Philips Experimental Robot Arm

User Instruction Manual v1.1

APT536-09-9962   2010-1-4

Philips Applied Technologies

Philips Experimental Robot Arm

User Instruction Manual v1.1

Mechatronics

Rob Rijs, Rob Beekmans, Sait Izmit, Dave Bemelmans

APT536-09-9962 2010-1-4

Published by: **Philips Electronics Nederland B.V.**
Philips Applied Technologies
Eindhoven, The Netherlands

Title:	Philips Experimental Robot Arm, User Instruction Manual v1.1
Author:	Rob Rijs, Rob Beekmans, Sait Izmit, Dave Bemelmans
Document code:	APT536-09-9962
Date:	2010-1-4
Distribution:	
Status:	

If you have any comment to make, please feel free to contact the author:

Philips Applied Technologies
Mechatronics
Rob Rijs
HTC-7
Eindhoven,5656AE, The Netherlands
Tel.: +31 6 12973807

Copies of this document can be ordered from the distributor:

Philips Applied Technologies
Mechatronics
Secretary
HTC-7
Eindhoven, 5656AE, The Netherlands
Tel.: +31 40 2748793

Contents

1 Intended use

The experimental robot arm intended use is as a research tool. The human kinematics and sense of force and position allow new developments in the field of robotics. The robot transmits sensor data as position and force to the PC on the user side. The PC is equipped with a basic test program to show a simple Point-to-Point motion application and to demonstrate sensor read-out. For research purposes the user can program this PC and use the sensor data to send actuator commands to the robot.

Example research applications typically include doing household tasks and caring for people. Research typically relates to the fields of artificial intelligence, control and dynamics.

The robot is intended to be used at a research laboratory and is not intended to be used in for example hospitals, wet environments or in a home environment.

2 Setting up the robot

2.1 SYSTEM OVERVIEW

The Philips Experimental Robot Arm delivery package contains the following items :

- Robot Arm
- Mounting plate and frame
- Power supply for Robot
- Emergency switch
- Desktop PC with Linux/Xenomai installation (incl. Keyboard & mouse)
- Linux Device Driver, User API and Example User Application
- Experimental Windows Driver and Matlab toolbox
- Example Matlab user application(s)
- 3rd party Windows application for uploading new firmware (inc. its documentation)
- Documentation
  - "Experimental Philips Robot Arm – User Manual"
  - "RT-Motion USB – User API Software Documentation"
  - Instruction day presentations

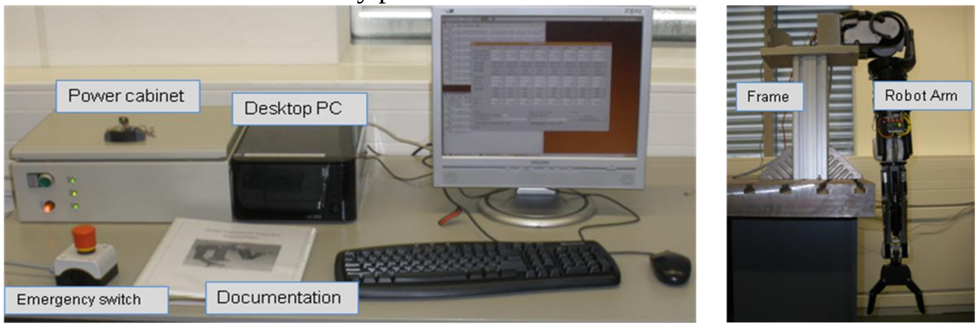


Figure 1 Overview of components

2.2 ROBOT MECHANICAL INSTALLATION

2.2.1 Mounting to frame

The user is responsible for mounting the robot in proper way. Guidelines are provided below.

The robot should be mounted with a strong enough connection to a frame that does not tumble in a worst-case posture with a worst-case load. A minimal guideline for that is to use a frame mass and dimension that complies with the balance rule (see also Figure 2 that shows the arm in the z-x plane):

M_{load} consists of the arm weight and is maximally 1.5 g
$L_{arm\text{-}base}$ is recommended as 0.9 m (so that the arm cannot reach the frame)
$L_{cog\text{-}base}$ is typically 0.2 m
M_{frame} is typically larger than 15 kg (taking a some safety margin into account)

Of course the force balance should also be OK in z-y plane.

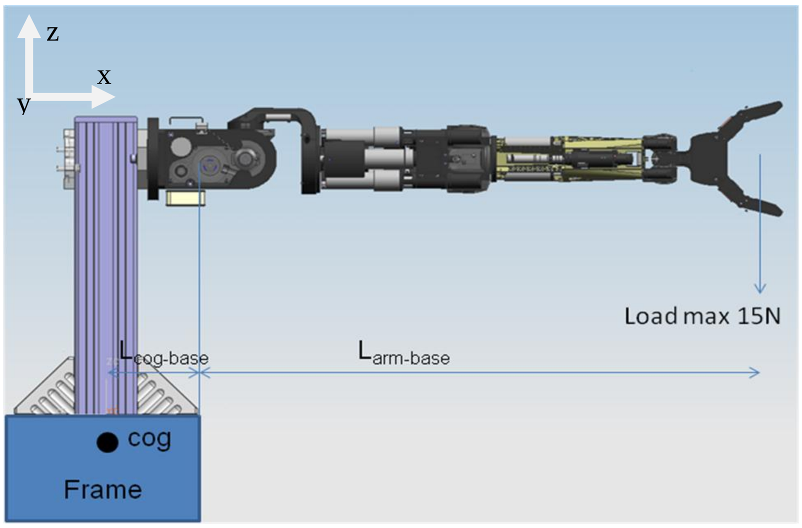


Figure 2 Force balance in worst case posture with load should be OK

Approximate weights:
Moving part
Upper arm 2.9 kg
Lower arm 0.8 kg
Hand 0.2 kg
Static part
Shoulder body: 3 kg

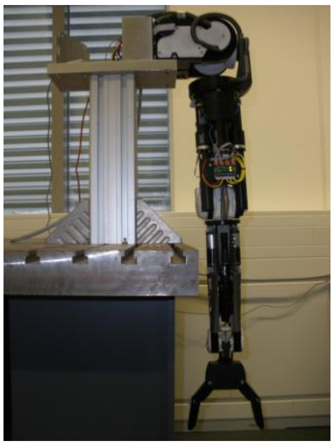


Figure 3 Support frame for both left and right arm.

A support frame is included with the arm, the support frame consist of a base plate that with M5 bolts connects the plate to the schoulder. A aluminum profile bar is used to obtain an uplifted z-height above the mounting plane. See Figure 4. The plate and aluminum profile are connected by M8 bolts. See Figure 4 for interface dimensions

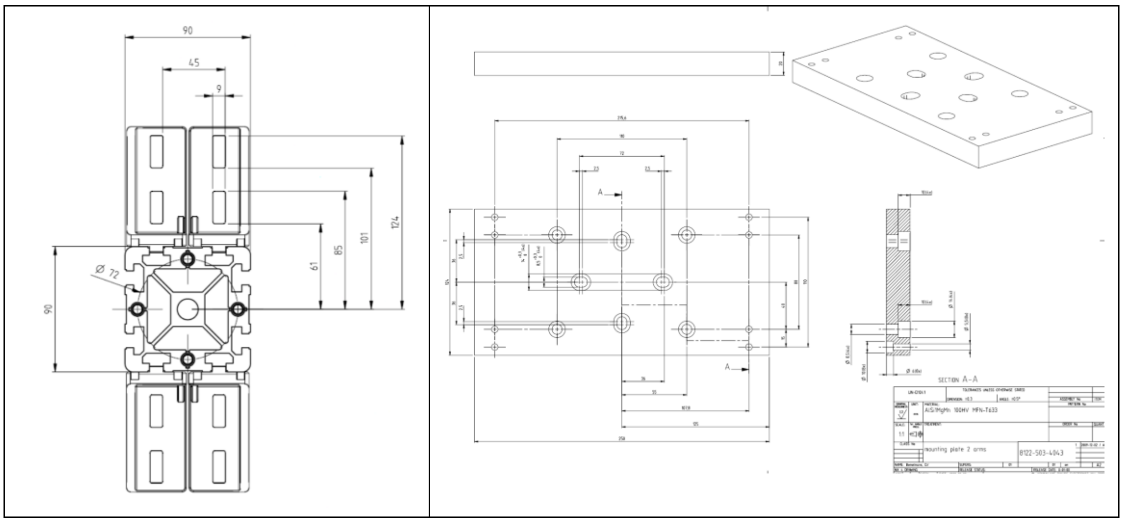


Figure 4 Drawings for mounting the robot arm

2.2.2 Safety fence

Whenever the power (48 and or 24 V) is switched on, the user must not be within reach of the robot. This means that a fence must be placed around the working radius of the robot when the power is on. Minimal radius for that fence is 0.8 m as is illustrated in Figure 5.

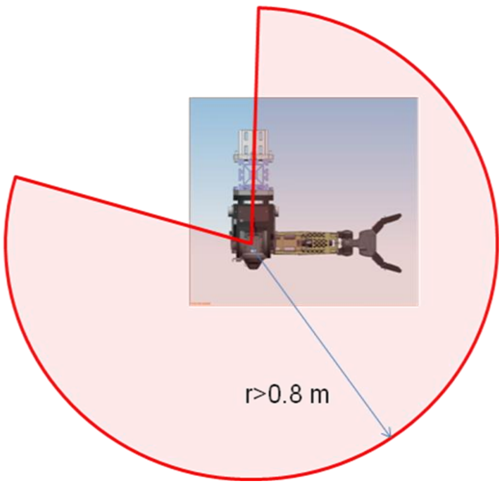


Figure 5 Top view of the working radius of the robot arm

2.3 ELECTRICAL CONNECTIONS

Before using the robotarm the system components have to be connected.
The pc should be connected as usual, the keyboard, mouse, screen and power cable can be connected to the dedicated ports.
The Robotarm USB cable should be connected to the USB port indicated in the figure below. Connect the power supply cabinet to 220V AC main. Do not forget to

earth the power supply using the provided cable. See Figure 6-Figure 8 for illustration.

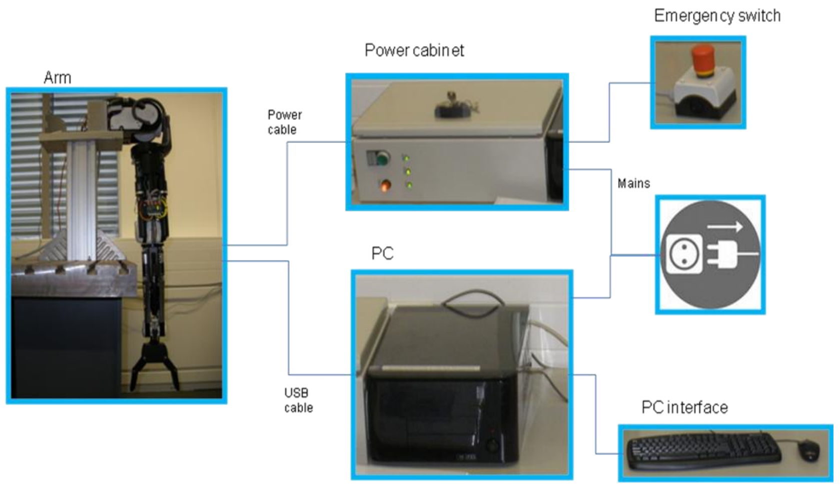


Figure 6 Connection overview

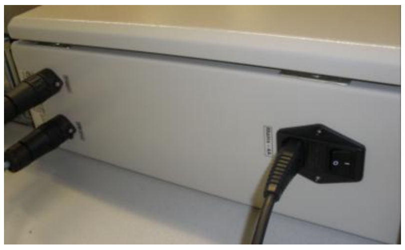


Figure 7 Power cabinet rear-side with the mains, robotpower and emergency switch connector

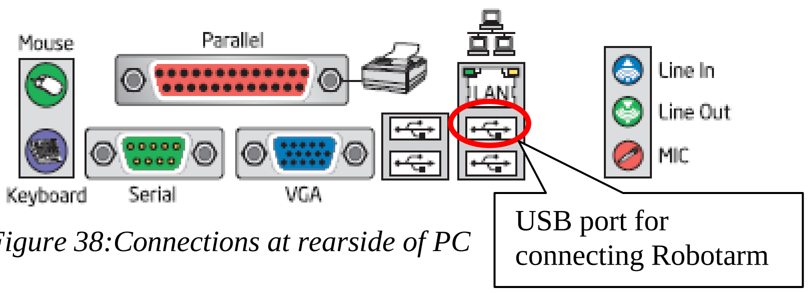


Figure 38:Connections at rearside of PC

The supply cabinet should be connected to the mains the robotarm and to the emergency stop.

2.4 USAGE

2.4.1 Safety

Warning:
This is a class A product. In a domestic environment this product may cause interference in which case the user may be required to take adequate measures.

- Stay out of reach when the power is on
  See also Figure 5. the fence should prevent users from coming into the working distance of the robot
- Do never stick fingers between shoulder, elbow and wrist joins as indicated in Figure 8. As the robot is able to lift 1.5 kg the force that can be exerted on body parts near the point of rotation is large. In all circumstances humans inserting body parts in these critical places should be avoided.

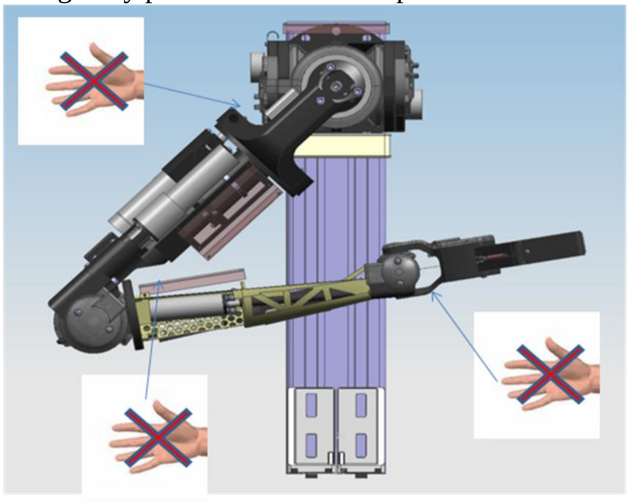


Figure 8 Never stick fingers or any other body parts in between shoulder moving parts or elbow moving parts.

- For certain teaching applications is will be convenient when the robot is moved manually. To do that, hold the robot only in the centre of the lower arm, centre of the upper arm and at the fingers. Only hold the robot when at least the 48 V and 24 V power supply is switched off (use the emergency switch)

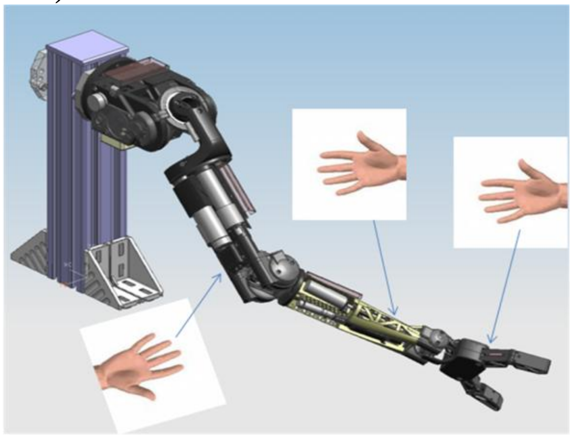


Figure 9 Safe holding place when the power (48 and 24 V is switched off)

2.4.2 Starting the system

To start working with the robot make sure that the USB reset switch and USB upload switch are in upper position. The switches are located at the shoulder front.

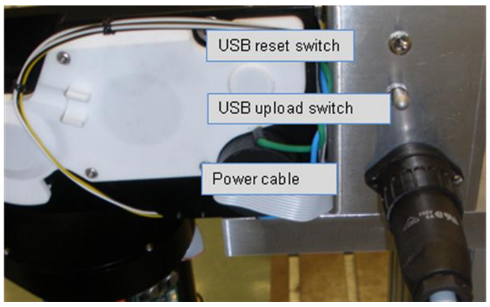


Figure 10 location of USB switches.

