PRENSILIA s.r.l.

Mia Hand Series

User Guide – v1.0

Prensilia SRL
Viale R. Piaggio 32
56025 Pontedera (PI)
ITALY

www.prensilia.com – info@prensilia.com

Table of contents

Welcome to Prensilia!

Get ready to experience the advanced motor and sensory features of your Prensilia Mia Hand Series. Setting up your hand is easy: simply connect the USB plug provided to your host controller and power on the hand with an external supply. Take some time to explore the features on your device. This guide provides instructions and tips to help you learn the basic quickly, to properly operate your hand and to perform basic troubleshooting. Thanks for acquiring this prototype! We hope it will boost your research.

WARNING: The Mia Hand you purchased is a prototype for research use only.

Usage conditions

- The prototype should be used and experimented for research only in laboratory.
- Do not use the prototype in any system on which people's lives depend (life support, weapons, etc.).
- In the case of human or animal experimentation, make sure to have the supply and the connections from/to the prototype electrically separated from the animal/human.
- Always connect the prototype to a host PC, through the USB cable provided.
- The prototype is not waterproof: keep it away from water or other liquids.
- If an Institutional Review Board approval (or other kinds of permission) is required by your Institution to carry out experiments with a non-FDA, non-CE approved prototype, you should obtain it before starting your experiments.
- Use a proper power supply: do not supply the prototype with electrical levels not compliant with the ratings described in this guide.
- The electronic boards of the prototype contain components that are sensitive to electro-static discharge. Care should be taken when handling the hand without the cover.
- Prensilia SRL does not warrant for the research usage that you may carry out, and for any other usage outside from the laboratory, or not complying with this user guide.

Applications examples

- Prosthetics;
- Humanoid, industrial, cognitive, rehabilitation robotics;
- Assistive robotics and technology;
- Tele-operation (remote grasping and manipulation).
- Human-robot interaction;
- Brain-Machine Interfaces (BMI);
- Bidirectional Human-Machine Interfaces (HMI);
- Neuroscience (sensorimotor control of the hand);
- Education and performing arts.

Additional information, technical support and repairs

Prensilia SRL provides technical and user support via email (support@prensilia.com). Support is provided for the lifetime of the equipment. Requests for repairs should be made through the Support department. For damage occurring outside of the warranty period or provisions (see the sales conditions), customers will be provided with cost estimates prior to repairs being performed.

Additional information on the components of the Mia Hand Series is available from their respective manufacturers:

- USB to UART cable from FTDI Ltd (https://ftdichip.com/)

Symbols in this manual

The boxes with a symbol below are used throughout this user guide to underline important information, such as danger levels or special notes.

NOTE:

This indicates important information to take into consideration while operating the robotic cell.

WARNING:

This indicates a potentially hazardous situation which, if not avoided, could result in injury or major damage to the equipment.

DANGER:

This indicates an imminently hazardous situation which, if not avoided, could result in death or serious injury.

1 General usage and safety considerations

This section provides important safety considerations related to the use of the robotic cell.

WARNING: Every time you intend to use the Mia Hand, please:

1. Make sure the Mia Hand is properly and securely bolted in place;
2. Make sure there is no external object that could impede the movements of the Mia Hand;
3. Do not wear loose clothing or jewellery when working with the Mia Hand. Make sure long hair is tied back when working with the Mia Hand;
4. Never use the Mia Hand if it is damaged, for example if the glove is broken or removed;
5. Make sure to warn people to keep their heads and faces outside the Mia Hand workspace or Mia Hand about to start operating;

2 Description of the Mia Hand

The Mia Hand is a five-fingered, self-contained anthropomorphic hand with three electrical motors. Each motor activates one Degree of Actuation (DoA); in particular:

- the independent flexion/extension of the thumb;
- the semi-independent abduction/adduction of the thumb and flexion/extension of the index finger;
- the simultaneous/synchronous flexion/extension of the last three fingers.

Each finger comprises a single phalanx. The distal part of each finger (i.e. the fingertip) is covered with rubber, promoting stability of the grip during object grasping.

