Protokollbeschreibung `rob6server` – Version 1.0a

Floris Ernst

11. November 2010

Inhaltsverzeichnis

1 Einleitung

In diesem Dokument wird das Kommunikationsprotokoll für die Kommunikation mit dem `rob6server` beschrieben.

2 Anmeldevorgang

Voraussetzung für die Anmeldung eines Clients ist der gestartete Roboterserver (`rob6server`). Der Startvorgang beinhaltet die Initialisierung des Roboters und des Servers. Nach erfolgreichem Starten wartet der Roboterserver auf Clients.

2.1 Anmeldung eines Clients am Server

2.1.1 Verbindungsaufbau mit dem Roboterserver

Über TCP-Sockets wird eine Verbindung zum Roboterserver aufgebaut. Standardmäßig verbinden sich Clients auf Port 5005, der Server ist **nicht** Multi-Client-fähig.

Bespiel:

start `telnet 127.0.0.1 5005`
receive `Trying 127.0.0.1...`
receive `Connected to 127.0.0.1.`
receive `Escape character is '^]'`
receive `Welcome to rob6server 0.1.01 !`

2.1.2 Initiale Kommunikation zwischen Server und Client

Um die Kommunikation mit dem Roboter zu initialisieren, muss das Kommando `Hello Robot` gesendet werden:

Bespiel:

send `Hello Robot`
receive `accepted`

Nun stehen dem Client alle Funktionen des Roboterservers zur Verügung.

3 Befehlsübersicht

3.1 Befehle zum Servermanagement

- `GetRobot`

  Gibt den Robotertyp zurück. Mögliche Antworten sind `ad850`, `kaw_fs`, `kr3rt`, `kr16rt` und (veraltet) `kr3`, `kr16`

  Beispiel:

send	`GetRobot`
receive	`ad850`

- `Is[Adept|Kuka|Kawa|KR3|KR16]`

  Liefert `true`, wenn der angeschlossene Roboter vom angefragten Typ ist.

  Beispiel:

send	`GetRobot`
receive	`ad850`
send	`IsAdept`
receive	`true`
send	`IsKawa`
receive	`false`

- `GetVersion`

  Liefert die Version des Servers.

  Beispiel:

send	`GetVersion`
receive	`0.1.01`

- `Quit`

  Beendet die Verbindung.

  Beispiel:

send	`Quit`
receive	`bye!`

- `Shutdown`

  Beendet die Verbindung und fährt den Server herunter.

  Beispiel:

send	`Quit`
receive	`bye!`
	shutting down ...

- `GetTimestamp`

  Fragt die momentane Zeit des Servers ab.

  Beispiel:

send	`GetTimestamp`
receive	`1285831711.121`

- `PingRobot num wait` Nur Adept-Roboter

  Bestimmt die Kommunikationslatenz zwischen Server und Roboter. Hier ist `num` die Anzahl der durchzuführenden Pings und `wait` die Wartezeit zwischen zwei Pings in Millisekunden.

  Beispiel:

send	`PingRobot 100 50`
receive	`0.003414 3659.8930000000 1285828592.9111907482 -0.00016902738 1.4625943e-05`

Hier ist die erste Zahl die mittlere Antwortzeit des Roboters (in Sekunden), die zweite Zahl der Nullpunkt der Roboterzeit, die dritte Zahl der Nullpunkt der Server-Zeit und die vierten und fünften Werte sind Koeffizienten eines Polynoms 2. Grades, um die Zeiten aufeinander zu kalibrieren.

- `CM_PING`

  Führt ein Ping aus.

  Beispiel:

  send `CM_PING`
  receive `PONG`

- `SetVerbosity num`

  Stellt die „Geschwätzigkeit“ des Roboter-Servers ein. Für `num` sind Werte zwischen 0 und 4 erlaubt. Die Bedeutung der Werte:

  0 — keine Ausgaben
  1 — Nur Fehler
  2 — Fehler & Warnungen
  3 — Fehler, Warnungen und Kommunikationsdaten
  4 — Alles

  Beispiel:

  send `SetVerbosity 4`
  receive `true`

3.2 Kommandos für den Bewegungsmodus

- `EnableAlter`

  Aktiviert den Echtzeit-Modus. Abhängig vom angeschlossenen Roboter und der auf dem Roboter laufenden Serversoftware ist dieser Modus mehr oder weniger „hart“. Echte Echtzeitsteuerung funktioniert bei `kr3rt` und `kr16rt`.