The power supply must be switched on (main switch is located on the rear side). The emergency switch mush be in outstretched position. If the green reset button is burning this button must be pressed to activate the power supply.

After doing so, all power supply leds (5 V, 24 V and 48 V) are green indicating that all power is on

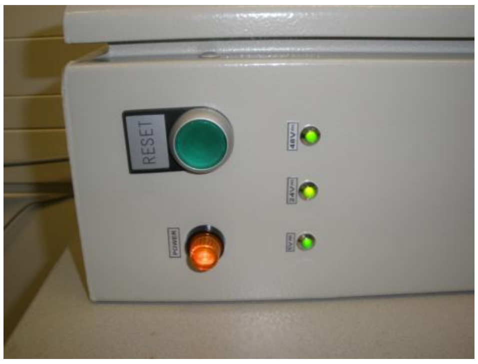


Figure 11 Photo of the power supply cabinet: All power is on if the 3 green leds are burning, the reset button is off and the power light is orange.

The emergency switch switches of the 48 and 24 power supply: the actuators can not work anymore. The 5 V power supply for the USB remains active so all sensors can be read. To reset the system after using the emergency switch: outstretch the emergency switch and press the reset button on the power supply cabinet.

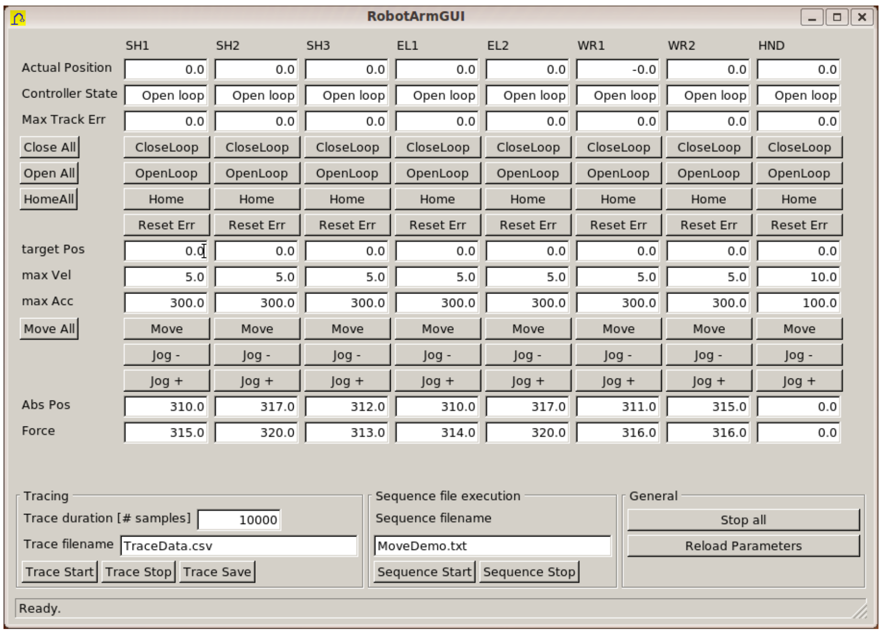
2.4.3 Using the example software

After starting the PC, the sequence below starts the example software:

- *Press emergency button*
- *Switch on the power of the supply cabinet (only 5 V active to supply the USB's)*

- Open a terminal shell
- Run the **cdservo** command to go to the directory with the controller executable
- Run the **ldlkm** command to load the USB driver
- Run the **ldxeno** command to load the Xenomai environment
- Before starting the application check that the ServoParam.ini file is in the current directory. If not, copy it from the
- Start the controller with the **sudo ./RobotArmApp** command
- Open another terminal shell for the userinterface
- Run the **cdgui** command to go to the directory with the gui executable
- Start the userinterface with the **./RobotArmGUI** command
- *Release the emergency button*
- *Press the reset button on the supply cabinet (see Figure 11)*

In the figure below a screenshot of the userinterface is shown



RobotArmGUI

	SH1	SH2	SH3	EL1	EL2	WR1	WR2	HND
Actual Position	0.0	0.0	0.0	0.0	0.0	-0.0	0.0	0.0
Controller State	Open loop	Open loop	Open loop	Open loop	Open loop	Open loop	Open loop	Open loop
Max Track Err	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Close All	CloseLoop	CloseLoop	CloseLoop	CloseLoop	CloseLoop	CloseLoop	CloseLoop	CloseLoop
Open All	OpenLoop	OpenLoop	OpenLoop	OpenLoop	OpenLoop	OpenLoop	OpenLoop	OpenLoop
HomeAll	Home	Home	Home	Home	Home	Home	Home	Home
	Reset Err	Reset Err	Reset Err	Reset Err	Reset Err	Reset Err	Reset Err	Reset Err
target Pos	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max Vel	5.0	5.0	5.0	5.0	5.0	5.0	5.0	10.0
max Acc	300.0	300.0	300.0	300.0	300.0	300.0	300.0	100.0
Move All	Move	Move	Move	Move	Move	Move	Move	Move
	Jog -	Jog -	Jog -	Jog -	Jog -	Jog -	Jog -	Jog -
	Jog +	Jog +	Jog +	Jog +	Jog +	Jog +	Jog +	Jog +
Abs Pos	310.0	317.0	312.0	310.0	317.0	311.0	315.0	0.0
Force	315.0	320.0	313.0	314.0	320.0	316.0	316.0	0.0

Tracing
Trace duration [# samples] 10000
Trace filename TraceData.csv
Trace Start | Trace Stop | Trace Save

Sequence file execution
Sequence filename MoveDemo.txt
Sequence Start | Sequence Stop

General
Stop all
Reload Parameters

Ready.

Figure 39: User interface screenshot

This simple user interface allows the user to control each axis separately but it is also possible to use all axes at the same time.

The units that are used on the user interface are degrees for positions, degrees per second for velocities and degrees per second ^ 2 for accelerations.

The top part of the user interface shows the servo controller status i.e. the actual position, the controller state and the max tracking error that has occurred. The controllers can have the following states: open loop, ready, moving and error.

Below the status fields there are four buttons that control the state of the servo controller. When the servo is in the open loop state, clicking the Close loop button will bring it into the ready state. The open loop button will brint the servo back to the open loop state. When the servo is in error, the reset error will bring it into the open loop state. The Home button resets the actual position to 0, independent of the actual position of the arm.

On the left side of the user interface there are buttons to perform the action for all axes.

The following three text fields and three buttons are for the move functions. The text fields contain the move parameters that are used when the buttons are clicked.

The two bottommost text fields are used to display sensor readings.

At the bottom of the window there are three group boxes for the tracing functions, the sequence file execution and some general commands.
The units for absolute position is bit value 0-1024 (uses a 10 bit ADC)
The units for force is bit value 0- 65536 (uses a 16 bit ADC). The hand force measurement uses a 10 bit ADC.

With the tracing function it is possible to capture controller data during a movement. It is possible to specify the number of samples that the capture should take. The maximum number of samples is 20.000. There are buttons to start a trace, stop the trace and save the captured data to disk. The header of the file describes the file format. The file directory can be reached by typing *cdservo* in a terminal shell

SEQUENCE FILE

The Sequence file execution can be used to run the robotarm through a sequence of moves. There are three commands that can be gives, A to set the acceleration, V to set the velocity and the P to start a point to point move. All commands are always set for all eight axes. An example of a sequence file is in the rtApp directory, the first lines are shown below. An example is shown in Figure
12.

```
A 300.0  300.0  300.0 300.0  300.0 300.0 300.0 300.0
V  15.0   15.0   15.0  15.0   15.0  15.0  15.0  15.0
P -30.0  -30.0    0.0  -5.0   -5.0   0.0   0.0   0.0
P -60.0    0.0    0.0  -5.0   -5.0   0.0   0.0   0.0
P -30.0  -30.0  -20.0  -5.0   -5.0   0.0   0.0   0.0
P -30.0  -30.0  -20.0 -90.0  -90.0   0.0   0.0   0.0
P -30.0  -30.0  -40.0 -45.0 -135.0   0.0   0.0   0.0
```

Figure 12 Example of the contents of a sequence file.

In the general group there are two buttons, the stop all moves button and the reload parameters button. When the stop all moves is clicked, a stop command is sent to all controllers.

When the reload parameters button is clicked, new servo controller parameters will be read from file and applied immediately in the controllers. The servo parameters are read from the file called ServoParam.ini, the fileformat consists of the name of the servo, followed by the values for P, I, D and the I saturation value. Below an example of this file.

The file is located in
/home/test/robotarm/robotarmforunies/build/xeno/debug/comp/controller/rtapp
Use cdservo to go to this directory,

```
SH1   10.00  10.00   0.05  20.00
SH2   10.00  10.00   0.05  20.00
SH3   20.00   0.00   0.04   5.00
EL1    2.00   0.00   0.02   5.00
EL2    2.00   0.00   0.02   5.00
WR1   10.00   0.00   0.03   5.00
WR2   10.00   0.00   0.03   5.00
HND   10.00   0.00   0.01   5.00
```

Figure 13 Contents of ServoParam.ini

2.4.4 Resetting the software

Occasionally, the USB firmware or software on the PC side may fail. This can be observed if one of the sensor readings in the GUI does not change when the arm is moving or some other unexpected behavior is observed.
The procedure to recover:

- Open all servo’s
- Press emergency button
- Close GUI window
- press [ctrl] c in the shell that is used to start the application (first opened terminal shell)
- reset the USB firmware (see Figure 10) by moving the rest switch downwards and after a short while upwards again.
- Repeat the sequence **ldlkm, ldxeno**, **sudo ./RobotArmApp** to start the application
- Start the GUI by using **./RobotArmGUI** in the GUI terminal shell
- Release the emergency button
- Press the reset button on the supply cabinet (see Figure 11)

2.5 ENVIRONMENTAL REQUIREMENTS

Climatic

Climatic conditions for transport and storage			
environment	Minimum temperature	Maximum temperature	Humidity
Transport and storage conditions	-25°C	+70°C	(5 – 95) %

Classification table for operational climatic conditions			
environment	Minimum temperature	Maximum temperature	Humidity
Temperature controlled	+10°C	+35°C	(20 – 80) %

Mechanical

Classification table for mechanical conditions on the product					
environment	Vibration			Shocks	
	Frequency	G value	Amplitude	G value	Pulse width
Stationary	(10 – 150) Hz	<2g	<0.15mm	<10g	(6 – 10) ms

Classification table for mechanical conditions on the packaging					
environment	Vibration			Shocks	
	Frequency	G value	Amplitude	G value	Pulse width
Stationary	(10 – 150) Hz	<2g	<0.15mm	<10g	(6 – 10) ms

2.6 MAINTENANCE

2.6.1 Cleaning

Clean the robot only when the power supply is switched off with a dry towel.

2.6.2 Repair

For fast repair it is recommended to have spare parts for the actuators and electronics, see ordering details in Appendix A.

For technical repair orders contact Apptech:
W. Enzing
Willy.Enzing@philips.com
+31 40 2793938

3 Robot Arm Outline

This chapter details information that will be of interest for path planning, control and application algorithms.

3.1 Mechanics

3.1.1 Differential drive building block

The robot arm contains 3 differential drives.
The inputs are two motor rotations φ_{motor1} and φ_{motor2}. The outputs of each differential drive are two orthogonal rotations:
$\varphi_{out1} = {}^1/_2\ (\varphi_{motor1} + \varphi_{motor2})$
$\varphi_{out2} = {}^1/_2\ (\varphi_{motor1} - \varphi_{motor2})$.
Motor torques relate to the output shaft by:
$T_{out1} = T_{motor1} + T_{motor2}$
$T_{out2} = T_{motor1} - T_{motor2}$.

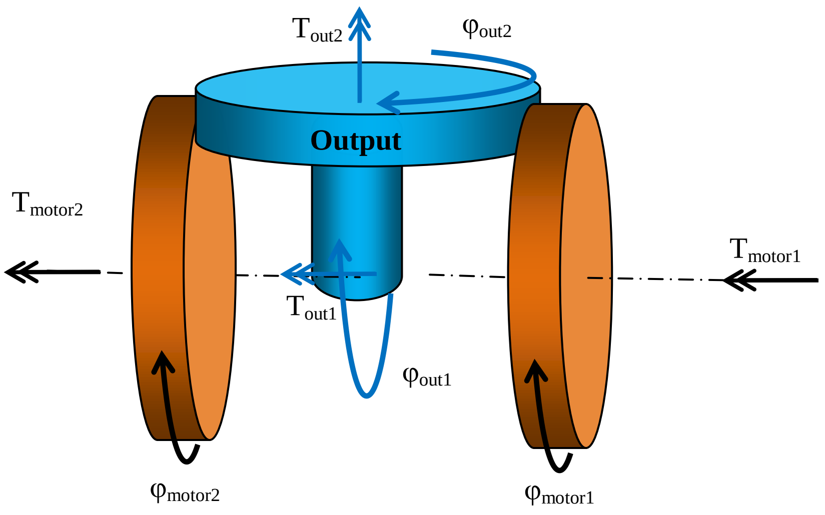


Figure 14 Schematic outline of differential drive.

3.1.2 Kinematics and range

The robot kinematics are outlined in Figure 15. The reach in world coordinates can be modified by changing the arm-to-frame mounting (see left-right arm mounting difference) . The upper arm length is 320 mm the lower arm length equals 280 mm. The hand length is approximately 200 mm

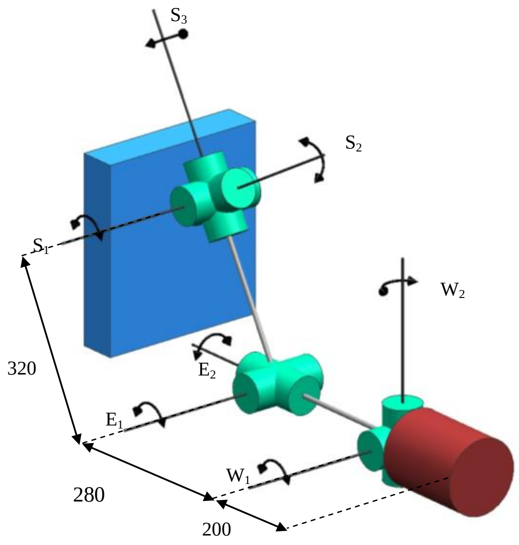


Figure 15 Kinematic overview: S_3 is a direct rotate, the others are outputs of a differential drive

