Series E2M3X Hand User Manual
E2P1X

Author:	Hugo Elias
Version:	1.2
Reviewer:	

This page accidentally left blank.

Table of Contents

1 Change list

Ver	Date	Person	Changes
1.2	25 Aug 2015	Hugo, Armando	Added elbow plate holes diagram, adjusted ranges.
1.1	18 Nov 2014	Hugo	Corrected muscle diagram.
1.0	20 Aug 2013	Hugo, Armando	Further small changes and bug corrections.
0.9	06 Dec 2012	Hugo, Armando	Lots of small changes from Ugo, Armando and Hugo.
0.8	14 Nov 2012	Andy Pether	Updated ROS GUI section to reflect recent changes.
0.7	11 Jan 2012	Hugo Elias	Completed tensioning instructions. Added diagram. Added Meaning of SPI in 2.1 Abbreviations. Improved Kinematic diagram. Cropped connection diagram tighter. Added joint ranges table, in degrees and radians. Fixed J0 / J2 problem in motor layout diagram Added Change list.
0.6	1 Dec 2011	Hugo Elias	Created document.

2 Introduction

This manual is for users of the EtherCAT version of the Shadow Dextrous Motor and Muscle Hands.

2.1 Abbreviations

Abbreviation	Meaning
ADC	Analogue to Digital Converter. A chip which reads an analogue voltage signal.
ASIC	Application Specific Integrated Circuit.
CAN	Controller Area Network. A 1Mb peer-peer network for vehicles and robots.
CPU	Central Processing Unit. A computer processor which runs software.
CRC	Cyclic Redundancy Check. An algorithm (or output of) for checking the correctness of incoming data.
DAC	Digital to Analogue Converter. A chip which produces an analogue voltage signal.
EC	EtherCAT
EEPROM	Electrically Erasable Programmable Read Only Memory.
FPID	Feed-forward Proportional Integral Derivative controller. An algorithm used to control something on a robot.
GUI	Graphical User Interface.
I/O	Input / Output
ICD3	In Circuit Debugger 3. A tool used to change the firmware on a PIC microcontroller.
IMU	Intertial Measurement Unit. Sensors used to measure acceleration and rotation.
LED	Light Emitting Diode. A small coloured light.
LVDS	Low Voltage Differential Signal. One of two possible electrical physical layers supported by EtherCAT.
mA	milliamps. A unit of electrical current.
MCU	Micro Controller Unit. A small, usually embedded, CPU.
MIPS	No longer an abbreviation; a word in its own right. A brand of CPU / MCU.
Nm	Newton Meters. A unit of torque.
PC	Personal Computer.
PCB	Printed Circuit Board.
PID	Proportional Integral Derivative controller. An algorithm used to control something on a robot.
PST	Pressure Sensor Tactile. A simple tactile sensor offered as standard.
PWM	Pulse Width Modulation. Digital method used to emulate an analog signal.
ROS	Robot Operating System, created by Willow Garage.
SPI	Serial Peripheral Interface, allowing an MCU to communicate with an ADC.

3 Setting up the hand

3.1 What's in the box?

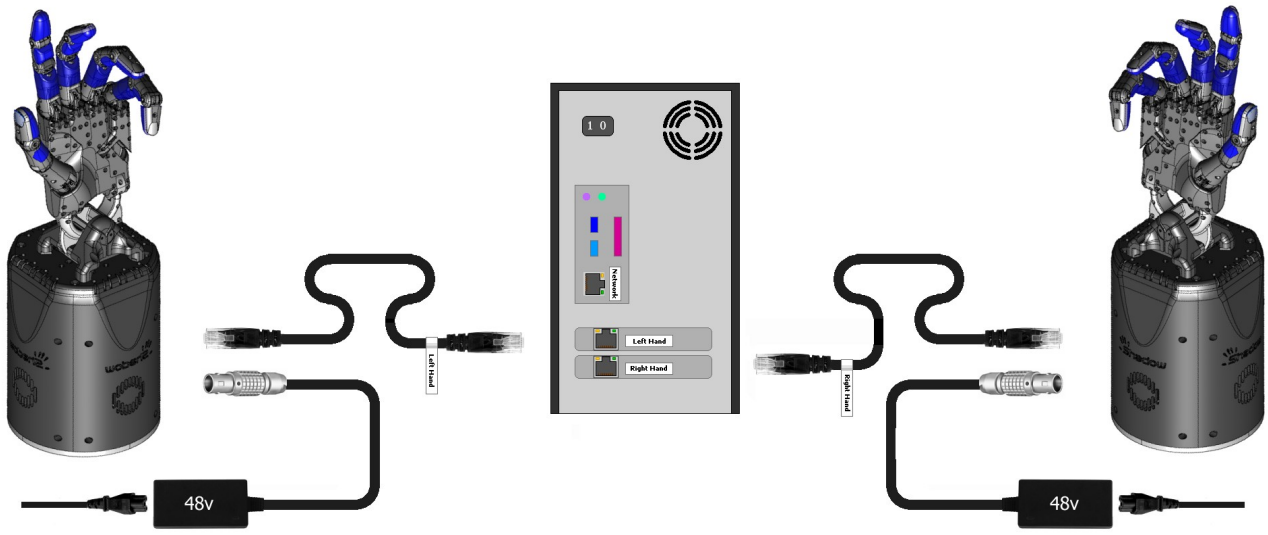
Item	Description
Shadow Hand E2M3 or E2PT	Hand Unit with motor or muscle actuators
PC	Host PC control unit for the hand
PSU for Hand	48v for motor hand, or 28v for muscle hand
Kettle Leads	To connect power supplies to mains
Power Cable	4-pin Large Lemo connector, already fitted to the hand
Air Pressure Regulator	Air regulator for Muscle Hand only
EtherCAT Extension Cable	50cm EtherCAT extension lead, already fitted to the Hand
Calibration Jigs	Bag containing calibration jigs for all joints
Toolbox	Contains hex drivers to perform required maintenance
User Manual	This document

3.2 Connecting Cables

There are two ways to connect the EtherCAT and power cables to the hand.

3.2.1 External Connections

If your hand already has cables fitted, then you can simply connect the EtherCAT and power connectors immediately.



EtherCAT: Connect the Ethernet cable to the hand's Ethernet socket, and connect the other end to the PC's second Ethernet port. **If you have a Bi-manual system, connect the Left and Right hands correctly to the labelled ports.** You have been supplied with a medium length Ethernet lead, but if you require a longer or shorter one, you can simply use a standard

commercial Ethernet Cat 5 cable, available from most computer parts suppliers.

Power: Connect the external power supply to the hand using the metal Lemo connector, making sure to line up the red dots. If you require a longer or shorter cable, please contact the Shadow Robot Company.

Air: If you have a muscle hand, you will also need to connect the 3.5 bar air supply to the 6mm push-fit connector.

3.2.2 Internal connections

If you are connecting the hand to a robot with internal cabling, then you may wish to use the internal connectors.

Turn the hand over, and use the orange and green hex drivers to remove the connector cover. Connect the two cables to their relevant sockets. Now affix the hand to your robot arm.

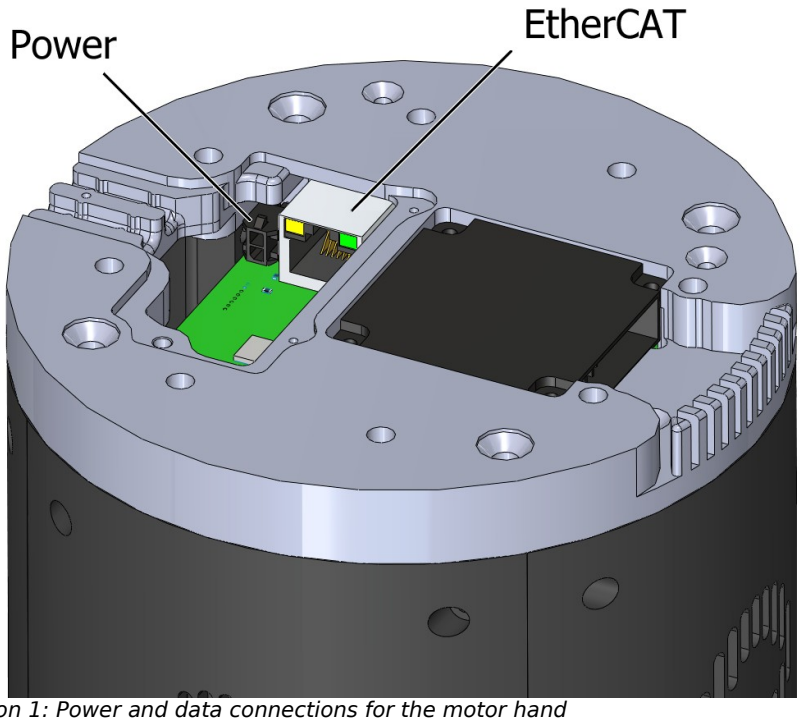
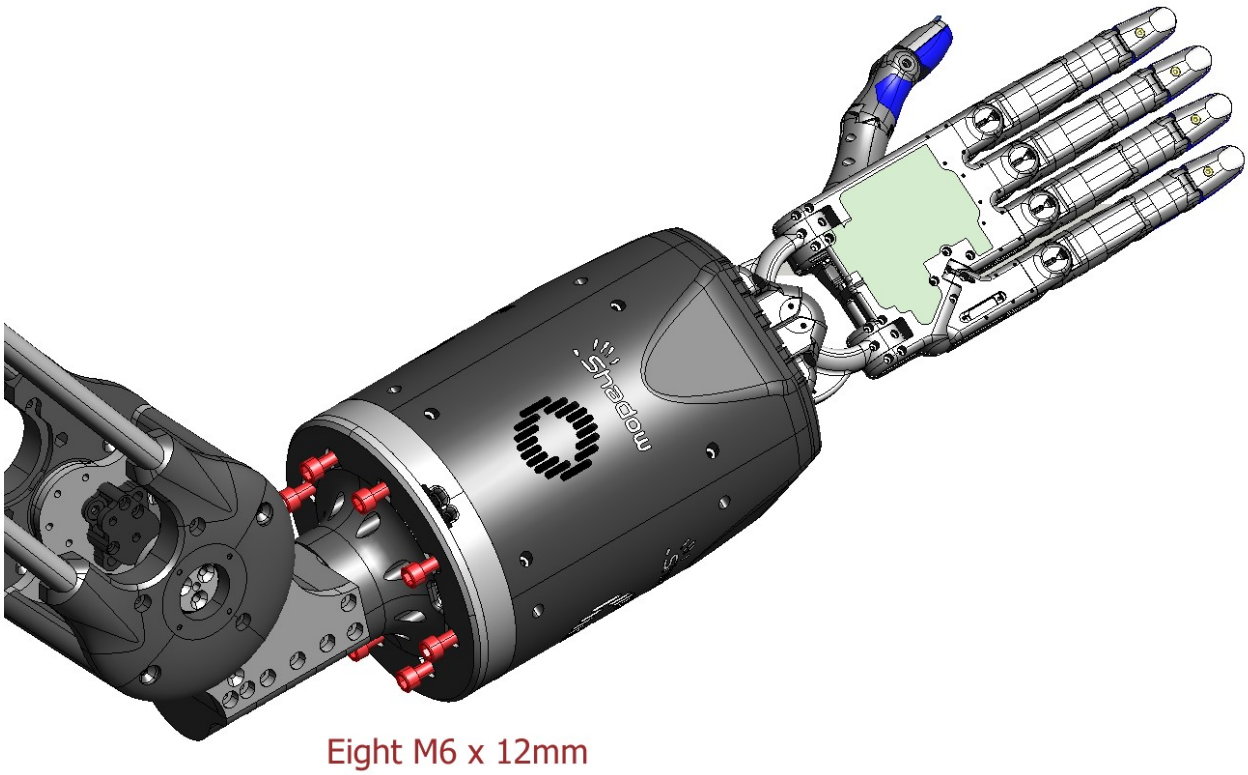


Illustration 1: Power and data connections for the motor hand

3.3 Mounting the hand

Your hand comes fitted with the appropriate fitting for your robot arm, and the mounting procedure will depend on this fitting.

3.3.1 Mounting to a Shadow Muscle Arm



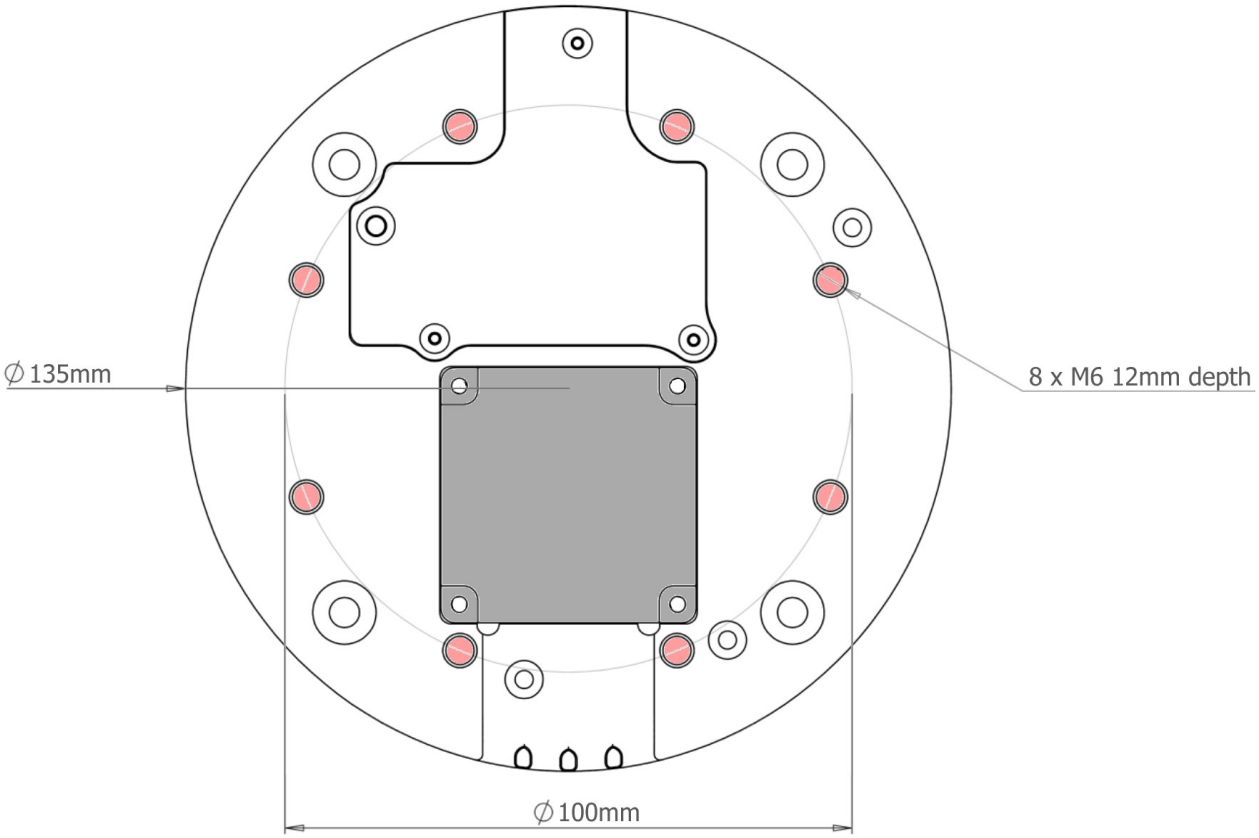
Connect the cables: An Ethernet cable and a power cable exit the arm at the elbow plate. Connect these to the inside of the hand's elbow plate first.

Attach the hand: Rotate the arm's elbow plate until the two muscles are equal length. Hold the hand up against the elbow plate with the palm facing 45º down and to the left, thumb upwards. Screw in the eight M6 bolts loosely by hand. Gently tighten each bolt finger tight. Then go around and tighten each bold **hand tight. Do not over-tighten the bolts.**

3.3.2 Mounting to other robot arms

Shadow Robot can supply an elbow adaptor plate to adapt the Hand to most other robot arms.