  Beispiel:

  send `EnableAlter`
  receive `true`

- `DisableAlter`

  Deaktiviert den Echtzeit-Modus. Bewegungen finden im Point-to-Point Modus statt, d.h., der Server wartet nach jedem Bewegungskommando, bis der Roboter die Zielposition erreicht hat.

  Beispiel:

  send `DisableAlter`
  receive `true`

- `EnableAdeptAlter` Nur Adept-Roboter

  Aktiviert den harten Echtzeit-Modus für den `ad850`, wenn `serveralter` läuft.

  Beispiel:

  send `EnableAdeptAlter`
  receive `true`

- `SetRTSpeedControl 0|1` Nur KUKA-Roboter

  Bestimmt das Verhalten im Realtime-Modus. Ist RTSpeedControl aktiviert, wird auch im Realtime-Modus mit Rampen gefahren, sonst werden die Gelenke so bewegt, dass sie gleichzeitig ankommen.

  Beispiel:

  send `SetRTSpeedControl 1`
  receive `true`

3.3 Kommandos zur Steuerung der Bewegung

3.3.1 Genauigkeit der Bewegung

- `SetAdeptFine` v Nur Adept-Roboter

  Aktiviert die Feinregelung der Positionierung und fordert eine Positioniergenauigkeit von v Prozent der Standardgenauigkeit der Gelenke.

  Beispiel:

  send `SetAdeptFine 50`
  receive `true`

- `SetAdeptFine` v Nur Adept-Roboter

  Aktiviert die Grobregelung der Positionierung und fordert eine Positioniergenauigkeit von v Prozent der Standardgenauigkeit der Gelenke.

  Beispiel:

  send `SetAdeptFine 50`
  receive `true`

3.3.2 Geschwindigket und Beschleunigung

- `SetAdeptSpeed` v Nur Adept-Roboter

  Bestimmt die Geschwindigkeit. Die Geschwindigkeit ist in Prozent des Maximalwerts anzugegen, es sind Werte bis 120 erlaubt.

  Beispiel:

  send `SetAdeptSpeed 100`
  receive `true`

- `SetAdeptAccel` a_1 a_2 Nur Adept-Roboter

  Bestimmt die Beschleunigungswerte. Es sind zwei Beschleunigungswerte (Anfahren und Bremsen) in Prozent des Maximalwerts anzugegen, es sind Werte bis 120 erlaubt.

  Beispiel:

  send `SetAdeptAccel 100 50`
  receive `true`

- `SetKukaRTSpeed` v_1 v_2 v_3 v_4 v_5 v_6 Nur KUKA-Roboter mit RT-Interface

  Bestimmt die Geschwindigkeit. Es werden sechs Parameter für die einzelnen Achsen erwartet.

  Beispiel:

  send `SetKukaRTSpeed 0.3 0.1 0.1 0.1 0.1 0.1`
  receive `true`

- `SetJointsMaxSpeed` v_1 v_2 v_3 v_4 v_5 v_6 Nur KUKA-Roboter

  Bestimmt die Geschwindigkeit. Es werden sechs Parameter für die einzelnen Achsen erwartet.

  Beispiel:

  send `SetJointsMaxSpeed 0.3 0.1 0.1 0.1 0.1 0.1`
  receive `true`

- `SetSingleJointMaxSpeed` j a Nur KUKA-Roboter

  Bestimmt die Geschwindigkeit für eine einzelne Achse.

  Beispiel:

  send `SetSingleJointMaxSpeed 3 0.3`
  receive `true`

- `SetJointsMaxAcceleration` a_1 a_2 a_3 a_4 a_5 a_6 Nur KUKA-Roboter

  Bestimmt die Beschleunigung. Es werden sechs Parameter für die einzelnen Achsen erwartet.