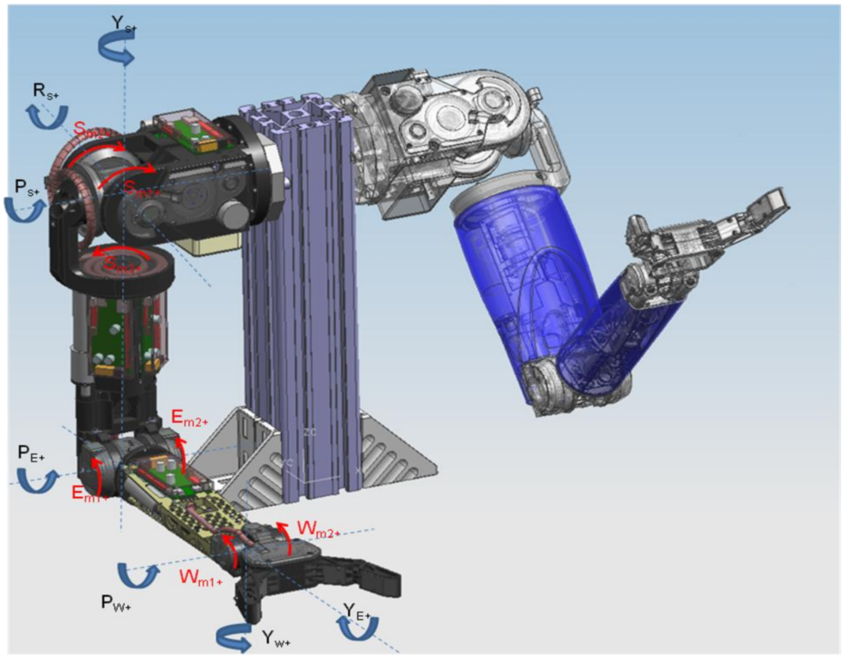


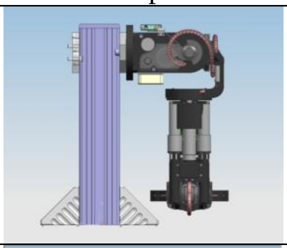
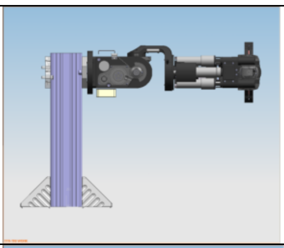
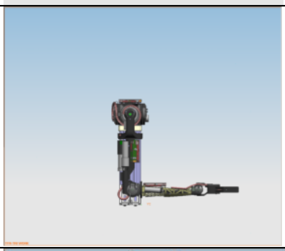
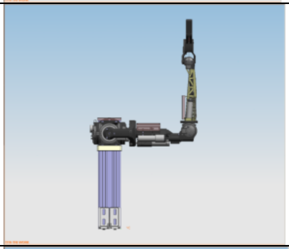
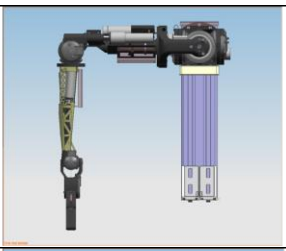
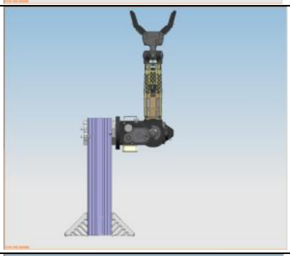
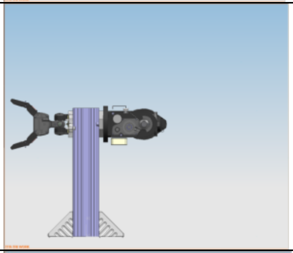
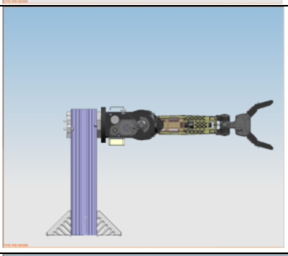
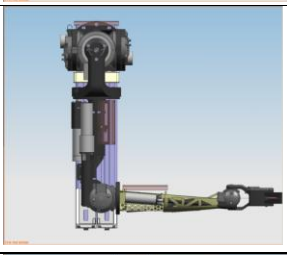
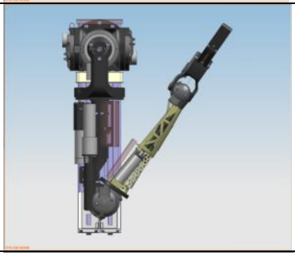
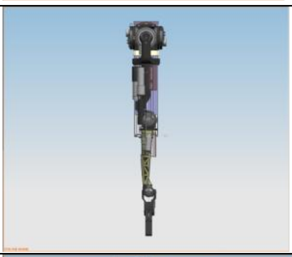
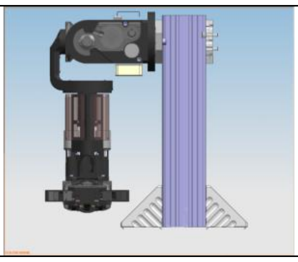
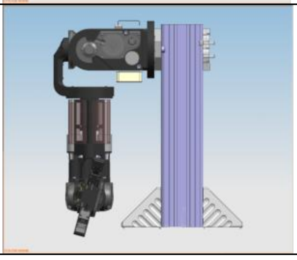
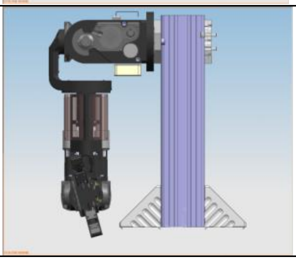
Figure 16 Robot arm outline of kinematics.

The robot reach is limited by mechanical end stops. The maximum rotational values are given in Table 1, the postures are shown in Figure 17
Range is given relative to the posture as shown in Figure 16.

Joint	**Rotation range**
R_S	0^0 , $+90^0$
P_S	-90^0 , $+90^0$
Y_S	-90^0 , $+90^0$

P_E	-90^0 , $+55^0$
Y_E	-105^0 , $+105^0$
P_W	-57^0 , $+57^0$
Y_W	-45^0 , $+45^0$

Table 1 Maximum reach of the robot for the orthogonal axis of rotations relative to the zero-posture as shown in Figure 16.

	Zero posture	+ max	- max
R_S			
P_s			
Y_s			
P_e			
Y_E			

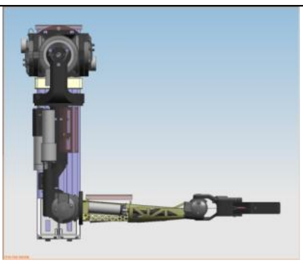
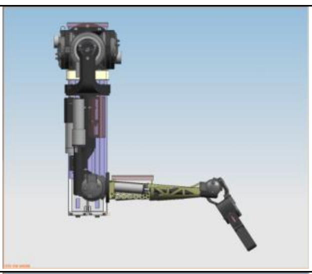
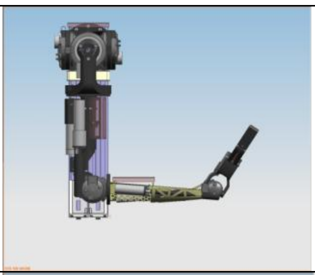
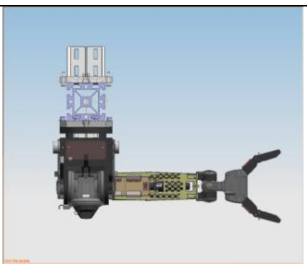
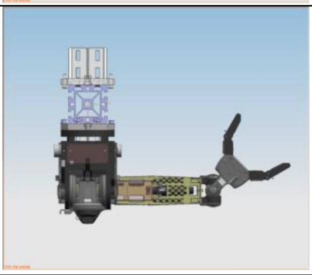
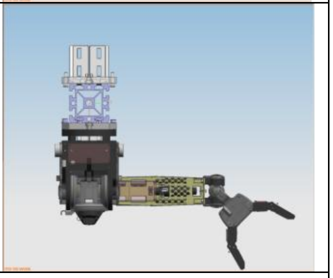
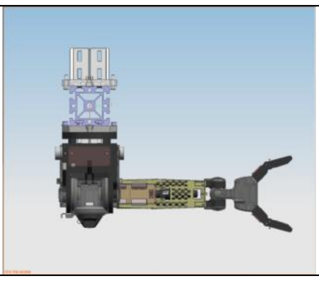
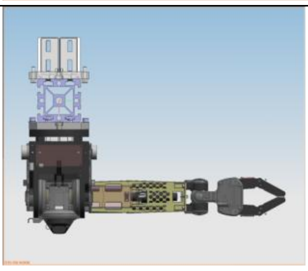
P_W			
Y_W			
			

Figure 17 Illustration of maximum joint positions

3.1.3 Technical specifications

The robot arm strength is designed such that it can lift a load of 1.5 kg (located at the centre of the fingers) with a fully stretched arm.
The elbow and wrist actuator need to be driven above their max continuous current to lift 1.5 kg. In rest with 1.5 kg at a fully stretched arm the position can be held continuously (if controlled properly) as friction offloads the needed motor current.
Motor speeds were taken by considering the maximum speed values during a set of household applications (see http://www.apptech.philips.com/robotics/index.html) that were based upon human motions.

Relevant specifications are summarized in Table 2. Motor inertia and other properties can be found on the Maxon website, see appendix A for motor ordering numbers.

	$M_{S1}+M_{S2}$	M_{S3}	$M_{E1}+M_{E2}$	$M_{W1}+M_{W2}$
$T_{DC10sec}$: [Nm] T_{DC}: Nm	13.4 13.4	9.8 9.8	5.9 4.5	1.25 1
Encoder resolution Counts per Turn	256	256	500	128
Total gearing [rad/rad]	1:550	1:371.25	1:348	1:290
Max Speed outgoing [rad/s]	1.47	2.27	2.8	3.54

Table 2 Robot arm parameters in motor coordinate space.

Take good care of the maximum speed. Even though the robot can reach higher speeds, this can lead to reduced life-time, immediate failure or missing encoder counts. USB firmware is not checking robot speed, this is the user responsibility.

3.1.4 Safety/overload precautions

To avoid damage to the robot drives and to improve safety a few measures for dealing with overload situations are taken. Measures are kept to a minimum to reduce interaction with the user application as much as possible.

Slip clutch in shoulder S_1 and S_2
The slip clutch protects the motor gearing from excessive torques that can be applied externally. The user can check this by comparing the sensor readings.
At firmware level only a check is made if excessive slippage occurs: this to protect the clutch from ongoing wear (and finally resulting in malfunction)

The slip clutch maximum torque is set a level that the robot is able to lift 1.5 kg.
The slip clutch maximum torque can be modified: contact Apptech to do that.

A software routine is running at the USB -processorboard that switches off the actuator loop if the difference between motor revelation and absolute angle sensor becomes larger than 2 revolutions. This is only implemented to avoid clutch wear in case of malfunctioning.

The 2 shoulder relative encoders on the shoulder motors and 2 shoulder absolute encoders can be used to calculate if there is a slip in the slip coupling. Similar to thermal watchdog task, since this task is also implemented as a non-real-time task, the user shouldn’t fully rely on this task for thermal protection. A higher level check in the user application should also be implemented to form a double layer of protection.

Motor overload checking in elbow and wrist actuators
Maximum current on all motors are limited in hardware on the RT-Motion USB boards. The wrist and elbow actuator can shortly be driven at a current above the maximum continuous motor current. In practical situations this can be required to accelerate a 1.5 kg load in the vertical direction.
The firmware on the USB board keeps track of the RMS current that is send to the actuators. If the RMS current during T seconds (set between 5 and 10 s depending on the actuator type) exceeds a certain threshold the motor loop is shut down.

For indication: at a worst case load of 1.5 kg a 30% duty cycle is allowed.
At higher loads or more intense duty cycles the thermal overload checking in the USB firmware may result in a system current saturation.

In order to make sure that the thermal time constants of the elbow and wrist motors are not exceeded, a non-real-time task called ‘Thermal Watchdog’ is implemented in the firmware. This task monitors the RMS current of the motor over time against the thermal time constant.

Whenever the (estimated) thermal limit is reached a firmware halt error is generated in order to disable the amplifiers. Be aware that independent of which motor generated the error, both amplifiers will be disabled in case of error.
Since this task is implemented as a non-real-time task, which tries to estimate the thermal behavior, the user shouldn't fully rely on this task for thermal protection. A higher level check in the user application should also be implemented to form a double layer of protection.

To avoid the thermal safety from intervening with the application it is recommended to implement an anti windup filter on the controller output on the user side, see Figure 18 for illustration

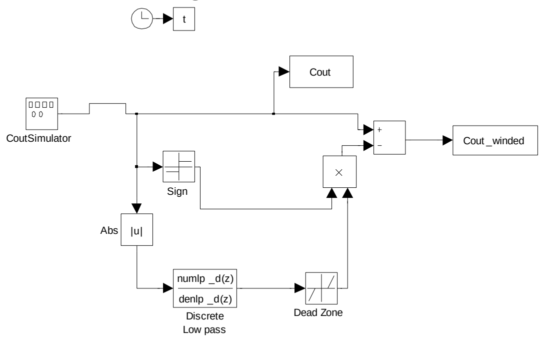


Figure 18 Anti windup filter example to avoid actuator overheating.

3.2 SENSORS

3.2.1 Force sensor

Each actuator degree of freedom contains a deformation sensor. Deformation of a compliant element in the drive chain is measured by an optical sensor. As deformation is approximately proportionally related to the load drive force this provides a measurement of force.

The force sensor is located at the load side of the gearing; by doing this, the gearing friction is not influencing the measured force that is transmitted to the load.

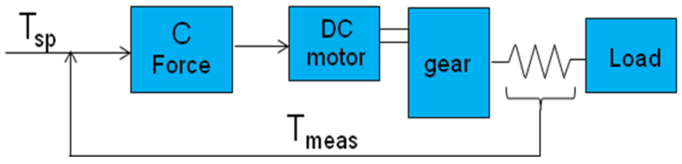


Figure 19 Schematics of force measurement and control

The force sensor is connected to the 16 bit AD converter

Sensor calibration

The optical reflectometer output $V_{measured}$ relates to deformation x via:

$$V_{measured} = \frac{K_{sensor}}{x + x_{offset}} - V_{offset}$$

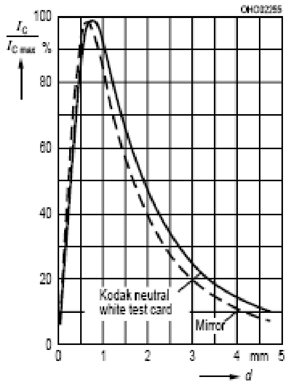


Figure 20 Sensor (Osram SFH 9202) output current in relation to the deformation. The output current is transformed to a voltage that is supplied the ADC of the processor board.

Approximate parameters are:
K_{sensor}=2.5 10^{-3}
X_{offset} = 0.25 10^{-3}
V_{offset} = 0.2
It is the user’s responsibility to calibrate these parameters: variation in sensor mounting and reflective material properties will cause variation in the sensor parameters.
The deformation translates to drive torque via a gain that is characterized by the stiffness. Approximate stiffness values are given in Appendix C.

Force control considerations
Force sensors are located in the actuator degrees of freedom. For feedback control purposes it is recommended to combine two force sensors to an orthogonal set of force measurements (T_{out1} and T_{out2}, see figure 1). This reduces interaction between the two servo loops.

SENSOR LOCATION

All force sensors measure the forces in the reaction path. This has practical advantages above measuring in the moving part of the arm. It can be shown by dynamic modeling (see example schematic of a 1 joint model) that these forces are dynamically identical.

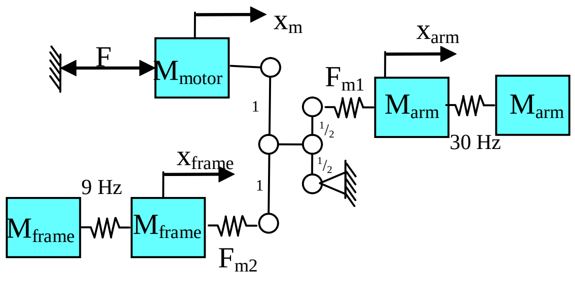


Figure 21 Dynamic model outline for measuring forces in the reaction path or moving part.

SHOULDER 1&2

See Figure 22: the compliant element and sensor are located near the second tooth gear (the same tooth gear that includes the slip clutch). See Figure 22.

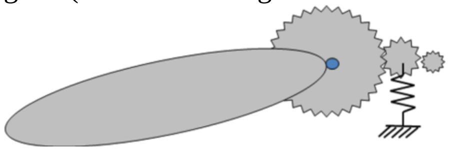


Figure 22 Illustration of measuring the torque in the upper arm

Expect a 700 Nm/rad stiffness that one would observe if the arm is felt in the Rs degree of freedom (see Figure 16).

SHOULDER 3

The shoulder 3rd degree of freedom has a static (wrt to shoulder differential) outer ring. The main body (upper arm) can rotate wrt to that ring. The actuator unit is flexibly suspended wrt to the upper arm main body, see Figure 23 for illustration.

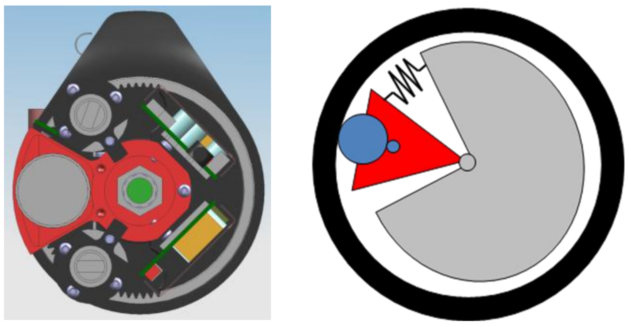


Figure 23 Shoulder 3: design and schematic of the compliant element

ELBOW AND WRIST.