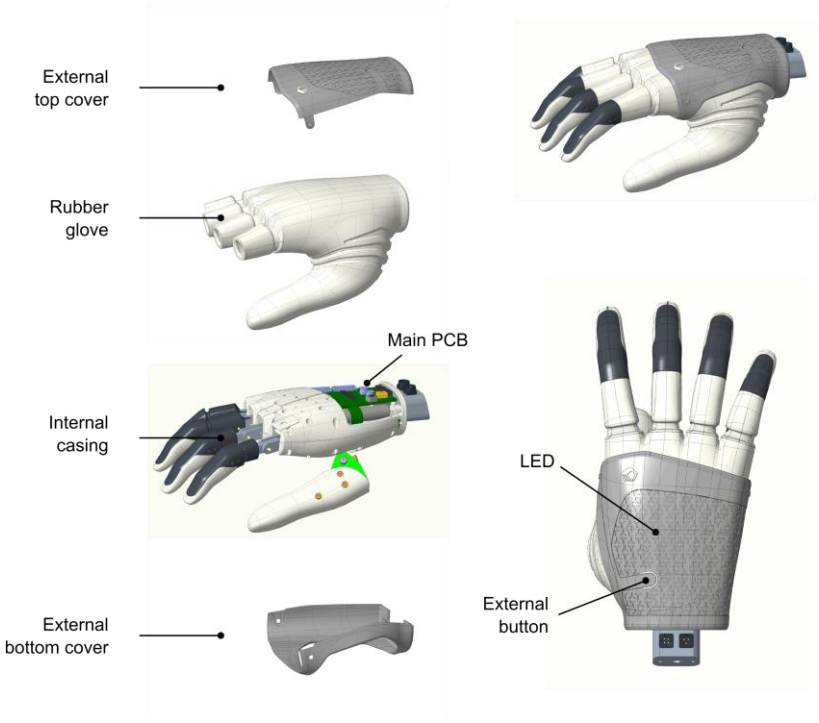


Figure 1 Exploded view of the Mia hand

As a safety feature, the actuation is non-back-drivable. This means that motion is not transmitted from the fingers to the motors. Thus, external actions (e.g. manual displacement, gravity) cannot move the fingers. Additionally, in the case of power supply failure, the hand will not reopen. This avoids any currently held object to fall. Mia Hand includes basic sensors for grasping, gesture and elementary manipulation:

- three motor digital encoders (one for each motor), that measure motors (and consequently fingers) positions;
- three motor current sensors that measure the electrical current flowing into the motors (hence the grasping force);
- six limit switches (two for each motor), that detect when the finger is fully flexed or extended;
- three bi-axial force sensors mounted on the thumb, index and middle fingers[1].

All sensors can be read, and all motors can be operated and controlled by means of the real-time communication protocol described in Chapter 4 – Communication protocol explained.

[1] Force sensors are not available on all hands.

NOTE: To learn about all the features of the Mia Hand, and hence its potential, it is worth describing the internal mechanisms and principles of operation. Take some time to read the following paragraphs.

2.1 Mechanism overview

Mia Hand is based on three identical motors and on a Geneva drive transmission. Its actuation architecture comprises:

- an independent flexion/extension of the thumb (motor M1),
- a simultaneous/synchronous flexion/extension of the last three fingers (motor M2),
- a semi-independent abduction/adduction (ab/ad) of the thumb and flexion/extension of the index finger (motor M3). This is called the Thumb-Index Semi-Independent Transmission (TISIT).

The TISIT combines a Geneva drive and a four-bar mechanism, which are both driven, in parallel, by motor M3 (Figure 2). In particular, the input of the TISIT (M3) is connected to both the drive wheel of the Geneva drive and the crank of the four-bar mechanism. The output of the Geneva drive (i.e. the driven wheel) rotates the flexion/extension plane of the thumb between two positions (ab/ad of the thumb), whereas the output of the four-bar mechanism (i.e. the rocker) flexes/extends the index finger in a continuous fashion. The operation of the TISIT can be divided in three phases (Figure 2):

- Phase 1: the index finger starts from fully flexed and, as the crank rotates, it extends.
- Phase 2 (switching phase): the index reaches its maximum extension. The thumb switches its opposition/reposition state.
- Phase 3: the index flexes again until complete flexion. In particular, the Geneva drive is synchronized with the four-bar mechanism such that the driven wheel switches between steps when the index is close to full extension.

Once calibrated, the position readout for the index finger (motor M3) will range between -255 (index flexed, thumb abducted) and 255 (index flexed, thumb adducted). The position for the thumb and the last three fingers will range between 0 (digit open) to +255 (digit closed).

NOTE: For approximation reasons, after calibration, the actual range of positions mentioned above could change slightly.

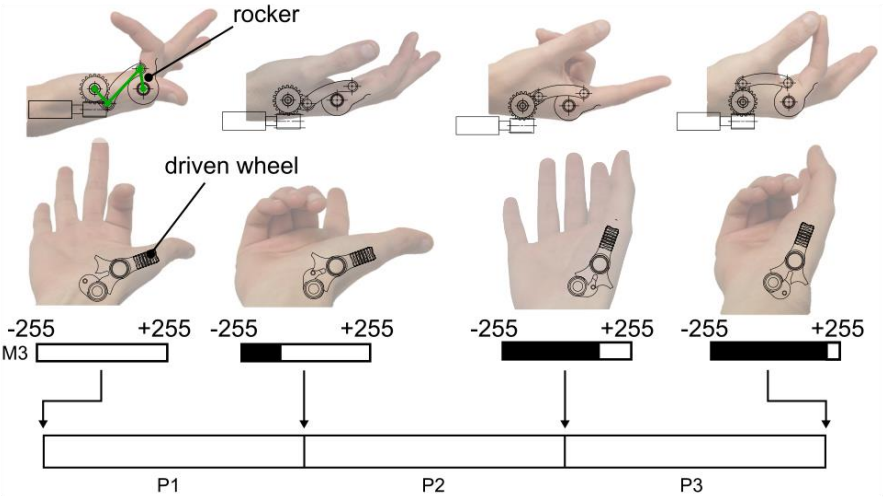


Figure 2 The target position of the index finger ranges from -255 to 255, being -255 the starting position, as indicated by the progress bar.