  Beispiel:

  send `SetJointsMaxAcceleration 0.005 0.001 0.01 0.01 0.01 0.05`
  receive `true`

- `SetSingleJointMaxAcceleration` j a Nur KUKA-Roboter

  Bestimmt die Beschleunigung für eine einzelne Achse.

  Beispiel:

  send `SetSingleJointMaxAcceleration 3 0.1`
  receive `true`

- `GetJointsMaxSpeed` Nur KUKA-Roboter

  Fragt die Geschwindigkeit der einzelnen Achsen ab.

  Beispiel:

  send `GetJointsMaxSpeed`
  receive `0.187200 0.187200 0.187200 0.396000 0.396000 0.738000`

- `GetJointsMaxAcceleration` Nur KUKA-Roboter

  Fragt die Beschleunigung der einzelnen Achsen ab.

  Beispiel:

  send `GetJointsMaxAcceleration`
  receive `0.005000 0.005000 0.005000 0.005000 0.005000 0.005000`

- `ResetJointsMaxSpeed` Nur KUKA-Roboter

  Setzt die Geschwindigkeit auf Standardwerte zurück.

  Beispiel:

  send `ResetJointsMaxAcceleration`
  receive `true`

- `ResetJointsMaxAcceleration` Nur KUKA-Roboter

  Setzt die Beschleunigung auf Standardwerte zurück.

  Beispiel:

  send `ResetJointsMaxAcceleration`
  receive `true`

3.3.3 Achsenlimits

- `GetJointsMaxChange`

  Liefert den maximalen Drehwinkel der Gelenke pro Bewegung zurück.

  Beispiel:

  send `GetJointsMaxChange`
  receive `370.000000 175.000000 284.000000 700.000000 250.000000 700.000000`

- `SetJointsMaxChange` α_1 α_2 α_3 α_4 α_5 α_6

  Setzt den maximalen Drehwinkel pro Bewegung der Gelenke. Es werden sechs Winkelwerte erwartet.

  Beispiel:

  send `SetJointsMaxChange 10 10 10 10 10 10`
  receive `true`

- `SetSingleJointMaxChange` j α

  Setzt den maximalen Drehwinkel pro Bewegung des Gelenks j auf α.

  Beispiel:

  send `SetSingleJointMaxChange 3 20`
  receive `true`

- `GetJointsMaxTurnMax`

  Liefert den maximalen Drehwinkel der Gelenke zurück.

  Beispiel:

  send `GetJointsMaxTurnMax`
  receive `185.000000 20.000000 154.000000 350.000000 125.000000 350.000000`

- `SetJointsMaxTurnMax` α_1 α_2 α_3 α_4 α_5 α_6

  Setzt den maximalen Drehwinkel der Gelenke. Es werden sechs Winkelwerte erwartet.

  Beispiel:

  send `SetJointsMaxTurnMax 10 10 10 10 10 10`
  receive `true`

- `SetSingleJointMaxTurnMax` j α

  Setzt den maximalen Drehwinkel des Gelenks j auf α.

  Beispiel:

  send `SetSingleJointMaxTurnMax 3 40`
  receive `true`

- `GetJointsMaxChange`

  Liefert den minimalen Drehwinkel der Gelenke zurück.

  Beispiel:

  send `GetJointsMaxTurnMin`
  receive `-185.000000 -155.000000 -130.000000 -350.000000 -125.000000 -350.000000`

- `SetJointsMaxTurnMin` α_1 α_2 α_3 α_4 α_5 α_6

  Setzt den minimalen Drehwinkel der Gelenke. Es werden sechs Winkelwerte erwartet.

  Beispiel:

  send `SetJointsMaxTurnMin -10 -10 -10 -10 -10 -10`
  receive `true`

- `SetSingleJointMaxTurnMin` j α

  Setzt den minimalen Drehwinkel des Gelenks j auf α.

  Beispiel:

  send `SetSingleJointMaxTurnMin 3 -40`
  receive `true`

- `ResetJointsMaxChange` j α

  Setzt den maximalen Drehbereich der Gelenke zurück.

