3. Rozpojené kontakty pokud je plně stisknuto

Používá se pro autorizace připojení ramene v manuálním módu. Pohyb je umožněn pouze při stisku ve střední pozici. Dvě krajní polohy způsobí vypnutí napájení ramene. Má vliv jen v manuálním módu. [3]



Obr. 10. Aktivační spínač[3]

#### 1.2.1.12 Tlačítka ovládání IO



V manuálním módu umožní změnu stavu zařízení připojených na digitální výstupy.

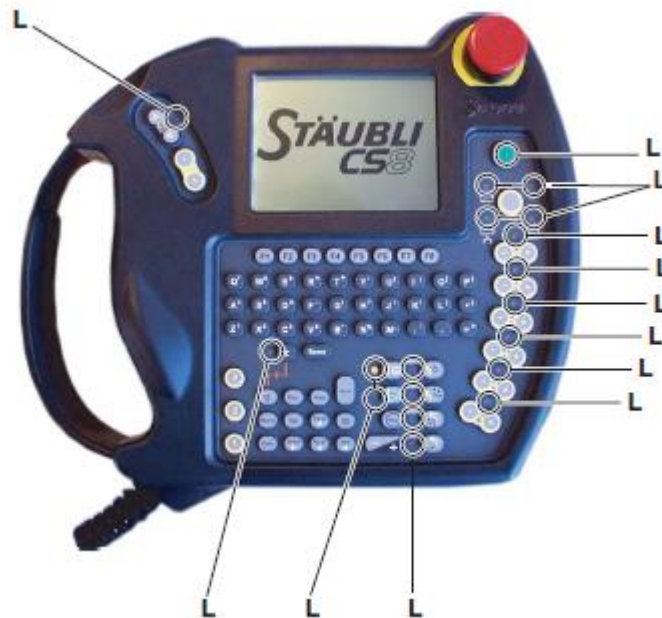
K tlačítkům lze tyto zařízení přiřadit přes MCP.

### ***1.2.1.13 Generování pohybu ramene v manuálním módu***

Tlačítka jsou přístupná v manuálním módu a umožní pohyb ramene v závislosti na zvoleném způsobu pohybu. [3]

### ***1.2.1.14 Osvětlení panelu***

MCP obsahuje osvětlená tlačítka, což umožňuje snazší přehled ve zvolených parametrech, čímž se práce s MCP usnadní. [3]



*Obr. 11. Body osvětlení[3]*

## **1.3 Kontrolér CS8C**

Je to inteligentní část robota, jeho součástí je procesor. Obsahuje všechny aplikace, spouští je a vykonává příkazy. Komunikuje s okolím pomocí různých rozhraní (Ethernet, USB, Sériová linka). Kontrolér je možné rozšířit o různé přídavné moduly podle potřeby uživatele nebo výroby. [3]



Obr. 12. Kontrolér CS8C[3]

Popis částí:

1. Digitálně řízené výkonové zesilovače. Jeden pro každou osu.
2. Generuje stejnosměrné napětí pro rychlostní převody a RSI desku
3. Generuje stejnosměrné napětí pro komponenty počítače
4. Zde jsou seskupeny obvody pro ochranu elektroniky
5. Procesor
6. Hlavní vypínač
7. Transformátor s ochranou. Může být jedno nebo třífázový [3]

## 2 POPIS ČELISTÍ

Čelisti se jako komplet skládají z úchopné hlavice a jejich kontroléru. Hlavice je typ ENG80 s kontrolérem MCS 12. [4]



Obr. 13. Čelisti[4]

### 2.1 Popis čelistí ENG80

Maximální hlučnost čelistí nepřesáhne úroveň 70dB. Další technická data udává následující přehledová tabulka. [4]

Popis	Označení
Označení kleští	306 100
Zdvih pro jeden prst	8mm
Maximální úchopná síla	400N
Stálá úchopná síla	400N
Doporučená hmotnost obrobku	2,1kg
Napájení přes MCS-12	24V DC
Čas otevření	0,35s
Čas sevření	0,35s
Opakovatelná přesnost v silovém režimu	±0,01mm
Přesnost polohování	±0,05mm
Maximální délka prstu	100mm
Hmotnost	0,84kg
Odběr proudu/odběr při zrychlení	2A/6,5A
Délka kabelu	5m
Rozsah operační teploty	5°-55°C

Tab. 2. Technické vlastnosti čelistí[4]

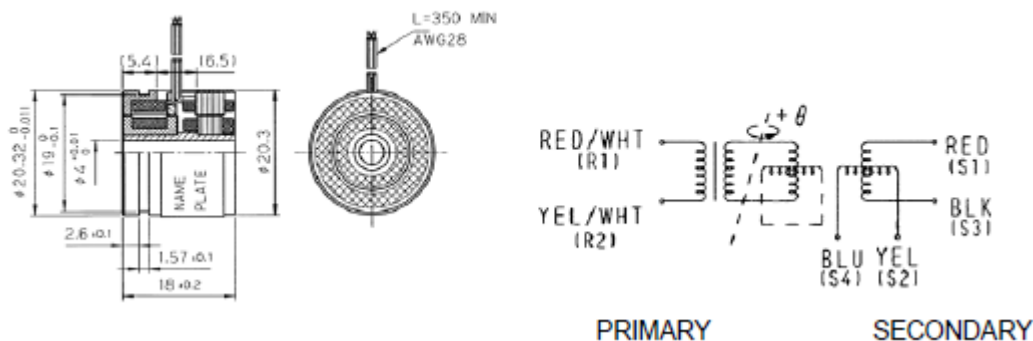
Minimální síla úchopu je přibližně 170N. Všechny síly mají udanou odchylku  $\pm 20\%$ .

Následující tabulka udává technické vlastnosti integrovaného motoru čelistí. [4]

Popis	Zkratka	Označení
Napětí terminálu	$U_{KL}$	24V
Jmenovitý krouticí moment	$M_N$	0,41Nm
Jmenovitý proud	$I_N$	6,0A
Špičkový proud	$I_{EFF}$	7,0A
Momentová konstanta	$K$	0,069Nm/A
Jmenovitá rychlost	$U_N$	1700ot/min
Maximální rychlost	$U_{MAX}$	2200ot/min
Maximální frekvence	$F_{MAX}$	256,7Hz
Výkonová ztráta při $M_N$	$P_V$	9W
Konstanta motoru	$K_M$	0,1N/W
Napětí na vodičích	$U_{ZK}$	24V
Odpor (fáze-fáze)	$R_{\pi 20}$	1,5 $\Omega$
Indukčnost (fáze-fáze)	$L_{\pi}$	1,4mH
Časová konstanta	$\tau_{\xi}$	0,93ms
Počet pólových páru		7

Tab. 3. Technické vlastnosti integrovaného motoru[4]

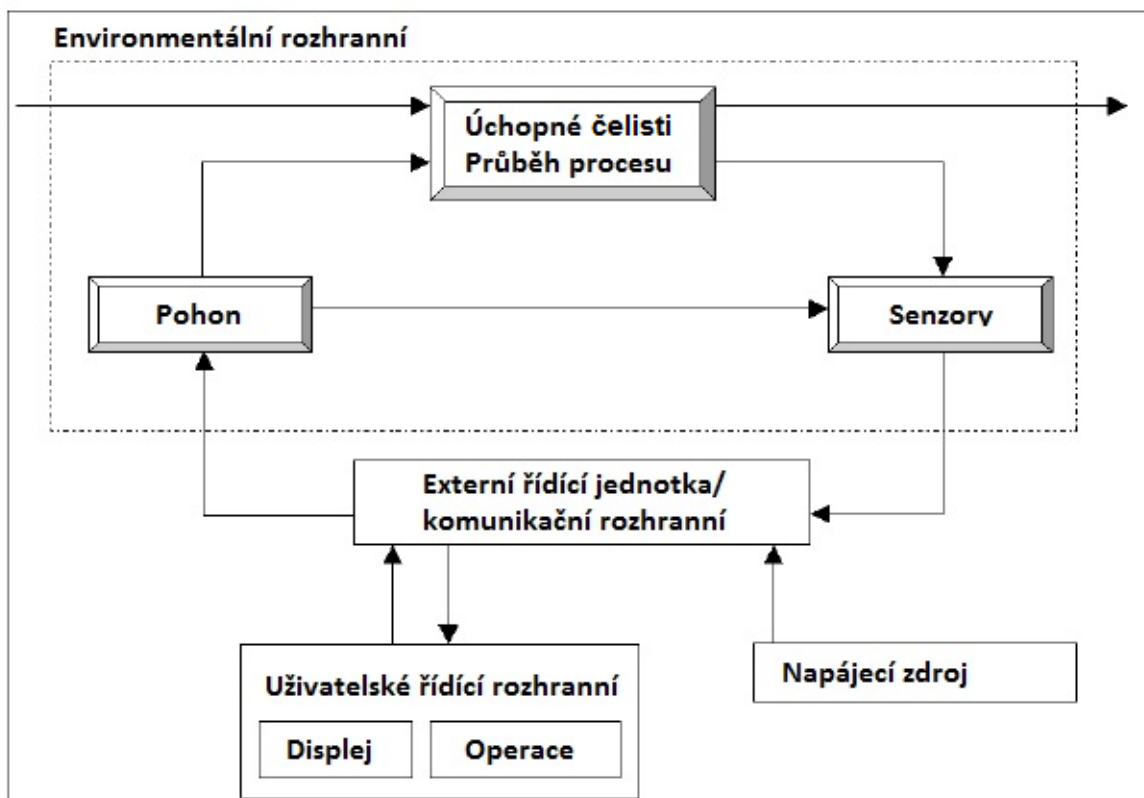
Pozornost je potřeba věnovat možnosti přehřátí motoru. Pokud motor neustále překračuje hodnotu 6,5A může dojít k jeho přehřátí a zničení. V čelistech je integrován dekodér (snímač polohy). Jeho rozměry a zapojení udává následující obrázek. [4]



Obr. 14. Rozměry a zapojení integrovaného dekodéru[4]