2.2 Sensory system overview

The Mia Hand integrates: 3 motor encoders for position measurements (one for each motor), 3 motor current sensors, 6 limit switches (two for each motor), and three bi-axial force sensors mounted on the index, middle and thumb fingers[2]. Encoders and force sensors (if present) are used for position, speed and force control.

2.2.1 Motor encoders

The three motors exploited in the Mia Hand are equipped with Hall effect sensors. These are used by the motor driver to correctly commutate between the motor phases. In addition to that, the information from the Hall effect sensors is also used by the main board of the Mia Hand to retrieve the position of each motor, mimicking the functionality of an incremental encoder. This provides information on the position of the motor. This position is relative to a reference position (i.e. 0) corresponding to when the related finger is completely extended. On top of this, absolute position information is retrieved after calibration. Calibrated positions are a processed information which is computed by the internal control system. Externally, absolute positions are the only available ones. These are encoded by 3-digit numbers plus their sign (Chapter 4 – Communication protocol explained).

2.2.2 Current sensors

Mia Hand integrates three motor current sensors (one for each motor). The current sensors consist of current shunt resistors, designed to monitor the current flow by measuring the voltage drop across a resistor placed in the current path. These sensors are embedded in the main board of the hand. The motor current I_M is acquired through a 10-bit ADC and is related to the ADC output (I_R) through the following law:

$$I_M = \frac{I_R}{750}$$

This is valid for all the motors. I_M is provided in Ampere. The nominal motor current resolution is 1.3 mA.

2.2.3 Limit switches

For safety reasons, Mia Hand fingers are stopped slightly before their actual mechanical stops. This is achieved by means of limit switches, which, when activated, trigger a stop of the related motor. Two different types of sensors are used as limit switches. Reflective optical sensors (Fairchild QRE1113) are used to signal

[2] Force sensors are not available on all hands

to the controller when the long fingers are fully flexed or extended. Hall effect digital proximity sensors (Allegro A3213 hall effect switches) are used, instead, for the thumb.

2.2.4 Force sensors[3]

The Mia Hand is equipped with bi-axial force sensors based on strain gauges technology and able to measure the normal (i.e. grasping) and the tangential (i.e. load) force exerted by the fingers. In the case of the index and middle fingers, the force sensor is a steel beam placed on the proximal part of the finger. Please note that the force sensor integrated in the long fingers does not measure pure force, but torque. Thus, the output of the sensor is sensitive to the lever arm of the applied force. This cannot be compensated in software as the location of the contact point between the object and the hand is not known.

For the thumb sensor, the sensing element is integrated at the base of the thumb, **proximally to the thumb flexion joint**. This means that the output of the sensor will change based on the flexion angle of the thumb. This issue is compensated via firmware by combining the information retrieved by the motor encoders (i.e. the angular position of the thumb) and the output of the force sensors. It is also worth noting that the normal force applied on the thumb does not depend on the point of application of such force (i.e. it is a purely force sensor).

This configuration allows to measure the grasping and the load force for all the available grasps (i.e. cylindrical, pinch and lateral grasps).

NOTE: the force sensor integrated in the long fingers is in fact a torque sensor, and thus the output of the sensor is sensitive to the lever arm of the force.

NOTE: for the thumb sensor, the sensing element is integrated at the base of the thumb, *proximally* to the thumb flexion joint. This means that the output of the sensor will change based on the flexion angle of the thumb.

2.3 Control system overview

The embedded controller in the Mia Hand is arranged in a hierarchical architecture consisting of one Master Controller (MC) and one Slave Controller (SC). The SC reads the information from the Hall effect sensors of the three motors and computes the relative position for each of them, as well as the rotation speed. This information is continuously transferred to the MC.

The SC is controlled by the MC, that regulates overall hand operation, implements all other functions (described below) and acts as interface with the external world. Although Mia Hand comes with the USB cable provided (see Chapter 3 – Installation), it uses a standard serial communication bus (UART, 3V3 TTL levels) to communicate with the external world. Mia Hand is also provided with non-volatile memory (EEPROM). This allows to memorize automatic grasp parameters (preshaping postures, hold off, etc.), PID gains (K_p, K_i, K_d) of the control algorithms, etc.

The main functionalities implemented by the MC are:

- **ASCII command interpreter**: a communication protocol is implemented to set/configure all parameters and trigger all functions of the board. The commands are transmitted over a serial communication bus.
- **PID controllers**: they control the three motors in position/speed/force mode. The control loop runs at 1 kHz and the PID gains can be changed in real time through dedicated ASCII commands.
- **Automatic grasps**: the controller allows to perform commonly used postures using single control commands (cylindrical, pinch and lateral grasps and relax posture). In particular, for each grasp, OPEN

[3] Force sensors are not available on all hands.

(i.e. preshape) and CLOSE (i.e. final) positions can be defined by the user. When the command is issued, the fingers synchronously move from the OPEN towards the CLOSE position.