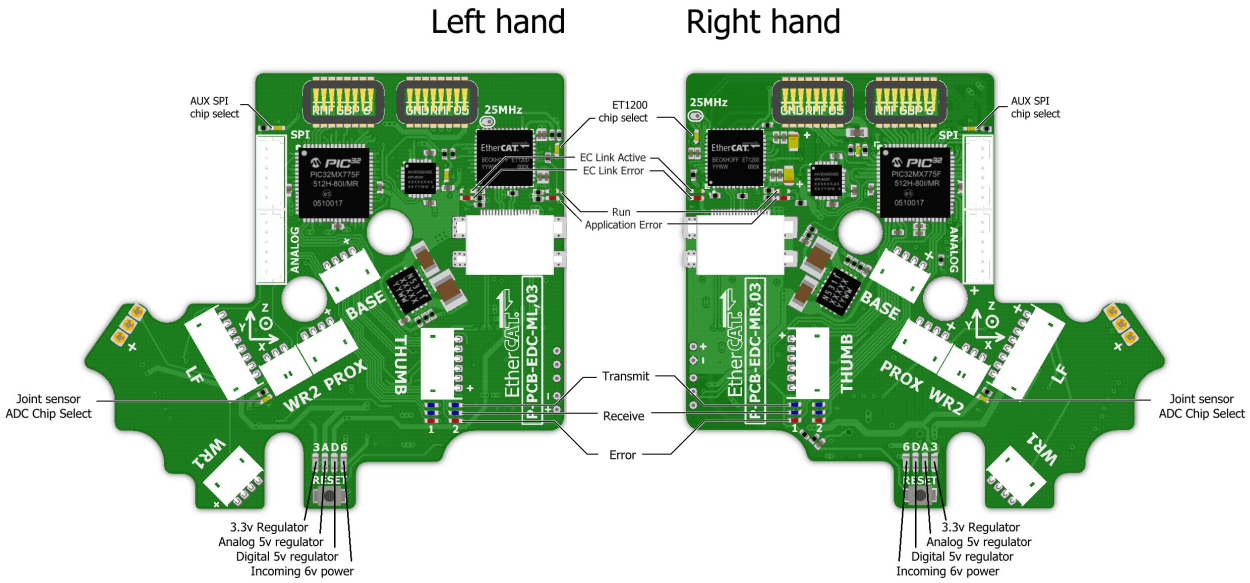
However, if you wish to make your own fitting for the Hand:



The Hand's elbow plate contains eight screw holes which accept M6 bolts to a depth of 12mm. The holes are spaced equally from the centre on a circle wth diameter 100mm. The overall diameter of the elbow plate it 135mm

3.4 Lights

Light	Colour	Activity	Meaning
Power LEDs x4	White	Off	Power bad
		On	Power good
EC Link Active	Green	Off	No link
		Flicker	EtherCAT packets flowing
		On	EtherCAT link established
EC Link Error	Red	Off	No EtherCAT link error
		On	EtherCAT link error detected
Run	Green	Off	Hand is in Init state
		Blinking	Hand is in Pre-Operational state
		Single Flash	Hand is in Safe Operational state
		On	Hand is in Operational state
Application Layer Error	Red	Off (while running)	No EtherCAT packet error
		On (while running)	EtherCAT packet error
		On (during boot)	Verifying ET1200 EEPROM
ET1200 chip select	Yellow	Off	Not communicating
		On	PIC32 communicating with ET1200
Joint sensor chip select	Yellow	Off	Joint sensors not being sampled
		On (dim)	Joint sensors being sampled
Aux SPI chip select	Yellow	Off	Aux SPI not in use
		On (dim)	Aux SPI in use
CAN 1 Transmit	Blue	Off	Palm not transmitting messages on CAN 1
		v.fast flicker	Palm transmitting messages on CAN 1
CAN 1 Receive	Blue	Off	Palm not receiving messages on CAN 1
		v.fast flicker	Palm receiving messages on CAN 1
CAN 1 Error	Red	Off	No errors detected on CAN 1
		Flickering	Errors detected on CAN 1
CAN 2 Transmit	Blue	Off	Palm not transmitting messages on CAN 2
		v.fast flicker	Palm transmitting messages on CAN 2
CAN 2 Receive	Blue	Off	Palm not receiving messages on CAN 2
		v.fast flicker	Palm receiving messages on CAN 2
CAN 2 Error	Red	Off	No errors detected on CAN 1
		Flickering	Errors detected on CAN 1



3.5 Powering up

You can power up the hand and PC in any order. You do not have to power up one before the other. When power is applied to the hand, the fans will be heard immediately.

3.5.1 Lights

On power up, the lights will be in the following state

Light	Colour	Activity	Meaning
Power LEDs	White	On	Power good
EC Link Active	Green	On	EtherCAT link established
EC Link Error	Red	Off	No EtherCAT link error
Run	Green	Off	Hand is in Init state
Application Layer Error	Red	On (during boot)	Verifying ET1200 EEPROM
		Then off	No EtherCAT packet error
ET1200 chip select	Yellow	On	PIC32 communicating with ET1200

Lights will also appear inside the motor or muscle base, indicating 5v, 6v and 24v (or 28v)supplies. In the motor hand, these can only be seen by removing the covers.

3.5.2 Jiggling

This applies to the motor hand only. On reset, all of the strain gauges (torque sensors) in the motors need to be zeroed. This happens automatically. The motors are driven back and forth to try to relieve any tension on the tendons. Then both gauges are zeroed. You will therefore see all joints of the hand move slightly on power up or reset or power up.

4 ROS

4.1 ROS Wiki

If you are unfamiliar with ROS, it is highly recommended that you read the ROS Tutorials:
http://www.ros.org/wiki/ROS/Tutorials

4.2 Shadow Wiki

The main documentation for developing Shadow Hand applications under ROS is online:

http://www.ros.org/wiki/shadow_robot

4.3 Starting the driver

4.3.1 ROS core

First start the ROS core using the desktop icon, or at a terminal, type:

```
roscore
```

4.3.2 Shadow Hand Driver

Next, launch the driver for the Shadow Hand using the desktop icon 'Hand Driver' or at a terminal, type:

```
sudo -s
roslaunch sr_edc_launch sr_edc.launch
```

Warning: This terminal now has root privileges, and the system is giving you permission to do things which can mess up the configuration of the PC. Do not run any other commands in this terminal window. Only use this terminal to launch the driver.

4.3.3 Lights on the hand

When the ROS driver is running you should see the following lights on the Palm:

Light	Colour	Activity	Meaning
Run	Green	On	Hand is in Operational state
CAN1/2 Transmit	Blue	V.fast flicker	Demand values are being sent to the motors
CAN1/2 Receive	Blue	V.fast flicker	Motors are sending sensor data
Joint sensor chip select	Yellow	On	Sensors being sampled

After killing the driver, the lights will be in a new state

Light	Colour	Activity	Meaning
Run	Green	Blinking	Hand is in Pre-Operational state
CAN 1/2 Transmit	Blue	Off	No messages transmitted on CAN 1/2
CAN 1/2 Receive	Blue	Off	No messages received on CAN 1/2
Joint sensor chip select	Yellow	Off	Sensors not being sampled

4.4 Robot Monitor

Now we can check that everything on the robot is working correctly using the robot monitor. Use the desktop icon, or at a terminal, type:

rosrun robot_monitor robot_monitor

This brings up a dialog box containing a tree of all parts of the robot. All parts should be marked with a green tick.

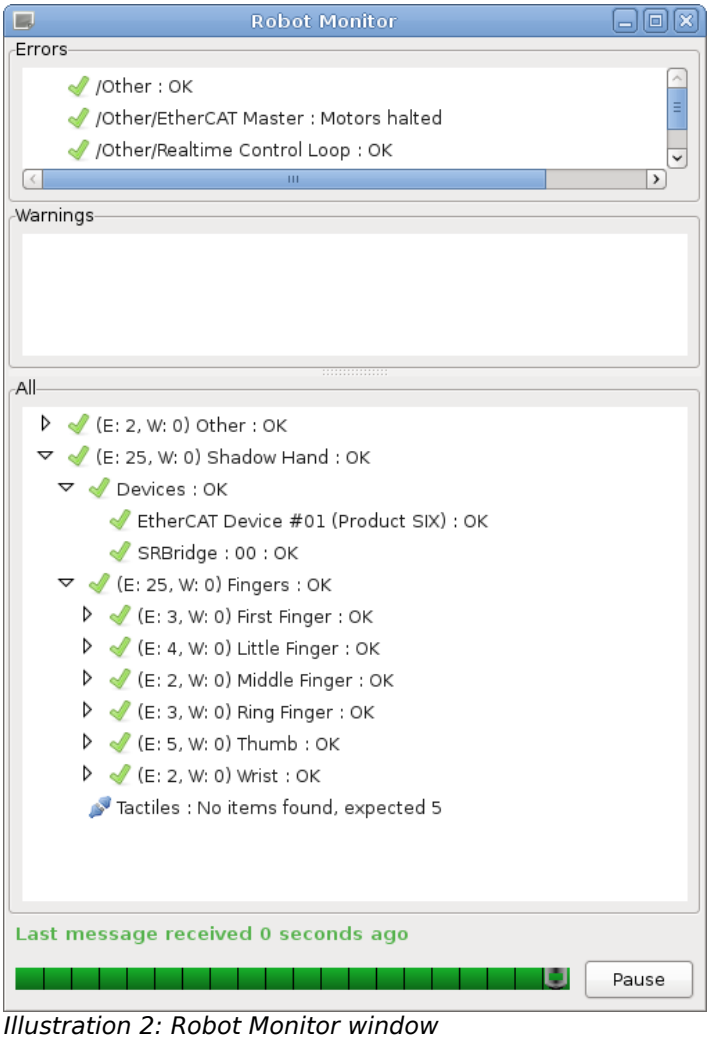


Illustration 2: Robot Monitor window

You can examine one motor in detail by double-clicking on it. This brings up the Motor Monitor dialog. This window can be used to check the status of a motor, or debug any problems.

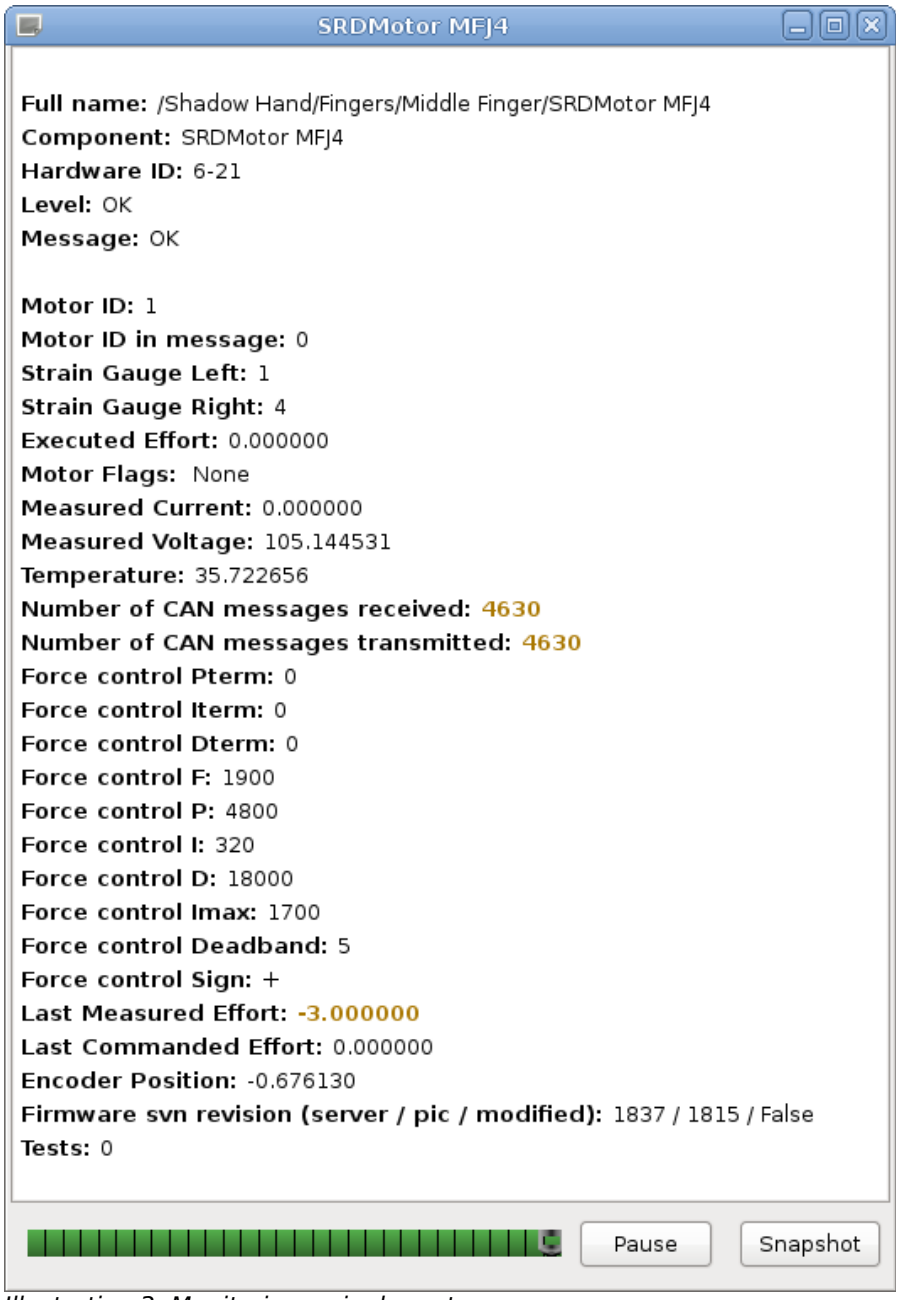


Illustration 3: Monitoring a single motor

Item	Description
Full Name	
Component	
Hardware ID	
Level	

Message	Any error or status messages
Motor ID	This is the motor number. Range [0..19]
Motor ID in message	For debugging only
Strain Gauge Left / Right	These are the ADC readings from the two gauges
Executed Effort	
Motor Flags	See motor flags table below
Measured current	Current flowing through the motor (Amps)
Measured Voltage	The motor power supply voltage. Not the voltage at the motor
Temperature	The temperature measured near the motor. The actual motor winding temperature will be higher than this. (ºC)
Number of CAN messages	Received messages should be twice the transmitted messages
Force control P, I, D terms	These are the PID terms from inside the motor's torque controller. They may be useful for debugging if plotted.
Force control F, P, I, D, Imax, Deadband, Sign	These are the FPID gain settings used by the motor's torque controller. They can be changed using the controller tuner.
Last Measured Effort	Difference between the two gauge readings (Torque)
Last Commanded Effort	Torque requested by the host-side control algorithms
Encoder Position	The angle of the joint in radians (ROS always calls this Encoder position, even if the robot uses Hall effect sensors)
Firmware svn revision	**1837:** The latest version of the firmware available at build time **1815:** The version of the firmware in the motor MCU **False:** There are no un-checked-in modifications to this firmware. This should never be true.

4.5 Graphical User interface

The the majority of functionality is provided by the software Application Programmer Interface (API). However, a few simple functions are provided in the Graphical User Interface (GUI) to test the hand, validate that it is working correctly, and adjust some of its settings.

4.5.1 Starting the interface

Now you may open the Graphical User Interface to try out some functions of the hand. Use the desktop icon, or at a terminal, type:

rosrun rqt_gui rqt_gui

The interface contains a number of plugins for interacting with the EtherCAT hand and arm. These are all available from the **Plugins → Shadow Robot** menu.

4.5.2 Controller tuner

It is possible to adjust the settings for any of the Position or Force controllers.

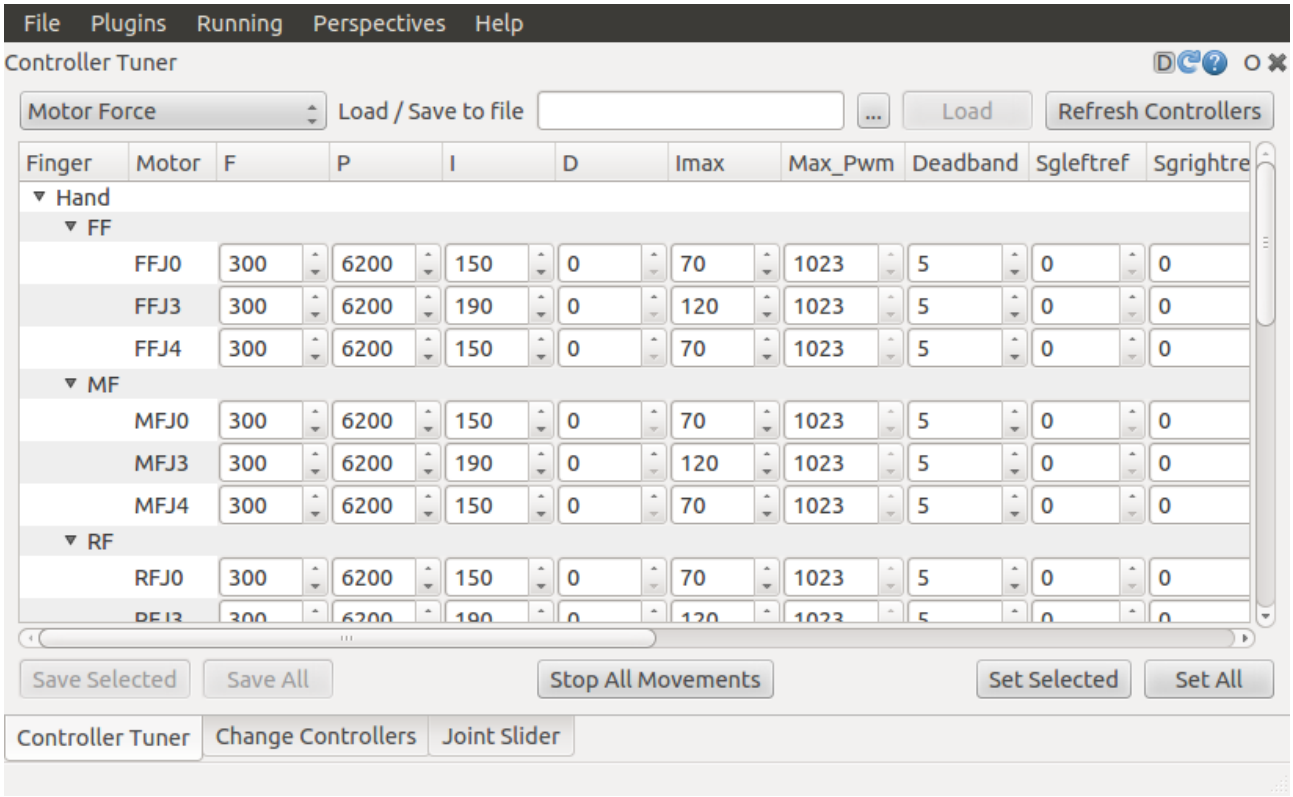


Illustration 4: Adjusting the torque controller settings.

Here you can select a finger or thumb, and adjust the F, P, I and D values for each joint. See 8.1 Control for details of these settings.

Click Set all of Set selected to send the new values to the motors and make them take effect.