## 2.2 Kontrolér MCS-12

Řídicí modul MCS-12 je zkonstruován tak, aby jej bylo možné využít s úchopnými čelistmi různých druhů. Čelisti je potřeba ke kontroléru jen správně připojit. Poté co je kontrolér připojen k čelistem, je řízen hlavním systémem (PC, kontrolér CS8C ). Čelisti poté vykonávají požadovaný mechanický pohyb. Jejich pozice je neustále monitorována. Informace o pozici jsou pak prostřednictvím kontroléru přeposlány do hlavního kontrolního systému. Ovládacímu modulu můžeme definovat různé parametry např. proud, pozice čelistí, rychlost, akcelerace. [4]



Obr. 15. Schéma rozhraní čelistí[4]

### 2.2.1 Technické údaje kontroléru MCS-12

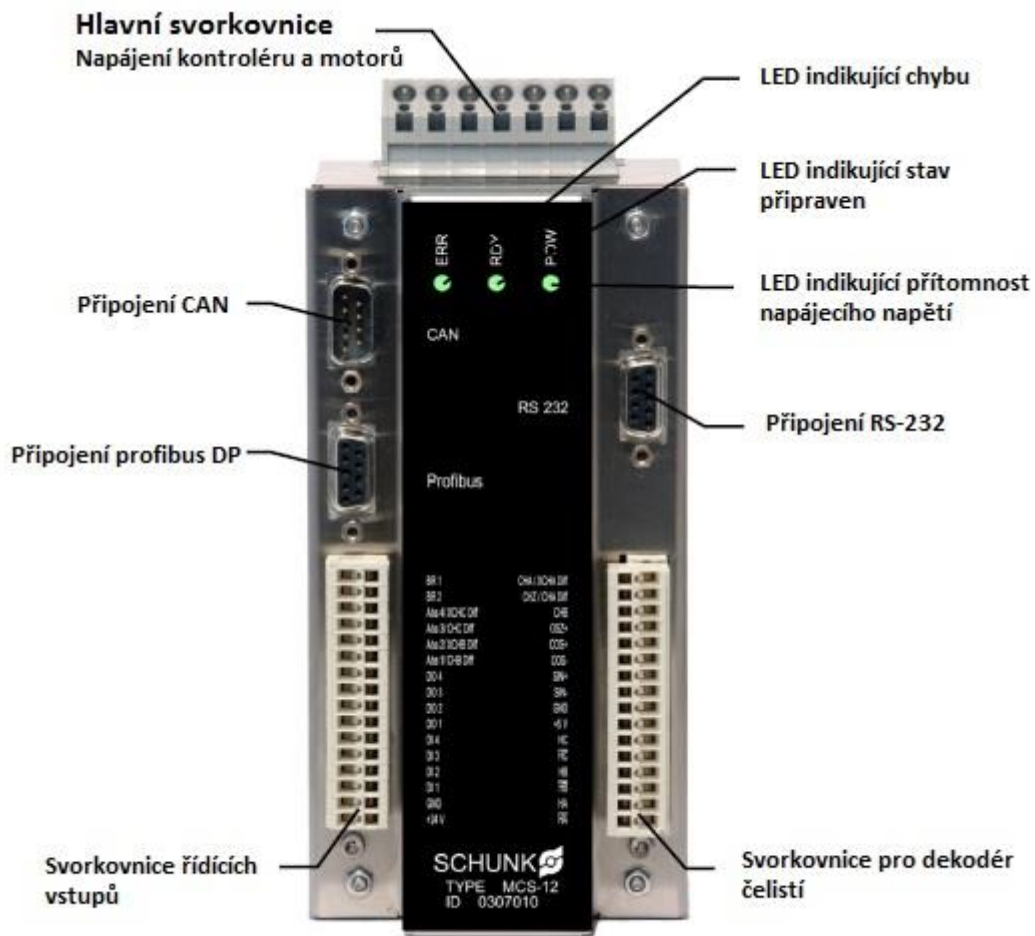
Technické údaje kontroléru udává následující tabulka.

Popis	Označení
Označení kontroléru	307 010
Napájecí napětí řídicí logiky	24V DC
Proudový odběr řídicí logiky	0,5A
Jmenovitý odebíraný proud	12A
Napájecí zdroj	24-48V DC
Hmotnost	0,98kg
Rozhraní	Profibus DP (1,5MBaud)
	RS-232 (9,600 Baud)
	CAN (max. 1MBaud)
Možnosti ovládání	PI-proudová regulace
	PI-rychlostní regulace
	PI-poziční regulace
Senzorový systém	Kodéry
	Dekodéry
Komunikace	Bloková
	Sinusová

*Tab. 4. Technické vlastnosti kontroléru MCS-12[4]*

### 2.2.2 Rozložení kontroléru

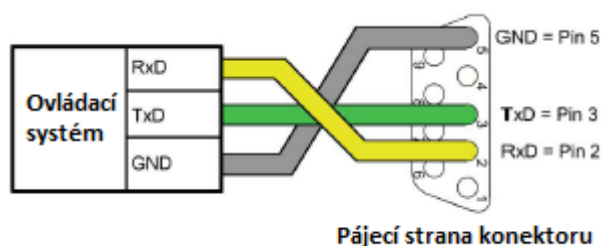
Rozložení kontroléru znázorňuje následující obrázek.



Obr. 16. Rozložení konektorů kontroléru MCS-12[4]

### 2.2.3 Propojení kontroléru MCS-12 s kontrolérem CS8C

Jako hlavní komunikační rozhraní byla zvolena sériová komunikace přes RS-232. Na kontroléru CS8C jsou přítomny dva sériové porty RS-232 s označením J203 (COM1) a J201 (COM2). Pro vlastní spojení je potřeba pouze křížený propojovací kabel RS-232. Komunikace probíhá na nejzákladnější úrovni (bez řízení toku dat atd.), proto jsou potřeba pouze 3 základní vodiče a to RxD, TxD a GND. Vlastní kabel je typu samec (do kontroléru MCS-12) a samice (do kontroléru CS8C). Zapojení znázorňuje následující obrázek. [4]

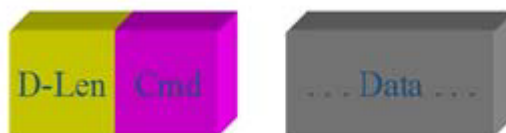


Obr. 17. Spojení kontroléru přes RS-232[4]

## 2.3 Komunikační protokol

Komunikační protokol definuje formát dat, která jsou posílána mezi kontrolérem MCS-12 a nadřazeným systémem (PC, PLC nebo CS8C). Komunikace pro každé zmíněné rozhraní se mírně liší v závislosti na daném rozhraní. Datový rámeček obsahuje vždy následující.

- délku dat (1byte, specifikuje délku následujících dat)
- příkaz (1byte, kód daného příkazu)
- data (potřebná data, délka je dána typem příkazu) [4]



Obr. 18. Společný datový rámeček[4]

### 2.3.1 Formát dat

Data jsou odesílána ve formátu Intel „little Endian“. Z toho plyne, že na prvním místě bude nejméně významný byte a na poledním nejvýznamnější byte. [5]

Číslo 49500000 převedeme do hexadecimální soustavy tedy **2f34f60**. Pro odeslání ve formátu „little Endian“ by toto číslo bylo po bytech zakódováno následovně.

**Little Endian formát - 60 4f f3 02**

### 2.3.2 Práce s desetinou čárkou

Pro zakódování čísel s desetinou čárkou se zde využívá norma IEEE 754. Ta byla vyvinuta na začátku osmdesátých let pro práci s desetinnými čísly na počítačích. Číslo je zde reprezentováno v základní přesnosti 32 bitů na číslo. Rozložení bitů je pak následovné. [5]

Znaménko + nebo -	Exponent	Mantisa
1 bit	8 bitů	23 bitů
s	e	f

Tab. 5. Formát IEEE 754[5]

Vzorec pro učení hodnoty čísla má pak tvar:

$$(-1)^s \cdot 2^{e-127} \cdot (1.f)_{bin}$$

Příklady zakódování.

	Znaménko	Exponent	Mantisa
	1 bitů	8 bitů	23 bitů
7/4	0	01111111	11000000000000000000000
-34.432175	1	10000100	00010011011101010001100
-959818	1	10010010	11010100101010010100000
+0	0	00000000	00000000000000000000000
-0	1	00000000	00000000000000000000000
$2^{-126}$ , nebo $1.175 \cdot 10^{-38}$ <b>Nejmenší kladné číslo</b>	0	00000001	00000000000000000000000
$(2 \cdot 2^{-23})2^{127}$ , nebo $3.403 \cdot 10^{38}$ <b>Největší kladné číslo</b>	0	11111110	1111111111111111111111111
Nekonečno	0	11111111	1111111111111111111111111
Není číslo (NaN)	0	11111111	Ne všechny „0“ nebo „1“

Tab. 6. Příklady zakódování do IEEE 754[5]

### 2.3.3 Komunikace přes RS-232

Pro komunikaci přes toto rozhraní se používá datový rámec mírně upravený. Tento rámec je znázorněn na následujícím obrázku. [5]



Obr. 19. Datový rámeček pro komunikaci přes RS-232 [5]

### **Group/ID**

Tvoří dvoubytovou informaci. Group 1. byte, který udává typ zprávy.

0x03 – značí chybovou zprávu

0x05 – zpráva master to module (slave)

0x07 – zpráva module (slave) to master (odpověď od modulu)

ID 2. byte udává adresu modulu. V továrním nastavení je tato adresa 12 tedy 0x0C. [5]

### **D-Len**

Jednobytová informace udávající délku zprávy. Do délky je započítán i kód příkazu tedy Cmd.

### **Cmd**

Opět jednobytová informace, která udává kód příkazu.

### **Data**

Jejich délka se liší podle typu příkazu. Může být i nulová.