The elbow and wrist force measurement share the same principle. Both are driven by a worm gear. Each wormwheel is coupled to one of the ingoing rotation of the the differential drive. The wormwheel shaft is suspended by leafsprings. For understanding the concept it is important to see that the contact point between worm and wormwheel acts as the pivotpoint.
See Figure 24 and Figure 25 for the schematic outlines.

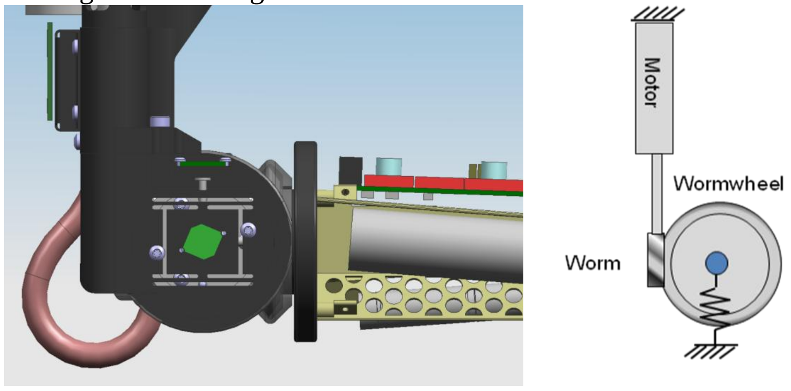


Figure 24 Outline of the elbow force measurement

It is important to notice that the hand pinching force is also felt by the wrist measurement. The hand pinching measurement needs to be processed together with the wrist measurements to obtain a measured of the actual wrist torques.

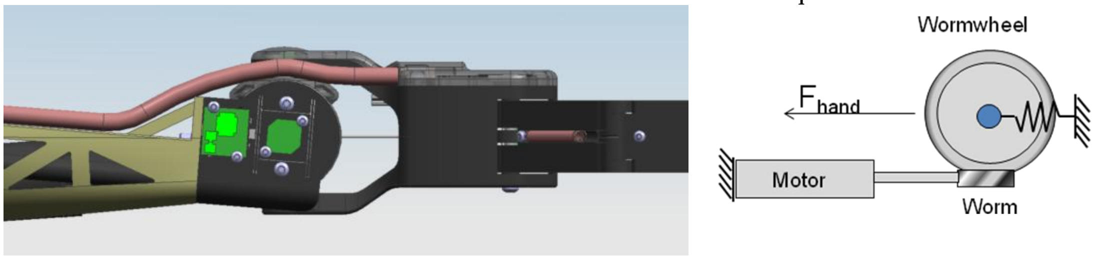
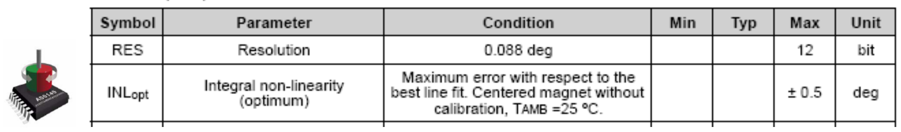


Figure 25 Outline of the elbow force measurement

The wrist torque measurement is limited to approximately 0.7-1 Nm. This means that in case of a worst-case combination of heavy load and huge hand pinching forces the sensor will saturate.

3.2.2 Hall angle sensor (absolute)

A hall position sensor is used in all loops to have means of absolute angle measurement. This prevents the necessity of homing routines and can also be used in a low bandwidth control loop.
For precise absolute applications (<0.5 degrees) it is recommended to write homing routines that use the mechanical end stops.



Symbol	Parameter	Condition	Min	Typ	Max	Unit
RES	Resolution	0.088 deg			12	bit
INL_{opt}	Integral non-linearity (optimum)	Maximum error with respect to the best line fit. Centered magnet without calibration, TAMB =25 °C.			± 0.5	deg

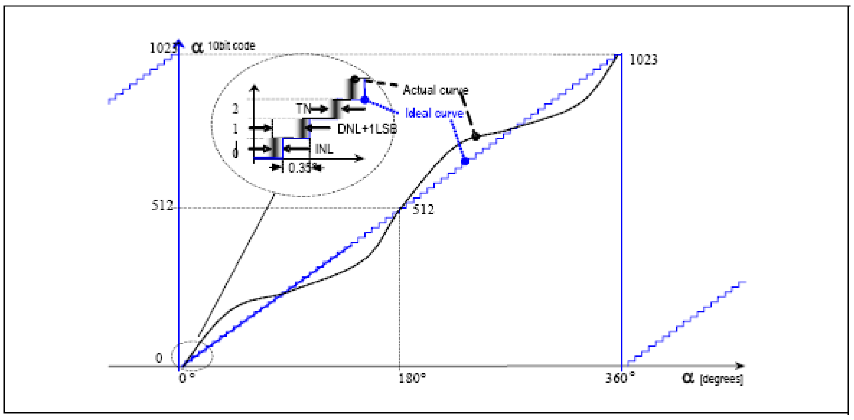


Figure 26 The sensor absolute errors can be influenced by mounting and magnetic field: use calibration methods to achieve good absolute accuracy.

Care should be taken of the 360 degree step that falls somewhere in its range.

The sensor uses a 10 bit AD convertor

Sensor location
All absolute sensors measure in the outgoing coordinate frame, see Figure 16

3.2.3 Motor position sensor (relative encoder)

Precise relative encoders are located on the motor axis. These precise sensor allow high bandwidth collocated control. See Table 2 for characteristics.

4 Electronics & Software background info

This chapter provides the necessary information for users to set up their own application software environment. Background on electronics and hardware is also provided to allow the user to further extend the robot functionality.

4.1 HARDWARE OVERVIEW

The hardware overview of the Experimental Philips Robot Arm is presented in Figure 27. The system contains four RT-Motion USB motion control boards from Philips Applied Technologies. The RT-Motion USB boards are explained in more detail in Section 4.2.

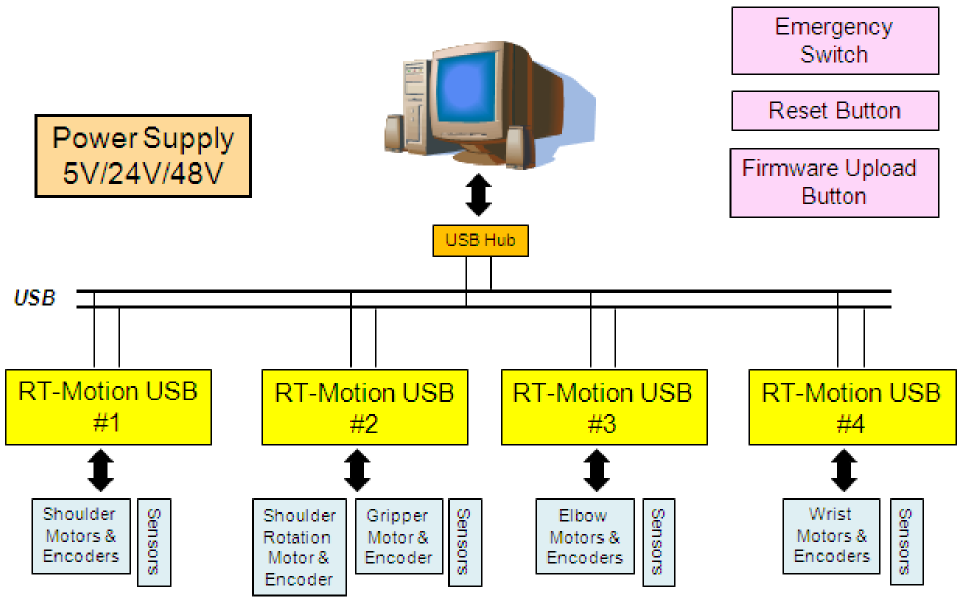


Figure 27:Robot Arm hardware Overview

The RT-Motion USB primary task in this application is 1) to read out al sensors and send them to the PC via the USB communication interface, 2) to transmit the user actuator command to the amplifiers 3) to check motor overheating and excessive slippage of the shoulder drives.

The four RT-Motion USB boards communicate with a central PC running Linux operating system with a Xenomai co-kernel, which provides real-time functionality.

RT-Motion USB boards interface with the following motors and encoders:

- **RT-Motion USB #1-** Referenced as board 0 in software, interfaces with the two shoulder motors and their encoders
- **RT-Motion USB #2-** Referenced as board 1 in software, interfaces with the two elbow motors and their encoders
- **RT-Motion USB #3-** Referenced as board 2 in software, motor channel X and encoder channel X interfaces with the shoulder rotation motor and encoder while motor channel Y and encoder channel Y interfaces with the gripper motor and encoder
- **RT-Motion USB #4-** Referenced as board 3 in software, interfaces with the two wrist motors and their encoders

There are also some absolute encoders and force sensors, which are interfaced to each RT-Motion USB board. Details of the sensors and their positions can be found in the detailed wiring diagram in appendix B.

The arm is powered by a power supply cabinet, which contains 5 volts, 24 volts and 48 volts outputs. The 5 volts output is used as the logic supply for the RT-Motion USB boards. All the motors on the arm are powered up using the 24 volts supply except the two shoulder motors, which are powered up by the 48 volts supply. An emergency button is connected to the 24 and 48 volts outputs in order to cut out the motor supply voltages in case of emergency. The 5 volts supply is intentionally left connected in order to enable communication with the RT-Motion USB boards for possibilities such as reading back firmware errors.

On the robot arm shoulder box, two switches are made available to the user. The reset switch is used to reset the RT-Motion USB firmware while the upload switch is used to upload new firmware to the RT-Motion USB boards. See Figure 10.

WARNING! Each RT-Motion USB board is specially modified for the defined position. Therefore,

- Boards shouldn’t be swapped with each other
- Motor wirings shouldn’t be modified
- Since other small PCBs are used due to ease of cabling, power limits of cables/components etc., the user should not modify the cabling

4.2 RT-MOTION USB

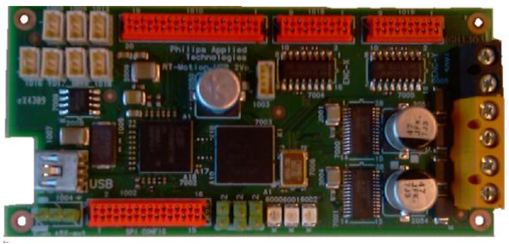


Figure 28: RT-Motion USB Motion Control Board

RT-Motion USB, which is presented in Figure 28, is a 2-axes motion control board with integrated drive units and extensive I/O. It is especially designed and developed to be used in high-speed USB-based distributed motion control applications. The electronics are supported by real-time software architecture, which is implemented both on firmware and PC-side driver levels.

RT-Motion USB motion control boards contain a 32-bit processor. An onboard CPLD is also included in the design for interfacing with incremental motor encoders. Two 150 watt DC motor amplifiers are used to drive two brushed DC motors. Five 10-bit and two 16-bit analog inputs together with 16 bit bidirectional digital I/O pins are also present on the RT-Motion USB motion control boards.

4.2.1 Functional Overview

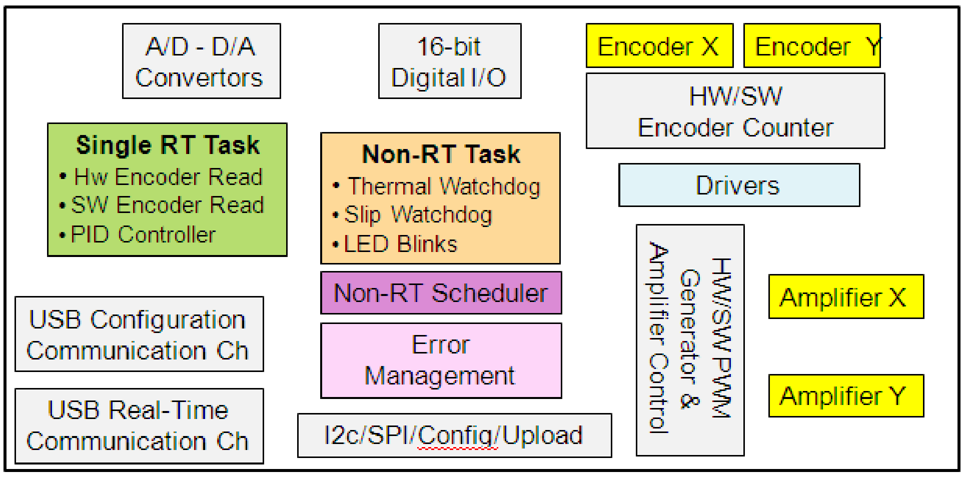


Figure 29: RT-Motion USB – Functional Overview

The functional overview of the RT-Motion USB motion controller is presented in Figure 29. The firmware can execute a single hard real-time task. In the example robot arm user example provided, a hard real-time task running at 5 kHz reads the encoder count values from the HW encoder counter and stores them in a buffer for the PC side to read them back. The user should be careful when choosing the execution frequency of the hard real-time task. Enough processing bandwidth should be left also for other software modules.

USB communication is handled by the communication layer, which will be explained in more detail in the next section. There is also a non-real-time task scheduler present in the firmware. Tasks such as thermal and slip watchdogs can be scheduled using this scheduler. There is also a error management module, which will be explained in more detail in the coming sections.

RT-Motion USB board contains also three LEDs. Even though the user can switch these LEDs from the user API, their main indications are as follows:

- **Green LED:** On after initialization, blinking whenever non-real-time scheduler is running.
- **Orange LED:** Off after initialization, indicates how much processing power the hard real-time task is using. It should never be fully on. There should be enough processor bandwidth for other software modules.
- **Red LED:** Off after initialization, indicates firmware error state.

4.2.2 USB Communication

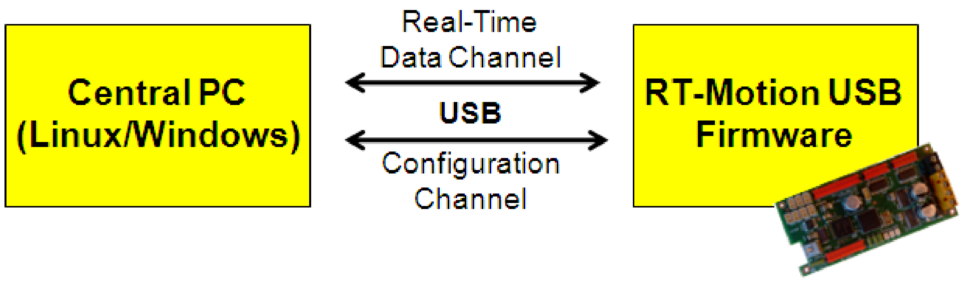


Figure 30: RT-Motion USB – Communication Overview

As shown in Figure 30, RT-Motion USB communicates with the central PC through two communication channels over USB. These channels are:

- **Configuration Channel** –Handles configuration related messages to serve user configuration requests.
- **Real-Time Data Channel –** Handles real-time data communication. Transfers input data of RT-Motion USB to the user while outputs user defined data using board outputs.

The communication layer is executed with a lower priority than hard real-time task. Therefore enough processor bandwidth should always be made available by the hard real-time task for USB communication.

Over USB, all transactions are initiated by the PC. Real-time data write (PC to Firmware) transactions also trigger the firmware to fill RT-Motion USB outgoing buffer for the next read (Firmware to PC) transaction. Therefore, it is advised to execute the control loop with one sample delayed actuation signal. A possible pseudo code such a scheme is as follows:

```
while(ControllerActive){
     WaitForNextSamplingTime();
     rtm_usb_SendStdMsg(Devnr,ActuationSignal);
     SensorInputs = rtm_usb_ReadStdMsg(Devnr);
     ActuationSignal = Controller(SensorInputs);
}
```

Message indexing is possible for real-time data verification. The user can increment the index before each write transaction. When a read transaction is performed, the firmware will also return the last received index. By comparing the two, user can verify correct message is being sent and received.

4.2.3 Firmware Errors

There is an error management module present in the RT-Motion USB firmware. There are four errors states:

- **No Error State –** Firmware is fully functional.
- **Non-Fatal Error State –** Error Code logged and Red LED is turned-on.
- **Halt Error State–** Error Code Logged and system is halted (all outputs are disabled). Red LED blinking task is set active.
- **Fatal Error State –** Firmware is terminated.