The following advanced values are less likely to need changing:

max_pwm: This puts a limit on the PWM value that will be sent to the motor by the torque controller. It can be useful when setting up a controller for the first time to limit the motor power to a safe level.

deadband: The error is considered to be zero if it is within ±deadband. This value should be set as a little more than the noise on the sensor. The units of deadband are the same as the value being controlled. So, the deadband for a torgue controller is in the units of the strain gauges. For a position controller, the units are radians.

sign: This defines the relationship between the sign of the output of the controller, and the direction of the motor. Range: integer 0=positive, 1=negative.

Click **Save** to save your settings.

4.5.3 Bootloader

The firmware in the motors MCUs can be updated from the PC, without opening up the motor base. This can be done from the GUI. The muscle hand does not support bootloading. Shadow will send you a new HEX if there is an update.

Plugins → Shadow Robot → EtherCAT → Motor Bootloader

You will see a window listing each motor board, along with its current firmware SVN revision number.

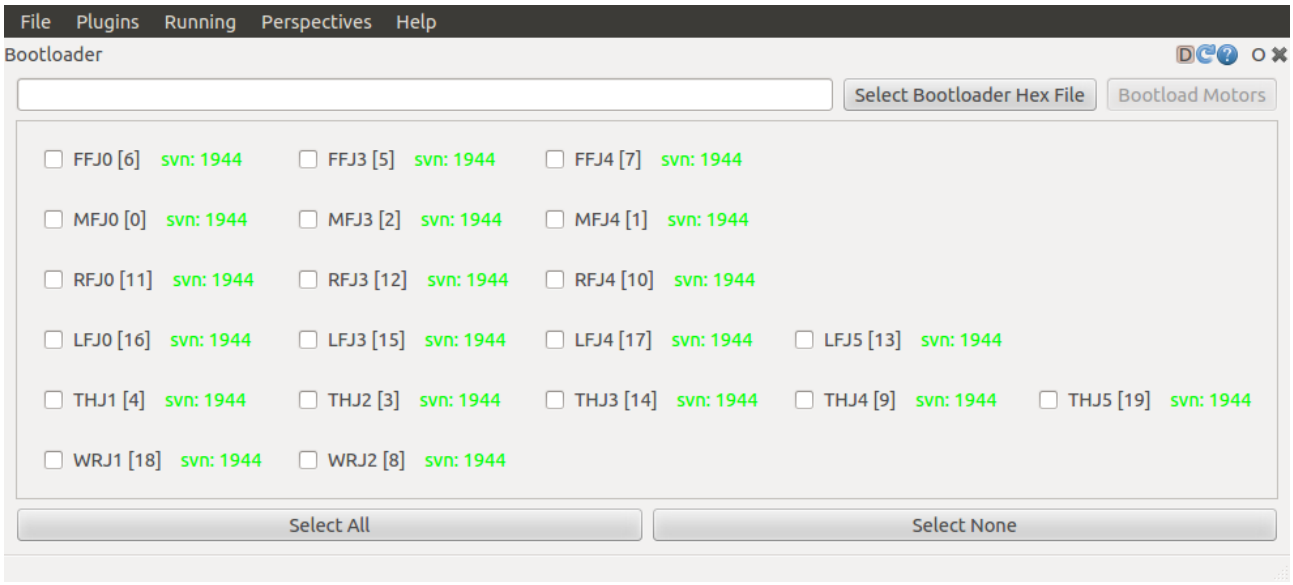


Illustration 5: Bootloading new firmware in to the motor microcontrollers.

Select Bootloader Hex File: Next, tell the plugin which firmware to use. You can find it in:

sr_external_dependencies/compiled_firmware/released_firmware

Select your motors: Now you may choose which motors to program. Either select one or more motors using the tick boxes, or click the Select/Deselect All button.

Program Motors: Now you can click the Bootload Motors button. The process is fairly slow, and takes about a minute per motor.

4.5.4 Change controllers

Use the *Change Controllers* plugin to load different types of controllers. Simply click on a controller type, and it will call a service from the pr2_controller_manager to unload the currently running controller if necessary, and load the one you've selected. See the chapter on control for details of these algorithms. See 8.1 Control for information about the different types of control.

NOTE: CURRENTLY THE ONLY FULLY SUPPORTED TYPES ARE POSITION PWM CONTROL, AND EFFORT TORQUE CONTROL. SELECTING OTHER TYPES MAY CAUSE UNPREDICTABLE RESULTS AND DAMAGE THE HARDWARE.

Illustration 6: Selecting different control strategies.

4.5.5 Motor Resetter

If for some reason you need to reset the firmware on a motor, you can either press the reset button on the PCB itself (which requires removal of the base covers), or use this plugin.

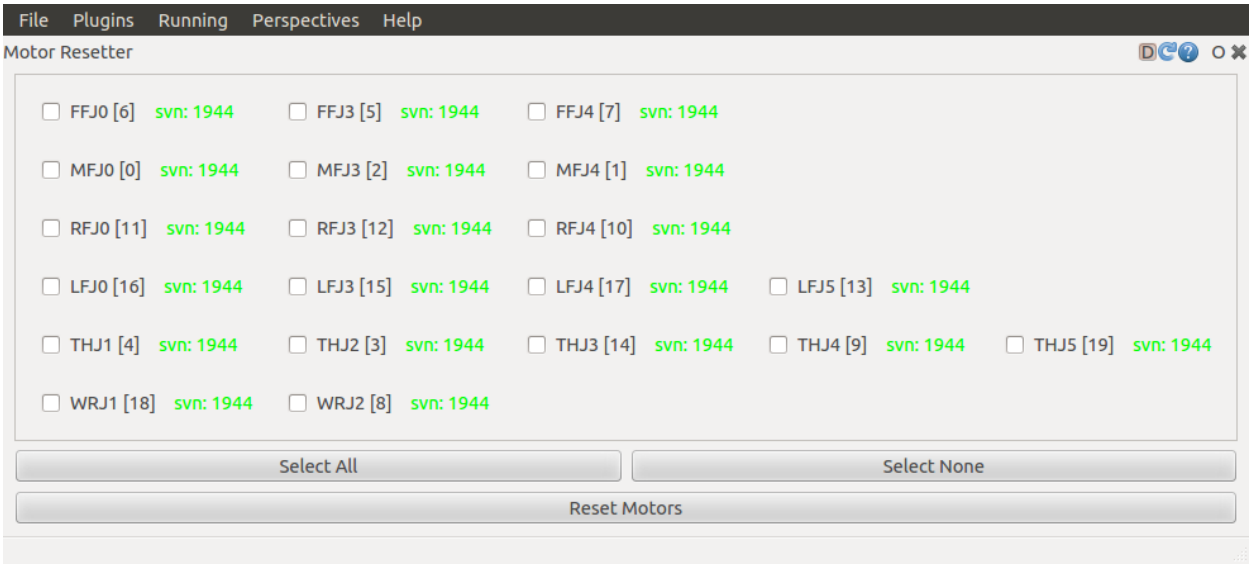


Illustration 7: Resetting the motor microcontrollers.

Tick the motors you wish to reset, and click **Reset Motors**. You should see the corresponding joints jiggle as the motors auto-zero the strain gauges.

4.5.6 EtherCAT Sliders

A simple interface has been provided to control the position of each joint using a slider (you have to start the position control first).

Plugins → Shadow Robot → Joint Sliders

A window with twenty sliders will appear. Moving any slider will cause the corresponding joint on the hand to move.

4.5.7 Hand Calibration

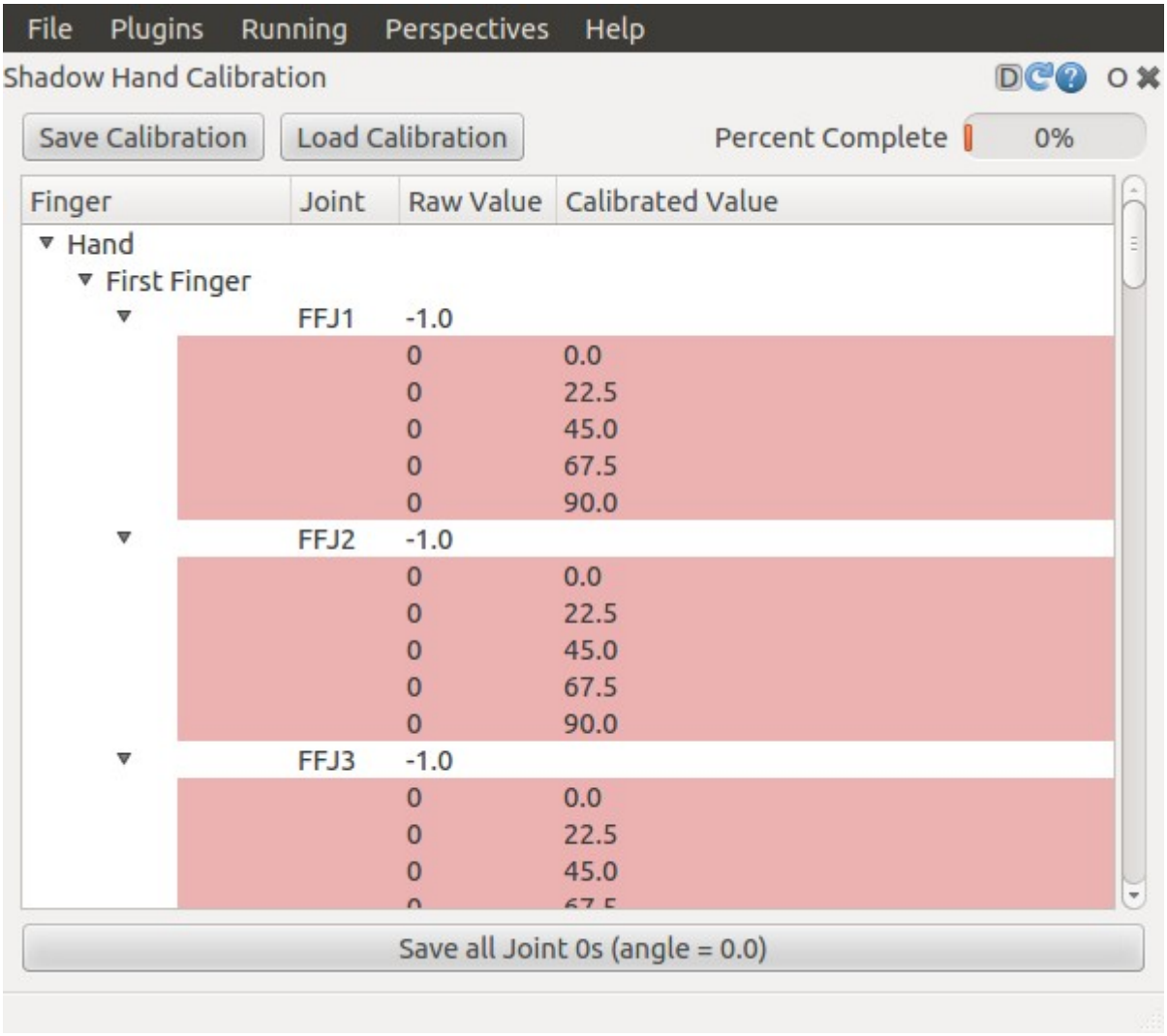


Illustration 8: Calibrating the joint sensors.

After some time, the joints on the hand may need to be re-calibrated. It is possible to re-calibrate one or more joints using this plugin. You do not need to calibrate all of the joints in the same session. Refer to the Calibration section in the Maintaining the Hand chapter for details.

4.6 Command line interface

All functions of the hand are available from the command line.

4.6.1 Using rostopic to view sensors

You can stream the value of a sensor or sensors to a terminal window using the ROS command rostopic:

rostopic echo /joint_states

You will see a lot of text scrolling up the screen.

```
Header:
   seq: 14597
   stamp:
         secs: 1244577
         nsecs: 4326734
   frame_id:
name: ['WRJ2', 'WRJ1', …. 'THJ2', 'THJ1']
position: [ … ]
```

The two items of interest are the name[] and position[] arrays. You can view these arrays individually:

roptopic echo /joint_states/name

writes just the joint names to the terminal window.

rostopic echo /joint_states/position

writes just the joint angles (in radians) to the terminal window.

4.6.2 Using rostopic to view debugging data

Type:

roptopic echo /debug_etherCAT_data

These are the debugging data available. Of particular interest is the /debug_etherCAT_data/sensors[] array. This contains the raw ADC sensor values from the hand. It is also useful for viewing the Aux analog sensor inputs (channels 32..35).

Number	Raw sensor value	Number	Raw sensor value	Number	Raw Sensor value
0	FFJ1	12	LFJ1	24	WR1B
1	FFJ2	13	LFJ2	25	WRJ2
2	FFJ3	14	LFJ3	26	ACCX
3	FFJ4	15	LFJ4	27	ACCY
4	MFJ1	16	LFJ5	28	ACCZ
5	MFJ2	17	THJ1	29	GYRX
6	MFJ3	18	THJ2	30	GYRY
7	MFJ4	19	THJ3	31	GYRZ
8	RFJ1	20	THJ4	32	AUX0
9	RFJ2	21	THJ5A	33	AUX1
10	RFJ3	22	THJ5B	34	AUX2
11	RFJ4	23	WRJ1A	35	AUX3

4.6.3 Reset motors

To reset individual motors, E.G. FFJ3:

rosservice call /realtime_loop/reset_motor_FFJ3

4.7 Moving joints

Ensure the hand is started and the position controllers are loaded. Then open a terminal and run the following commands:

This moves a single joint:

rosrun sr_edc_muscle_tools example_move_joint.py

This shows how to read position and pressure from a muscle:

rosrun sr_edc_muscle_tools example_reading_sensors.py

This moves a number of joints:

rosrun sr_edc_muscle_tools example_sendupdate.py

To see the source code for these Python scripts:

roscd sr_edc_muscle_tools
ls

4.8 Writing controllers

Rather than use the ROS topics to access sensor data, you will need to write a plugin for the PR2 Controller Manager. This will give you access to the sensor data at the full 1kHz rate, and allow you to create your own control algorithms for the hand. Please see this page for more information about the PR2 Controller Manager:

http://ros.org/wiki/pr2_controller_manager

The Controller Manager is the node that talks to the hardware via EtherCAT and provides a facility for hosting plugins. The valve and position controllers you have already used are examples of this. Note that the Controller Manager can host any number of running controllers but one should be loaded at a time for a given joint so they don't fight for control.

4.9 Deeper settings

4.9.1 Editing PID settings

The motor controller PID settings are stored in a YAML file. To edit this file directly:

roscd sr_ethercat_hand_config/controls/

gedit host_side/sr_edc_position_joint_controllers.yaml

or

gedit motors/motor_board_effort_controllers.yaml

4.9.2 Changing motor data update rates

Each motor can return two sensor readings every 2ms. The first is always the measured torque. The second is requested by the host. This allows the host to decide on the sensor update rate of each sensor. Currently, the rates cannot be adjusted at run-time, and are specified in a file which you can edit. To edit the file:

roscd sr_robot_lib/config

gedit motor_data_polling.yaml

The complete list of motor sensors appears in the file, along with a number

Number	Meaning
-2	Read once when the driver is launched
-1	Read as fast as possible
0	**Do not use zero**
> 0	Read period in seconds

Sensors set to -1 will be read in turn, unless it's time to read another sensor. Usually 5 sensors are set to -1, meaning that they are sampled at 100Hz.

5 Mechanical Description of the hand

5.1 Dimensions

The Hand has been designed to be similar to a typical male hand, however the fingers are all the same length, although the knuckles are staggered to give comparable fingertip locations to the human hand.