### **CRC**

Cyklický redundantní součet. Je to hashovací funkce, která se používá pro detekci chyb při přenosu či ukládání dat. Zde se využívá typ CRC16-IBM. Tedy CRC je dvoubytový. [5]

### 2.3.4 Funkce CRC16

K datovému rámci se vygeneruje bitová posloupnost, která se k rámci připojí, tak aby jejich spojení bylo beze zbytku dělitelné stanoveným číslem. [6] Toto číslo se nazývá generující polynom. Zde se využívá CRC16- IBM kde je generující polynom  $x^{16}+x^{15}+x^2+x^0$  tedy hodnota 0x8005. [5]

#### Uvedu příklad jednoduchého výpočtu CRC:

Máme zprávu určenou k odeslání:

11001001

Dále máme generující polynom:

1101

Výpočet probíhá následovně:

Vynásobíme zprávu  $2^n$ , kde  $n$  je stupeň generujícího polynomu. Zde to je 3. Tím se nám zpráva doplní zprava nulami. Poté provedeme dělení a určíme zbytek. Ten pak při odesílání připojíme ke zprávě. [6]

$$11001001 * 2^3 = 11001001000$$

$$11001001000 / 1101 = 10010010 \text{ zbytek } \mathbf{010}$$

**Vlastní výsledek pro nás není zajímavý. Zajímá nás zbytek.**

Po výpočtu pak tedy odešleme posloupnost zpráva+zbytek tedy:

11001001**010**

Tento postup zaručí, že po přijetí zprávy bude tato beze zbytku dělitelná generujícím polynomem. Pokud dojde k tomu, že zbytek nebude nulový, tak byla zpráva přijata s chybou. [6]

### 2.3.5 Příklad komunikace přes RS-232

Na dalším obrázku je uveden krátký příklad komunikace přes toto rozhraní. Příklad ukazuje použití příkazu, přesun na pozici „Move to position“. Komunikace probíhá s modulem, který má adresu 1. [4]

Příkaz „master to modul“ Pohni se na pozici 10mm								
ID	D-Len	Cmd	Data	CRC16				
0x05	0x01	0x05	0xB0	0x00	0x00	0x20 0x41	0x48	0x80
Odpověď od modulu. Pozice dosáhnu za 3,358s. Pohyb začal.								
ID	D-Len	Cmd	Data	CRC16				
0x07	0x01	0x05	0xB0	0xEE	0xEE	0x56 0x40	0x7B	0xE4
Časově zpožděná odpověď od modulu. Pozice dosažena. Jsem na pozici 9,9969mm.								
ID	D-Len	Cmd	Data	CRC16				
0x07	0x01	0x05	0x94	0xB6	0xF3	0x1F 0x41	0x7E	0xD5

Obr. 20. Příklad komunikace přes RS-232[4]

Na příkladu je patrné, že modul po přijetí příkazu odpovídá okamžitě, oznamuje začátek pohybu a zároveň odešle vypočtený čas konce pohybu. Druhá zpráva přichází opožděně (po cca 3,5s) a oznamuje dokončení pohybu a hlásí svoji pozici.

### 2.3.6 Variace příkazů čelistí

Čelisti jsou schopny zpracovat mnoho různých typů pohybových a informačních nebo řídicích příkazů. Vybrané příkazy budou uvedeny spolu s popisem jejich činnosti. [5]

Některé pohybové příkazy:

**Referenční pohyb** – čelisti provedou předem daný referenční pohyb

**Pohyb na pozici** - zadanou absolutně nebo relativně

**Pohyb na pozici ve vymezeném čase**

**Pohyb ve smyčce** – čelisti se pohybují cyklicky mezi výchozí a zadanou pozicí

**Pohyb s daným proudovým zatížením** – po dosažení požadovaného proudu se pohyb přerušuje

**Úchop** – čelisti uchopí a drží s daným proudem

Dále je možné nastavovat požadované rychlosti, zrychlení, časy dosažení, trnutí, požadovaný proud. Samozřejmostí jsou příkazy zastavení nebo příkazy, které vyvolají odeslání informačních zpráv. Také jsou obsaženy sady příkazů pro programování firmware.

[5]

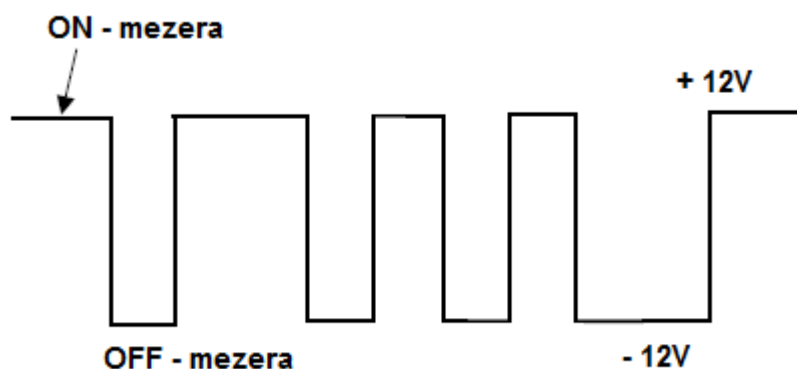
### 3 POPIS ROZHRANNÍ RS-232

Zde je uveden stručný popis tohoto rozhraní. Zařízení jsou připojována k počítačům pomocí rozhraní. Rozhraní RS-232 se už na dnešních počítačích objevuje méně. Bývá nahrazováno sběrnici USB. Nicméně za pomoci převodníku, připojeného do USB, může být dostupné na každém počítači. K propojení se využívá konektorů D-Sub 9Pin (samec na straně počítače samice na straně zařízení). Někdy může být i konektor s 25 Piny. Data jsou u RS-232 přenášena sériově tj. v bitech za sebou. Toto rozhraní se nejčastěji využívá k propojení počítače a zařízení, někdy i ke komunikaci dvou počítačů mezi sebou. Rozhraní je bipolární. Používají se následující úrovně. [7]

Úroveň +3V až +12V pro stav ON.

Úroveň -3V až -12V pro stav OFF.

V mnoha aplikacích se nepracuje se zápornou hladinou, pak tedy úrovně jsou +12V pro stav ON a 0V pro stav OFF. [7]



Obr. 21. Ukázka signálu RS-232[7]

Zde se využívá rozhraní označované jako mini RS-232. Kde se využívá pouze třívodičového zapojení. Rozhraní pak obsahuje vodiče TD (Transmit data), RD (Receive data), Ground (zem). Signály RD a TD jsou pak při komunikaci překříženy. Přenášené úrovně signálu jsou srovnávány proti zemi. [7]

Jednotlivé byty jsou pak v komunikaci rozděleny start a stop bity, kdy stop bit (logická 1) označuje klidový stav linky a start bit (logická 0) označuje začátek odesílání každého bytu.

Data mohou být doplněna kontrolní paritou. Komunikační rychlost musí být na obou stranách nastavena stejně. Toto rozhraní se asynchronní, protože zde neexistuje žádný vodič s řídicím hodinovým signálem.

**Výhody rozhraní:**

- jednoduchost
- dostupnost

**Nevýhody:**

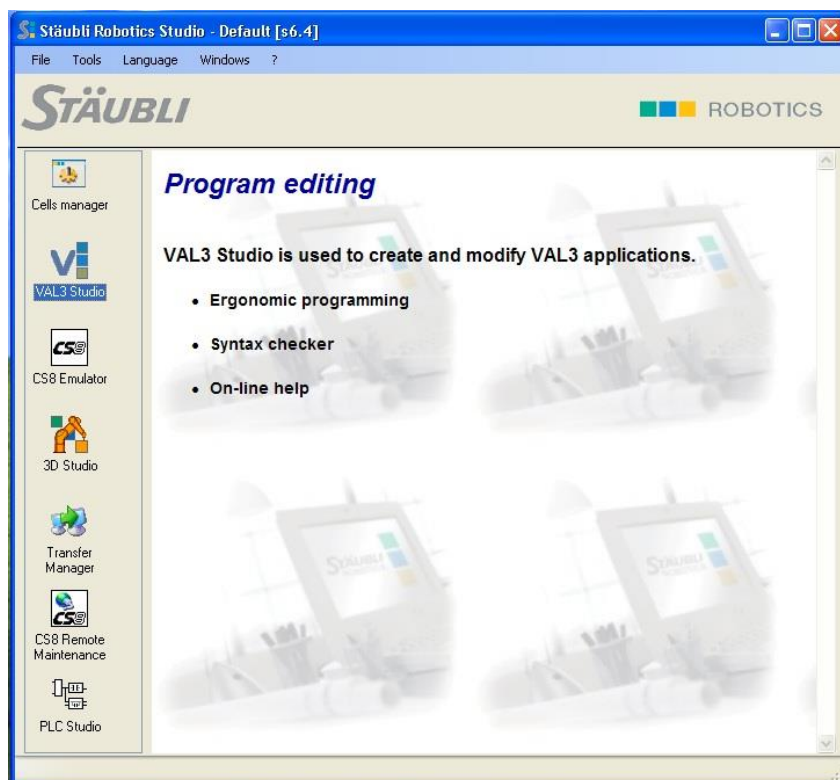
- pouze třívodičové spojení
- četné možnosti selhání
- minimální možnost analýzy selhání
- možnost pouze detekce nikoliv korekce datové chyby
- chybí protokol, kdy by se zařízení dorozumělo s opačnou stranou, že není schopno se přizpůsobit parametrům přenosu

## 4 STÄUBLI ROBOTICS STUDIO

Je soubor programů pro tvorbu uživatelských aplikací, jejich ladění a ovládání robotického ramene. Je dodáván od výrobce jako vývojový software k robotu. Balík programů obsahuje:

1. Cell manager
2. VAL3 studio
3. CS8 emulator
4. 3D studio
5. Transfer manager
6. CS8 remote maintenance
7. PLC studio

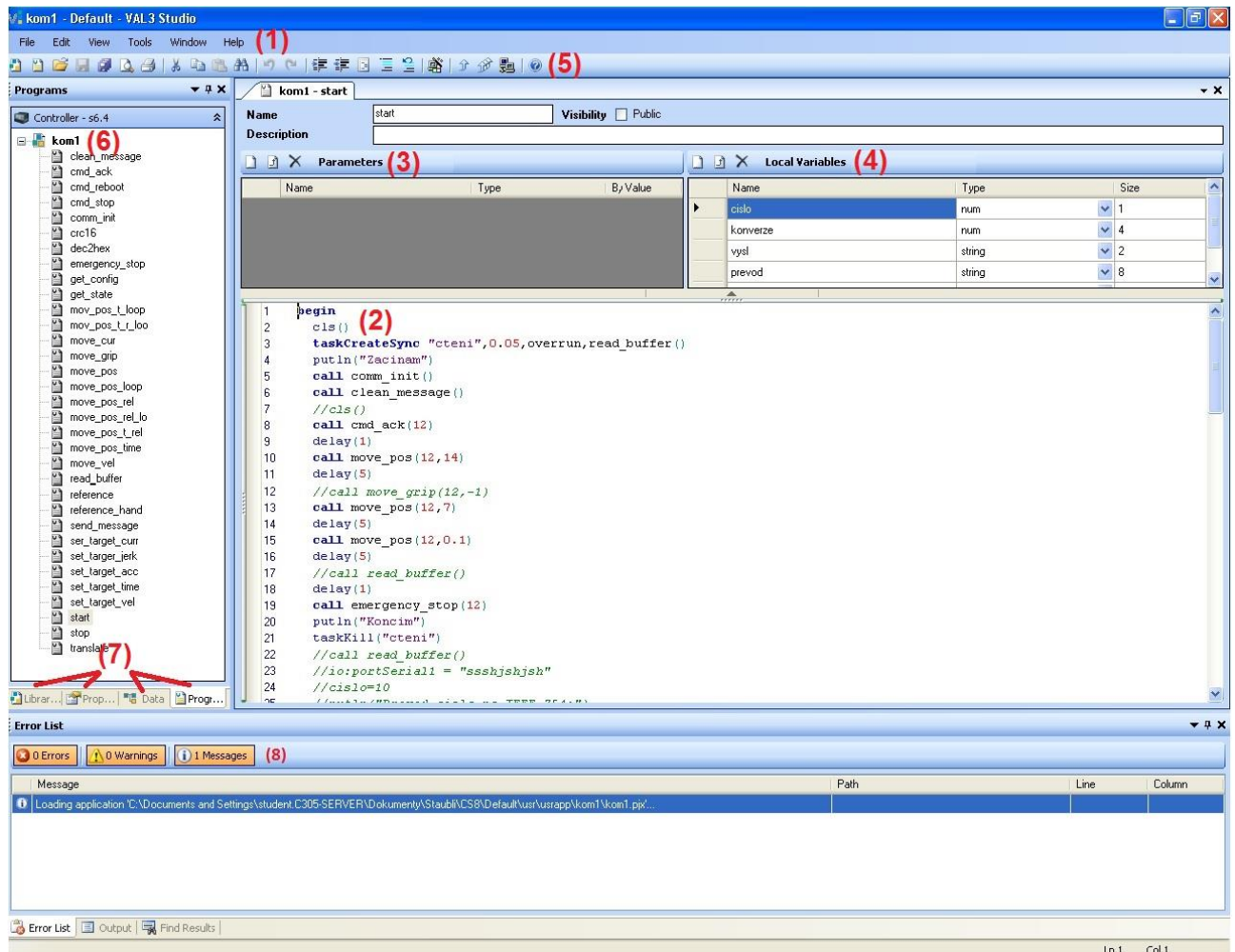
Z hlediska vytváření programů je nejdůležitější VAL3 studio, které poskytuje prostředí pro snadnou tvorbu aplikací a také transfer manager, který umožňuje přenos vytvořeného programového vybavení z PC do řídicí jednotky ramene. [1]



Obr. 22. Stäubli robotics studio

## 4.1 VAL3 studio

Umožňuje vlastní programování v jazyce VAL3. Dokáže provést kontrolu syntaktických chyb v programu. Dále umožňuje vytvářet, editovat a ukládat projekty. Samozřejmostí je online nápověda. Lze z něj rychle přistoupit do transfer manageru. [1]



Obr. 23. VAL3 studio

### 1. Hlavní menu VAL3 studia

Umožňuje pomocí rolovacích nabídek přístup k funkcím VAL3 studia

### 2. Okno pro psaní a editaci vlastní aplikace

### 3. Přehled vstupních parametrů programu

### 4. Přehled lokálních proměnných programu

### 5. Panel rychlých přístupů k funkcím VAL3 studia

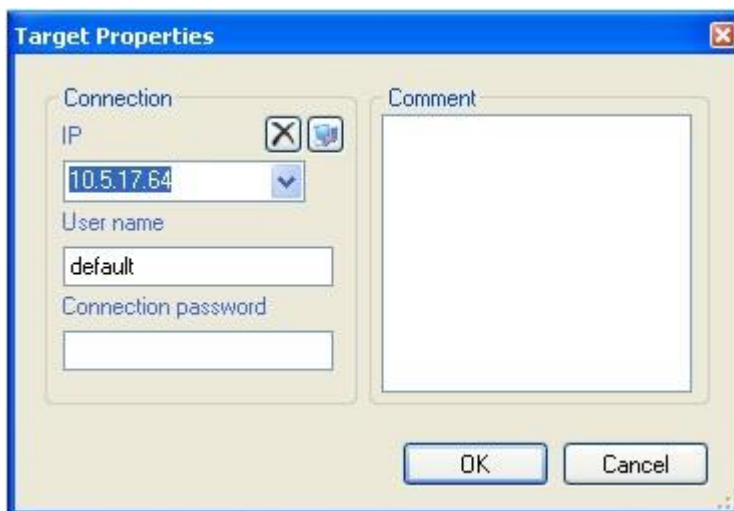
### 6. Přehled zvolené nabídky, různé nabídky se volí pomocí přepínačů (7)

7. Volba nabídek. Je možné volit mezi přehledem programů, globálních dat, možností nebo zvolených knihoven

## 4.2 Transfer manager

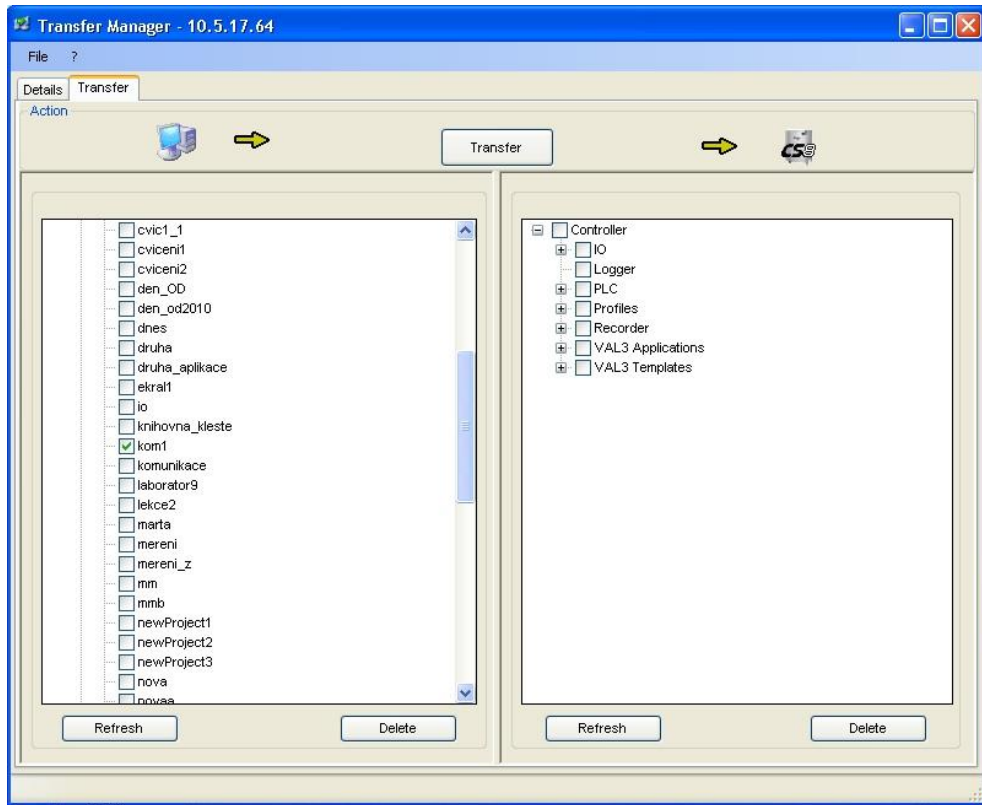
Umožňuje přenos vytvořeného programového vybavení z PC do řídicí jednotky ramene a také naopak. Vyžaduje, aby kontrolér robota byl zapnutý. Ke spojení využívá ethernet. [1]

Po zadání správné IP adresy a přihlašovacích údajů umožní výměnu dat.



Obr. 24. *Transfer manager - přihlášení*

Poté je již ovládání jednoduché a intuitivní. Levá strana znázorňuje data umístěná v PC a pravá strana data v kontroléru. Po výběru požadovaných položek a stisknutí tlačítka transfer se uskuteční přenos dat. Data je možno přenášet oběma směry



Obr. 25. Transfer manager

### 4.3 Programovací jazyk VAL3

VAL 3 je vyšší programovací jazyk navržený pro ovládání robotů stáubli v různých aplikacích. Jazyk VAL3 kombinuje základní funkce se standardními programovacími jazyky. Zaměřuje se především na: [8]

- nástroje ovládání robota
- geometrické modelovací nástroje
- nástroje pro kontrolu vstupů a výstupů

Instrukce VAL3 studia spadají do těchto základních kategorií:

- základní prvky jazyka
- základní datové typy a proměnné
- uživatelské rozhraní
- úlohy
- knihovny

- řízení robota
- pozice ramene
- řízení pohybu [8]

### 4.3.1 Aplikace

VAL3 aplikace je softwarový balíček, který obsahuje soubor dat a instrukcí pro ovládání robota připojeného ke kontroléru. Aplikace VAL3 obsahuje soubor programů (instrukce, které mají být provedeny), soubor globálních proměnných a soubor knihoven, což je vlastně soubor instrukcí a globálních proměnných mimo naší aplikaci. Když aplikace běží, může také obsahovat soubor úloh, které mají být vykonávány současně. [8]

Aplikace VAL3 vždy obsahuje programy start(), stop(), proměnné rámeček (typ rámeček) a nástroj (typ nástroj), který definuje nástroj upevněný na robotickém rameni.