  Beispiel:

  send `ResetJointsMaxChange`
  receive `true`

- `ResetJointsMaxTurn` j α

  Setzt die maximalen und minimalen Drehwinkel der Gelenke zurück.

  Beispiel:

send `ResetJointsMaxTurn`
receive `true`

3.3.4 Bewegung

- `MoveMinChangeRowWiseStatus` $m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,4}$ $m_{2,1}$ $m_{2,2}$ $m_{2,3}$ $m_{2,4}$ $m_{3,1}$ $m_{3,2}$ $m_{3,3}$ $m_{3,4}$ `flip|noflip|toggleHand|noToggleHand` `up|down|toggleElbow|noToggleElbow` `lefty|righty|toggleArm|noToggleArm`

  Dieses Kommando führt eine PTP-Bewegung aus, bei der die Zielmatrix (homogene Koordinaten, zeilenweise) und die gewünschte/erlaubte Roboterkonfiguration vorgegeben wird. Ist das Erreichen der Zielposition in mehreren Konfigurationen möglich, und sind diese Konfigurationen durch die Status-Flags auch zulässig, so wird die Position angefahren, bei der die Gelenkwinkeländerung am geringsten ist.

  Beispiel:

  send `MoveMinChangeRowWiseStatus 0 0 -1 1768 0 -1 0 0 -1 0 0 640 flip toggleElbow toggleArm`
  receive `true`

- `MovePTPJoints` j_1 j_2 j_3 j_4 j_5 j_6

  Bewegt den Roboter im PTP-Modus an eine neue Stellung der Gelenke.

  Beispiel:

  send `MovePTPJoints 10 0 0 0 0 0`
  receive `true`

- `MoveRTHomRowWise` $m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,4}$ $m_{2,1}$ $m_{2,2}$ $m_{2,3}$ $m_{2,4}$ $m_{3,1}$ $m_{3,2}$ $m_{3,3}$ $m_{3,4}$

  Dieses Kommando führt eine RT-Bewegung durch, bei der die Zielmatrix (homogene Koordinaten, zeilenweise) mit der momentanen Konfiguration angefahren wird.

  Beispiel:

  send `MoveRTHomRowWise 0 0 -1 1768 0 -1 0 0 -1 0 0 640`
  receive `true`

- `MoveRTHomRowWiseStatus` $m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,4}$ $m_{2,1}$ $m_{2,2}$ $m_{2,3}$ $m_{2,4}$ $m_{3,1}$ $m_{3,2}$ $m_{3,3}$ $m_{3,4}$ `flip|noflip|toggleHand|noToggleHand` `up|down|toggleElbow|noToggleElbow` `lefty|righty|toggleArm|noToggleArm`

  Nur für KUKA, KUKA RT und Adept im Soft-RT-Modus

  Dieses Kommando führt eine RT-Bewegung aus, bei der die Zielmatrix (homogene Koordinaten, zeilenweise) und die gewünschte/erlaubte Roboterkonfiguration vorgegeben wird. Ist das Erreichen der Zielposition in mehreren Konfigurationen möglich, und sind diese Konfigurationen durch die Status-Flags auch zulässig, so wird die Position angefahren, bei der die Gelenkwinkeländerung am geringsten ist.

  Beispiel:

  send `MoveRTHomRowWiseStatus 0 0 -1 1768 0 -1 0 0 -1 0 0 640 flip toggleElbow toggleArm`
  receive `true`

- `MoveRTJoints` j_1 j_2 j_3 j_4 j_5 j_6 Nur für KUKA, KUKA RT und Adept im Soft-RT-Modus

  Dieses Kommando führt eine RT-Bewegung aus, bei der das Ziel in Gelenkwinkeln angegeben ist.

  Beispiel:

  send `MoveRTJoints 10 20 10 10 10 10`
  receive `true`

3.3.5 Positionsabfrage

- `GetPositionHomRowWise`

  Liefert die momentane Position als homogene Matrix (zeilenweise) zurück.