When Mia Hand is connected to a PC, a graphical user interface (GUI) allows quick access to the commands implemented in the communication protocol.

2.3.1 Calibration routines

Mia Hand features two calibration routines that can be issued by sending the appropriate commands over the communication bus or by acting on the physical button on the dorsum of the hand. These are referred as “complete” calibration routine and “fast” calibration routine.

The complete calibration is intended to be issued occasionally. Indeed, the complete calibration routine maps the open/close positions of the DoA to the position values of 0 (or -255, for the index finger) and 255, respectively, and hence to readout consistent calibrated position values. Once issued, every DoA fully closes and opens so that the maximum and minimum relative positions obtained from the encoders are stored in the EEPROM, for keeping the mapping after the hand restarts.

The fast calibration, as the name recalls, is a quicker routine: all DoA fully open and their relative position is re-aligned with the ones stored during the last complete calibration. Then, the index finger position is set to 40. The fast calibration can be issued when a complete calibration was already completed with success. The useful behind this is that if, after a successful complete calibration, for some reasons finger physical positions result to be “shifted” from absolute ones (e.g., finger completely flexed does not correspond anymore to absolute position 255), a fast calibration removes the need to issue another complete calibration, which is slower.

NOTE: Mia Hand stores in the EEPROM the current position of all DoAs every 1 (one) second. This allows the hand to know the status of the fingers even after it is restarted. This removes the need to issue calibration routines frequently. On the other hand, if the Mia Hand is moving in the moment when it is switched OFF, it will probably require re-calibration at the following startup.

NOTE: fast and complete calibration routines can be issued also by pressing and holding the button placed on the back of the hand. If the button is pressed for 1 s, a blue LED is switched on. Then, if the button is released, a fast calibration is issued. If, instead, the button is pressed for 3 s, the LED switches from blue to green and a complete calibration is issued as soon as the button is released.

2.3.2 Automatic grasps

Mia Hand can perform grasps in an automatic, stereotypical (pre-programmed) manner. These grasps are modelled on the natural hand’s ones and are triggered by simple commands. The original package includes five prehension patterns (Figure 3):

- ‘C’: cylindrical grasp
- ‘P’: pinch (i.e. bi-digital) grasp
- ‘L’: lateral grasp
- ‘S’: spherical grasp
- ‘T’: tridigital grasp

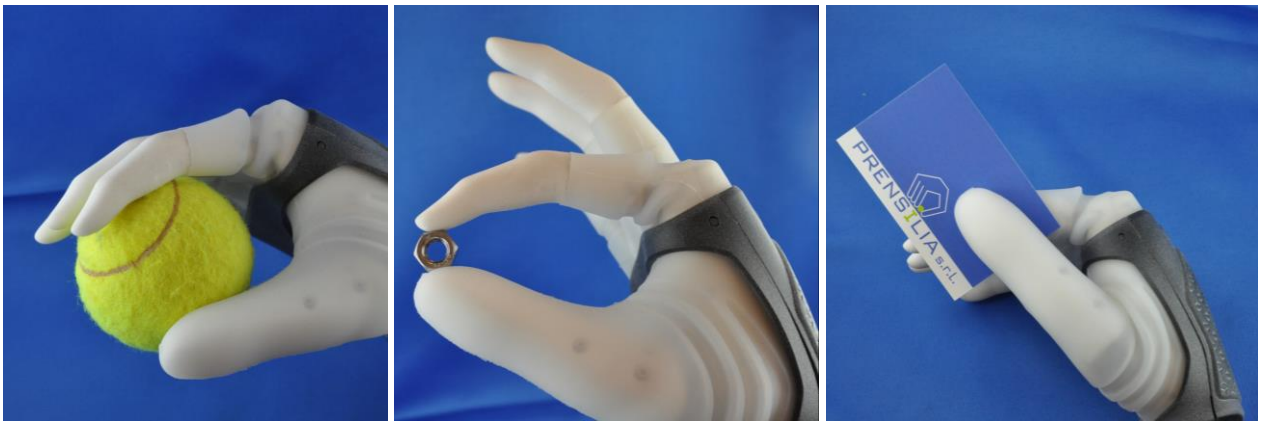


Figure 3 The cylindrical, pinch and lateral grasps.

Each grasp is defined through an “open” REST position (which may differ, for each motor, from the actual maximum opening position) and a “closed” POS position (which may differ, for each motor, from the actual maximum closing position), defined for each of the three motors. When all parameters are set, it is possible to correctly perform a grasp issuing the “GRASP” command.

The HOLDOFF parameter is used to ensure proper coordination between different fingers (Figure 4). Indeed, it delays the movement of the target motor of a percentage of the total time needed to perform the grasp. This can be calculated as:

$$t_{delay} = \frac{HOLDOFF \cdot t_{grasp}}{100}$$

Two grasp modes are available:

- The automatic mode: issuing a single command, Mia Hand performs a complete grasp
- The manual mode: issuing a single command, Mia Hand performs only a specified percentage (called grasp-step) of the movement needed to complete the grasp. This mode is useful to finely control the position of the fingers during a stereotypical grasp.