Aplikace mohou být načteny, odstraněny, odstartovány nebo zastaveny pomocí MCP. Když je aplikace spuštěna, začne se vykonávat program start(). Aplikace končí po vykonání všech instrukcí v tomto programu. Poté se začne vykonávat program stop(). Všechny úlohy, které byly vytvořeny knihovnamí, jsou zastaveny v obráceném pořadí, než byly spuštěny. [8]

### 4.3.2 Programy

Program je sekvence po sobě jdoucích instrukcí jazyka VAL3. Program tvoří soubor instrukcí, soubor vstupních parametrů a soubor lokálních proměnných programu. Program může být vykonáván v různých částech aplikace i několikrát a může být volán rekurzivně. Počet instrukcí programu je omezen pouze velikostí zásobníku systému. [8]

Program start() je program, který je volán na začátku běhu aplikace. Je vždy bez vstupních parametrů. Typicky obsahuje deklaraci proměnných, které se použijí v aplikaci, deklaraci výstupů a spuštění úloh. Program start může být také volán vícekrát v jedné aplikaci.

Program `stop()` je volán, když aplikace končí. Nemůže mít žádné vstupní parametry. Tento program obsahuje vše, co je potřebné ke správnému ukončení běžící aplikace tzn. znovuuštění výstupních proměnných a ukončení běžících úloh ve správném pořadí. [8]

### 4.3.3 Datové typy

Všechny proměnné VAL3 mají svůj typ. To umožňuje okamžité zjištění chyb programu. Kontrolou se zjišťuje, zda byl daný typ správně použit. [8]

Základní datové typy jsou:

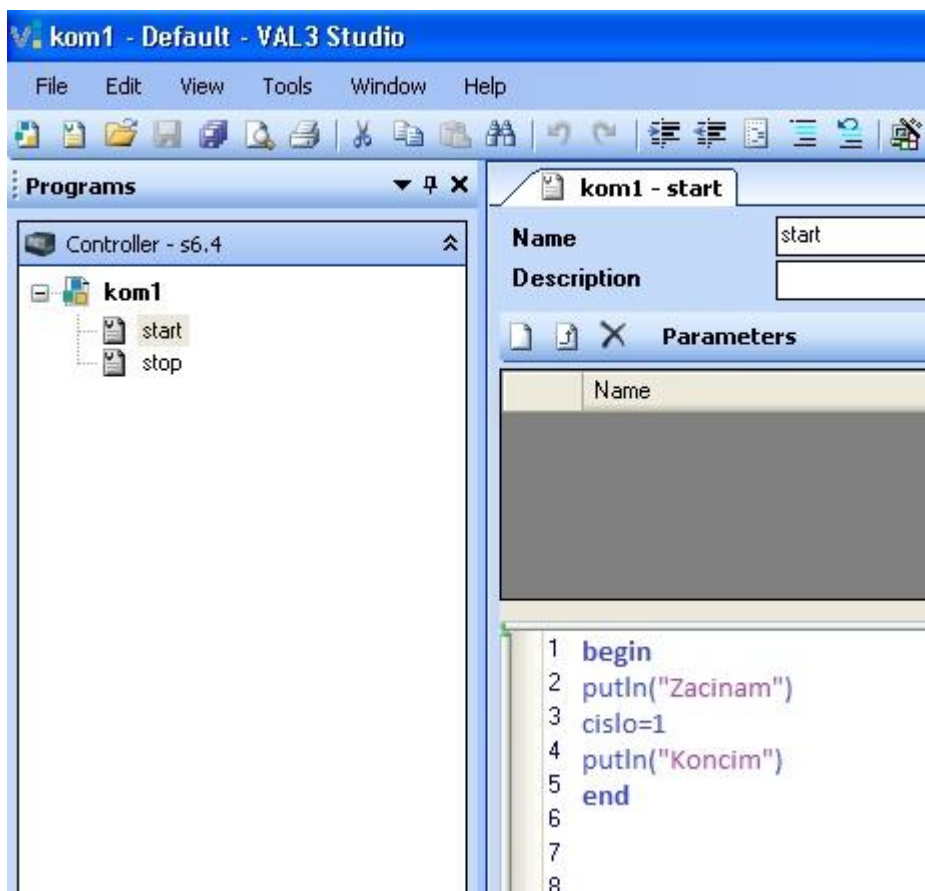
- typ bool** (typ logická proměnná, může nabývat hodnot true nebo false)
- typ num** (typ pro vyjádření čísel)
- typ string** (typ pro vyjádření textových řetězců)
- typ dio** (typ reprezentující digitální vstupy a výstupy)
- typ aio** (typ reprezentující analogové vstupy a výstupy)
- typ sio** (typ reprezentující vstupy a výstupy sériové linky nebo ethernetu)

Strukturované typy jsou:

- typ trsf** (kartézské geometrické transformace)
- typ frame** (kartézské geometrické rámce)
- typ tool** (typ pro nainstalované nástroje)
- typ point** (kartézské pozice nástroje)
- typ joint** (pozice natočení robota)
- typ config** (nastavení robota)
- typ desc** (pro nastavení parametrů pohybů robota) [8]

#### 4.3.4 Příklad jednoduchého programu

Zde je uveden příklad jednoduchého programu v jazyce VAL3.



Obr. 26. Příklad programu

Program start() obsahuje sadu instrukcí a 1 lokální proměnnou „cislo“. Na začátku se vypíše do konzole text „Zacinam“ poté je do proměnné uložena hodnota 1 a na závěr se vypíše do konzole text „Koncim“. Poté celá aplikace končí, protože program stop() je prázdný, tedy neobsahuje žádné instrukce.

#### 4.3.5 Knihovna v jazyce VAL3

VAL3 knihovna je vlastně aplikace, která obsahuje proměnné a programy. Ty mohou být dále využity jinou VAL3 aplikací. Obsah knihovny je identický s obsahem aplikace popsané v bodě 4.3.1. [8]

Programy nebo proměnné v knihovně mohou být nastaveny jako veřejné nebo soukromé. Pouze veřejné programy a proměnné pak mohou být použity z vnějšku knihovny a je

možné k nim přistupovat. K soukromým proměnným nebo programům můžeme přistoupit pouze v rámci knihovny. Přístupem je myšleno u proměnných čtení nebo zápis u programů to je jejich použití. [8]

#### 4.3.6 Vybrané instrukce pro práci se sériovou linkou

Typ **sio** se u VAL3 aplikací používá pro spojení proměnné se sériovou linkou nebo ethernetem. Typ **sio** je charakterizován parametry specifickými pro určitý typ komunikace, časovou prodlevou komunikace a koncem typu řetězec. Proměnná musí být nejdříve spojena pomocí instrukce s fyzickým vstupem/výstupem, jinak ji není možné využívat.

Zde jsou uvedeny vybrané instrukce pro práci se sériovou linkou a na konci tabulky i instrukce pro práci s binárními daty. [8]

Vybrané instrukce pro práci se sériovou linkou		
Příkaz	Syntaxe	Popis příkazu
SioLink	void <b>sioLink</b> (sio& siVariable, sio siSource)	Spojí proměnnou „siVariable“ s fyzickým sériovým vstupem/výstupem „siSource“. Funkce nic nevrací.
ClearBuffer	num <b>clearBuffer</b> (sio siInput)	Vyprázdní vstupní vyrovnávací paměť sériové komunikace. Vrací počet vymazaných znaků.
SioGet	num <b>sioGet</b> (sio siInput, num& nData)	Přečte znak nebo pole znaků ze sériového vstupu (vyrovnávací paměti) a uloží jej do pole typu num. Vrací počet přečtených znaků. Při vypršení času pro čtení vrátí -1.
SioSet	num <b>sioGet</b> (sio siOutput, num& nData)	Odešle znak nebo pole znaků typu num přes sériovou linku. Vrací počet odeslaných znaků. Pokud byl překročen čas daný pro odeslání, vrací -1.
SioCtrl	num <b>sioCtrl</b> (sio siChannel, string nParameter, value)	Nastaví parametry specifické pro každou komunikaci. <b>Příklad:</b> <b>sioCtrl</b> (io:portSerial1, “baudrate“, 9600) Nastaví pro sériový port 1 rychlost přenosu na 9600 bitů za sekundu.
toBinary	num <b>toBinary</b> (num nValue, num nValueSize, string sDataFormat, num& nDataByte )	Funkce byly navrženy, aby umožnily výměnu dat mezi zařízeními při komunikaci přes sériovou linku nebo ethernet. toBinary zakóduje číslo do posloupnosti bytů a fromBinary zase umožní posloupnost bytů dekodovat. Způsob zakódování/dekodování udává parametr sDataFormat.
fromBinary	num <b>fromBinary</b> (num nDataByte, num nDataSize, string sDataFormat, num& nValue )	

Tab. 7. Vybrané funkce pro práci se sériovou linkou[8]

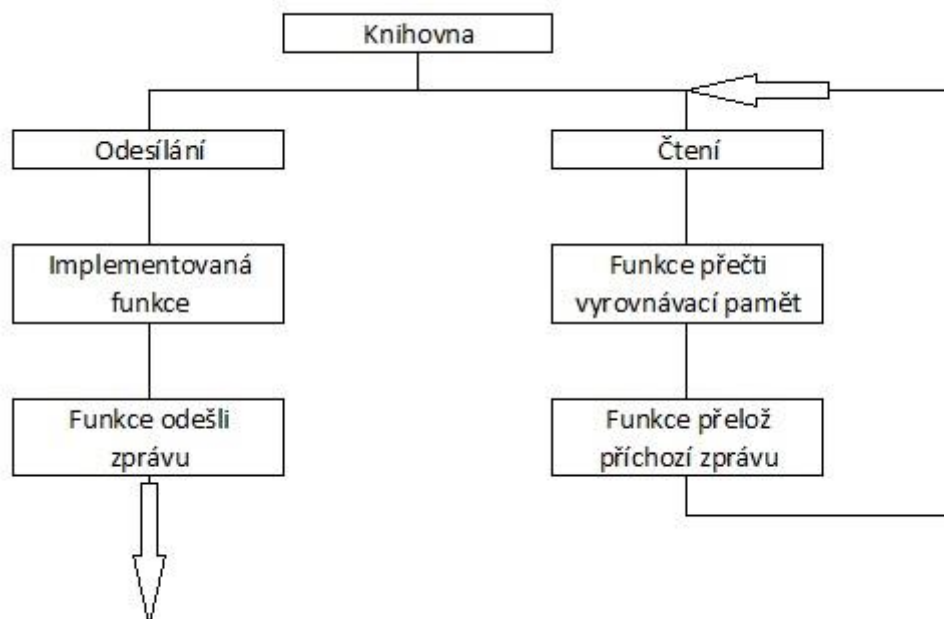
## **II. PRAKTICKÁ ČÁST**

## 5 POPIS VYTVOŘENÉ KNIHOVNY

Knihovna je vlastně ovladač k úchopným čelistem UG20. Je v ní implementována většina příkazů, které jsou podle manuálu (Motion control manual) k dispozici. Knihovna slouží jako ovládací rozhraní mezi robotem a čelistmi. Je napsána v jazyce VAL3. To, že byla tato aplikace napsána jako knihovna, umožní její napojení na kteroukoliv aplikaci psanou k robotu. Z toho plyne možnost ovládání čelistí z jakékoliv uživatelské aplikace.