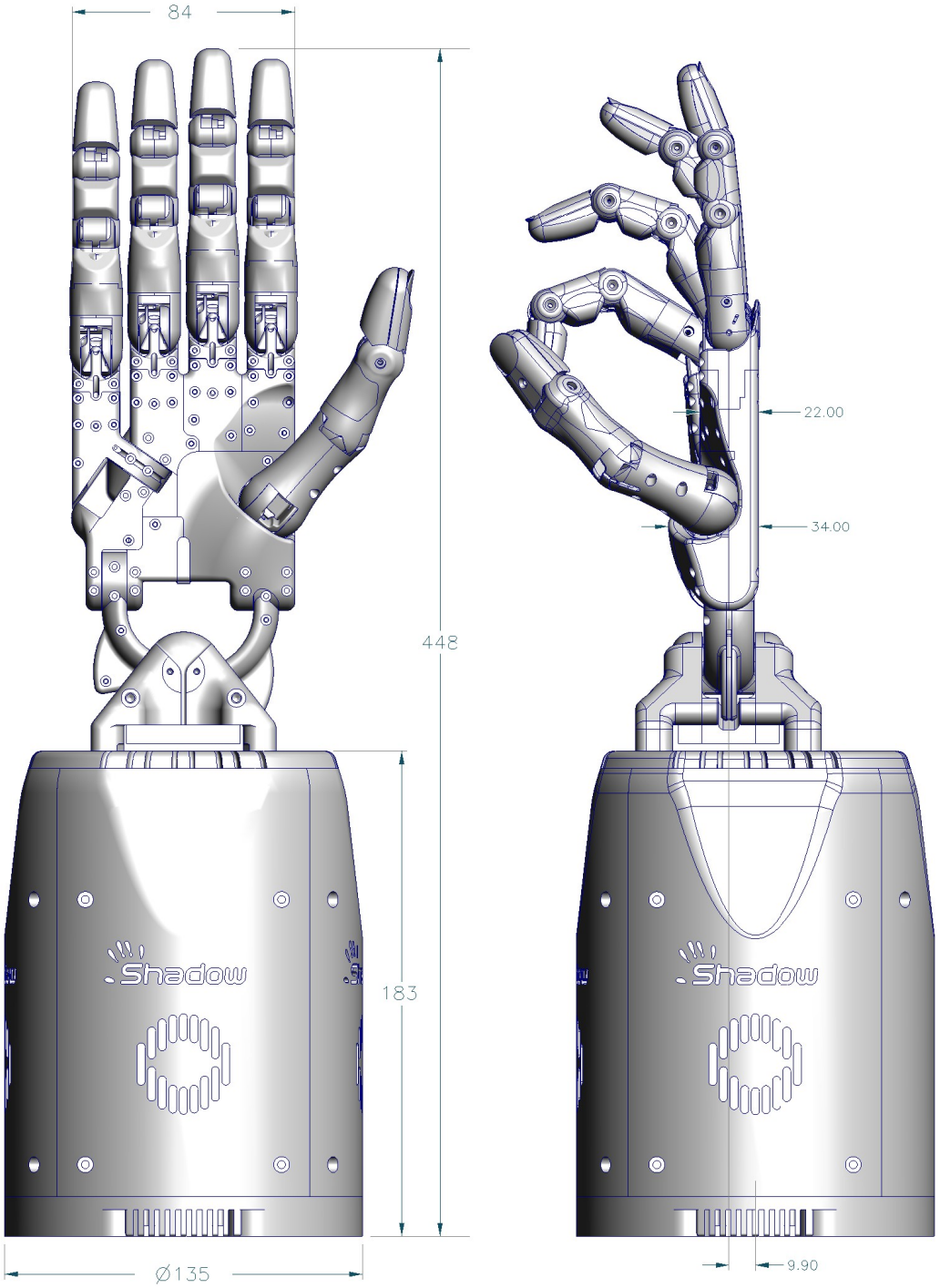
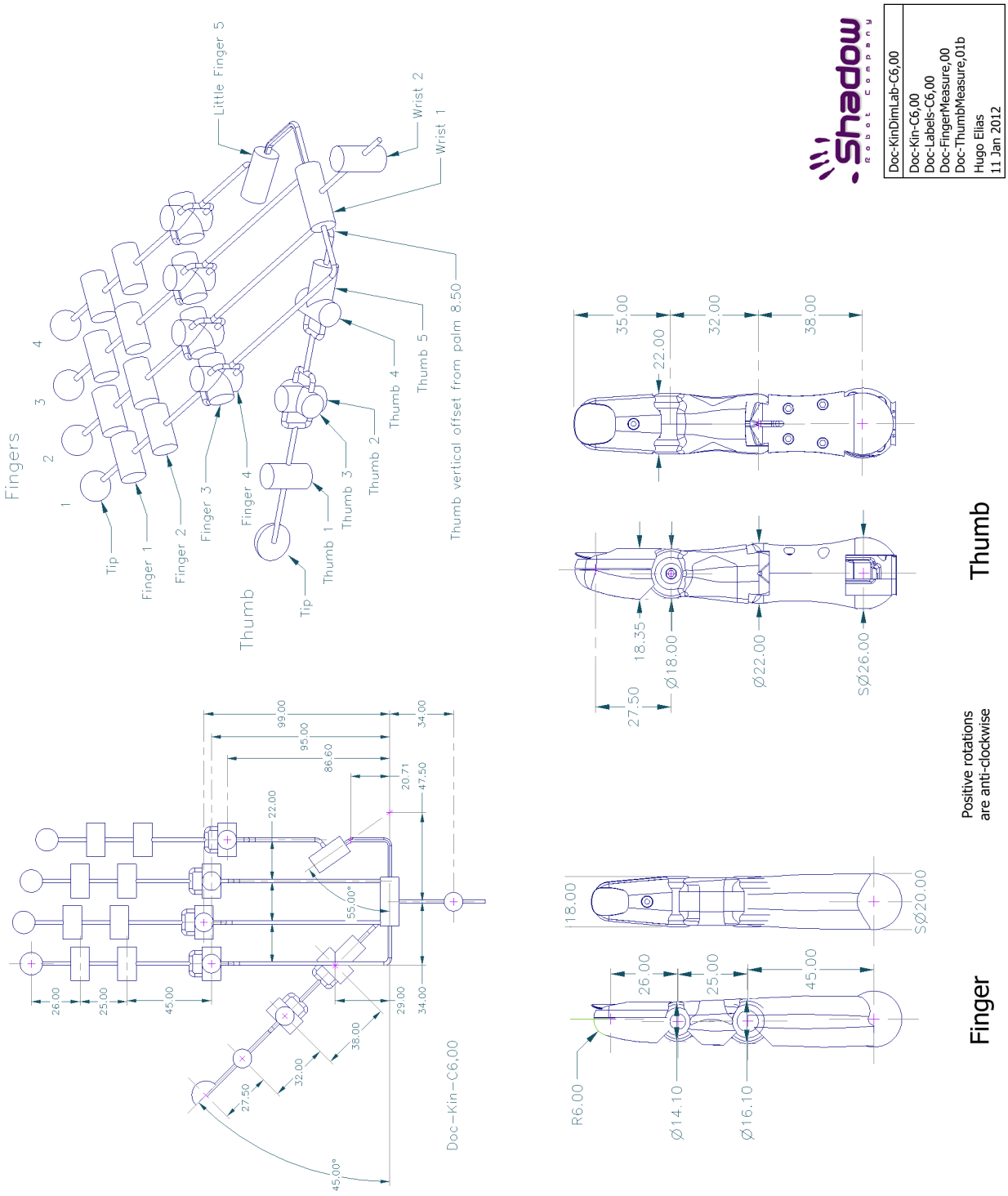


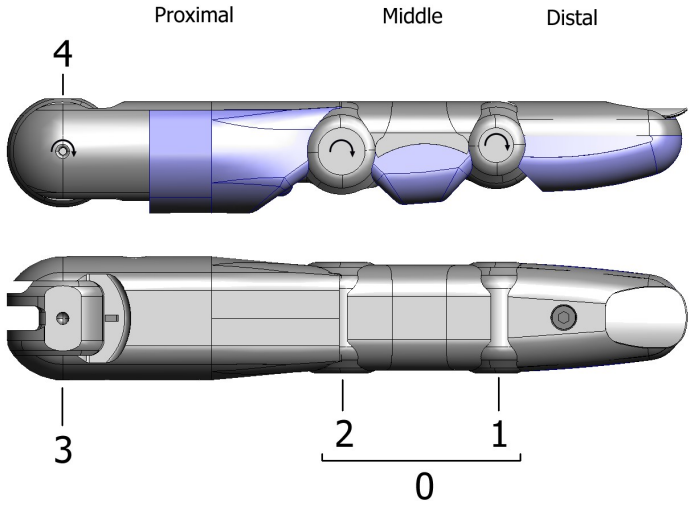
Illustration 9: Dimensions of the motor hand unit.

5.2 Kinematics



5.3 Finger

The four fingers are named according to the UK convention: First, Middle, Ring, Little.



5.3.1 Naming and angle conventions

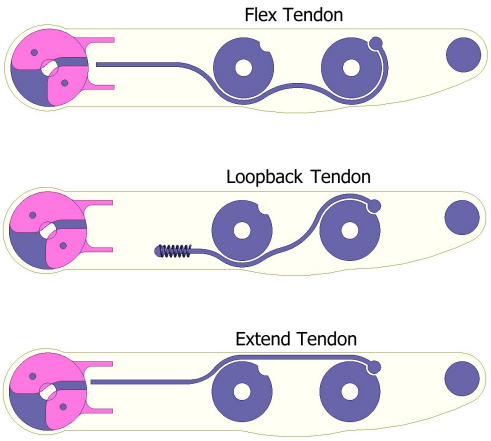
The four finger joints are the distal (finger tip), middle, proximal (nearest the palm), and the adduction/abduction joint (sideways movement) which is coplanar with the proximal joint. Joints are numbered from 1, starting at the distal end. Arrows on the diagram show the direction of positive rotation. Lines show the axis of rotation. For joint 4s, spreading the fingers is negative rotation. i.e. for FF and MF, anti-clockwise is positive, and for RF and LF, clockwise is positive.

5.3.2 Loopback tendons and J0 coupling

In order to reduce the number of actuators in the forearm, joints 1 and 2 of the fingers are coupled together such that:

joint1 angle <= joint2 angle

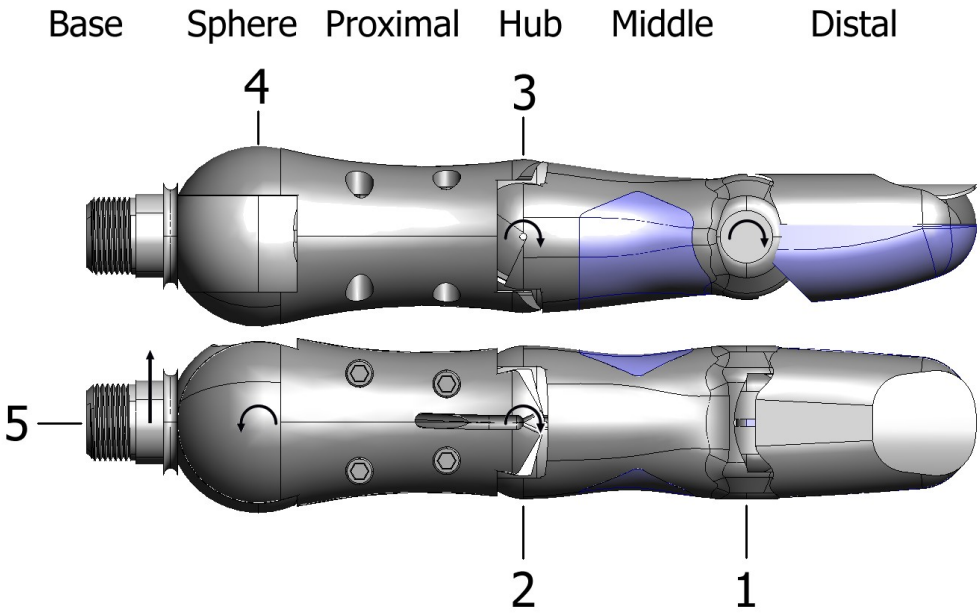
This coupling is achieved through a loopback tendon connected between the two joints as shown in the diagram below.



Any flexion of joint 1, beyond the angle of joint 2, forces joint 2 to flex to maintain the constraint. Joints 1 and 2 are together connected to one motor. For the purposes of control, they are considered to be a single joint, 0 (zero).

5.4 Thumb

There is one thumb which is mechanically different to the fingers.



5.4.1 Naming and angle conventions

The thumb has five degrees of freedom, and is stronger than the fingers, partly due to its larger diameter pulleys, and partly due to the larger motors actuating joints 4 and 5. All five joints of the thumb are independently actuated, and there is no intentional coupling between joints.

5.5 Ranges

Joint(s)	Min deg	Max deg	Min rad	Max rad	Notes
FF1, MF1, RF1, LF1	0	90	0	1.571	Coupled
FF2, MF2, RF2, LF2	0	90	0	1.571	
FF3, MF3, RF3, LF3	0	90	0	1.571	
FF4, MF4, RF4, LF4	-20	20	-0.349	0.349	
LF5	0	45	0	0.785	
TH1	0	90	0	1.571	
TH2	-30	30	-0.524	0.524	
TH3	-12	12	-0.209	0.209	
TH4	0	70	0	1.222	
TH5	-60	60	-1.047	1.047	
WR1	-40	28	-0.698	0.489	
WR2	-28	8	-0.489	0.140	

5.6 Position sensors

The angle of each joint in the finger is measured by a Hall effect sensor moving through the magnetic field of either a diametrically polarised ring magnet, or an axially polarised button magnet. An ADC in the proximal phalange samples the two sensors for joints 1, 2 and 3, while an ADC in the palm samples the sensor for joint 4.

The angle of each joint in the thumb is measured by a Hall effect sensor moving through the magnetic field of either a diametrically polarised ring magnet, or an axially polarised button magnet. An ADC in the middle phalange samples the distal two sensors, while an ADC in the palm samples the rest.

A pair of Hall effect sensors are used to measure Joint 5. They are placed 90º apart, and the sampled values from each sensor are added together at the host to improve the signal to noise ratio.

5.6.1 J0 sensors

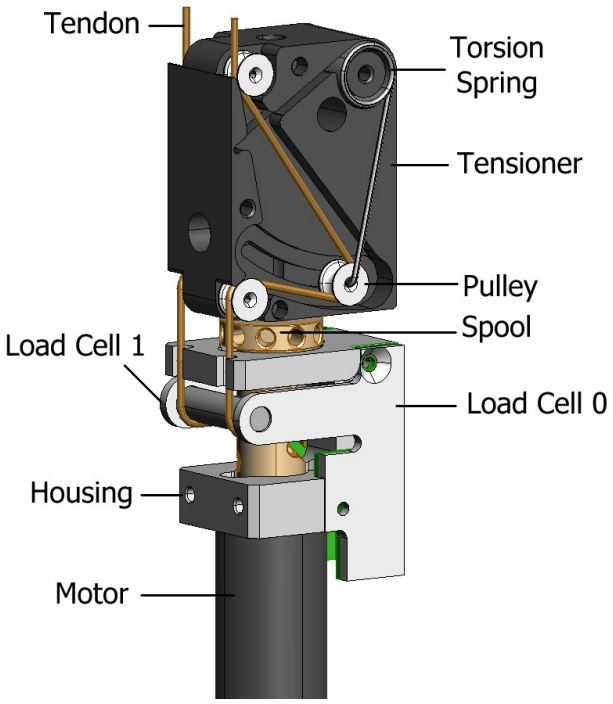
The distal and middle joints in the fingers (J1 and J2) are internally summed and the result transmitted as J0. This is because the coupling of the joints means that they should be measured and controlled as a single joint.

5.7 Motor unit

In a motor hand, every degree of freedom is driven from the array of twenty motors mounted on the forearm frame. Each motor drives two tendons to give a pull/pull control.

Force sensing is integrated into the tendons at the motors, and is used to provide compliant movements. Each pair of tendons couples a motor to a joint.

Each motor is managed by the Hand PC and completely controlled by the on-board electronics.



5.7.1 Small vs Large

Two types of motors are used in this hand design.

Small Motors: Sixteen small motors are used for all finger joints, and most thumb joints.

Part	Value	Units
Motor	Maxon 118608	
Gear	Maxon 352367	
Motor Power	3	W
Gear ratio	131:1	
Max Output Torque	405	mNm
Max Tendon Load	90[1]	N
Max continuous current	217	mA

Large Motors: Four large motors are used for the two wrist joints, and thumb joints 4 and 5.

Part	Value	Units
Motor	Maxon 110151	
Gear	Maxon 143988	
Motor Power	6	W
Gear ratio	128:1	
Max Output Torque	810	mNm
Max Tendon Load	145[2]	N
Max continuous current	350	mA

5.7.2 Tensioner

Each motor unit (except for the wrist) includes a tensioner unit. These exist purely to maintain a little tension on each tendon at all times, so that the tendons wind tidily onto the motor spool. The tension applied by the tensioner is very slight, and does nothing to compensate for backlash.

5.7.3 Torque measurement

The tendons exit the motor spool, and take a 90º turn around a metal bar. This bar is held at each end by a load cell, which is set up to measure the vertical component of any load exerted on the bar. The torque is calculated as the difference between the two readings.

If the tendons are not correctly adjusted, the tensioners may not be able to maintain tension on the spool. In this case, the tendons can become tangled around the spool.

5.7.4 Slack adjustment and tensioning.

Refer to the section: 9 Maintaining the hand.

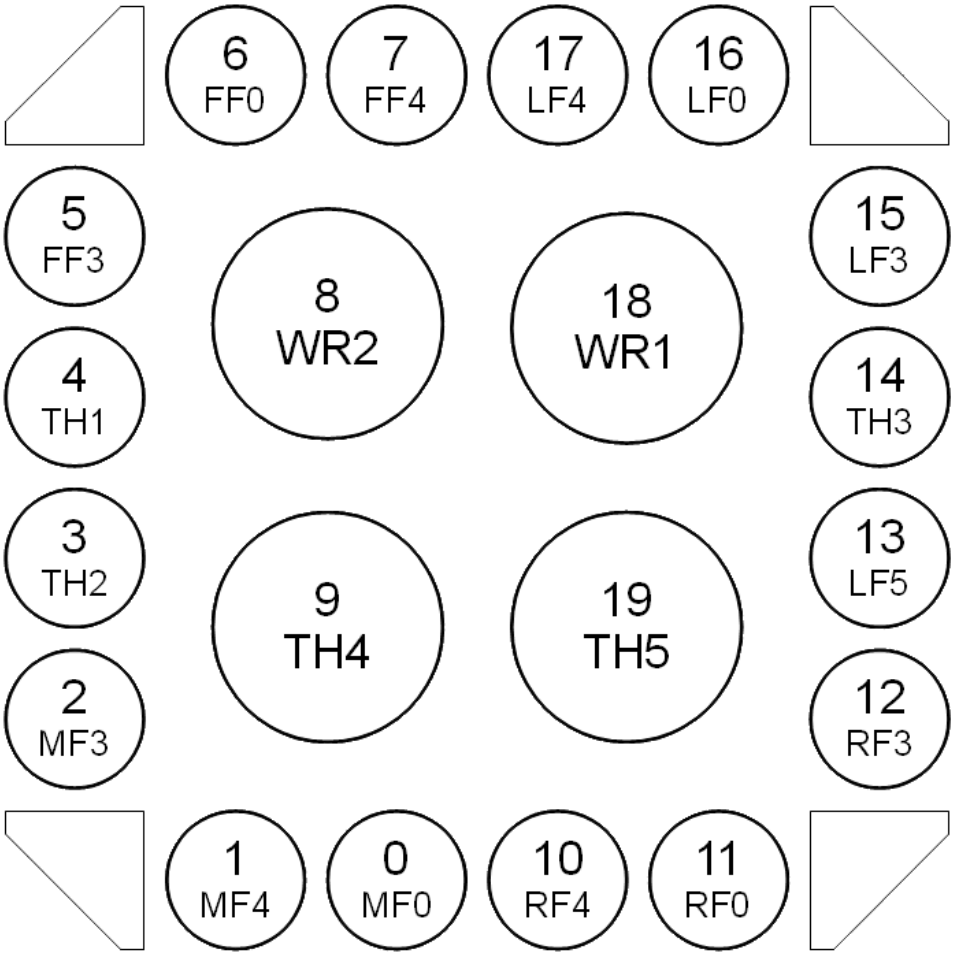
1 Verified by measurement - 2012-11-06.
2 Calculated as double the torque of the small motor.