Using the user API, user can always read back from the firmware, the error count, the error state of the firmware and the last 4 error codes that are logged. User can also reset these values via the user API. Whenever *Halt Error* occurs, error needs to be cleared for output access. Safety critical tasks such as thermal and slip coupling watchdog tasks (see chapter 3) will generate halt errors.

4.2.4 I/O Configuration & Pinning

RT-Motion USB motion controller boards contain the following I/O:

- 16 bit digital bi-directional I/O (3.3V – 5V input tolerant)
- 1x10-bit 5ch analog inputs (0-3V), 2x16-bit analog inputs (0-3.3V)
- 2x16-bit analog outputs (0-2.7V) (used to drive amplifiers)
- 2 brushed DC motor amplifier outputs (each 150 watt max.)
- 2 differential encoder input
- 3 LEDs

Table 4.1: RT-Motion USB Connector List

Connector ID	**Description**
1000	Amplifier Power Supply (50V/ 6A max)
1001	Amplifier X (0) Output
1029	Amplifier Y (1) Output
1002	Configuration/SPI/i2c Connector
1004	Power Source Jumper
1007	Mini USB Connector
1010	Digital I/O Connector (bits 0-15)
1003	Analog Output Connector
1008	Analog Input 0
1009	Analog Input 1
1012	Analog Input 2
1014	Analog Input 3
1015	Analog Input 4
1016	Analog Input 5
1017	Analog Input 6
1018	Encoder X (0) Differential Input
1019	Encoder Y (1) Differential Input
6000	Green LED
6001	Orange LED
6002	Red LED

The type of I/O connectors on RT-Motion USB motion controllers are listed in Table 4.1 above.

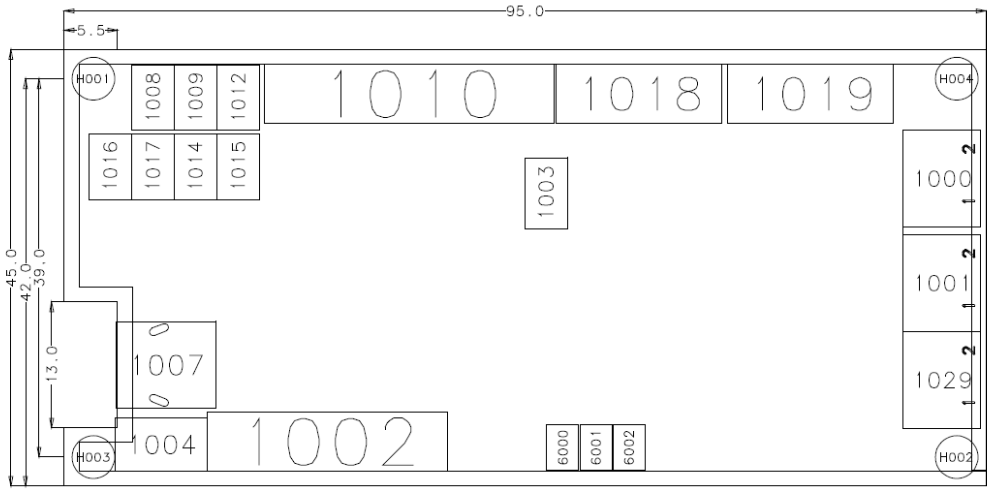


Figure 31: RT-Motion USB – Connector Layout (Top View)

The layout of these connectors is presented in Figure 31 together with the mechanical dimensions of the board.

Table 4.2: Digital I/O Connector 1010

Pin #	Description	Pin #	Description
1	GND	11	Digital I/O 9
2	Digital I/O 0	12	Digital I/O 10
3	Digital I/O 1	13	Digital I/O 11
4	Digital I/O 2	14	Digital I/O 12
5	Digital I/O 3	15	Digital I/O 13
6	Digital I/O 4	16	Digital I/O 14
7	Digital I/O 5	17	Digital I/O 15
8	Digital I/O 6	18	No Connection
9	Digital I/O 7	19	3.3 V Output
10	Digital I/O 8	20	5 V Output

The pinning for the digital I/O connector 1010 is presented in Table 4.2 above. These are non-isolated 3.3 volt bi-directional pins. Even though the pins are 5 volt input tolerant, the user should be careful when interfacing with 5 volt logic.

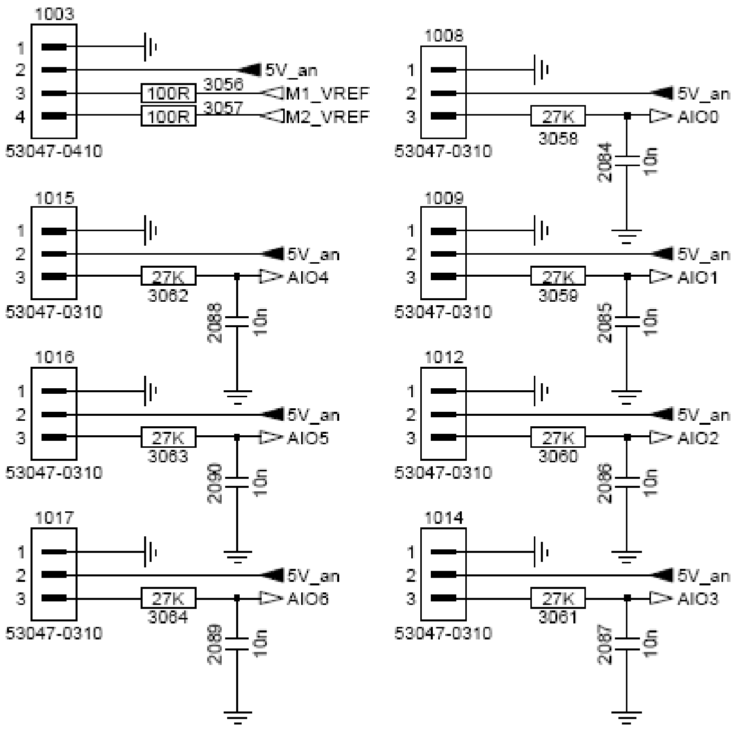


Figure 32: RT-Motion USB – Analog I/O Pinning

The pinning for the analog I/O connectors is presented in Figure 32. The signals M1_VREF and M2_VREF represent the analog outputs, which are also used to derive the motor amplifiers. Analog inputs 0 to 4 correspond to the 10-bit analog converter channels while analog inputs 5 and 6 correspond to the 16-bit analog converter inputs.

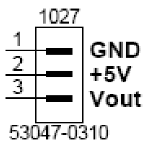


Figure 33: Absolute encoder and force sensor output connector

The absolute encoder and force sensors in the robot arm has similar pinning with the RT-Motion USB analog input connectors as it can be seen in Figure 33.

The range for the 10-bit analog inputs is in between 0 and 3 volts while that range is in between 0 and 3.3 volts for the 16-bit analog inputs. All the inputs are connected through an anti-aliasing filter to the converters. These anti-aliasing filters are implemented as 1st order low-pass filters with a corner frequency of 500 Hz.

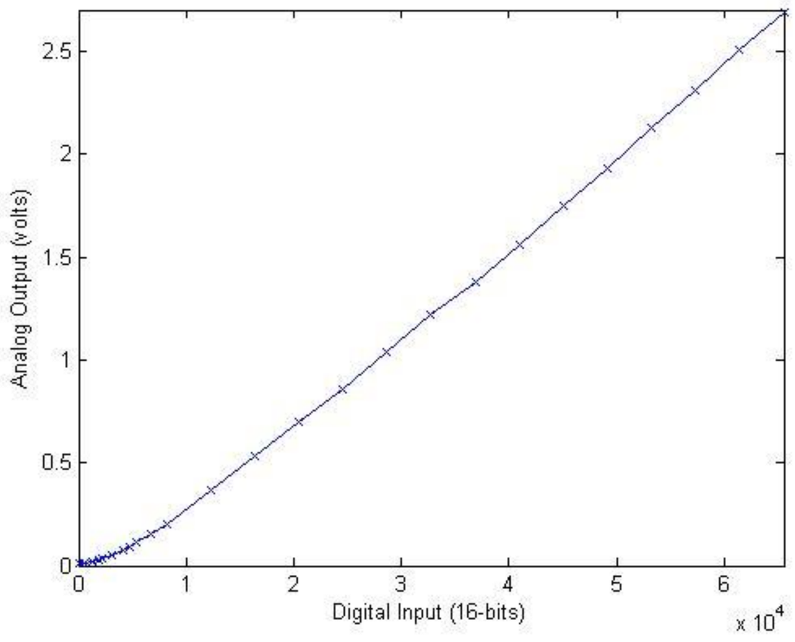


Figure 34: RT-Motion USB - Output Curve of the 16-bit D/A Converter

The analog output can generate a voltage value between 0 and 2.7 volts. Output behavior of the 16-bit D/A converter is presented in Figure 34. As it can be observed from the figure, the output of the converter is pretty linear with a small non-linear region close to zero. When the amplifiers are disabled, the firmware accepts 16-bit unsigned short integer (range 0 to 65535) as input for the D/A outputs while it accepts 16-bit signed short integer (range -32768 to 32767) when amplifiers are enabled. The sign information is used the set the direction of the amplifiers.

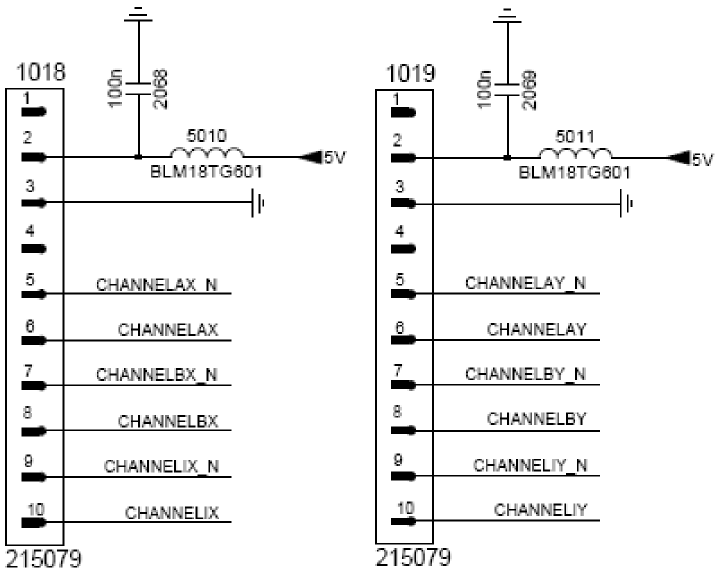


Figure 35: -Motion USB – Differential Encoder Pinning

The pinning for the two differential encoder connectors is presented in Figure 35. Even though the index signal is also shown in the connectors, the current encoder counter disregards them.

Table 4.3: Configuration/SPI/i2c Connector 1002

Pin #	Description	Pin #	Description
1	i2c Data	9	SPI Data In
2	GND	10	5V
3	i2c Clock	11	3.3V
4	5V Output	12	SPI Data Out
5	Reset	13	GND
6	5V External	14	SPI Clock
7	Program	15	GND
8	GND	16	SPI Chip Select

Connector 1002 contains configuration pins and SPI/i2c communication pins. The pinning for this connector is presented in Table 4.3 above. The Reset and the Program pins are already made available to the user on the shoulder of the robot arm through switches. 5V External is also connected to the external power supply to power up the boards. When using external supply, the Power Source Jumper 1004 should be put on the right position.

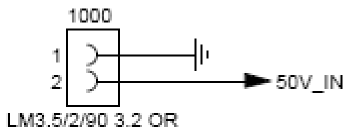


Figure 36: RT-Motion USB – Amplifier Power Supply Pinning

The pinning for the amplifier power supply connector 1000 is presented in Figure 36. This connector can handle maximum 50 volts and 6 amps continuous.

4.2.5 Amplifier Tuning for Robot Arm

The RT-Motion USB motion controller amplifiers are tuning for each of the motors in the robot arm in order to achieve the best performance. The user should ONLY use the configuration described below and should not play with the settings under no circumstance. Doing some may result in damage to electronics, motors and/or mechanics.

RT-MOTION USB #1 (REFERRED AS BOARD #0 IN SW)

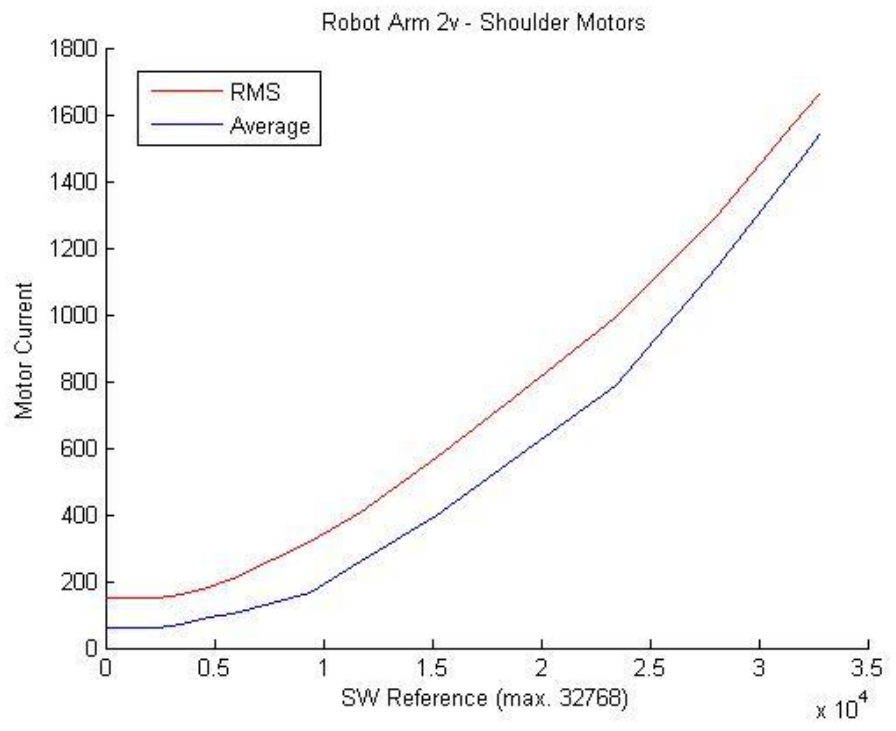


Figure 37: RT-Motion USB – Shoulder Amplifier Behavior

The two shoulder motors are connected to this motion controller board. The input-output behavior that is achieved using the tuning parameters below is presented in Figure 37.