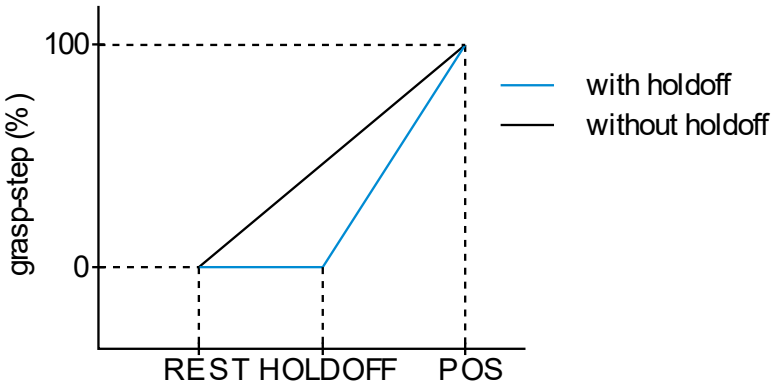


Figure 4 Different behaviour of finger trajectory with and without holdoff during an automatic grasp

NOTE: grasps names are assigned by convention only. It is indeed possible to perform any movement with any grasp, by properly setting the REST and POS positions. This specific naming

allows to easily remember the characteristics of each grasp, compared to a generic counter-based naming (e.g. grasp1, grasp2, etc.).

2.3.3 EMG Decoder[4]

Mia Hand can be controlled through a pair of EMG electrodes (EMG1 and EMG2). The EMG decoder uses the same control technique of the GRASP in manual mode, automatically calculating the grasp-step. However, differently from it, grasp steps range from 0 to 250. The EMG decoder working principle can be summarized as follows:

- If EMG1 is above a threshold for a certain period of time, grasp-step is increased (Mia Hand closes).
- If EMG2 is above a threshold for a certain period of time, grasp-step is decreased (Mia Hand opens).
- When the hand is completely open, if EMG2 is greater than a threshold for more than 2 s, another grasp is selected. Three grasps are currently available within the EMG decoder, in the following order: Cylindrical, Pinch, Lateral.

When the EMG decoder is active, the currently selected grasp is identified by a specific LED colour. Cylindrical, Pinch and Lateral grasps are, respectively, signalled by a green, blue and red LED.

The speed used to close or open the hand depends on the difference between the EMG signals and their respective threshold, and on a gain. In particular, the time to perform the grasp (or to open the hand) is set to:

$$t = 100 - \frac{(EMG - TH)K}{16}$$

Where:

- t: is the resulting time expressed in ms
- EMG: is one of the analog inputs (EMG1 or EMG2), ranging from 0 to 1024.
- TH: is the (opening or closing) threshold specified in the command
- K: is the EMG gain

When the hand is close to the open position (grasp-step lower than 20), and both EMG1 and EMG2 are lower than their respective thresholds, Mia Hand moves to a relaxed pose. As soon as EMG1 or EMG2 overcome their respective threshold, the hand moves to the calculated grasp-step of the last selected grasp.

2.4 Communication protocol overview

Mia Hand can be controlled from a host PC (or any other controlling system able to communicate on the same bus) in the form of sequences of ASCII characters called packets. The communication is carried over a serial bus, accessible by means of the USB cable provided (see Chapter 3 – Installation). Consecutive ASCII characters sent to Mia Hand are temporarily stored by the MC in a 32-bytes buffer, following a FIFO logic. Thus, after sending the first 32 characters, each newly sent character will overwrite the oldest one sent. Moreover, this command buffer is flushed when Mia Hand is switched OFF. So, if the hand is restarted during a communication, all the previously sent characters are lost.

Mia Hand will process only packets recognized as valid. A packet is recognized as valid if it is 18 characters long, starts with "@" and ends with "*\r", where "\r" denotes the CR character. As better described in Chapter 4 (Communication protocol explained), after receiving a valid package, the hand replies with an acknowledgement message. This happens even when the valid package does not contain a valid command. If so, the hand will only acknowledge the valid package, doing nothing more. Invalid packets, instead, are simply ignored.

2.4.1 Serial communication settings

The serial communication settings required for communicating with Mia Hand are listed in Table 1.

[4] This functionality is not available in all hands.

Table 1 Asynchronous Serial communication specifications

Serial wires used	Tx, Rx, GND
Baud Rate	115200 bit/s
No. bits	8
Stop bits	1
Parity	None
Voltage levels	TTL (0, 3V3)

3 Installation

The prototype you purchased includes a wrist that can be used to mechanically connect the hand with external devices, like robot arms. Besides mechanical connection (through three threaded holes), the wrist is also equipped with two connectors providing electrical connection to the hand. Details of such interface are reported in Chapter 8 – Mechanical interface.

NOTE: in the case of connection on a robot arm pay particular attention to the wiring from/to the hand.

A typical configuration for operating the Mia Hand requires (Figure 5):

- A host PC for controlling the hand following the communication protocols detailed in this document;
- A bench power supplier with fixed regulated output (8.5 V) and a peak current of 5 A (not provided);
- A TTL to USB Serial converter cable (provided)[5] hereafter called USB cable
- Adaptors cables to connect the hand to the power supply and to the USB cable (provided).