The spool is split into two halves which can rotate relative to each other, but are normally held by a bolt fixedly with respect to each other. To take up any tendon slack, simply loosen the bolt, rotate the top half of the spool, and re-tighten the bolt.

Only one tendon is tight at any one time, while the other tendon is fairly loose. However, the loose tendon is kept under very slight tension by the tensioner to help it wind around the spool tidily. The wrist motors do not have tensioners.

5.8 Motor layout

This is the layout of the motors in the base, as seen from the hand end.



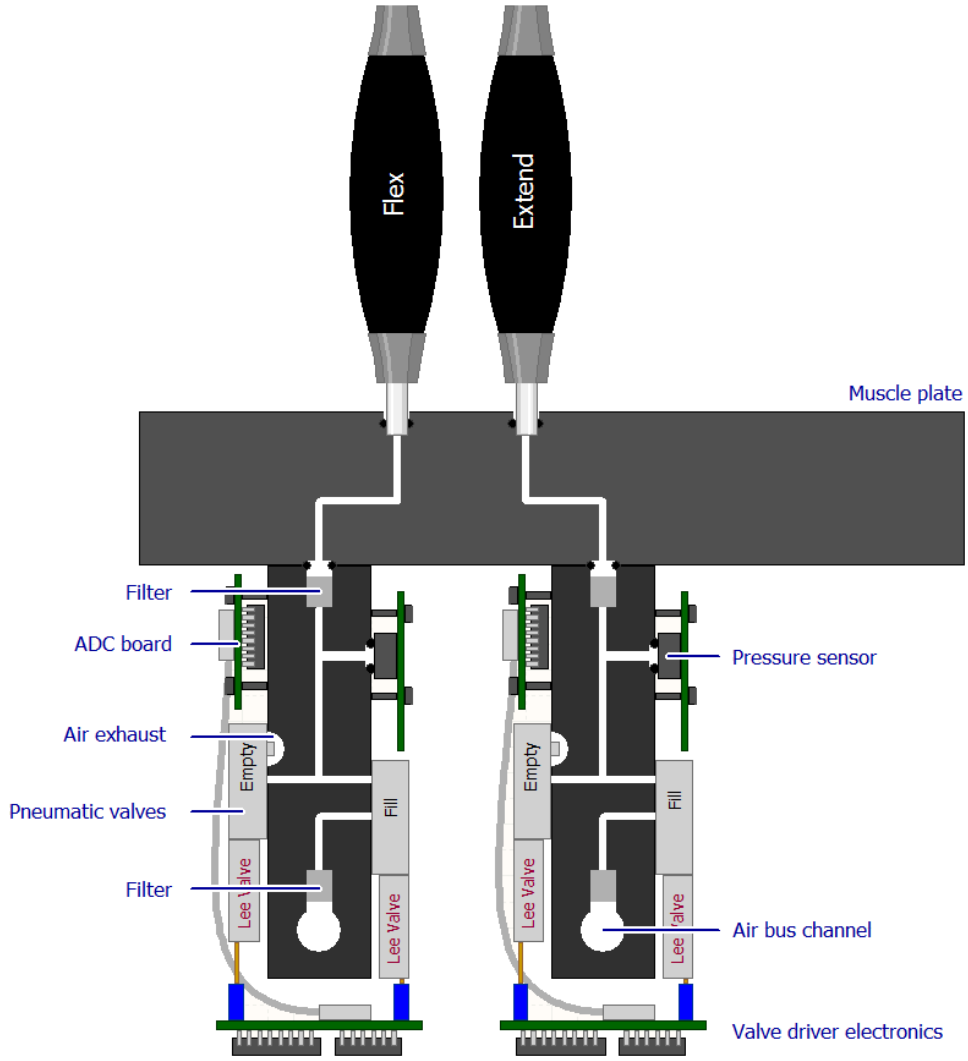
Back of Hand

5.9 Muscle system

In a muscle hand, every degree of freedom is driven from the array of forty muscles mounted

on the forearm frame. Each muscle drives one tendon to give a pull/pull control.

Pressure sensing is integrated into the tendons at the motors. Each joint is driven by a pair of muscles: flex and extend. Each muscle is driven by a pair of valves: fill and empty.



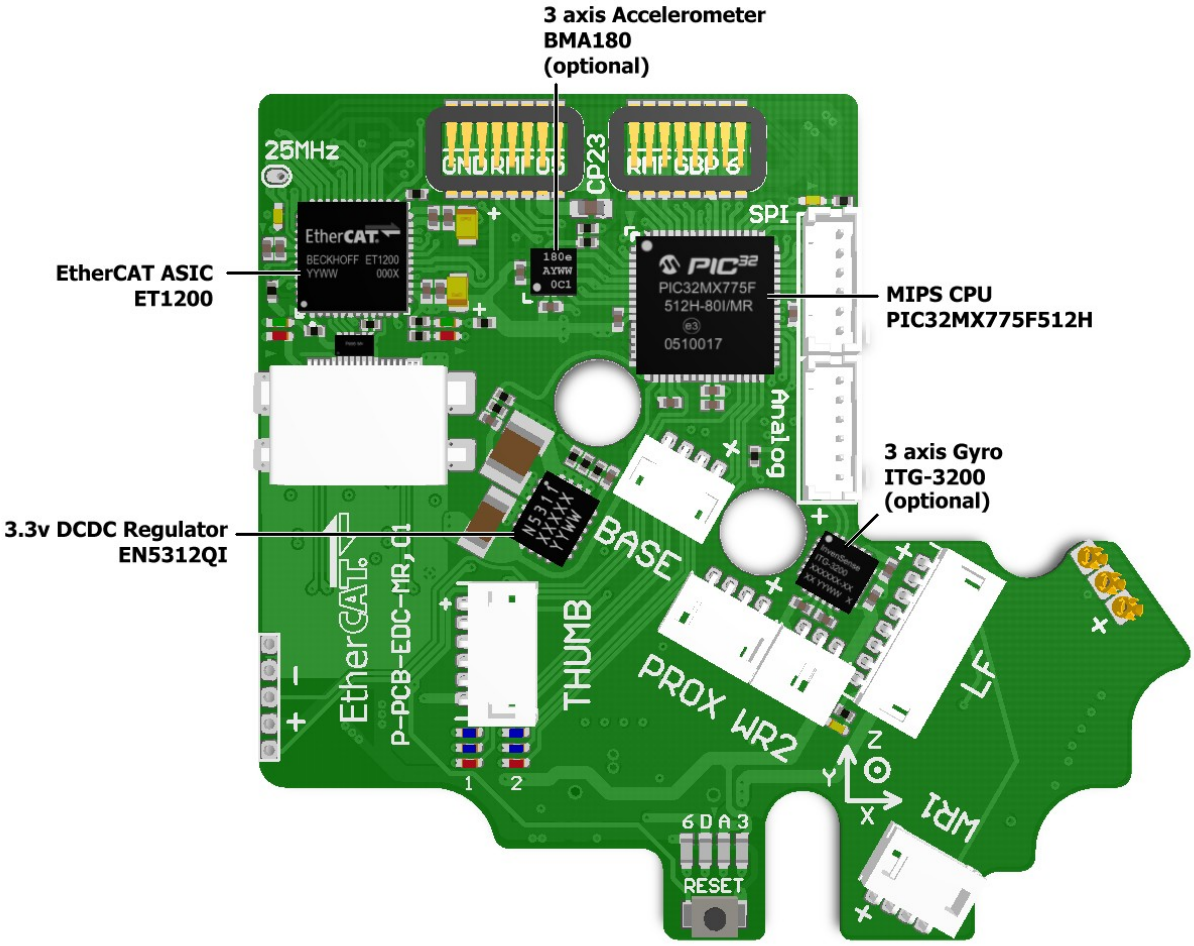
Most tendons are driven by a single muscle. However the wrist tendons, which require more force, are driven by two or four muscles.

5.9.1 Torque measurement

There is no explicit torque measurement in the muscle hand. Joint torque can be inferred from muscle pressure and joint position.

6 Electrical Description of the hand

6.1 Chipset



PIC32: This is a 32-bit MIPS CPU which performs all of the sensor sampling, and handles the flow of all the data in the hand.

ET1200: This is the EtherCAT ASIC, dealing with the reception and transmission of EtherCAT packets to and from the PC.

MCP3208: These are 12-bit ADCs used throughout the hand. There are two in the palm, and one in each finger. There are two on the underside of the Palm PCB, and one in each finger.

A1321: These are Hall effect sensors used to measure the position of all joints of the hand.

EN5312QI: This is an ultra high efficiency switching regulator which powers the PIC32 and the ET1200.

BMA180: (Optional) This is a 3-axis accelerometer which may be useful to some customers.

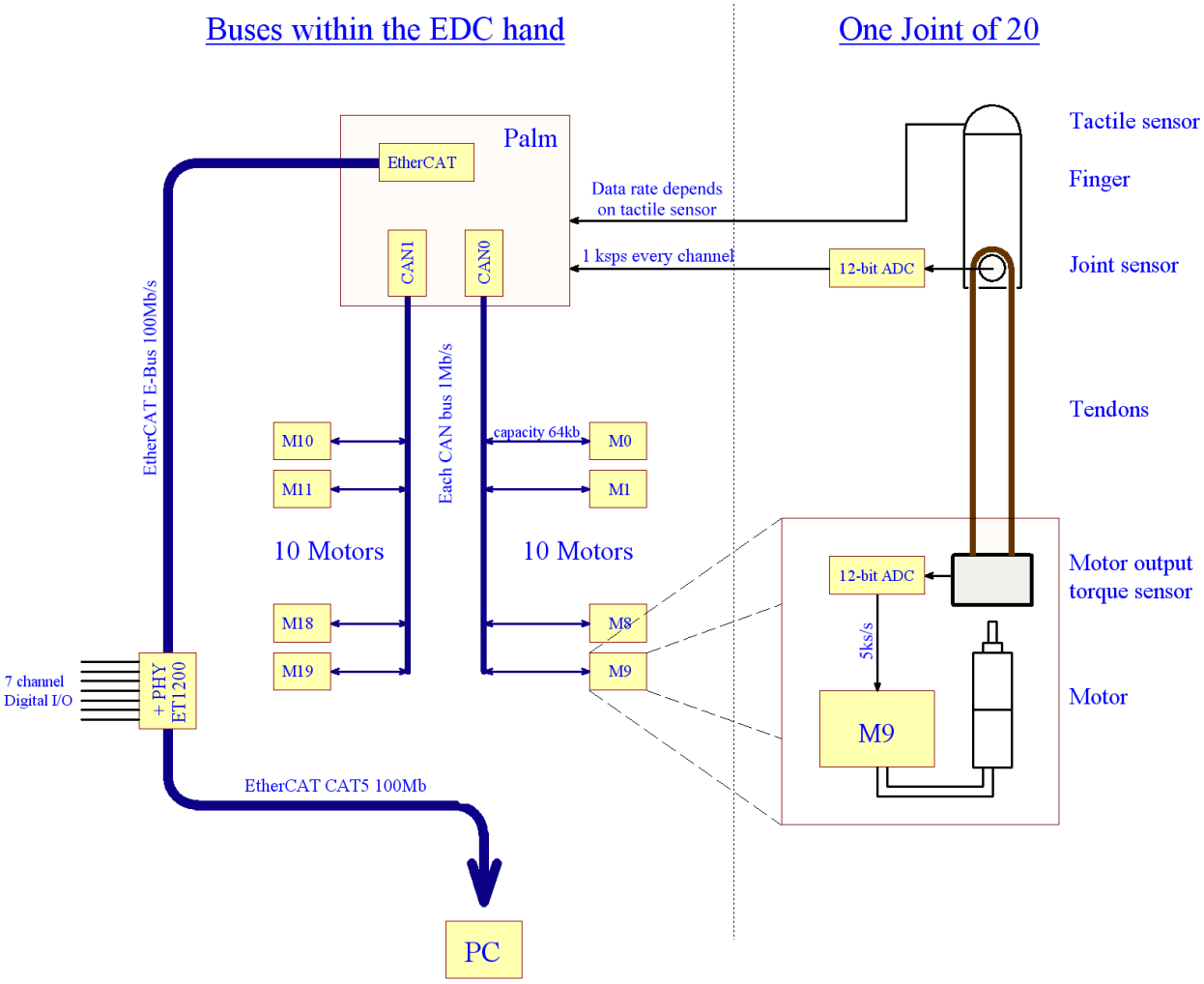
I**TG-3200:** (Optional) This is a 3-axis gyroscope which may be useful to some customers.

6.2 Data flow

6.2.1 Overview

The only data connection between the PC and the hand is a 100Mb EtherCAT cable. EtherCAT is an open high performance Ethernet-based fieldbus system. The EtherCAT cable enters the hand at the elbow, and connects to an EtherCAT bridge which converts the signal to E-Bus (LVDS) which is suitable for connection to the palm as it needs no magnetics.

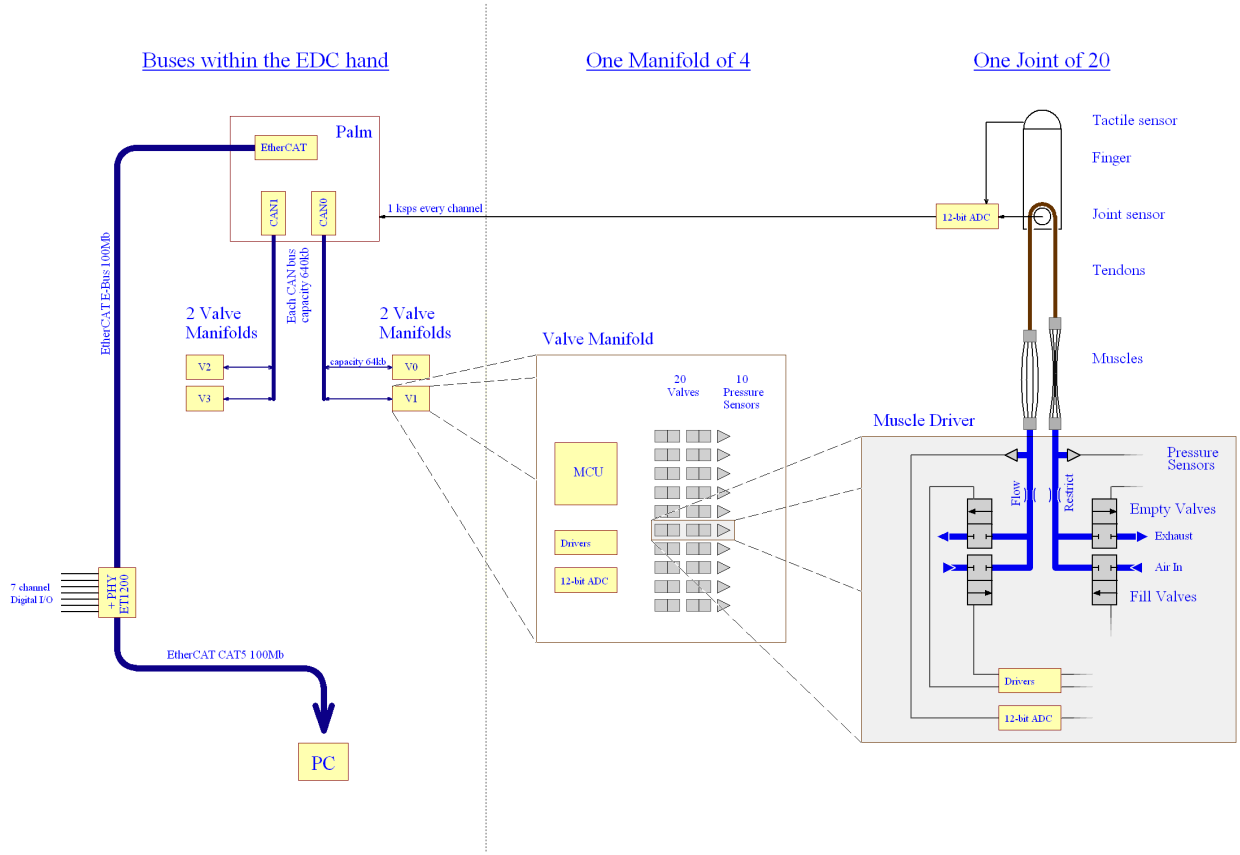
Motor Hand



Two CAN buses are connected to the palm, and each serves 10 motors, giving a total of 8kB/s bandwidth to/from each motor.

An SPI bus serves each finger, allowing the palm to sample all joint sensors at 1000Hz.

Muscle Hand



Two CAN buses are connected to the palm, and each serves 20 muscles, giving a total of 4kB/s bandwidth to/from each muscle.

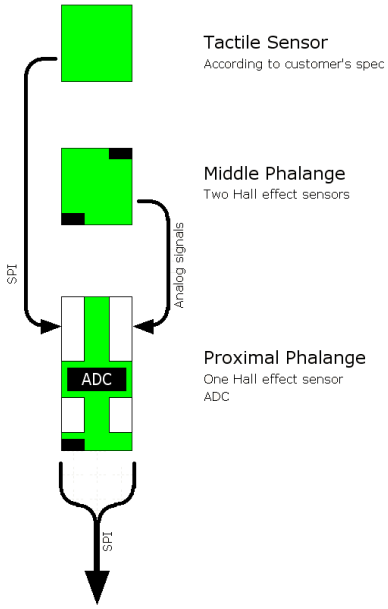
An SPI bus serves each finger, allowing the palm to sample all joint sensors at 1000Hz.