Amplifier tuning for both amplifiers through user API should be set as follows:

- Decay Mode: 2 (Mixed Decay 48% Mode)
  - rtm_usb_Amp_DecMode(0,0,2);
  - rtm_usb_Amp_DecMode(0,1,2);
- Blank Pin: 1
  - rtm_usb_Amp_Blank(0,0,1);
  - rtm_usb_Amp_Blank(0,1,1);
- ExtMode: 0
  - rtm_usb_Amp_ExtMode(0,0,0);
  - rtm_usb_Amp_ExtMode(0,1,0);
- Amplifier PWM: 2 (No Amplifier PWM)
  - rtm_usb_Amp_SetEnableSigPath(0,0,1);
  - rtm_usb_Amp_SetEnableSigPath(0,1,1);
  - rtm_usb_Amp_SetPwmSrc(0,0,2);
  - rtm_usb_Amp_SetPwmSrc(0,1,2);
  - rtm_usb_Amp_EnablePwm(0,0);
  - rtm_usb_Amp_EnablePwm(0,1);
- No thermal checking
- Slip coupling check
  - rtm_usb_Sched_SetTick(0,1);

- rtm_usb_Slip_Config(0,0,4,-1,2,1,1564,90);
- rtm_usb_Slip_Config(0,1,3,1,2,1,1564,90);
- rtm_usb_Slip_SchedNonRt(0,100);
- rtm_usb_Sched_Enable(0,1);

RT-MOTION USB #2 (REFERRED AS BOARD #1 IN SW)

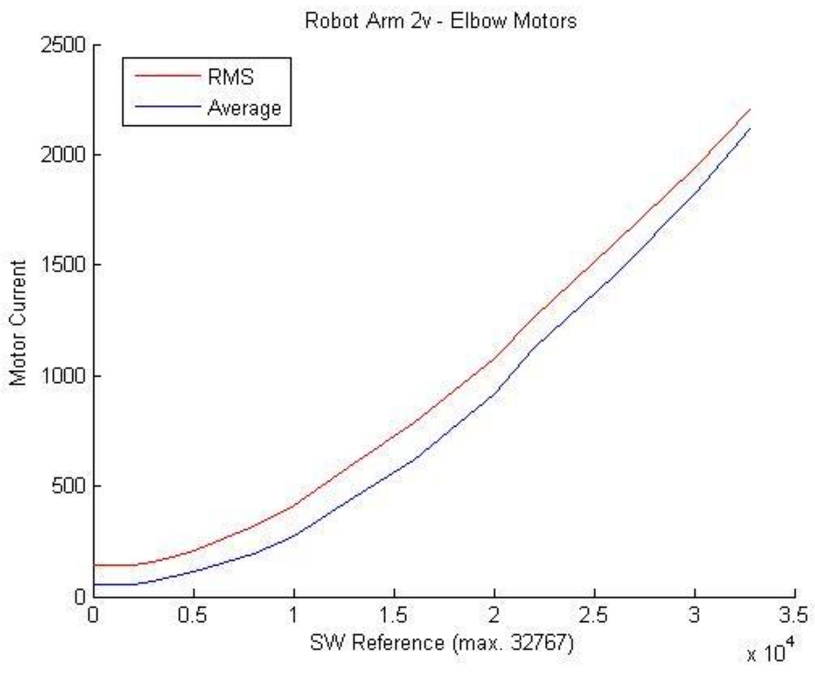


Figure 38: RT-Motion USB – Elbow Amplifier Behavior

The two elbow motors are connected to this motion controller board. The input-output behavior that is achieved using the tuning parameters below is presented in Figure 38.

Amplifier tuning for both amplifiers through user API should be set as follows:

- Decay Mode: 2 (Mixed Decay 48% Mode)
  - rtm_usb_Amp_DecMode(1,0,2);
  - rtm_usb_Amp_DecMode(1,1,2);
- Blank Pin: 1
  - rtm_usb_Amp_Blank(1,0,1);
  - rtm_usb_Amp_Blank(1,1,1);
- ExtMode: 0
  - rtm_usb_Amp_ExtMode(1,0,0);
  - rtm_usb_Amp_ExtMode(1,1,0);
- Amplifier PWM: 2 (No Amplifier PWM)
  - rtm_usb_Amp_SetEnableSigPath(1,0,1);
  - rtm_usb_Amp_SetEnableSigPath(1,1,1);
  - rtm_usb_Amp_SetPwmSrc(1,0,2);
  - rtm_usb_Amp_SetPwmSrc(1,1,2);
  - rtm_usb_Amp_EnablePwm(1,0);
  - rtm_usb_Amp_EnablePwm(1,1);

- Thermal checking
  - rtm_usb_Sched_SetTick(1,1);
  - rtm_usb_Thermal_SchedNonRt(1,10);
  - rtm_usb_Thermal_Config(1,0,6,27000);
  - rtm_usb_Thermal_Config(1,1,6,27000);
  - rtm_usb_Sched_Enable(1,1);
- No Slip coupling check

RT-MOTION USB #3 (REFERRED AS BOARD #2 IN SW)

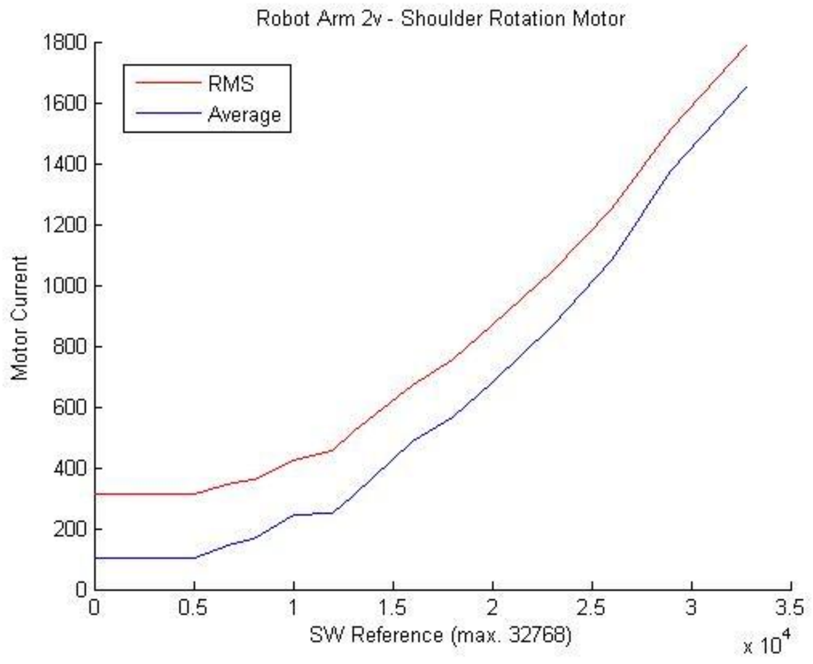


Figure 39: RT-Motion USB – Shoulder Rotation Amplifier Behavior

The shoulder rotation motor is connected to the X channel of this motion controller board. The input-output behavior that is achieved using the tuning parameters below is presented in Figure 39.

Amplifier tuning for amplifier channel X through user API should be done as follows:

- Decay Mode: 1 (Mixed Decay 18% Mode)
  - rtm_usb_Amp_DecMode(2,0,1);
- Blank Pin: 1
  - rtm_usb_Amp_Blank(2,0,1);
- ExtMode: 0
  - rtm_usb_Amp_ExtMode(2,0,0);
- Amplifier PWM: 2 (No Amplifier PWM)
  - rtm_usb_Amp_SetEnableSigPath(2,0,1);
  - rtm_usb_Amp_SetPwmSrc(2,0,2);
  - rtm_usb_Amp_EnablePwm(2,0);
- No thermal checking
- No Slip coupling check

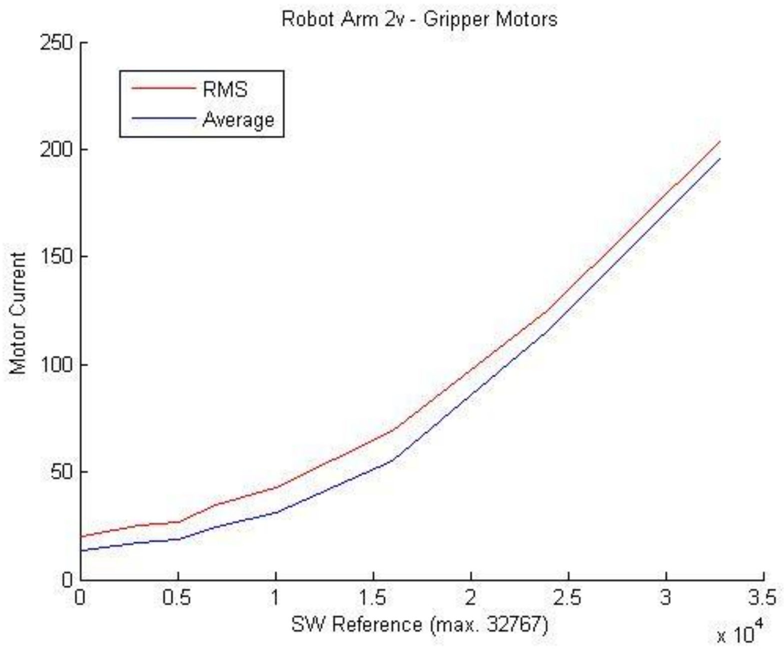


Figure 40: RT-Motion USB – Gripper Amplifier Behavior

The gripper motor is connected to the Y channel of this motion controller board. The input-output behavior that is achieved using the tuning parameters below is presented in Figure 40.

Amplifier tuning for amplifier channel Y through user API should be done as follows:

- Decay Mode: 1 (Mixed Decay 18% Mode)
  - rtm_usb_Amp_DecMode(2,1,1);
- Blank Pin: 0
  - rtm_usb_Amp_Blank(2,1,0);
- ExtMode: 0
  - rtm_usb_Amp_ExtMode(2,1,0);
- Amplifier PWM: 2 (No Amplifier PWM)
  - rtm_usb_Amp_SetEnableSigPath(2,1,1);
  - rtm_usb_Amp_SetPwmSrc(2,1,2);
  - rtm_usb_Amp_EnablePwm(2,1);
- No thermal checking
- No Slip coupling check

RT-MOTION USB #4 (REFERRED AS BOARD #3 IN SW)

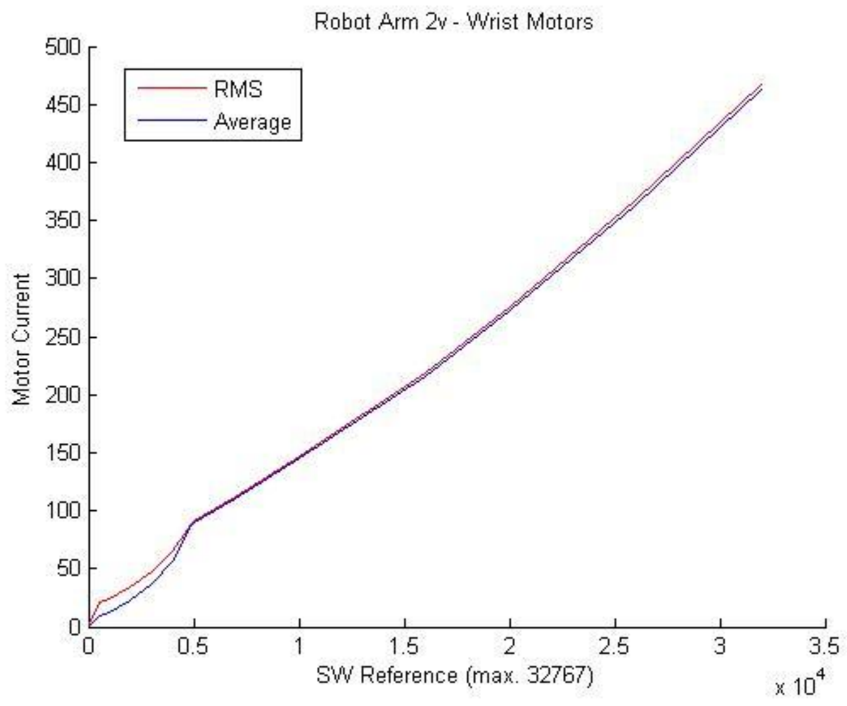


Figure 41: RT-Motion USB – Wrist Amplifier Behavior

The two elbow motors are connected to this motion controller board. The input-output behavior that is achieved using the tuning parameters below is presented in Figure 41.

Amplifier tuning for both amplifiers through user API should be set as follows:

- Decay Mode: 0 (Slow Decay Mode)
  - rtm_usb_Amp_DecMode(3,0,0);
  - rtm_usb_Amp_DecMode(3,1,0);
- Blank Pin: 0
  - rtm_usb_Amp_Blank(3,0,0);
  - rtm_usb_Amp_Blank(3,1,0);
- ExtMode: 0
  - rtm_usb_Amp_ExtMode(3,0,0);
  - rtm_usb_Amp_ExtMode(3,1,0);
- Amplifier PWM: 1 (HW PWM), Threshold: 5000, Gain: 1/1
  - rtm_usb_Amp_SetEnableSigPath(3,0,1);
  - rtm_usb_Amp_SetEnableSigPath(3,1,1);
  - rtm_usb_Amp_SetPwmThreshold(3,0,5000);
  - rtm_usb_Amp_SetPwmThreshold(3,1,5000);
  - rtm_usb_Amp_SetPwmThresholdGain(3,0,1,1);
  - rtm_usb_Amp_SetPwmThresholdGain(3,1,1,1);
  - rtm_usb_Amp_SetPwmSrc(3,0,1);
  - rtm_usb_Amp_SetPwmSrc(3,1,1);
  - rtm_usb_Amp_EnablePwm(3,0);
  - rtm_usb_Amp_EnablePwm(3,1);

- Thermal checking
  - rtm_usb_Sched_SetTick(3,1);
  - rtm_usb_Thermal_SchedNonRt(3,10);
  - rtm_usb_Thermal_Config(3,0,6,23000);
  - rtm_usb_Thermal_Config(3,1,6,23000);
  - rtm_usb_Sched_Enable(3,1);
- No Slip coupling check

4.2.6 Firmware Upload

Firmware updates can primarily be performed by Apptech personnel and tooling. Alternatively new firmware can be uploaded to RT-Motion USB boards via USB using the 3rd party Windows application Mass DFU, which is provided “As Is”.

The procedure for uploading new firmware through USB is as follows (See Mass DFU documentation for more details):

- Turn-off power and set the upload switch on the shoulder box
- Turn-on power. The orange and red LEDs on the RT-Moption USB boards should be on while the green ones are off
- Start DFU application on Windows PC, go to File→ Setup
- Select firmware binary file, click new topology
- Connect USB cable to Windows PC (install drivers if asked)
- After 4 boards appear in topology, File→Program
- There should be four boxes on the screen and they should appear blue if they are recognized by Windows PC
- Click “Start Automatic Programming”
- When finished, turn-off power, switch off program switch
- Turn-on power

CPLD images, on the other hand, can only be changed by Apptech personnel and tooling

4.3 PC-SIDE SOFTWARE

This section explains the PC side software of the robot arm.

4.3.1 Linux & Xenomai

Xenomai is a real time co-kernel which cooperates with Linux via Adeos. Xenomai enables the implementation of hard real time applications. Because of this fact Linux (Ubuntu) in cooperation with Xenomai is used as the real-time PC platform for the robot arm application.

4.3.2 Architectural Overview

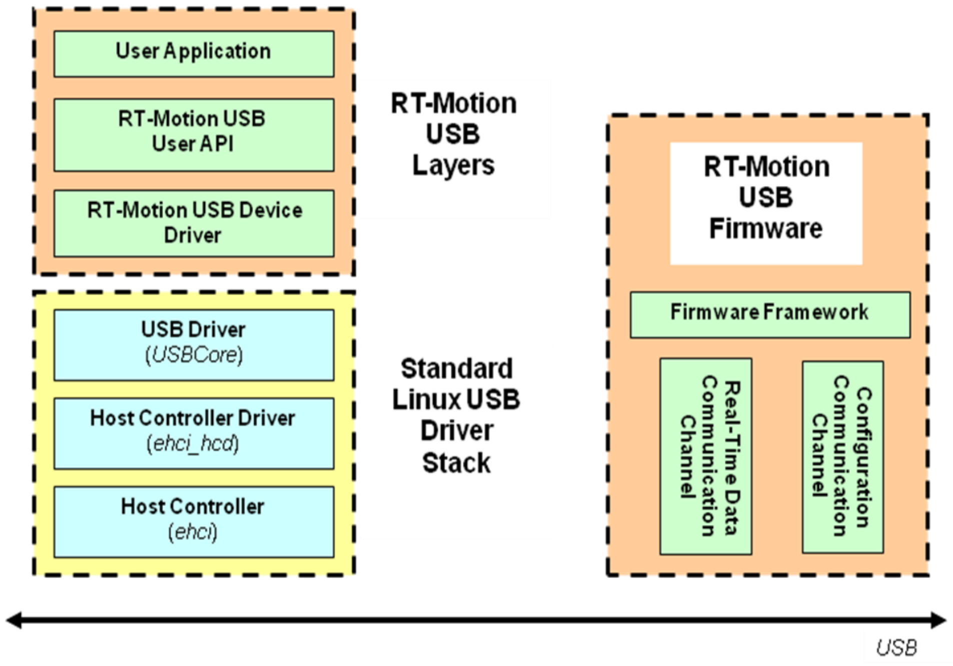

User Application
RT-Motion USB User API
RT-Motion USB Device Driver
RT-Motion USB Layers
USB Driver (USBCore)
Host Controller Driver (ehci_hcd)
Host Controller (ehci)
Standard Linux USB Driver Stack
RT-Motion USB Firmware
Firmware Framework
Real-Time Data Communication Channel
Configuration Communication Channel
USB

Figure 42: RT-Motion USB – Architectural SW Overview

The architectural software overview of RT-Motion USB and therefore that of robot arm is presented in Figure 42.