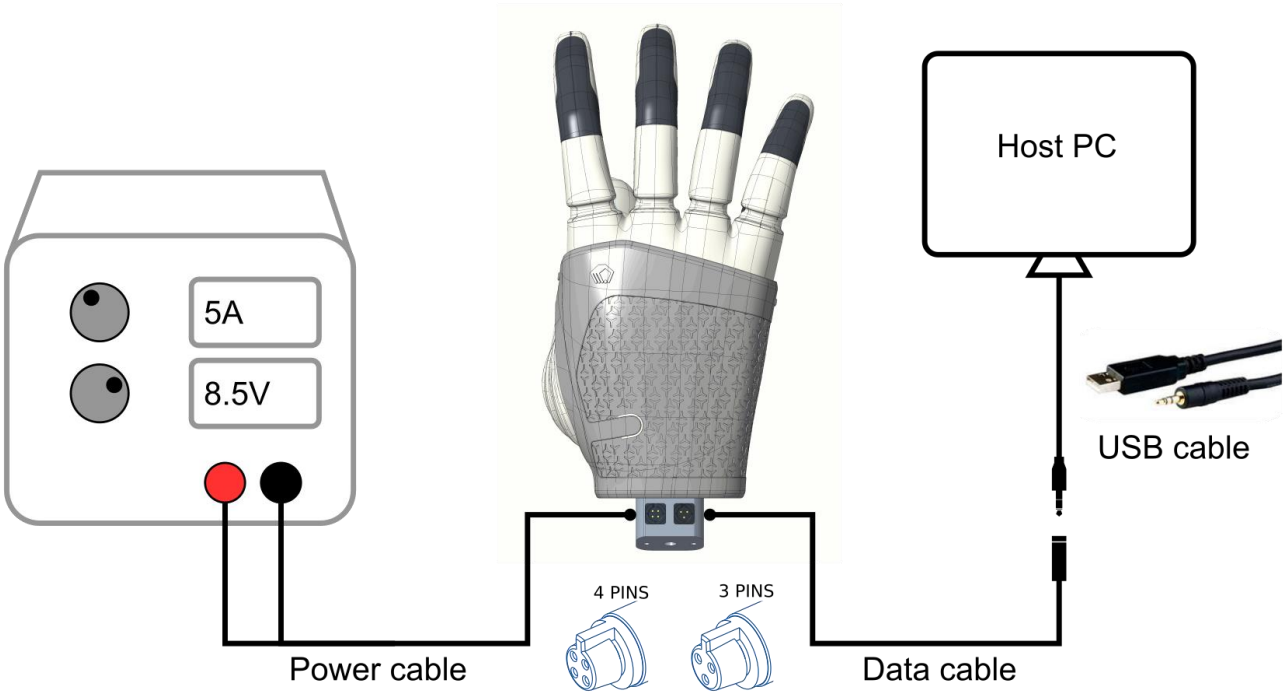


Figure 5 Connection scheme between Mia Hand, USB cable, host PC and external power supply.

DANGER: failing to properly set the power supply with the correct output parameters may result in a fire hazard. Always double check that the power supply is setup to provide the correct voltage and current levels to Mia Hand.

[5] Manufacturer (FTDI) code: TTL-232R-3V3-AJ (https://ftdichip.com/products/ttl-232r-3v3-aj/). Spare parts available for purchase on main electronics distributors (e.g. Farnell, RS) or Prensilia SRL.

3.1 Install the USB cable

To operate the hand, you first need to install the USB cable drivers on your host PC. Drivers for the USB cable are freely available for all major operating systems (Windows, MacOS, Linux) on the manufacturer website[6] or from Prensilia. They are necessary to make the USB cable (and so Mia Hand) appear as a virtual COM port. USB cable installation guides are freely available here[7].

NOTE: we recommend to install the drivers manually before plugging the USB cable in the host PC (this avoids Windows live update searching for the drivers automatically). If communication problems arise once the drivers are installed, please contact us at support@prensilia.com.

The name/number assigned COM ports is specific to the operating system in use. The easiest way to find out which COM port is being assigned to Mia Hand is to check which COM port appears when the USB cable is plugged in (provided the drivers have been already correctly installed on the PC). You can determine and setup the COM Port on a PC running Windows OSs by following these steps:

1. Open Device Manager. To do so, click: Start ► Control Panel ► Device Manager
2. Under the list of devices, expand the “Ports” and check if any new COM port is listed after plugging in the USB cable. This is usually in the following form: USB Serial Port (COM #), where # is the COM port number.
3. Setup the COM port correctly. Right click on the COM port and then click Properties ► Port Settings ► Advanced. Then, set “Latency Timer (msec)” to 1 (one). Click OK twice to save the new settings.

The last step may or may not be needed on other operating systems.

WARNING: Failing to properly set up the COM port as described above may result in an unresponsive device and/or unexpected behaviour.