6.2.2 Palm

The palm acts as the main data hub, dealing with all EtherCAT, CAN, and SPI communication.

6.2.3 Fingers

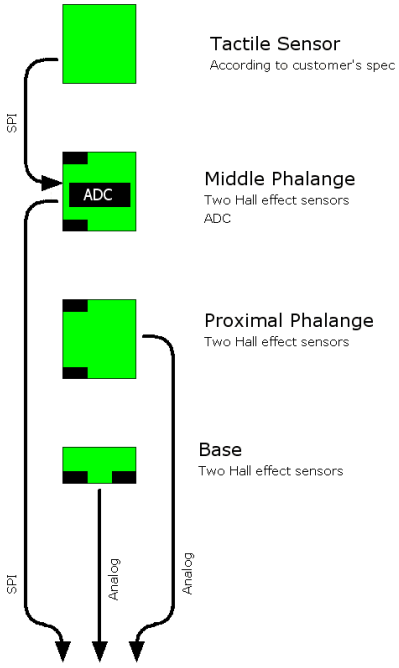
Each finger is identical. It contains four analogue Hall effect sensors which measure the position of each of its four joints. Three of these sensors are sampled by an ADC in the finger, while the fourth (J4) is sampled by an ADC in the palm.



The proximal phalange contains the ADC, which is connected to the palm by SPI. It also houses the connectors for the middle phalange and the tactile sensor.

6.2.4 Thumb

The thumb contains six Hall effect sensors to measure the position of its five joints. The base joint with its large angle range needs two sensors to fully cover the range.



Only the middle phalange of the thumb contains an ADC. The other two phalanges return analog signals which are sampled by an ADC on the palm.

6.2.5 Motors

When in torque control mode, each motor MCU samples its own torque sensor at 5000Hz, and updates the motor PWM signal at the same rate, according to its internal PID control loop. New force demands are sent to the motor MCU every millisecond.

When in position control mode, the force control loop is switched off, and the host simply sends PWM demand values to the MCU every millisecond. These are implemented immediately.

Requests for sensor data are sent to the motor MCU every 2 milliseconds, and corresponding values are returned immediately.

6.2.6 Analogue sensor filtering

Hall effect sensors naturally produce noise, and so need a simple RC filter. The filter produces a flat response to 100Hz, allowing for an undisturbed joint angle signal up to rotation speeds of 90º per 0.025sec.

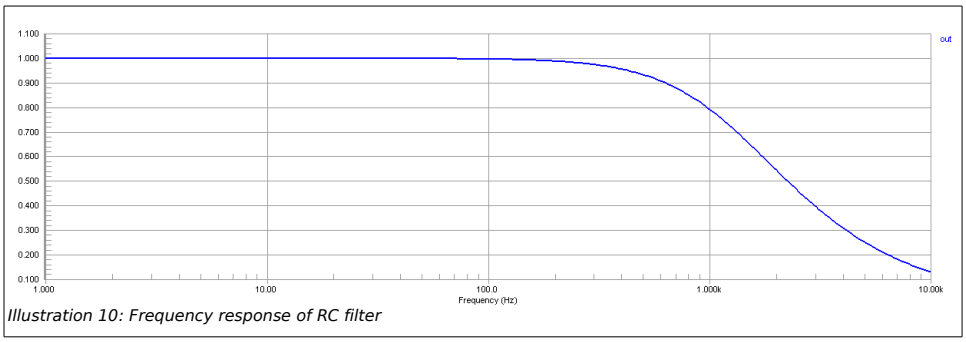
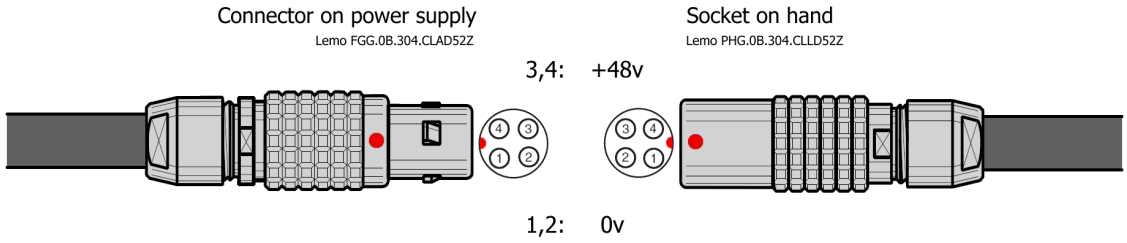


Illustration 10: Frequency response of RC filter

7 Connectors and Pinouts

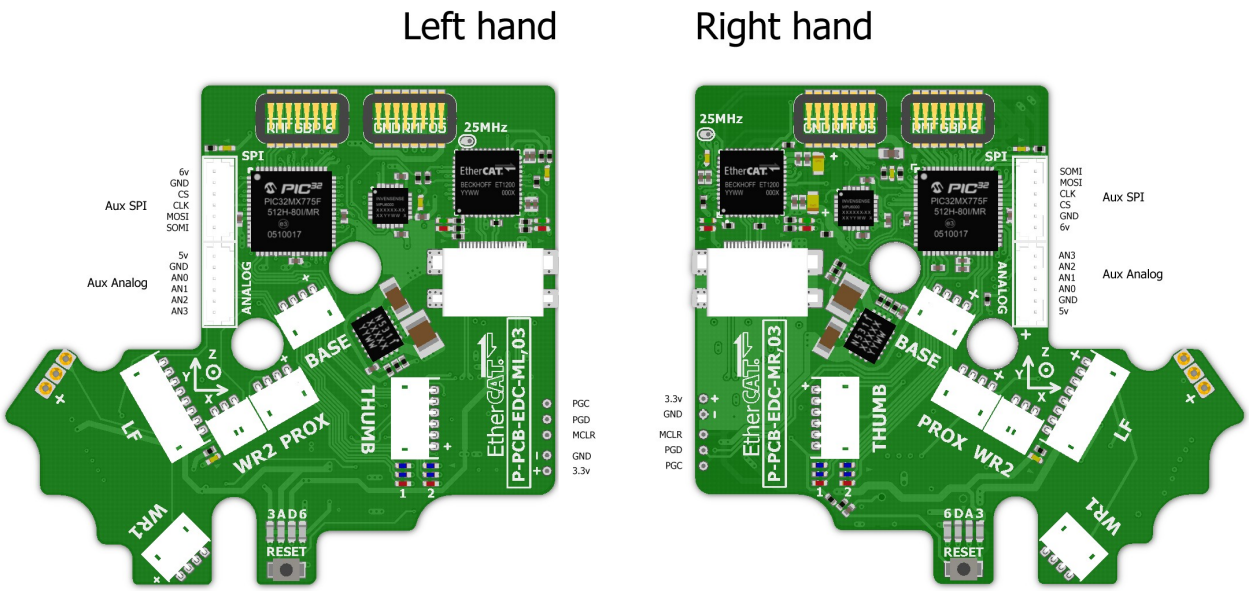
7.1 External Connectors

7.1.1 Power



7.1.2 Peripheral Connectors

The following connectors are available for connecting external peripherals.



Auxiliary Analog: This connector allows you to add up to four extra analog sensors to the hand. These sensor channels are always sampled at 1000Hz, and are available at the host. They are currently published in /debug_etherCAT_data/sensors[26 .. 28]. To assemble a connector for this socket, use the following part numbers:

Harwin M30F1100600, Harwin M30-1010046.

1. 5v regulated supply
2. GND regulated supply
3. Analog input channel 0
4. Analog input channel 1
5. Analog input channel 2
6. Analog input channel 3

Auxiliary SPI: An external SPI device may be connected here, e.g. an ADC, DAC, or I/O expander. The palm may be able to auto-detect the type of device connected, and inform the host. Currently, the palm supports only three devices:

- MCP3208 - 8 channel, 12-bit ADC
- MCP3204 - 4 channel, 12-bit ADC
- MCP3202 - 2 channel, 12-bit ADC

The auto-detection of these devices is not completely reliable, since the MCP320x chips have no explicit autodetection mechanism. The palm attempts to read all eight channels. If it reads values other than 0x000 or 0xFFF, it assumes an ADC exists. Therefore, if the analogue sensors are giving exactly 0v or 5v on every channel, the palm may fail to autodetect.

If the palm fails to autodetect a device, it assumes the device is an MCP3208.

To assemble a connector for this socket, use the following part numbers: Harwin M30F1100600, Harwin M30-1010046.

1. 6v supply
2. GND supply
3. Chip Select
4. Clock
5. Master out, Slave In
6. Slave Out, Master In

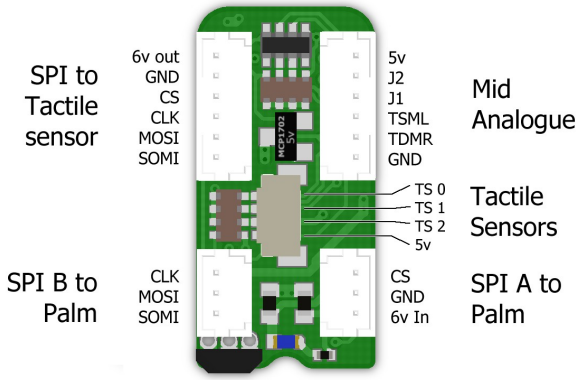
ICD3 Socket: This allows you to re-program new firmware into the Palm's PIC32 MCU. See the section on re-programming the palm. You have been supplied with an adaptor to connect the ICD3 programmer here.

7.2 Internal Connectors

The following connectors are detailed here for your information, but are not available without dismantling the hand.

7.2.1 Fingers

The connectors in the fingers are can be exposed by removing the four screws on the back of the proximal phalange, using an orange hex driver. The board is available in two versions



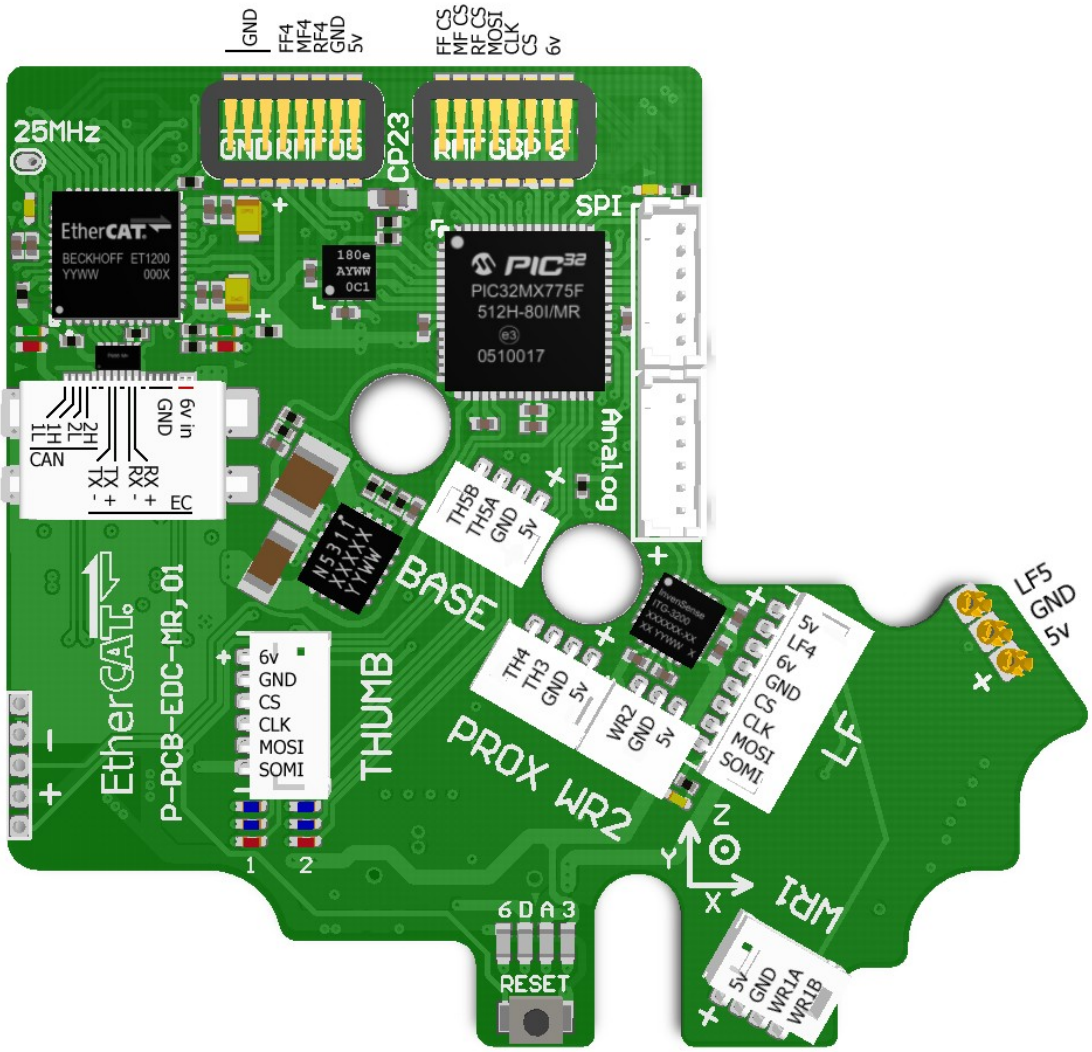
SPI to Palm: This consists of two 3-way connectors, and connects the Palm MCU to the finger.

MID Analog: This connects the two Hall effect sensors in the middle phalange to this PCB.

SPI to Tactile Sensor: This connects the tactile sensor to this PCB.

Tactile Sensor Connector: This is an analogue input, accepting three resistive sensors.

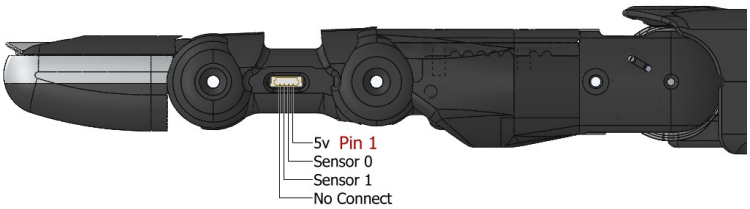
7.2.2 Palm

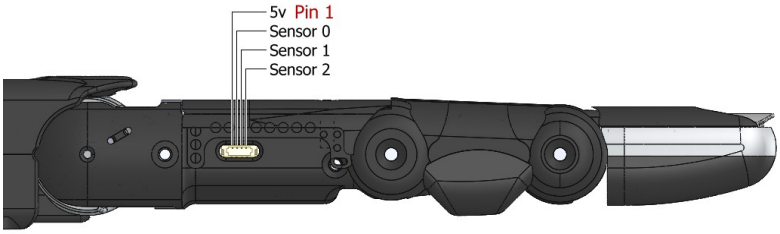


EtherCAT Dual CAN connector: This connector supplies power to the palm, as well as connecting the EtherCAT LVDS and both CAN buses.

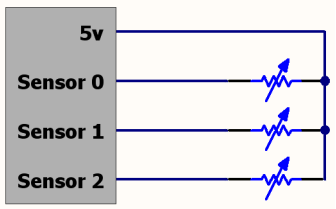
7.2.3 Fingers

Some hands have connectors for resistive sensors connected into the fingers. **These connectors are supported on some hands only. If you would like to use these sensors, please contact armando@shadowrobot.com.**





Simple variable resistive sensors can be connected to the fingers. The Middle Phalange supports two sensors, while the Proximal supports three.



The connectors are SUR series from Japan Solderless Terminals.
The mating part number is **04SUR-04SUR-32W150**

8 Software description of the hand

There are four parts to the software that controls the hand:

ROS: This is the Robot Operating System, written by Willow Garage.

Shadow Hand ROS Driver: This is the interface between ROS and the Shadow Hand, written by the Shadow Robot Company.

Palm Firmware: The palm firmware deals with data flowing between the joint sensors, PC and motors.

Motor Firmware: The motor firmware implements the torque control loop, and handles motor sensors and safety.

Valve array Firmware: The valve array firmware implements simple timed pulses to pneumatic valves to fill or empty muscles. It also reads pressure sensors connected to the muscles. It provides no internal PID control. All control loops go through the host.

8.1 Control

8.1.1 Effort and Torque

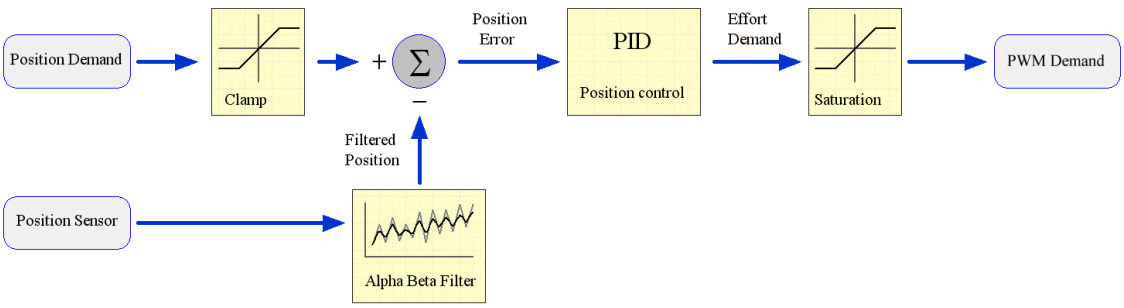
ROS uses the concept of effort as something that actuators provide. The word *effort* us used, rather than *torque*, because it can be applied to any type of actuator (rotary, linear, pressure, etc.), whereas torque only applies to rotary actuators. Since all motors on the Shadow hand are rotary, we use the words effort and torque interchangeably.