Knihovna sestává ze dvou oddělených částí. Účel první části je správa odesílání zpráv. Jádrem každé implementované funkce je naplnění číselného pole, které představuje rámec, odesílaný přes sériovou linku. Tento rámec má svoji danou strukturu, jak bylo znázorněno v teoretické části práce. Obsah rámce se liší podle dané funkce a jejich vstupních dat. Na konci každé výstupní funkce je volán program, který zajistí odeslání výstupní zprávy. V tomto programu se kód větví podle velikosti rámce určeného k odeslání. Po určení počtu bytů, které se mají odeslat, je celá výstupní zpráva přesunuta do pole, které odpovídá její přesné velikosti, kvůli omezení funkcí SioSet. Poté je zpráva pomocí této funkce odeslána. Při výpočtu množství bytů se využívá skutečnosti, že pole určené pro práci s výstupní zprávou je přednaplněno hodnotami většími než 256. Potom všechny hodnoty, které jsou menší než hodnota 256, jsou považovány za část zprávy. Knihovna proto obsahuje privátní funkci, která je volána na začátku každé implementované funkce. Tato funkce zajišťuje přednaplnění pracovního pole těmito hodnotami.

Druhá část knihovny má na starost příjem zpráv. Zpráva je z modulu čelistí odeslána pokaždé jako reakce na příchozí příkaz. U pohybových funkcí dojde k odeslání zprávy hned po příjmu příkazu a následně po jeho dokončení. Některé např. informační zprávy mohou přicházet cyklicky. Z předchozích informací je patrné, že čtení vstupu tedy musí probíhat neustále. Proto je čtecí funkce spuštěna při inicializaci knihovny jako synchronní úloha. Ta se spouští pokaždé v daném časovém intervalu. Tato úloha nejprve přečte celou vyrovnávací paměť po jednotlivých přijatých zprávách. Postupné čtení každé zprávy umožňuje reakci na každou z nich. To je provede tak, že zpráva je uložena do pole a toto pole poslouží jako vstupní parametr funkce, která má za úkol reakci na příslušnou zprávu. Po přečtení všech příchozích zpráv je vyrovnávací paměť vyprázdněna. V dalším cyklu úlohy se toto znova opakuje.



Obr. 27. Znáznornění funkce knihovny

## 5.1 Vybrané příkazy a popis jejich funkce

Zde jsou uvedeny vybrané příkazy implementované v knihovně. Popis jejich syntaxe a příklad použití.

### 5.1.1 Referenční pohyb

Příkaz zajistí vykonání referenčního pohybu. Typ referenčního pohybu je zvolen v konfiguračních datech. [5]

#### Syntaxe:

reference(num module\_address)

module\_address – adresa modulu, který má příkaz zpracovat

#### Příklad použití:

Příkaz:

reference(12) – požaduji vykonání referenčního pohybu u modulu s adresou 12

Odpověď:

Pokud byl pohyb úspěšný modul odpoví „OK“.

### 5.1.2 Přesun pozici

Příkaz zajistí přesun na zadanou pozici. Pozice musí být zadána ve správném jednotkovém systému, který je právě zvolen. Příkaz také přihlíží k právě nastavenému pohybovému profilu. [5]

#### Syntaxe:

`move_pos(num module_address, num position)`

`module_address` – adresa modulu, který má příkaz zpracovat

`position` – žádaná pozice čelistí

#### Příklad použití:

Příkaz:

`move_pos(12, 10)` – požaduji vykonání pohybu na pozici 10mm u modulu s adresou 12

Odpověď:

1. Pozice bude dosažena za 8s.
2. Po ukončení pohybu – pozice byla dosažena. Jsem na pozici 9,989mm.

### 5.1.3 Úchop

Příkaz zajistí, že čelisti uchopí předmět požadovanou silou. Znaménko zde značí směr, tedy jestli provádíme úchop nebo rozevření. Je potřeba si uvědomit, že stisknuto bude do té doby, než bude pohyb zastaven příkazem „STOP“. [5]

#### Syntaxe:

`move_grip(num module_address, num grip_A)`

`module_address` – adresa modulu, který má příkaz zpracovat

`grip_A` – provede se úchop nebo rozevření s tímto proudem, udaným v ampérech

**Příklad použití:**

Příkaz:

`move_grip(12, -5)` – požaduji uchopení s proudem 5A u modulu s adresou 12

Odpověď:

1. „OK“ – pohyb začal
2. Po dosažení požadovaného proudu je odeslána zpráva – proud dosažen.

#### 5.1.4 Nastavení požadované rychlosti

Příkaz nastaví do konfiguračního profilu danou rychlost, kterou se budou řídit všechny pohybové příkazy. Tato rychlost zde bude uložena, dokud nebude změněna nebo nedojde k restartu modulu. [5]

**Syntaxe:**

`set_target_vel(num module_address, num velocity)`

`module_address` – adresa modulu, který má příkaz zpracovat

`position` – žádaná rychlost pohybu

**Příklad použití:**

Příkaz:

`set_target_vel(12, 20)` – uložím požadovanou rychlost 20mm/s do konfiguračního profilu

Odpověď:

„OK“ rychlost uložena.

### 5.1.5 Nouzové zastavení

Činnost je zastavena nejrychleji, jak je to možné. Pokud je v modulu nakonfigurována brzda, je tato aktivována. Po zadání příkazu začne modul v cyklech odesílat chybovou zprávu nouzového zastavení. Pro pokračování v činnosti je nutno použít příkaz obeznámení s chybou. [5]

**Syntaxe:**

```
emergency_stop(num module_address)
```

module\_address – adresa modulu, který má příkaz zpracovat

**Příklad použití:**

Příkaz:

```
emergency_stop(12) – požadavek okamžitého zastavení čelistí s adresou 12
```

Odpověď:

Chybová hláška nouzového zastavení.

### 5.1.6 Zjištění konfigurace

Modul odešle informační zprávu, která obsahuje různá konfigurační data modulu.

**Syntaxe:**

```
get_config(num module_address)
```

module\_address – adresa modulu, který má příkaz zpracovat

**Příklad použití:**

Příkaz:

```
get_config(12) – zjistí konfiguraci modulu s adresou 12
```

Odpověď:

Typ modulu, objednáací číslo, verze firmware, verze protokolu, verze hardware a datum vytvoření firmware.

## 5.2 Struktura vybraných příkazů

Zde bude uvedena struktura některých příkazů a zároveň bude vysvětleno, jak jsou programy vytvořeny. Ve výběru budou uvedeny klíčové programy knihovny.

### 5.2.1 Program pro inicializaci knihovny

```
comm_init (num module_address)  
Initialize communication  
begin  
    sioLink(comport,io:portSerial1)  
    sioCtrl(comport,"mode","RS232")  
    sioCtrl(comport,"baudRate",9600)  
    sioCtrl(comport,"bits",8)  
    sioCtrl(comport,"stopBits",1)  
    sioCtrl(comport,"parity","none")  
    sioCtrl(comport,"flowControl","none")  
    sioCtrl(comport,"timeout",0)  
    clearBuffer(comport)  
    busy=true  
    ready=false  
    taskCreateSync "cteni",0.05,overrun,read_buffer()  
    putln("Connection is being established")  
  
    while(ready!=true)  
        call cmd_ack(module_address)  
        delay(1)  
    endwhile  
  
    call clean_message()  
    putln("Connection is established")  
end
```

Obr. 28. Program pro inicializaci komunikace

V programu jsou nejprve nastaveny parametry komunikace přes RS-232. Poté je vyprázdněna vyrovnávací paměť čtení. Pracovní příznaky, které vyjadřují připravenost modulu, jsou nastaveny na výchozí hodnoty. Je spuštěna synchronní úloha čtení vstupu. V cyklu „while“ dochází k navázání komunikace s modulem. Poté, co byla přijata odpověď na příkaz „cmd\_ack“, dojde k nastavení pracovních příznaků vně této funkce. Podmínka cyklu je tedy splněna a komunikace byla navázána. To je uživateli sděleno výpisem v konzole.

Funkce je volána s parametrem inicializovaného modulu a nemá žádné lokální proměnné.

## 5.2.2 Program pro výpočet CRC16

```

crc16 (num &msg, num length, num &crc)
Vypocet CRC16
num i
num pomlength
num temp
num temp1
num temp2
num temp3
string vysl[4]
begin
  pomlength=0
  if length==0
    for i=0 to size(msg)-1 step 1
      if msg[i]<256
        pomlength=pomlength+1
      else
        endIf
    endFor
  else
    pomlength=length
  endIf

  //Vlastní určení CRC16

  crc=0
  for i=0 to (pomlength-1) step 1
    temp=bAnd(crc, 65280)/256
    temp1=bAnd(crc, 255)
    temp2=bAnd(msg[i], 255)
    temp3=bXor(temp1, temp2)
    crc=bXor(temp, tbl[temp3])
  endFor
end

```

Obr. 29. Program pro výpočet CRC16

Program určí hodnotu CRC16-IBM vypočtenou ze zprávy „msg“ a výsledek je uložen do proměnné „crc“. Pokud je zadána i délka zprávy, počítá se s délkou zadanou. Pokud je ale délka zprávy rovna 0, pak se určí v počátečním cyklu „for“. Opět se využívá skutečnosti, že pracovní pole je předem naplněno číselnými hodnotami většími než 256. V tomto případě byl zvolen algoritmus výpočtu CRC s tabulkou předem vygenerovaných hodnot, která je globální proměnnou. Algoritmus výpočtu je pak pevně dán. Po určení CRC16 pomocí logických operací a hodnot v tabulce je toto uloženo do proměnné „crc“. Pomocí postupného dělení pak z této hodnoty snadno určíme výsledné dva byty CRC16.