  Beispiel:

```
send     GetPositionHomRowWise
receive  0.000000 -0.173648 -0.984808 1741.140107 ↩
         0.000000 -0.984808 0.173648 -307.009978 ↩
         -1.000000 -0.000000 -0.000000 640.000000
```

- `GetPositionJoints`

  Liefert die momentanen Gelenkwinkel.

  Beispiel:

```
send     GetPositionJoints
receive  10.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```

- `GetStatus`

  Liefert die momentane Konfiguration zurück. Die Rückgabe erfogt als `flip|noflip up|down lefty|righty`

  Beispiel:

```
send     GetStatus
receive  noflip down lefty
```

3.4 Greifer-Kommandos

- `HasGripper` Nur Adept-Roboter

  Prüft, ob der serielle Greifer angeschlossen ist und funktioniert.

  Beispiel:

```
send     HasGripper
receive  true
```

- `GripperGoHome` Nur Adept-Roboter

  Bewegt den Greifer in die Referenzposition (etwa 4 cm weit offen).

  Beispiel:

```
send     GripperGoHome
receive  State: 01000000000000010000000000000000
         true
```

- `GripperMove amp` Nur Adept-Roboter

  Bewegt den Greifer mit der in `amp` angegebenen Ampere-Zahl. Negative Werte schließen den Greifer, positive Werte öffnen den Greifer.

  Beispiel:

```
send     GripperMove -1
receive  Start moving with amperage -1.000
         true
```

- `GripperMoveToPosition pos` Nur Adept-Roboter

  Bewegt den Greifer an die angegebene Position `pos` (Öffnung in m)

  Beispiel:

```
send     GripperMoveToPosition 0.025
receive  State: 01000010010000010000000000000000
         true
```

3.5 Weitere Befehle

- `ForwardCalc` j_1 j_2 j_3 j_4 j_5 j_6

  Berechnet die einer Gelenkstellung entsprechende homogene Matrix.

  Beispiel:

send `ForwardCalc 10 0 0 0 0 0`
receive `0.000000 -0.173648 -0.984808 1741.140107` ↩
`0.000000 -0.984808 0.173648 -307.009978` ↩
`-1.000000 -0.000000 -0.000000 640.000000`
`noflip up lefty`

- `BackwardCalc` $m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,4}$ $m_{2,1}$ $m_{2,2}$ $m_{2,3}$ $m_{2,4}$ $m_{3,1}$ $m_{3,2}$ $m_{3,3}$ $m_{3,4}$ ↩
  `flip|noflip up|down`

  Berechnet aus einer angebenen Matrix und der korrespondierenden Roboterkonfiguration die zugehörigen Gelenkwinkel.

  Beispiel:

  send `BackwardCalc` ↩
  `0.000000 -0.173648 -0.984808 1741.140107` ↩
  `0.000000 -0.984808 0.173648 -307.009978` ↩
  `-1.000000 -0.000000 -0.000000 640.000000` ↩
  `noflip up lefty`
  receive `10.000001 -0.000055 0.000110 0.000000 -0.000055 0.000000`

- `DirectAdeptCmd MSG` Nur Adept-Roboter

  Sendet `MSG` an den Adept und führt das Kommando dort aus.

  Beispiel:

  send `DirectAdeptCmd jmove 0,0,0,0,0,0`
  receive `true`

Protokollhistorie

1.0a Fehlerkorrektur (`SetVerbosity` war nicht dokumentiert)
1.0 Initiale Version

Achtung! Zur Zeit sind folgende Kommandos implementiert aber nicht dokumentiert:

`DisableLin`, `DisableRoutetest`, `DoAlterCart`, `DoAlterJoint`, `EnableLin`, `EnableRoutetest`, `GetAllowedStatus`, `GetJointsRowWise`, `GetMinChangeWeights`, `IsPossible`, `MoveLINHomRowWise`, `MovePTPHomRowWise`, `MovePTPHomRowWiseStatusTurn`, `MovePTPJointsStatus`, `ResetAllowedStatus`, `ResetMinChangeWeights`, `RoutetestJoints`, `RoutetestRowWiseStatus`, `SetAllowedStatus`, `SetMinChangeWeights`, `GetRTSpeedControl`