8.1.2 Controller options

The host supports two types of control for the Shadow Hand: torque (effort) control or position control.

Effort: No control is implemented on the host. The Effort demand is sent to the motor which implements it using a 5kHz control loop. See 8.3.4 Control for details of the Effort control algorithm.

Position: This uses ROS's standard PID position controller. The output of the host side PID controller is sent to the motor as a PWM demand. No effort controller is used for position control.



8.2 Palm firmware

Three versions of the palm firmware exist which support different hardware.

	Distal Tactile Sensors			Other Tactile	Auxiliary Inputs	
Hardware	PST	BioTac	UBI	Mid / Prox	Aux SPI	Aux Analogue
Firmware type						
Motor Hand						
Motor Hand with advanced tactile sensors						
Muscle hand						

Currently supported	
May support in the future	
Will not support	

The palm firmware is responsible for the following:

- Managing the ET1200 EtherCAT ASIC and EtherCAT state
- Receiving command data from the ET1200
- Transmitting the contents of the command packet to the motors
- Receiving data from the motors
- Sampling data from the joint sensors
- Sampling data from the Tactile sensors
- Loading status data into the ET1200

8.2.1 Command data for motor hand

Command data is sent from the host, and received by the palm. It consists of the following:

Item	Size	Description
EDC_Command	32 bits	Used for switching the palm into test mode
Motor data type request	32 bits	Which sensor data should the motors return?
Even or Odd motors?	16 bits	Which motors should return data?
Type of motor demand	32 bits	Are we demanding torque or PWM? Can also be used to send config values to the motors.
Motor demand data	16 bits x 20	All 20 torque or PWM demands. May also contain config data for the motors.
Tactile sensor data type request	32 bits	Which type of sensor data should the tactile sensors return?

8.2.2 Status data for motor hand

Status data is sent from the palm, and received by the host. It consists of the following:

Item	Size	Description
EDC_Command	32 bits	Copy of the same value from command data
Joint sensor / IMU data	16 bits x 37	All of the joint sensors, the Auxiliary Analog channels, and the IMU sensors.
Motor data type	32 bits	Copy of the same value from command data
Even or Odd motors?	16 bits	Copy of the same value from command data
Which motor data arrived	32 bits	Flags indicate which CAN messages were seen
Which motor data had errors	32 bits	Flags indicate that the wrong type of data was sent by this motor.
Motor data	16 bits x 2 x 10	Torque + one other sensor from 10 motors.
Tactile sensor data type	32 bits	Copy of the same value from command data
Which tactile data is valid ?	16 bits	Flags indicate which tactile data is valid.
Tactile sensor data	16 bits x 8 x 5	

8.2.3 Command data for muscle hand

Command data is sent from the host, and received by the palm. It consists of the following:

Item	Size	Description
EDC_Command	32 bits	Used for switching the palm into test mode
Muscle data type request	32 bits	Which sensor data should the muscles return?
Type of muscle demand	32 bits	Are we demanding torque or PWM? Can also be used to send config values to the motors.
Valve on-times	4 bits signed x 40	Signed valve on-times for all 40 muscles. Positive means fill, negative means empty.
Tactile sensor data type request	32 bits	Which type of sensor data should the tactile sensors return?

8.2.4 Status data for muscle hand

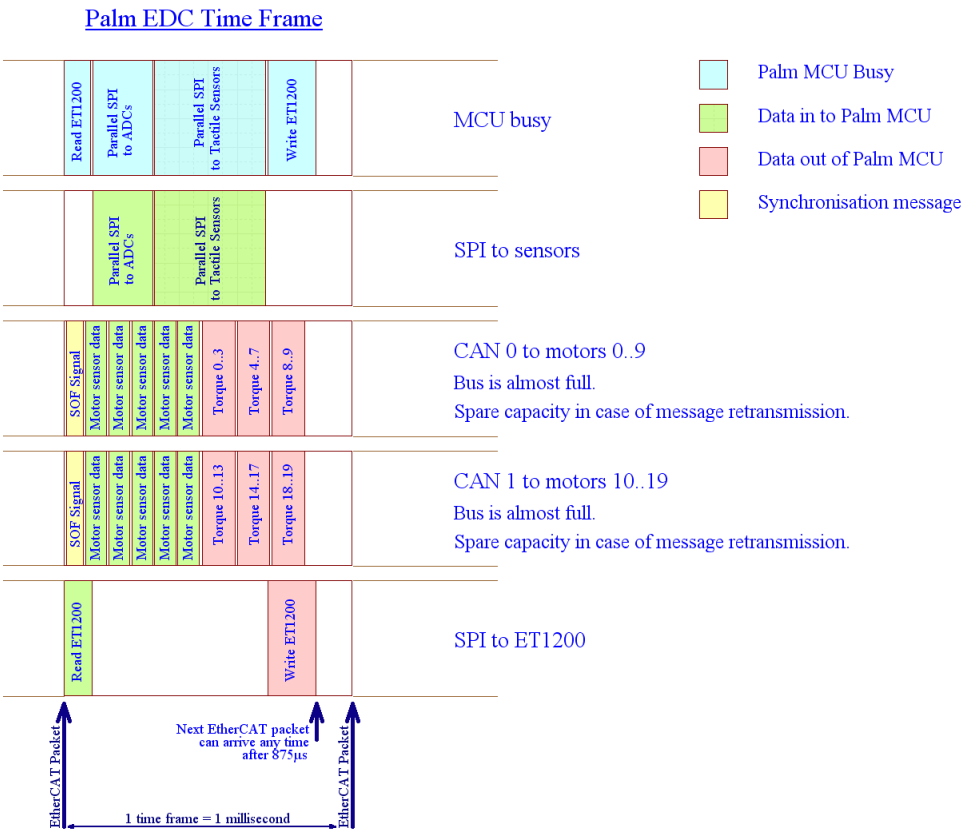
Status data is sent from the palm, and received by the host. It consists of the following:

Item	Size	Description
EDC_Command	32 bits	Copy of the same value from command data
Joint sensor / IMU data	16 bits x 37	All of the joint sensors, the Auxiliary Analog channels, and the IMU sensors.
Muscle data type	32 bits	Copy of the same value from command data
Which muscle data arrived	32 bits	Flags indicate which CAN messages were seen
Which motor data had errors	32 bits	Flags indicate that the wrong type of data was sent by this motor.
Muscle data	12 bits x 40	Usually pressure sensor data.
Tactile sensor data type	32 bits	Copy of the same value from command data
Which tactile data is valid ?	16 bits	Flags indicate which tactile data is valid.
Tactile sensor data	16 bits x 8 x 5	

8.2.5 Time frame

The Palm firmware has a considerable amount of work to complete in the 1 millisecond time frame:

- Detect the incoming EtherCAT packet
- Download the command data from the ET1200
- Request sensor data from the motors / muscle drivers
- Sample all of the joint sensors
- Request data from the tactile sensors
- Receive sensor data from the motors / muscle drivers
- Transmit demand data to the motors / muscle drivers
- Upload status data into the ET1200



In this diagram, we can see a breakdown of the time frame.

MCU Busy: We can see that the MCU is busy for most of the time, communicating with the ET1200, sampling sensors etc.

SPI to Sensors: The SPI bandwidth is really the limiting factor in the time frame. Data cannot be written back to the ET1200 until it has been collected by the MCU.

CAN buses: The CAN buses are close to maximum utilization. A little time is left during each frame to allow for re-transmission attempts. The time frame begins with a request-for-data message from the palm. The motors / muscle drivers respond immediately with their data. As soon as all 10 messages have been received, the palm sends out the demand values to all motors / muscle drivers.

SPI to ET1200: All of the data must be written to the ET1200, before the next EtherCAT packet arrives. If it does not, then the packet's status data will be filled with zeros.

8.2.6 Tactile sensors

The palm firmware supports different types of tactile sensor. The type of sensor is automatically detected, and the correct protocol is used between the hand and the sensor. The host PC is also informed of the sensor type so that it can interpret the data correctly. If more than one type of sensor is connected, then it is not possible to communicate with any of them, and no tactile sensor information will be available. The host will be informed of the conflict.

See 10.1 Distal Tactile Sensors for a list of supported tactile sensors.

8.3 Motor firmware

The motor firmware is responsible for the following:

- Ensuring the safety of the motor
- Actively controlling the force applied to the tendons by the motor
- Returning sensor data to the host

8.3.1 Safety

The motor will be halted under the following circumstances:

- The measured temperature of the motor exceeds 60ºC
- The A3950 H-bridge reports a fault
- The CRC for the FPID configuration is bad
- No demand values are seen for 20ms

8.3.2 Sensors

Every motor returns two 16-bit sensor values every 2ms. The first sensor value is usually Torque, and the second is requested by the host. Therefore it is possible for the host driver to modify the transmission rates of the sensors on the fly. By default, the rates are set in the file **sr_robot_lib/config/motor_data_polling.yaml** , and can be changed by the customer. The customer may also wish to modify the driver to have real-time control over the rates.

8.3.3 Demands

Two types of demand may be sent to the motors, depending on the type of control / debugging desired.

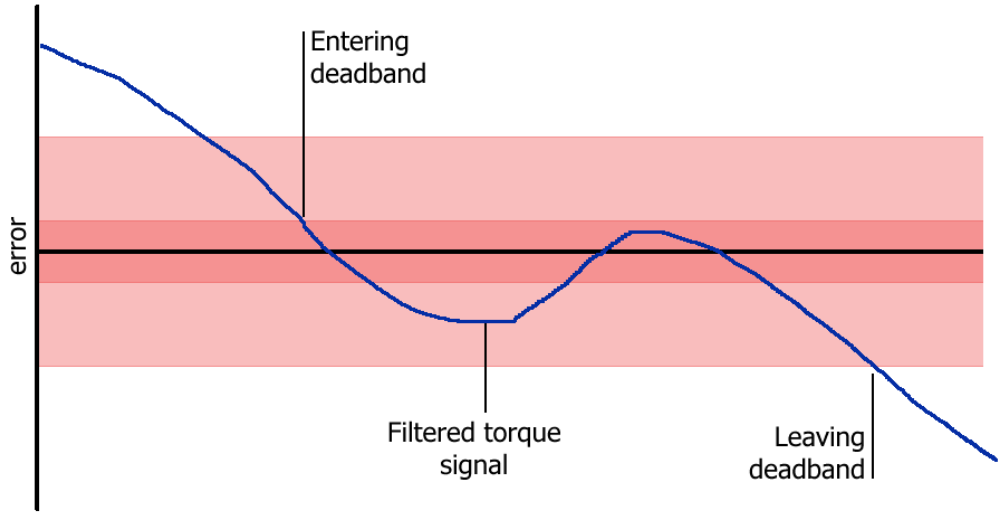
PWM demand: This is used for basic position control, and is used by default on a new hand. The PWM demand value is sent straight to the motor, unless there is a safety cutout.

Torque demand: This is an alternative method of control. The motor MCU will use its FPID algorithm to maintain the demanded torque at the tendons.

8.3.4 Control

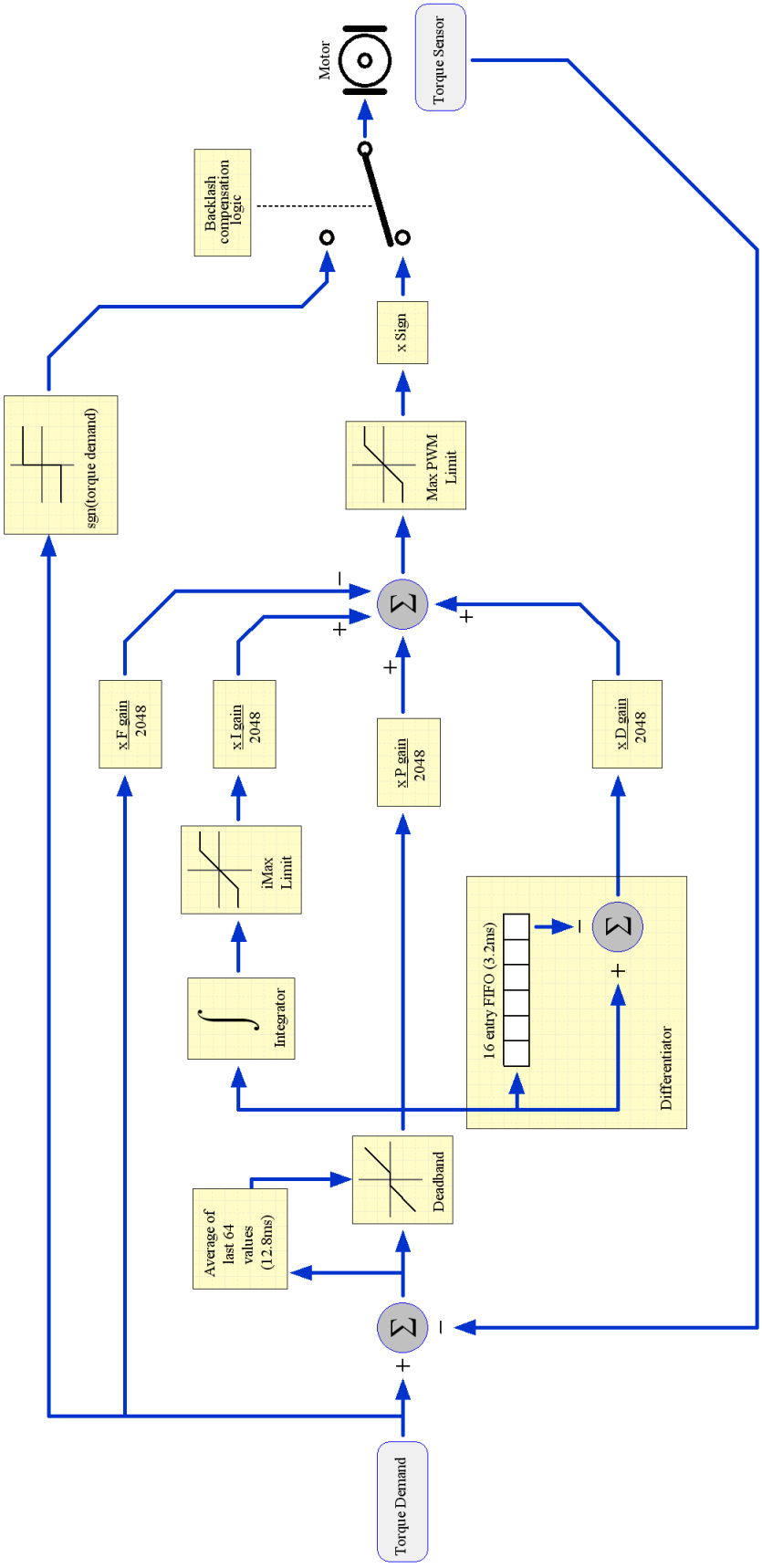
The motor firmware implements an FPID algorithm, running at 5kHz. FPID is a Feed-forward, Proportional, Integral, Derivative algorithm, where a proportion of the torque demand is fed forward to the output. The algorithm supports a number of other features to ensure the safety of the motor, stability of the control and speed of response. See next page for a flow diagram of the control algorithm.

Deadband: When the torque is sufficiently close to its target position, ideally we would like the motor to stop, drawing no power, and preventing oscillation. This is achieved with the deadband. This deadband algorithm uses the average of the last 64 torque readings (equivalent to 12.8ms) to decide whether or not the torque target has been reached. It also includes hysteresis to prevent chattering when close to the deadband.



Derivative: The derivative is implemented using a 16-entry FIFO (equivalent to 3.2ms). The derivative is the difference between the first and last entries in the FIFO.

Backlash Compensation: Due to the mechanical nature of the hand, there must be some slack in the tendons. When the motor changes direction, there will be a short time period while the spool winds in the slack. This is known as backlash, and is a known problem in machine control. Therefore, in order to improve the response time of the controller, the motor is driven at full power when the torque demand changes sign. This takes up the slack as fast as possible. Normal control is resumed as soon as tension is felt on tendon.


Motor
Torque Sensor
Backlash compensation logic
x Sign
Max PWM Limit
sgn(torque demand)
x F gain 2048
x I gain 2048
x P gain 2048
x D gain 2048
iMax Limit
Integrator
16 entry FIFO (3.2ms)
Differentiator
Deadband
Average of last 64 values (12.8ms)
Torque Demand

9 Maintaining the hand

9.1 Mechanical maintenance

9.1.1 Inspecting for wear and tear

There are three types of wear and tear that might happen to your hand over the course of its use.

Settling: Parts and fittings may settle over time. In particular, tendons may stretch slightly, and the spools on the motors may shift, increasing the amount of slack in the system. This can be remedied by re-tensioning the tendons. Hall effect sensors may shift in position slightly. This may be remedied by re-calibrating the joint.