3.2 Connect the cables and power ON the hand

Once the USB cable drivers are successfully installed, Mia Hand can be controlled following these steps:

1. Check that the external power supply voltage and maximum current match the recommended values specified above;
2. Connect the USB cable to the host PC;
3. Connect the USB cable to Mia Hand through the adaptor cable;
4. Connect the black/red cable (provided) to the external power supply with regulated voltage output (+8.5V and GND), and to the power connector on Mia Hand;
5. Switch ON the power supply;
6. The hand is now operative and ready to communicate with the host PC.

NOTE: The USB cable and the power supply cable can be connected in any order. However, if the USB cable is connected after the hand is switched ON, some spurious signals could reach the hand through the serial port while physically connecting the plug. This could potentially issue a command or part of it and could leave in the host input queue spurious bytes as well, causing

[6] Third party virtual COM port drivers: https://ftdichip.com/drivers/vcp-drivers/
[7] Third party installation guide: https://ftdichip.com/document/installation-guides/

communication problems (if not correctly handled). It is suggested to switch ON the hand after plugging in the USB connection.

WARNING: Do not touch the metallic parts of the connectors (which are both connected to the system ground) once plugged in. The hand is a Static Sensitive Device!

4 Communication protocol explained

The Mia Hand communication protocol is based on human-readable ASCII commands, to simplify the debugging of your application. For your convenience, an ASCII code chart reporting the correspondence between ASCII characters and HEX value is provided in Chapter 9 – ASCII code chart.

More in details, the communication is based on 18-bytes ASCII messages structured as follows:

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	<*destination*>	<*command*>	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0	'*'	CR

For each correct message received, the device sends an acknowledgement reply with the same message, replacing '@' with '<' and CR with an LF character ('\n').

The field <*destination*> indicates the component to which the command is directed to, and can be:

- '**1**': motor 1 (thumb flexion/extension)
- '**2**': motor 2 (middle, ring and little – indicated as MRL – fingers flexion/extension)
- '**3**': motor 3 (index flexion/extension and thumb opposition)
- '**A**': entire (ALL) hand
- '**E**': EEPROM (i.e., data storage)
- '**S**': general system

The field <*command*> depends on <*destination*>:

4.1 Commands with destination '1', '2' or '3' – motors:

4.1.1 'P': SET TARGET POSITION – position control of the target motor

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'1'	**'P'**	'+'	TARGET POS				PWM		IGNORED						'*'	CR
		'2'		'-'														
		'3'																

Byte 0 = '@' : header character
Byte 1 : target motor (admits: '1', '2', '3')
Byte 2 = 'P' : command is "SET TARGET POSITION"
Byte 3 : sign of the selected target position (admits: '+', '-'. Negative positions are only allowed for motor '3', i.e., index finger)
Byte 4-7 : target position in ASCII (admits from '0000' to '0255').
Byte 8-9 : max PWM duty cycle (DC) during the motion (admits from '00' to '99')
Byte 10-15 : Ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

Example @1P+025050000000*\r
Description Motor 1 (thumb) moves to position 250 with maximum PWM set to 50

After sending the command, the target motor moves to the requested position. During the motion, the PID controller applies a PWM DC that saturates to the threshold specified in the command (bytes 8-9).

Notes on the '**P**' command:

- This command is executed only if the hand is calibrated (see commands '**K**': COMPLETE CALIBRATION and '**F**': FAST CALIBRATION).

- Different types of movements towards the target position are performed, depending on the temporal delay between the last and the previous '**P**' command. If this delay is smaller than 2.5 s, the movement is divided into small steps. This is called *stepper motion*. The movement is thus performed through a fixed number of smaller steps. This allows to obtain a lower, but more controlled, motion speed, with respect to a direct motion towards the final position. Under the stepper motion, in fact, the final position is reached in a time equal to the delay between consecutive '**P**' commands. If the delay is, instead, larger than 2.5 s, the finger performs a direct movement (*direct motion*): the target position specified in the '**P**' command is directly set as the PID target position, without intermediate steps. Thus, the motor reaches it with an average higher speed than during the stepper motion.
- As a safety feature, both stepper and direct motion are subject to a watchdog timer that starts when the hand acknowledges a '**P**' command. Specifically, if the motor does not reach the target position within a specific time, it is stopped. For the stepper motion, the time limit corresponds to the delay from the previous '**P**' command. For the direct motion, instead, the time limit is 2 s.

4.1.2 '**S**': SET TARGET SPEED - speed control of the target motor

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'1' '2' '3'	**'S'**	'+' '-'	IGNORED				SPEED		PWM		IGNORED				'*'	CR

Byte 0 = '@' : header character
Byte 1 : target motor (admits: '1', '2', '3')
Byte 2 = 'S' : command is "SET TARGET SPEED"
Byte 3 : sign of the selected target speed (admits: '+', '-')
Byte 4-7 : Ignored
Byte 8-9 : target speed in ASCII (admits from '00' to '99'). The measurement unit is encoder counts per 16 ms
Byte 10-11 : max PWM duty cycle (DC) during the motion (admits from '00' to '99')
Byte 12-15 : Ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

Example @1S+000050750000*\r
Description Motor 1 (thumb) closes with speed 50 and with maximum PWM set to 75

As a result of the command, the target motor is moved with the specified speed (if allowed by the maximum PWM set through bytes 10-11). As for the '**P**' command, also the '**S**' command is subject to a watchdog timer. That is, the movement is stopped 2 s after the acknowledgement is sent. If continuous motion is desired, another '**S**' command should be sent before the watchdog is activated.