The PC communicates with RT-Motion USB motion controller board(s) via USB using the standard Linux USB driver stack. RT-Motion USB Linux device driver is implemented as a native Linux driver the running in kernel space. Motion control user applications, on the other hand, should be implemented as real-time Xenomai task(s) in the user space.

4.3.3 User API

The user API provides an interface between the user application and the RT-Motion USB firmware. It provides several firmware configuration and real-time data communication functions. More detail about these functions and their parameters can be found in the “RT-Motion USB – User API Software Documentation”. Both the Linux and Windows user API is provided as a binary file.

4.3.4 Example code

EXAMPLE CODE OVERVIEW

The application consists of two components, the controller and the userinterface. The controller contains the servocontrollers and performs the communication with the Robotarm hardware. The userinterface shows a graphical userinterface that allows the user to control the servocontrollers in the controller component and shows the actual status of the servocontrollers.

For the communication between the components a CORBA interface is used. The controller component has a CORBA interface which is used by the userinterface. By using a CORBA interface it is possible to run the userinterface on a different system than the controller.

The following sections describe the design of the components; the directory structure; building the components and starting the components.

SOFTWARE DESIGN

The software in divided in two components, the realtime controller software and the userinterface. The communication between the components is implemented using the CORBA protocol. An overview of this is shown below. Since the connction between the components is a network connection they can run on different systems.

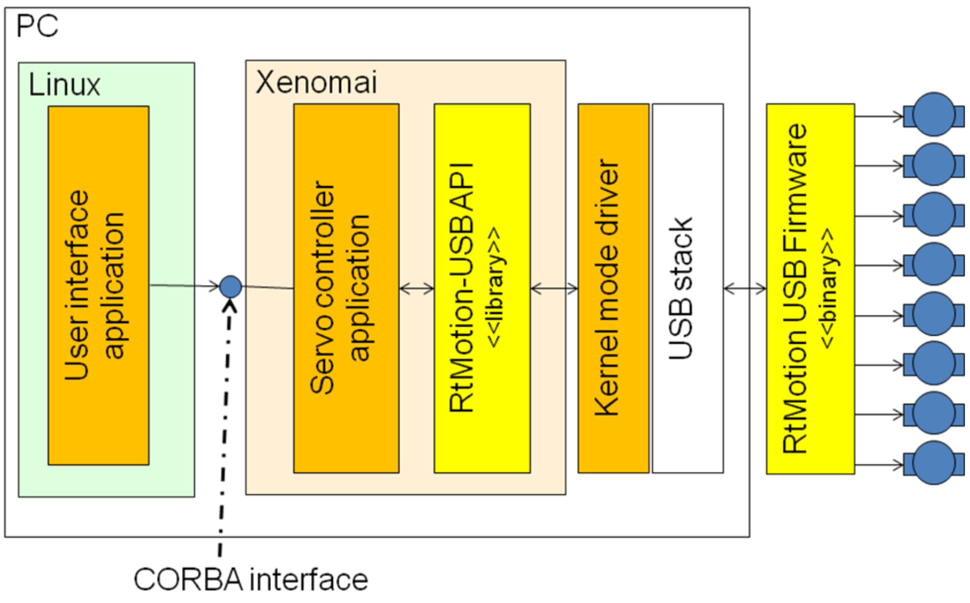

PC
Linux
User interface application
Xenomai
Servo controller application
RtMotion-USBAPI <<library>>
Kernel mode driver
USB stack
RtMotion USB Firmware <<binary>>
CORBA interface

Figure 43: SW-Components/Layers

Userinterface component design
The userinterface is a simple application that uses the FOX GUI toolkit (www.fox-toolkit.org) and CORBA for the communication with the controller component. The userinterface consists of a single window that contains all the controls (see Figure). It is a singlethreaded application that uses a timer to create periodic update events for the status information.

At startup the application queries the controller component for the number of servo controllers available, it creates a column with textfields and buttons for each controller. The buttons that are tied to a specific controller all receive the controller index number in their userdata field. When these buttons are clicked, this index is used to give a command to the correct servocontroller.

Controller component design
The controller component runs the realtime loop for the eight servo controller for the axes and the tracing utility. It also performs the application startup and configuration. The internal structure of the controller is shown below.

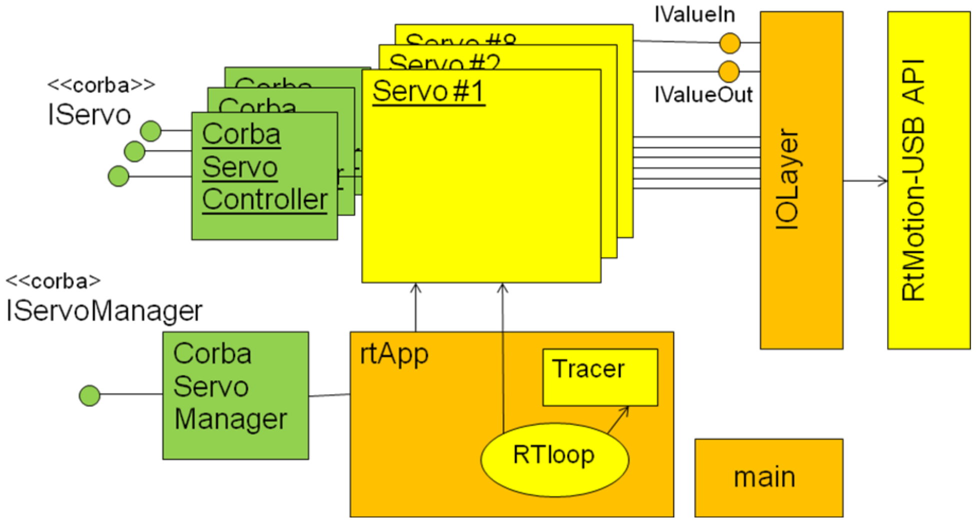


Figure 44: Controller component structure

The controller component contains eight servos, the internal structure is shown below. Each axis can be in the following states: Open loop, Ready (closed loop), Moving or Error. The axis goes into error when the tracking error (setpoint position – actual position) becomes too large or when the USB firmware indicates an error condition. The unit that is used internally is angular degrees. The constants Kact and Kenc are used to convert the externally used units into degrees.

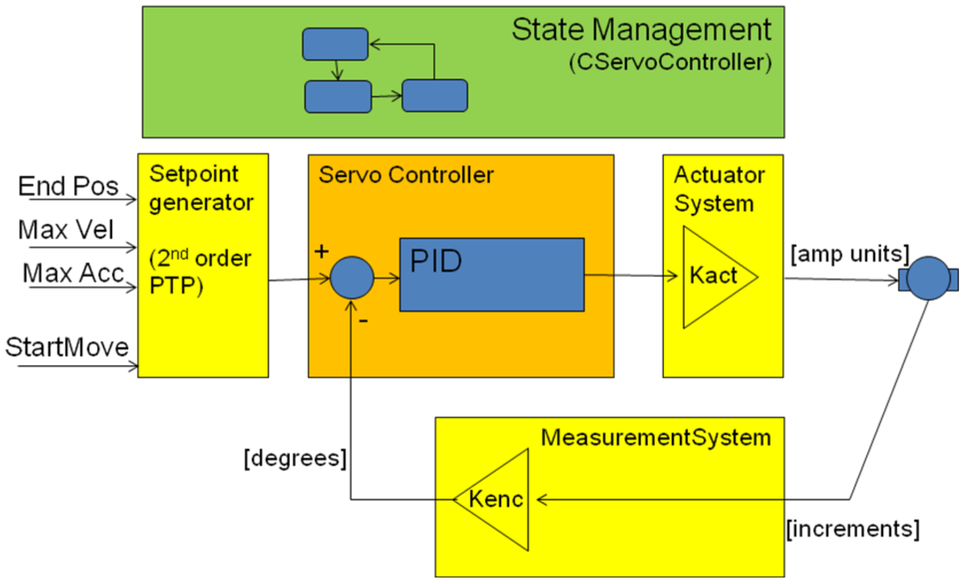


Figure 45: Internals of the Servo for each axis

The figure below shows the details of the PID loop. The constants Kenc, Kact, Kp, Ki, Kd and Ilim are set in the App.cpp file in the rtApp/src directory. These values can be different for each axis. The Kp, Ki, Kd and Ilim values are also read from the ServoParam.ini file. By updating this file and reloading the parameters (only when all servos are openloop) the servo parameters can be updates easily.

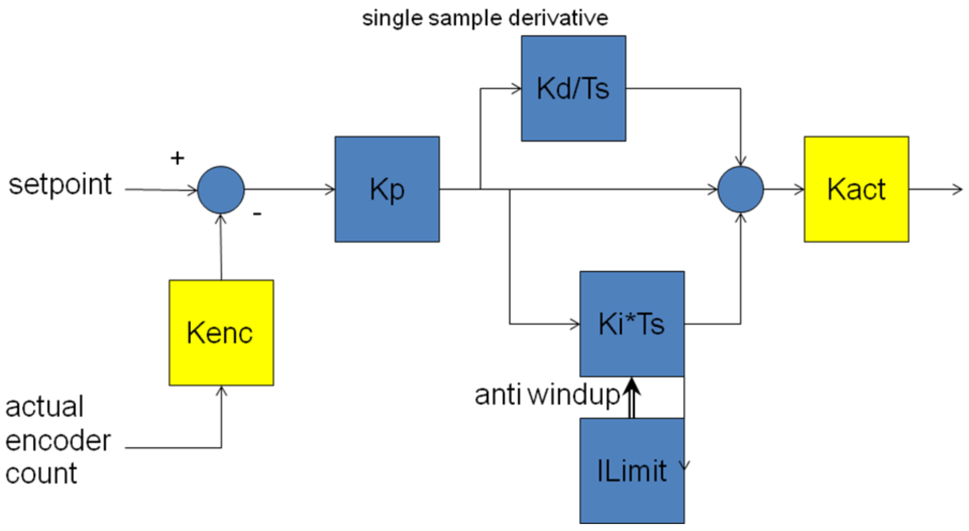
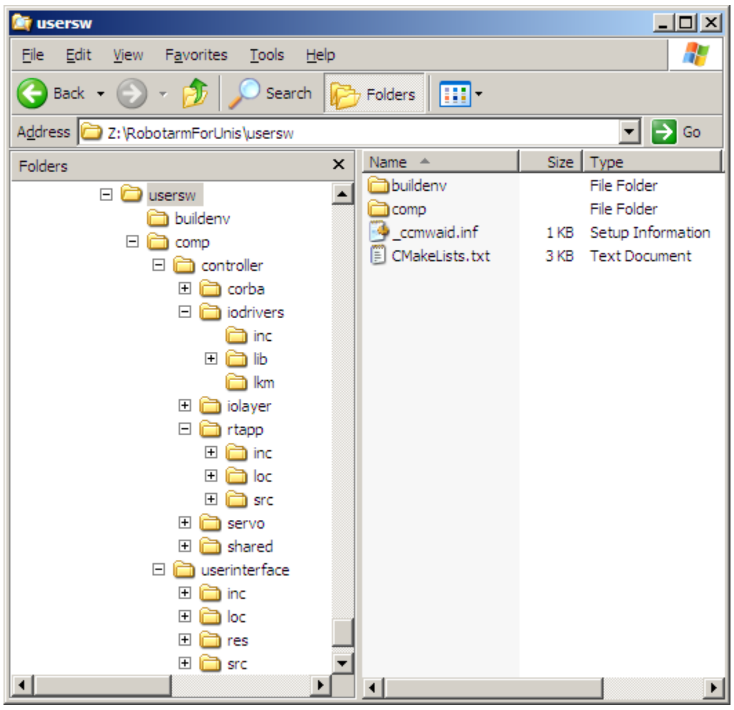


Figure 46: PID controller structure

DIRECTORY STRUCTURE

The figure below shows the relevant directories.


usersw
File Edit View Favorites Tools Help
Back Search Folders
Address Z:\RobotarmForUnis\usersw
Go
Folders

usersw
- buildenv
- comp
  - controller
    - corba
    - iodrivers
      - inc
      - lib
      - lkm
    - iolayer
    - rtapp
      - inc
      - loc
      - src
    - servo
    - shared
  - userinterface
    - inc
    - loc
    - res
    - src

Name	Size	Type
buildenv		File Folder
comp		File Folder
_ccmwaid.inf	1 KB	Setup Information
CMakeLists.txt	3 KB	Text Document

Figure 37: Example Software directory structure

BUILDING THE EXAMPLE SOFTWARE

The software for the robot arm consists of the user software example code and several libraries.

The following steps should be taken to(re)build the software:
- Open a terminal shell
- Run the **cdbuild** command to go to the directory with the build scripts
- Run the **./xeno_generate.sh** command to generate makefiles for xenomai app
- Run the **./xeno_build.sh** command to build the xenomai app
- Run the **./linux_generate.sh** command to generate makefiles for linux app
- Run the **./linux_build.sh** command to build the linux app

Note: the userinterface depends on some files in the controller, e.g. the rtServoIF.idl and some files in the shared directory of the controller are used by both the controller and the userinterface components.

STARTING THE COMPONENTS

CHANGES TO THE SOFTWARE

With a small change in the software it is possible to run the userinterface on a different system than the controller. This is be done by changing the CORBA initialisation string that is on line 119 of the MainWindow.cpp file of the userinterface. In the current string `"-RemoteORBEndpoint 127.0.0.1:7012"` replace the value 127.0.0.1 with the name or ipnumber of the system running the controller.

With some small changes to the cmake scripts it is possible to compile the user interface for windows. Of course this requires windows versions of the libraries that are used, i.e. ACETAO (http://www.dre.vanderbilt.edu/~schmidt/) and FOX (http://www.fox-toolkit.org) libraries.

4.4 RECOMMENDED USE

Some recommended use aspects, which need attention, are presented next.

4.4.1 PC Processor Bandwidth & Sampling Time Choice

The user should be careful when choosing the sampling period. He or she should make sure that the processor has enough bandwidth to run the controllers without over running previous sampling instances.

Since some of the software runs on Linux domain and some on Xenomai domain, processor bandwidth should be checked at 2 levels. Xenomai commands can be used to see how much processing bandwidth the real-time user application is using. Be aware that Ubuntu System Monitor will not show this. However since the USB driver is in the Linux space, it must be made sure that Linux domain has enough processing bandwidth. This can be checked using Ubuntu System Monitor.

For reference, the robot arm example application is configured for a sampling frequency of 500 Hz.

4.4.2 RT-Motion USB Amplifier Tuning

Under no circumstances, the suggested amplifier configuration should be changed. By doing so, the user may risk serious damage and safety.

As shown earlier, the input-output response of the amplifiers are not perfectly linear. Even though it is sufficient for many controllers, if required, the user can implement a look-up table in the user application to compensate for amplifier non-linearity.

4.4.3 Recovering from Firmware Errors

Whenever an error occurs, the user can read back error information such as error state, error count and the last four error codes. Even though the user can clear errors, especially after halt errors, it is advised to reset the firmware in order to avoid firmware instability. It is not guaranteed after error clear, the firmware is still stable.

4.4.4 Maximum Speed

The robot arm should never be used beyond the speed limitation specs defined in this documentation. Especially when the shoulder is moving downwards with the whole weight of the arm, it is physically possible that the shoulder motors can rotate faster than the maximum defined motor speed specification. If maximum motor speed is exceeded, the motor(s) will generate more voltage than the power supply. Even though some safety circuits are included in the power supply, it is not guaranteed that the power supply will clip the over voltage. Be aware that over voltage may destroy RT-Motion USB motion controller boards.

4.4.5 Thermal & Slip Coupling Watchdogs

Since the thermal and slip coupling watchdog tasks are also implemented as a non-real-time tasks, the user shouldn’t fully rely on these tasks for full thermal and slip coupling damage protection. Higher level checks in the user application should also be implemented to form a double layer of protection.

4.4.6 USB Usage

External USB devices connected to the central PC may affect real-time communication behavior. Because of this, the Experimental Philips Robot Arm should be used with the delivered non-USB keyboard and mouse. It is not advised to connect any other USB device (incl. webcam, keyboard and mouse) to the Linux/Xenomai PC.