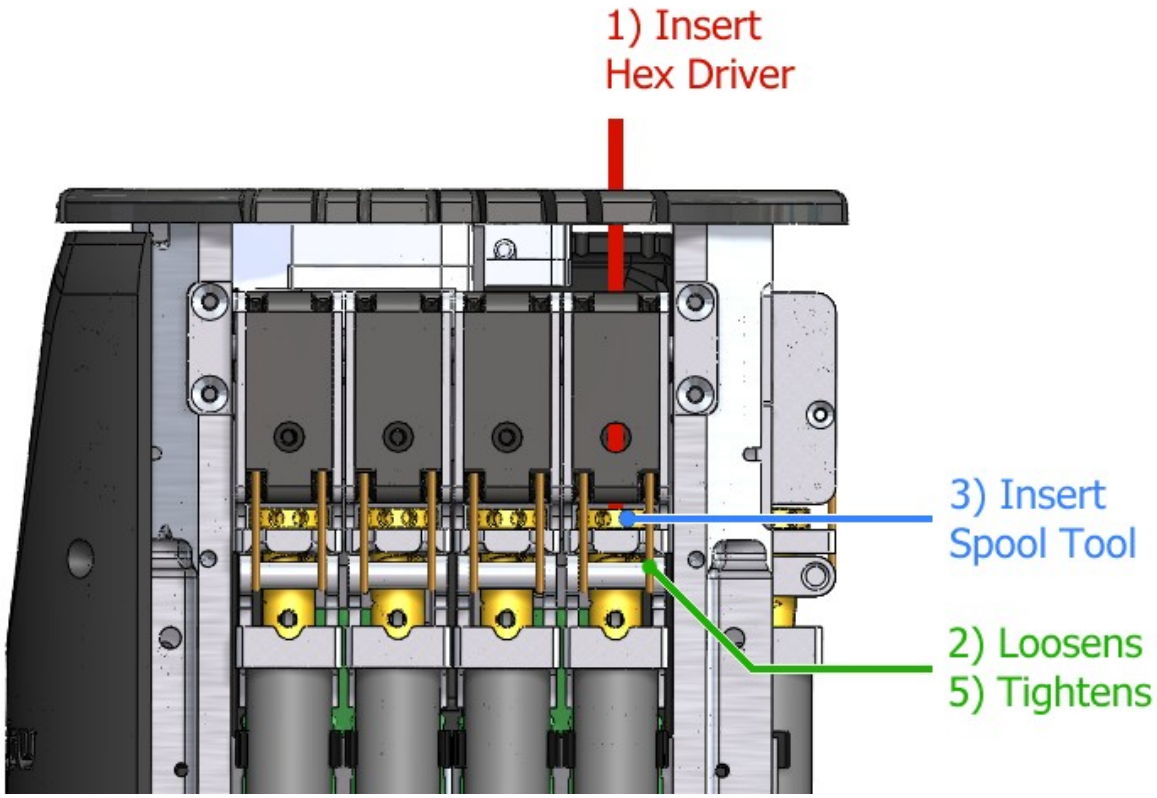
Wearing: In use, some tendons are subject to wear. In particular, wherever a tendon travels over a pulley, it may show signs of wear, appearing slightly fluffy. Normally, this isn't a problem, but if excessive wear is apparent, then the tendon might need to be changed.

Failing: As many parts of the hand are subject to very high loads at times, it is possible for some parts to fail. The exact symptoms of the failure depend on the particular part in question. Most such failures will need to be fixed by service personnel.

9.1.2 Re-tensioning

From time to time, you may find that the amount of slack in the tendons has increased, causing backlash and deteriorating the control. The tendons can be re-tightened at the motor.

Look at the Motor Layout diagram in this document to find the position of the motor that needs tightening. Using the green hex driver, remove the black motor cover; you can unplug the fan.



1. Insert the blue hex driver through the hole in the wrist plate, and push it down until you feel it engage with the hex socket in the screw in the motor spool.
2. Rotate the hex driver clockwise a little way. This will tighten the left tendon, and loosen the right tendon.
3. Insert the spool-tool into one of the 2mm holes in the top of the spool.
4. Use the hex driver to loosen the screw. The top of the spool should now be free to rotate.
5. Use the spool-tool to rotate the top of the spool to the right. You should see the right tendon tighten as you do this.
6. Re-tighten the bolt using the hex driver.

9.1.3 Broken tendon

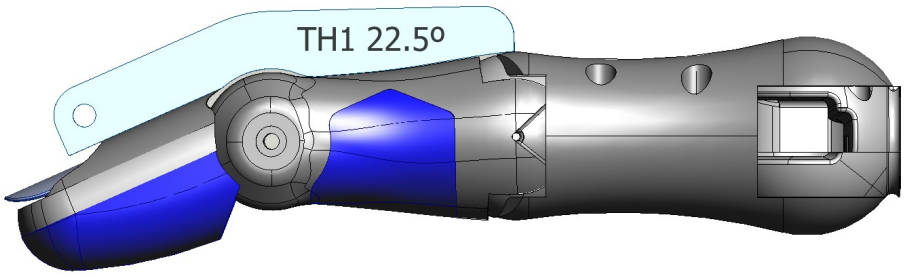
Fixing a broken tendon can be easy or extremely difficult depending on the joint involved. Complete instructions for replacing tendons is beyond the scope of this document. Please contact the Shadow Robot Company if this happens.

9.1.4 Tangled tendon

If a pair of tendons has become extremely slack, then they may become entangled at the spool. If this happens, rotate the motor until they are no longer entangled, and follow the procedure for re-tensioning.

9.1.5 Re-calibrating

The calibration tables in the host allow it to convert from the raw ADC readings sent by the Palm, into real angles in radians required by ROS. It is possible that, due to normal wear and tear, some joints may need to be re-calibrated.

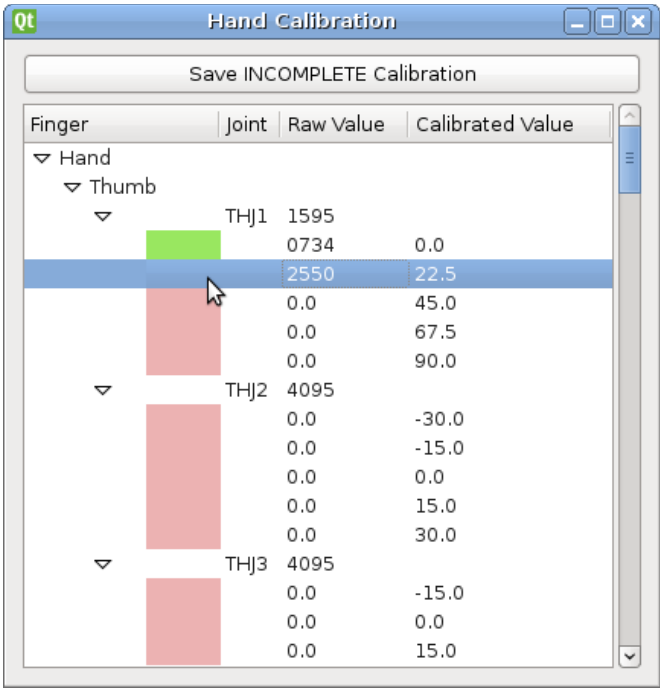


Run the GUI and start the Hand Calibration Plugin.

Find the box containing the calibration jigs for the joint you wish to calibrate. Each jig is labelled with the joint name and its angle in degrees. Not all joints have jigs.

Start with the smallest or most negative one, and place it on the joint so that it fits snugly as in the illustration above.

Double click on the first entry under the joint in the calibration plugin.



The new sensor value will appear on that line. Repeat the process for each jig (usually 5 jigs), double-clicking on the corresponding line in the calibration window each time. You may want to get a friend to help you with this process.

Lastly, click SAVE to save your new calibration.

9.2 Electronic maintenance

9.2.1 Strain Gauge failure

If one strain gauge appears to have failed, it is possible that the connector has come loose. Swing out the motor, and examine the connectors. Re-insert with a small pair of tweezers. If this was not the cause of the problem, please contact Shadow for advice.

9.2.2 Position sensor failure

If a position sensor appears to be returning incorrect values, or a constant value, first check that the raw sensor data is working:

rosrun echo /debug_etherCAT_data/sensor[?]

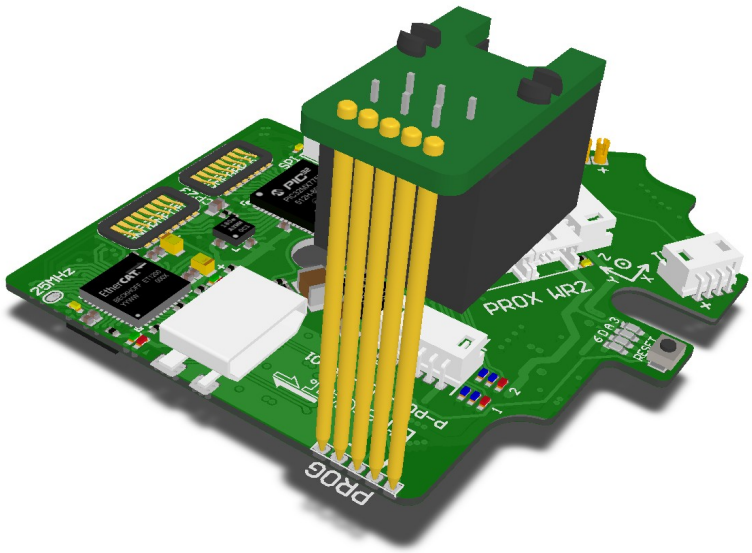
9.3 Reprogramming firmware

9.3.1 Motors

See the section 4.5.3 Bootloader for details on using the GUI to bootload new firmware.

9.3.2 Palm

There is currently no way to reprogram the palm MCU from the PC. The only way to do this is to connect a Microchip ICD3 to the palm PCB via the supplied gold pin connector.



Locate the supplied gold-pin programming adaptor, and connect it to your ICD3.

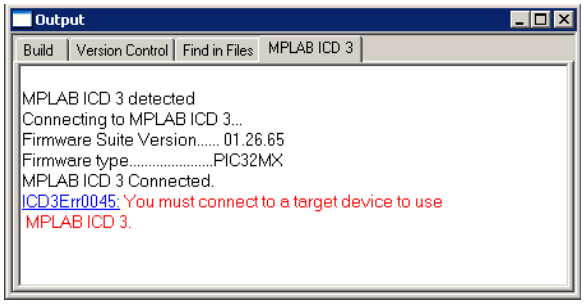
Remove the four screws holding the palm's polycarbonate cover, and remove the cover.

Open MPLAB, and open the **EDC_Palm.mcw** workspace.

Import the firmware. **File → Import**. Locate the .HEX file.

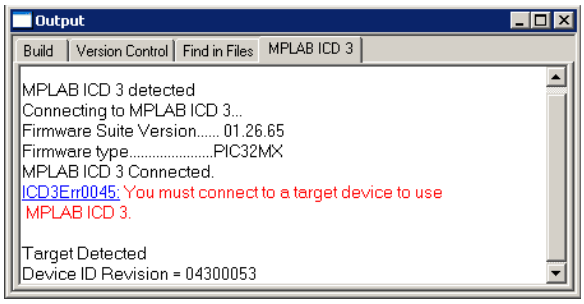
Connect to the ICD2. **Programmer → Select Programmer → MPLAB ICD3**

Wait for MPLAB to connect to the ICD3.

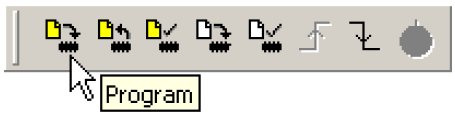


Push the gold-pin connector into the holes in the palm PCB as shown above.

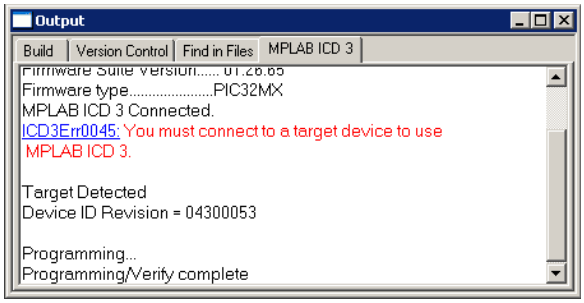
MPLAB should say “Target detected” in the Output window.



Click Program.



Wait until MPLAB says “Programming/Verify complete”.



Remove the gold-pin connector. You may need to press the reset button or power-cycle the hand for it to re-boot correctly. You will also need to re-start the ROS driver if it is currently running.

10 Peripherals

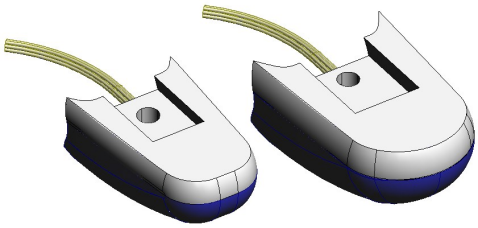
The Shadow hand can be fitted with several different types of tactile sensors, though they must

all be the same type. A mixture is not possible.

10.1 Distal Tactile Sensors

10.1.1 PST sensor

These are simple sensors, fitted as standard, which measure the air pressure within a bubble at the finger tip. When the finger tip presses on an object, the pressure in the bubble increases. The sensor incorporates an automatic drift and temperature compensation algorithm (essentially a high pass filter with an extremely low cut off frequency).



PST_F,01 PST_T,01

Illustration 11: Pressure Sensor Tactile.

Sensor	Update rate
Pressure	500Hz
Temperature	500Hz

10.1.2 Biotac

The BioTac® is a biologically inspired tactile sensor from SynTouch LLC. It consists of a rigid core surrounded by an elastic skin filled with a fluid to give a compliance similar to the human fingertip. The BioTac is capable of detecting the full range of sensory information that human fingers can detect: forces, microvibrations, and thermal gradients. The skin is an easily replaced, low-cost, moulded elastomeric sleeve.

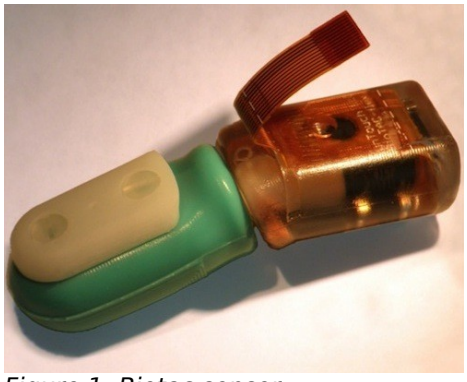


Figure 1: Biotac sensor

Sensor	Update rate
Pressure AC signal	2000Hz

Pressure DC signal	90Hz
Temperature AC & DC	90Hz
19 Normal force sensors	90Hz each

Note: Since the BioTac sensor replaces the joint sensing electronics in the finger, a Hall effect sensor is integrated into it which measures the position of finger and thumb joint 2. If a BioTac is removed from the hand, then position control will not work for that finger's joint 2. Please refer to the BioTac user manual for information on this sensor.

10.1.3 ATI Nano 17

The ATI Nano 17 sensor provides 6-axis force-torque measurement for each finger and thumb. These sensors are very sensitive, and able to detect forces of a few hundredths of a Newton. They are able to determine the location of touch anywhere on the black elipsoid, as well as the direction of applied force and local torque. The ATI Nano 17 sensor is large in comparison to the finger tip, and each has a large cable that must be routed back to data acquisition cards in a PC.

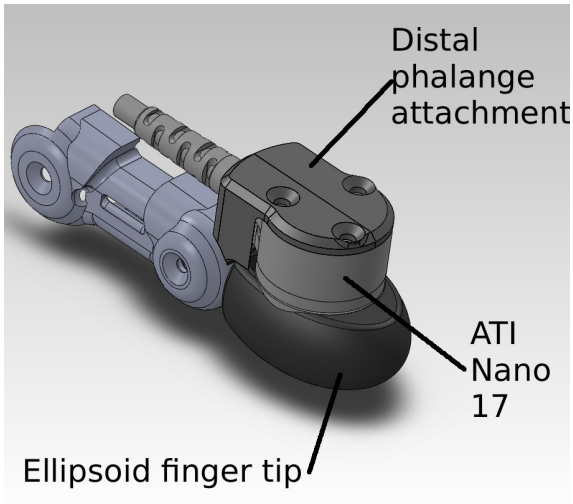
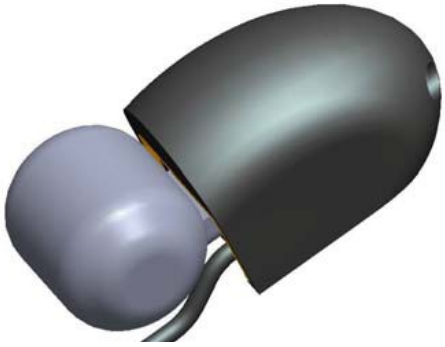


Figure 2: ATI Nano 17 sensor and ellipsoid finger tip

10.1.4 Sunrise 6-axis sensor

Currently under investigation is a similar 6 axis force torque sensor custom made by SRI (Sunrise Auto Safety Technology & Sunrise Instruments) that will have integrated amplification and sampling, removing the need for the large external cables of the ATI Nano 17.



10.1.5 Unversity of Bielefeld (UBI) Tactile Sensors

Some hands support the UBI distal tactile sensors. These provide 16 taxels on each fingertip and thumbtip. If you would like to use these sensors on your hand, please contact Armando De La Rosa T. at: hand@shadowrobot.com.

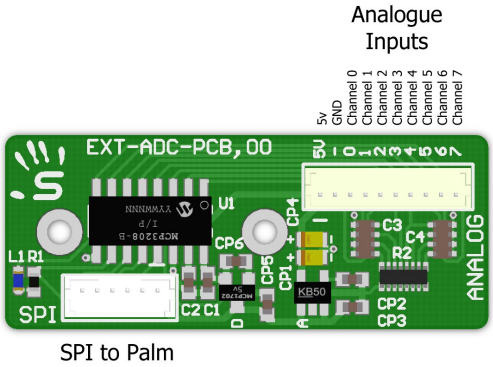
10.2 Mid / Prox Tactile Sensors

Some hands support additional tactile sensors. If you would like to use these sensors on your hand, please contact Armando De La Rosa T. at: hand@shadowrobot.com.

10.3 Palm Auxiliary Sensors

10.3.1 8-Channel 12-bit ADC

This external board connects to the Palm/s Auxiliary SPI port and provides an 8 channel, 12-bit ADC. The inputs have the same filters as the palm's Hall effect sensors. See 6.2.6 Analogue sensor filtering.



SPI to Palm