### 5.2.3 Program pro čtení vstupu

```

read_buffer ()
Read data from buffer
num i
num input[100]
num j
num pole_data[20]
num pom1
num pom2
string prevod[4]

begin
  while (true)
    pom1=0
    pom2=0
    pom1=sioGet (comport, input)
    if pom1!=0
      while (pom1>pom2)
        call dec2hex (input [pom2], prevod)
        put (prevod)
        pom2=pom2+1
        put (" ")
        call dec2hex (input [pom2], prevod)
        put (prevod)
        pom2=pom2+1
        put (" ")
        j=0

        for i=input[pom2]+2 to 0 step -1
          pole_data[j]=input [pom2]
          call dec2hex (input [pom2], prevod)
          put (prevod)
          put (" ")
          pom2=pom2+1
          j=j+1
        endFor
        putln (" ")
        call translate (pole_data)

      endwhile
      clearBuffer (comport)
    endif
  endwhile
end

```

Obr. 30. Program pro čtení vstupu

Celý program je uzavřen v nekonečném cyklu, protože se při použití spouští jako synchronní úloha. Nejprve jsou inicializovány hodnoty pomocných proměnných, poté je přečten vstup. Pokud nebyla načtena data, program nevykoná nic. Při načtení dat se tyto zprávy vypisují do konzole (čísla jsou převedena do hexadecimálního tvaru pomocí

programu „dec2hex“) a zároveň v cyklu „for“ se naplní pole příchozími daty vždy jedné příchozí zprávy. Toto pole je pak předáno ve funkci „translate“, která umožní podle typu zprávy příslušně reagovat. Program překládá a vypisuje zprávy do bodu než jsou všechny byty zprávy zpracovány.

#### 5.2.4 Program pro odeslání zprávy

```
send_message (num &out_mess)  
Sends message over serial port.  
  
num i  
num length  
num pole10[10]  
num pole11[11]  
num pole6[6]  
  
begin  
    length=0  
  
    for i=0 to size(out_mess)-1 step 1  
        if out_mess[i]<256  
            length=length+1  
        else  
            endIf  
        endFor  
    switch length  
        case 6  
            for i=0 to length-1 step 1  
                pole6[i]=out_mess[i]  
            endFor  
            sioSet(comport,pole6)  
            break  
        case 10  
            for i=0 to length-1 step 1  
                pole10[i]=out_mess[i]  
            endFor  
            sioSet(comport,pole10)  
            break  
        case 11  
            for i=0 to length-1 step 1  
                pole11[i]=out_mess[i]  
            endFor  
            sioSet(comport,pole11)  
            break  
        default  
            putln("Nekonzistentni zprava")  
            break  
    endSwitch  
    delay(0.1)  
end
```

Obr. 31. Program pro odeslání zprávy

Program zajistí odeslání zprávy přes sériovou linku. Pole definovaná jako lokální proměnné mají délku typických zpráv, protože funkci „sioSet“ musí být předána pole s přesnou délkou zprávy. Funkce totiž odešle vždy pole celé a nedokáže rozlišit, které byty už nejsou obsahem zprávy. Na začátku se tedy určí délka zprávy s využitím skutečnosti, že pracovní pole je předem naplněno číselnými hodnotami většími než 256. Poté je pomocí přepínače program rozvětven podle délky zprávy. Zde je pak zpráva přesunuta do odpovídajícího pole a odeslána pomocí funkce „sioSet“.

### 5.2.5 Program vybrané výstupní funkce

```

move_pos (num module_address, num position_mm)
Move to a defined position
num i
num konverze[4]
num pom1
num pom2

begin
  call clean_message()
  message[0]=5
  message[1]=module address

  message[2]=5
  message[3]=176
  toBinary(position_mm, size(position_mm), "4.01", konverze)
  for i=0 to 3 step 1
    message[4+i]=konverze[i]
  endFor
  //crc16
  call crc16(message, 0, crc)
  pom1=roundDown(crc/256)
  pom2=crc-((roundDown(crc/256))*256)
  message[8]=pom2
  message[9]=pom1
  busy=true
  call send_message(message)
end

```

Obr. 32. Ukázkový program výstupní funkce

Nejprve se „vyčistí“ pracovní pole výstupní zprávy. V tomto případě to znamená, že je předem naplněno číselnými hodnotami většími než 256. Na dalších řádcích je v tomto poli sestavena zpráva odpovídající funkci „přesun na pozici“. Tedy první byte má hodnotu 5, což označuje zprávu od řídicího systému pro podřadný. Druhý byte obsahuje adresu

modulu. Třetí byte udává délku dat sečtenou s instrukcí příkazu. Zde tedy jeden byte pro označení příkazu a další čtyři byty pro pozici zakódovanou do IEEE 754 jednoduché přesnosti s desetinou čárkou ve formátu „little endian“. Čtvrtý byte je kód příkazu v tomto případě 176 (označuje příkaz přesun na pozici). Další čtyři byty jsou tedy zakódovaná pozice v milimetrech. K zakódování byla využita funkce, kterou poskytuje jazyk VAL3 „toBinary“. A poslední dva byty tedy „message[8] a amessage[9]“ jsou CRC16 celé zprávy. Hodnotu je pomocí postupného dělení třeba rozdělit do dvou bytů a ještě zaměnit jejich pozici (formát „little endian“). To je provedeno na řádcích, kde pracujeme s pomocnými proměnnými „pom1 a pom2“. Do zprávy pak ukládáme v opačném pořadí jak je patrné z obrázku, „pom2“ do „message[8]“ a „pom1“ do „message[9]“. Takto naplněná zpráva je pak předána funkci, která zajistí její odeslání.

## 6 UKÁZKOVÝ PROGRAM S VYUŽITÍM KNIHOVNY

Zde je uveden příklad aplikace s využitím knihovny k čelistem UG20. Příklad byl zvolen tak, aby představoval situaci uchopení předmětu. Tedy nejdříve musí dojít k rozevření čelistí, poté se robot přesune na správné místo, předmět se uchopí a nakonec se robot přesune s uchopeným předmětem na jiné místo. U aplikace musí být samozřejmě přidána knihovna pro ovládání čelistí mezi použitými knihovnami. Po zavedení systému musíme kontrolér robota a kontrolér čelistí spojit kabelem RS-232.

<i>Tool</i>				
<i>Public</i>	<i>Name</i>	<i>Father</i>		
	flange			
<i>Joint (Rx)</i>				
<i>Public</i>	<i>Name</i>		<i>Public</i>	<i>Name</i>
	j1			j2

<i>start ()</i>	<i>stop ()</i>
<pre> begin   cls()   call ug20:comm_init(12)   if (ug20:ready==true)     call ug20:move_pos(12,14)     do       until (ug20:busy!=true)     movej (j1, flange, mNomSpeed)     waitEndMove ()     do       until (isSettled()==true)     call ug20:move_pos(12,1)     do       until (ug20:busy!=true)     movej (j2, flange, mNomSpeed)     waitEndMove ()     do       until (isSettled()==true)     endIf   end end </pre>	<pre> begin   call ug20:emergency_stop(12)   call ug20:kill_ctení ()   resetMotion () end </pre>

Obr. 33. Ukázková aplikace s využitím knihovny

Z obrázku je patrné, že jako nástroj je použita standardní příruba a také zde máme definovány dvě polohy robota. Pro jednoduchost příkladu nebyl nadefinován nový nástroj čelistí. Čelisti jsou ale k robotu připojeny.

Na začátku aplikace je vymazán veškerý výpis na uživatelské konzole. Dále je zavolán program, který zajistí inicializaci knihovny. O úspěšném nebo neúspěšném spojení jsme informováni v konzole. Další podmínka zajistí, že pokud nebyla navázána komunikace, program uvnitř podmínky nebude proveden. Dále se rozevřou čelisti na šířku 14mm a

počká se tak dlouho, dokud provádění neskončí. To je zajištěno cyklem „do...until“. Robot se poté přesune do bodu „j1“ se standardním nástrojem a nominální rychlostí. Program nebude pokračovat, dokud robot nedokončí pohyb na souřadnice daného bodu. To je opět zajištěno cyklem. Poté se čelisti sevřou a opět se čeká na dokončení pohybu. Robot se pak přesune na pozici „j2“. V programu stop() jsou zastaveny čelisti a ukončena úloha čtení vstupu. Tím program bezchybně skončí.

Rozdíl mezi funkcemi isSettled() a waitEndMove() spočívá v tom, že první funkce vrací „true“, pokud je robot stabilizovaný ve své cílové pozici. Na rozdíl od druhé, která čeká, jen dokud se pozice odeslaná do jednotky motoru shoduje s cílovou (naposledy zadanou) pozicí.

Proměnné knihovny „ready“ a „busy“ jsou informativní příznaky. Příznak „ready“ je „true“ pokud bylo navázáno spojení a s čelistmi je možné komunikovat. Druhý příznak je „true“ pokud jsou čelisti v pohybu, když jsou v klidu příznak nabývá hodnoty „false“ .

## ZÁVĚR

Práce se zabývá zejména průmyslovým robotem Stäubli Unimation TX40 a nástrojem, který je k němu připojen konkrétně úchopnými čelistmi UG20. Popisuje robota a vysvětluje jak jej ovládat. Obeznamuje čtenáře s problematikou komunikace s čelistmi.

V teoretické části práce byl popsán robot a jeho řídicí jednotka a byly uvedeny základy jeho ovládání přes MCP. V práci byly také podrobně popsány čelisti, uvedeny jejich technické vlastnosti. Poměrně podrobně byl také popsán způsob komunikace s čelistmi. Také byla probrána problematika komunikace přes rozhraní RS-232. Hlavní funkce knihovny byly znázorněny také pomocí ukázkových příkladů.