4.5 ALTERNATIVES

This software present the software alternatives provided by the robot arm. These alternatives are provided ‘As Is’, **USE THEM AT YOUR OWN RISK!**

4.5.1 Windows Driver

A proprietary windows user API is provided in binary format. The user supplied with .h, .lib and .dll files. There is also a WinUSB driver provided for Windows to recognize the RT-Motion USB motion controller(s). So far the Windows driver and the user API is only tested in Windows XP. This is an experimental driver.

4.5.2 Matlab Toolbox

An experimental Matlab toolbox, which calls functions from RT-Motion USB Windows user API, is also provided with the robot arm. RT-Motion USB Matlab toolbox accesses the Windows user API, which is described in the previous section. Before first time use, the user should modify the rtm_usb_init() function to point to right .h, .lib and .dll files.

Using the toolbox, the user can call RT-Motion USB user API functions using the same names and function parameters. Example Matlab user application for reading all relative encoders of the robot arm is also provided. Unfortunately it is not possible achieve high sampling loops when closing the loops in Matlab in non-real-time manner.

There is a known bug with the toolbox. If, after communicating to the RT-Motion USB board, rtm_usb_terminate function is called, the user should unplug and plug the USB cable before calling the rtm_usb_init function again. Otherwise Matlab will crash.

In some other occasions, Matlab crashes unexpectedly. Some cases are not fully explained. Therefore use it at your own risk!

4.5.3 Local PID Controller Template

There is a highly experimental local controller template embedded in the RT-Motion USB firmware. This is a PID position loop template with set-point feed-forward. Set-points and feed-forward gains can be sent via USB using the real-time data communication channel. Two position loops can be run at a sampling frequency close to 4 kHz. The arm is provided with example Matlab .m file, which configures the local PID controller for the wrist and sends set-points over USB.

The firmware doesn’t contain any set point profiling. This should be done on Linux/Windows/Matlab side. The profiling should limit the maximum speed according to robot arm specs. BE ESPECIALLY CAREFULL WITH MAXIMUM SPEED SPECS OF THE SHOULDER WHEN USING LOCAL PID.

Use at your own risk!

5 Licensing

The RT-Motion USB firmware is proprietary software of Koninklijke Philips Electronics N.V. The firmwares of the RT-Motion USB boards are already flashed before the delivery of the robot arm and any future firmware updates will be delivered in binary format. The licensing statement for the firmware is as follows:

> Copyright (c) 2008-2010 Koninklijke Philips Electronics N.V.
> All rights reserved. Reproduction in whole or in part is prohibited
> without the written consent of the copyright owner.
>
> In particular, under no circumstances is this software to be combined with any Open Source Software in any way or placed under an Open Source License of any type without the express written permission of Koninklijke Philips Electronics N.V.
>
> This source code and any compilation or derivative thereof is the proprietary information of Koninklijke Philips Electronics N.V. and is confidential in nature.

The licensing statement of the RT-Motion USB Linux device driver, which can also be found in the source code is as follows:

> Copyright Royal Philips Electronics NV 2008-2010.
> This software is licensed under the GNU General Public License version 2 as published by the Free Software Foundation. See the file COPYING for details.
>
> This software is supplied without any warranties. Use it at your own risk.
> This software is based on "USB Skeleton driver - 2.0" (see copyright below)
>
> USB Skeleton driver - 2.0
>
> Copyright (C) 2001-2004 Greg Kroah-Hartman (greg@kroah.com)
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2.
>
> This driver is based on the 2.6.3 version of drivers/usb/usb-skeleton.c but has been rewritten to be easy to read and use, as no locks are now needed anymore.

RT-Motion USB Linux/Windows user API is proprietary software of Koninklijke Philips Electronics N.V. It is delivered in binary format. The licensing statement for the user API is as follows:

> Copyright (c) 2008-2010 Koninklijke Philips Electronics N.V.
> All rights reserved. Reproduction in whole or in part is prohibited

6 Removal old product

[In Dutch]
Uw product is vervaardigd van kwalitatief hoogwaardige materialen en onderdelen, welke gerecycled en opnieuw gebruikt kunnen worden.

Als u op uw product een doorstreepte afvalcontainer op wieltjes ziet, betekent dit dat het product valt onder de EU-richtlijn 2002/96/EC.

Win inlichtingen in over de manier waarop elektrische en elektronische producten in uw regio gescheiden worden ingezameld.

Neem bij de verwijdering van oude producten de lokale wetgeving in acht en plaats deze producten niet bij het gewone huishoudelijke afval. Als u oude producten correct verwijderd voorkomt u negatieve gevolgen voor het milieu en de volksgezondheid.

Appendix A: Spare part list of buy-in components

	T_{S1}+ T_{S2}	T_{S3}	T_{E1} + T_{E2}	T_{W1}+T_{W2}
Motor	RE30, 268216	RE30, 268214	RE25 118752	A-max 22 110164
Gearing Max cont load	166940 1:66 7.5 Nm	166940 1:66 7.5 Nm	166938 1:33 2.25 Nm	143979 1:29 1.3 Nm
Encoder Counts per Turn	225783 256 CPT	225783 256 CPT	225778 500 CPT	228177 128 CPT

Table 3 Actuator, motor encoder and gearing types

RT-Motion USB Motion Controller Boards
They can be ordered from Philips Greenhouse (part number GH1303) by Philips Applied Technologies personnel, who can do the required modifications.

Functional sensors:
Osram SFH9202 (for force)
AustriaMicrosystems AS5145 magnetic encoder

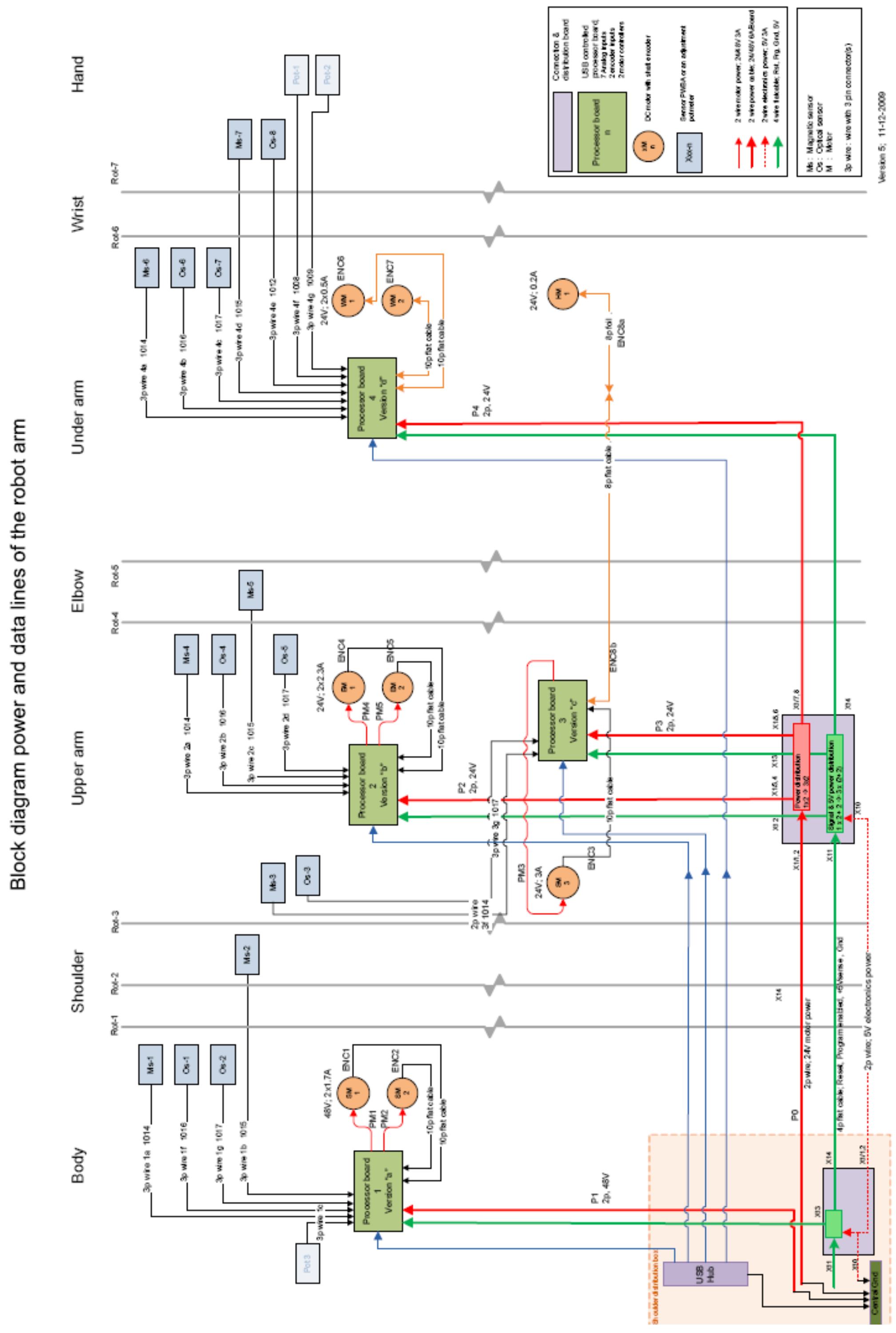
PC

Motherboard:	Intel D945GCLF2
Memory:	Corsair 1GB PC-5300 DDR2 SDRAM DIMM
DVD Drive:	Samsung DVD Writer EIDE/ATA Black
Handdisk:	Western Digital Caviar SE WD3200AAJS 320Gb 7200RPM
Enclosure:	InWin BM639 120W ITX Tower Black/Silver USB Audio
KB + mouse:	Microsoft Basic Black Value Pack EN PS2

Power supply:
Meanwell RS-25-5 (5V, 5A)
Meanwell HRP-300-24 (24V, 14 A)
Meanwell HRP-300-48 (48V, 7A)

USB-hub
Eminent High performance 4 Port USB hub 480/12/1.5 Mbit/s

Appendix B: Robot Arm Wiring Diagram


Block diagram power and data lines of the robot arm
Body
Shoulder
Upper arm
Elbow
Under arm
Wrist
Hand
Rot-1
Rot-2
Rot-3
Rot-4
Rot-5
Rot-6
Rot-7
Processor board 1 Version "a"
Processor board 2 Version "b"
Processor board 3 Version "c"
Processor board 4 Version "d"
Ms-1
Os-1
Os-2
Ms-2
Ms-3
Os-3
Ms-4
Os-4
Ms-5
Os-5
Ms-6
Os-6
Os-7
Ms-7
Os-8
Pot-1
Pot-2
Pot3
ENC1
ENC2
ENC3
ENC4
ENC5
ENC6
ENC7
48V; 2x1.7A
24V; 2x2.3A
24V; 3A
24V; 2x0.5A
24V; 0.2A
P0
P1 2p, 48V
P2 2p, 24V
P3 2p, 24V
P4 2p, 24V
10p flat cable
8p flat cable
USB Hub
Shoulder distribution box
Control unit
2p wire; 24V motor power
4p flat cable; Reset, Programenabled, nBusWake, Gnd
2p wire; 5V electronics power
Power distribution
Signal & 5V power distribution
Connection & distribution board
USB controlled processor board; 7 Analog inputs 2 encoder inputs 2 motor controllers
DC motor with shaft encoder
Sensor PWM or an adjustment potmeter
2 wire motor power; 24/48V/3A
2 wire power cable; 24/48V/6A/board
2 wire electronics power; 5V/3A
4 wire flatcable; Rst, Prg, Gnd, 5V
Ms : Magnetic sensor
Os : Optical sensor
M : Motor
3p wire : wire with 3 pin connector(s)
Version 5; 11-12-2009

Appendix C: Arm compliance

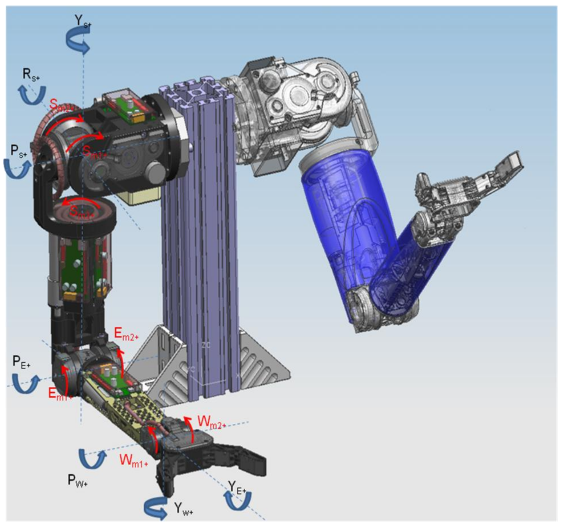


Figure 47 Coordinateframe

The shoulder compliance is characterized by the measurement in Figure 48. The compliance in the Ps degree of freedom can be derived from that measurement (see chapter 3 for dimension of the differential)

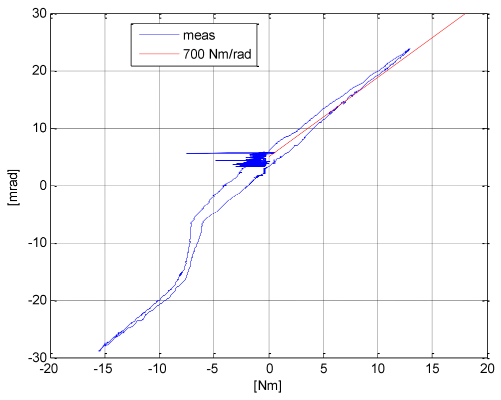


Figure 48 Stiffness measurement in the R_s degree of freedom

Elbow and wrist stiffness is shown in respectively Figure 49 and Figure 50. The shoulder 3rd degree of freedom was not measured at time of writing.

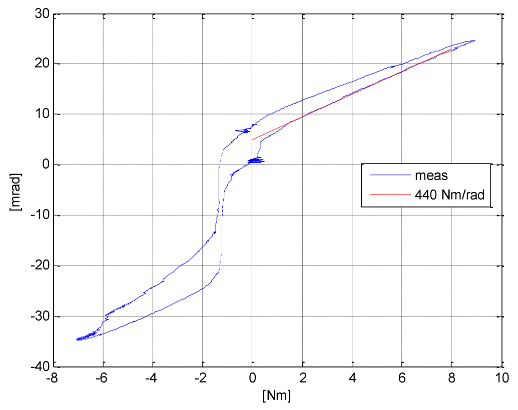


Figure 49 Stiffness measurement in the P_e degree of freedom

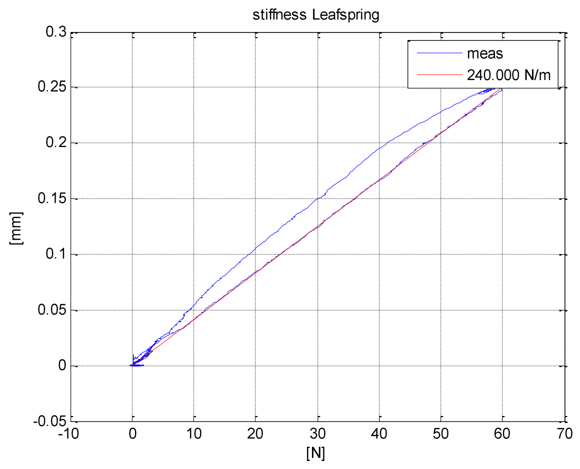


Figure 50 Stiffness measurement of the wrist spring: this relates to 70 Nm/rad for the P_w degree of freedom (see Figure 51 for dimensions)

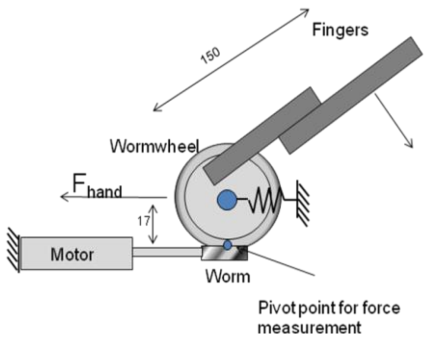


Figure 51 Illustration of wrist compliance