V praktické části byl podrobně uveden popis vytvořené knihovny. Bylo znázorněno, jak knihovna pracuje. Tato část také obsahuje popis jednotlivých příkazů, konkrétně jejich syntaxe a vysvětluje reakce čelistí na vybrané příkazy. Byly popsány i klíčové funkce knihovny a na ukázkové aplikaci byla znázorněna funkce knihovny.

Práce rozšiřuje znalosti čtenáře v problematice průmyslových robotů, jejich nástrojů a také způsobů komunikace s nimi. Slouží také jako manuál k vytvořené knihovně a vysvětluje, jak se s ní musí pracovat.

## ZÁVĚR V ANGLIČTINĚ

The bachelor thesis focus on description and explanation of the control of the industrial robot Stäubli Unimation TX40 mainly its component – the grasping end-effector UG20, and the possibilities of communication with it.

The theoretical part of thesis is about the robot and its control unit and the introduction of its managing by MCP. The grasping end-effector were described with its technical attributes and possibilities of communication with them and communication by interface RS-232. The main functions of the library are showed by examples.

The practical part of thesis describes the establishment of the library and the way it works. This part also includes a description of each command, namely their syntax and explains the reaction of grasping end-effector on selected commands. The main functions of the library were described and showed in sample application.

This work should extend the reader's knowledge of the problematics of the industrial robots, their components and the possibilities of communication with them. It could be also used as a manual for the established library and explain how to work with it.

**SEZNAM POUŽITÉ LITERATURY**

- [1] *Stäubli – Textile Machinery, Connectors and Robotics* [online]. 2007 [cit. 2013-05-08].  
Dostupné z: <http://www.staubli.com/>
- [2] *Arm – TX series 40 family*. Faverage: Stäubli, 2007. 84 s.
- [3] *CS8C Controller : Instruction manual*. Faverage: Stäubli, 2009. 248 s.
- [4] *EGN\_Manual\_V06: Assembly and operation manual*. Lauffen/ Neckar, 2008. 44 s.
- [5] *MotionControl\_Eng: SCHUNK MOTION*. SCHUNK company, 2009. 152 s.
- [6] *Cyklická redundantní kontrola. Fakulta strojního inženýrství VUT Brno* [online].  
1998 [cit. 2013-05-13]. Dostupné z: <http://drogo.fme.vutbr.cz/~jroupec/site/crc.html>
- [7] DOC. ING. KOTÁSEK, CSC., Zdeněk. *RS232*. VUT Brno, 2009. Dostupné z:  
<http://www.fit.vutbr.cz/study/courses/ITP/public/itp07/rs232.pdf>
- [8] *VAL3 Reference Manual*. Stäubli Faverges, 2008. 186 s.
- [9] *Automatizace a robotizace I*. SPŠs a JŠ Haverova 191, Kolín, 2009. 74 s.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

MCP	Motion Control Panel
IP	Ingress Protection
LCD	Liquid Crystal Display
LED	Light Emitting Diode
USB	Universal Serial Bus
DC	Direct Current
PC	Personal Computer
RxD	Receive Data
TxD	Transfer Data
GND	Ground
PLC	Programmable Logic Controller
IEEE	Institute of Electrical and Electronics Engineers
CRC	Cyclic Redundancy Check
3D	Three Dimensional
IP	Internet Protocol
SIO	Seriál Input Output
AIO	Analog Input Output
DIO	Digital Input Output

## SEZNAM OBRÁZKŮ

<i>Obr. 1. Robotické rameno[2]</i> .....	11
<i>Obr. 2. Znárodnění možností pohybů ramene[2]</i> .....	13
<i>Obr. 3. Manuální programovací panel[3]</i> .....	14
<i>Obr. 4. Výběr módu robota[3]</i> .....	15
<i>Obr. 5. Výběr módu pohybu ramene[3]</i> .....	17
<i>Obr. 6. Ovládání pohybu v módu kloub[3]</i> .....	18
<i>Obr. 7. Souřadnicový systém v módech rám a nástroj[3]</i> .....	18
<i>Obr. 8. Ovládání pohybu v módu rám a nástroj[3]</i> .....	19
<i>Obr. 9. Znárodnění rotace kolem osy[3]</i> .....	19
<i>Obr. 10. Aktivační spínač[3]</i> .....	22
<i>Obr. 11. Body osvětlení[3]</i> .....	23
<i>Obr. 12. Kontrolér CS8C[3]</i> .....	24
<i>Obr. 13. Čelisti[4]</i> .....	25
<i>Obr. 14. Rozměry a zapojení integrovaného dekodéru[4]</i> .....	26
<i>Obr. 15. Schéma rozhraní čelistí[4]</i> .....	27
<i>Obr. 16. Rozložení konektorů kontroléru MCS-12[4]</i> .....	29
<i>Obr. 17. Spojení kontroléru přes RS-232[4]</i> .....	30
<i>Obr. 18. Společný datový rámec[4]</i> .....	30
<i>Obr. 19. Datový rámec pro komunikaci přes RS-232 [5]</i> .....	32
<i>Obr. 20. Příklad komunikace přes RS-232[4]</i> .....	34
<i>Obr. 21. Ukázka signálu RS-232[7]</i> .....	36
<i>Obr. 22. Stäubli robotics studio</i> .....	38
<i>Obr. 23. VAL3 studio</i> .....	39
<i>Obr. 24. Transfer manager - přihlášení</i> .....	40
<i>Obr. 25. Transfer manager</i> .....	41
<i>Obr. 26. Příklad programu</i> .....	44
<i>Obr. 27. Znárodnění funkce knihovny</i> .....	49
<i>Obr. 28. Program pro inicializaci komunikace</i> .....	53
<i>Obr. 29. Program pro výpočet CRC16</i> .....	54
<i>Obr. 30. Program pro čtení vstupu</i> .....	55
<i>Obr. 31. Program pro odeslání zprávy</i> .....	56

---

<i>Obr. 32. Ukázkový program výstupní funkce .....</i>	<i>57</i>
<i>Obr. 33. Ukázková aplikace s využitím knihovny .....</i>	<i>59</i>

**SEZNAM TABULEK**

<i>Tab. 1. Tabulka vlastností ramene.....</i>	<i>12</i>
<i>Tab. 2. Technické vlastnosti čelistí.....</i>	<i>25</i>
<i>Tab. 3. Technické vlastnosti integrovaného motoru.....</i>	<i>26</i>
<i>Tab. 4. Technické vlastnosti kontroléru MCS-12.....</i>	<i>28</i>
<i>Tab. 5. Formát IEEE 754.....</i>	<i>31</i>
<i>Tab. 6. Příklady zakódování do IEEE 754.....</i>	<i>31</i>
<i>Tab. 7. Vybrané funkce pro práci se sériovou linkou.....</i>	<i>46</i>

## SEZNAM PŘÍLOH

- PI Výstup VAL3 studia – obsah knihovny
- PII Výstup VAL3 studia – ukázková aplikace
- PIII CD-ROM – seznam všech příkazů implementovaných v knihovně, jejich popis a výpis

## PŘÍLOHA P I: VÝSTUP VAL3 STUDIA – OBSAH KNIHOVNY

Project properties	
<b>Parameters</b>	version:s6.4, unit:millimeter, stackSize:5000
<b>Program Files</b>	clean_message.pgx cmd_ack.pgx cmd_reboot.pgx cmd_stop.pgx comm_init.pgx crc16.pgx dec2hex.pgx emergency_stop.pgx get_config.pgx get_state.pgx mov_pos_t_loop.pgx mov_pos_t_r_loo.pgx move_cur.pgx move_grip.pgx move_pos.pgx move_pos_loop.pgx move_pos_rel.pgx move_pos_rel_lo.pgx move_pos_t_rel.pgx move_pos_time.pgx move_vel.pgx read_buffer.pgx reference.pgx reference_hand.pgx send_message.pgx ser_target_curr.pgx set_target_jerk.pgx set_target_acc.pgx set_target_time.pgx set_target_vel.pgx start.pgx stop.pgx translate.pgx
<b>Data Files</b>	kom1.dtx
<b>Libraries</b>	io:io

<i>Tool</i>				
<i>Public</i>	<i>Name</i>	<i>Father</i>		
	flange			

<i>Frame</i>				
<i>Public</i>	<i>Name</i>	<i>Father</i>		
	world			

<i>MDesc</i>				
<i>Public</i>	<i>Name</i>		<i>Public</i>	<i>Name</i>
■	mNomSpeed			

<i>Bool</i>				
<i>Public</i>	<i>Name</i>		<i>Public</i>	<i>Name</i>
■	overrun		■	ready
■	busy			

<i>Num</i>				
<i>Public</i>	<i>Name</i>		<i>Public</i>	<i>Name</i>
	crc			message [20]
	tbl [256]			

<i>String</i>				
<i>Public</i>	<i>Name</i>		<i>Public</i>	<i>Name</i>
	sbirka [50]			

<i>Sio</i>				
<i>Public</i>	<i>Name</i>		<i>Public</i>	<i>Name</i>
	comport			

## PŘÍLOHA P II: VÝSTUP VAL3 STUDIA – UKÁZKOVÁ APLIKACE

Project properties	
<b>Parameters</b>	version:s6.5.1, unit:millimeter, stackSize:5000
<b>Program Files</b>	start.pgx stop.pgx
<b>Data Files</b>	zkouskaknihovny.dtx
<b>Libraries</b>	io:io ug20:kom1

Tool		
Public	Name	Father
	flange	

Frame		
Public	Name	Father
	world	

Joint (Rx)				
Public	Name		Public	Name
	j1			j2

MDesc				
Public	Name		Public	Name
■	mNomSpeed			

Num				
Public	Name		Public	Name
	kom			

```

start ()
begin
  cls ()
  call ug20:comm_init(12)
  if (ug20:ready==true)
    call ug20:move_pos(12,14)
    do
      until (ug20:busy!=true)
        movej (j1, flange, mNomSpeed)
        waitEndMove ()
    do
      until (isSettled()==true)
        call ug20:move_pos(12,1)
        do
          until (ug20:busy!=true)
            movej (j2, flange, mNomSpeed)
            waitEndMove ()
          do
            until (isSettled()==true)
              endIf
            end
          end
        end
      end
    end
  end
end

```

```

stop ()
begin
  call ug20:emergency_stop(12)
  call ug20:kill_cteni ()
  resetMotion ()
end

```