4.1.3 '**K**': SET POSITION PID GAINS – modify the gains of the PID position controller of a target motor

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'1'	**'K'**	'+'	Kp		'+'	Ki		'+'	Kd		IGNORED				'*'	CR
		'2'		'-'			'-'			'-'								
		'3'																

Byte 0 = '@' : header character
Byte 1 : target motor (admits: '1', '2', '3')
Byte 2 = 'K' : command is "SET POSITION PID GAINS"
Byte 3 : sign of Kp gain (admits: '+', '-')
Byte 4, 5 : Kp gain expressed in ASCII (admits from '00' to '99')
Byte 6 : sign of Ki gain (admits: '+', '-')
Byte 7, 8 : Ki gain expressed in ASCII (admits from '00' to '99')
Byte 9 : sign of Kd gain (admits: '+', '-')
Byte 10, 11 : Kd gain expressed in ASCII (admits from '00' to '99')
Byte 12-15 : ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

As a result of the command, the PID gains used in the position controller of the target motor are overwritten with the specified values. Please note that the PID gains set with this command are discarded when the hand is switched OFF, unless they are stored in the EEPROM with the command "SAVE PARAMETERS TO EEPROM".

4.1.4 'k': READ POSITION PID GAINS – read current gains of the PID position controller of the selected motor

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'1'	**'k'**	IGNORED													'*'	CR
		'2'																
		'3'																

Byte 0 = '@' : header character
Byte 1 : target motor (admits: '1', '2', '3')
Byte 2 = 'k' : command is "READ POSITION PID GAINS"
Byte 3-15 : ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

When the "READ POSITION PID GAINS" command is received, the hand replies with the following message:

Byte	0 … 6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	'Ppid : '	'+' / '-'	Kp		' ; '			'+' / '-'	Ki		' ; '			'+' / '-'	Kd		LF

4.1.5 'H': SET SPEED PID GAINS - modify the gains of the PID speed controller of a target motor

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'1'	**'H'**	'+'	Kp		'+'	Ki		'+'	Kd		IGNORED				'*'	CR
		'2'		'-'			'-'			'-'								
		'3'																

Byte 0 = '@' : header character
Byte 1 : target motor (admits: '1', '2', '3')
Byte 2 = 'H' : command is "SET SPEED PID GAINS"

Byte 3 : sign of Kp gain (admits: '+', '-')
Byte 4, 5 : Kp gain expressed in ASCII (admits from '00' to '99')
Byte 6 : sign of Ki gain (admits: '+', '-')
Byte 7, 8 : Ki gain expressed in ASCII (admits from '00' to '99')
Byte 9 : sign of Kd gain (admits: '+', '-')
Byte 10, 11 : Kd gain expressed in ASCII (admits from '00' to '99')
Byte 12-15 : ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

As a result of the command, the PID gains used in the speed control of the target motor group are overwritten with the specified values. Please note that the PID gains set with this command are discarded when the hand is switched off, unless they are stored in the EEPROM with the command "SAVE PARAMETERS TO EEPROM".

4.1.6 'h': READ SPEED PID GAINS – read current gains of the PID speed controller of the selected motor

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'1'	**'h'**	IGNORED													'*'	CR
		'2'																
		'3'																

Byte 0 = '@' : header character
Byte 1 : target motor (admits: '1', '2', '3')
Byte 2 = 'h' : command is "READ POSITION PID GAINS"
Byte 3-15 : ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

When the "READ SPEED PID GAINS" command is received, the hand replies with the following message:

Byte	0 … 6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	'Vpid : '	'+' '-'	Kp		' ; '			'+' '-'	Ki		' ; '			'+' '-'	Kd		LF

4.1.7 'G': SET GRASP PARAMETERS - set new grasp parameters for the specified motor

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'1' '2' '3'	**'G'**	'C' 'P' 'L' 'S' 'T'	'+' '-'	REF REST			'+' '-'	REF POS			I	HOLDOFF			'*'	CR

Byte 0 = '@' : header character
Byte 1 : target motor (admits: '1', '2', '3')
Byte 2 = 'G' : command is "SET GRASP PARAMETERS"

Byte 3 : grasp to be modified:

- ‘C’: cylindrical grasp
- ‘P’: pinch grasp
- ‘L’: lateral grasp
- ‘S’: spherical grasp
- ‘T’: tridigital grasp

Byte 4 : sign of REST (open) reference position (admits ‘+’ or ‘-’. Negative positions are only allowed for motor ‘3’, i.e., index finger)

Byte 5-7 : REST reference position (admits from ’000’ to ‘255’)

Byte 8 : sign of POS (close) reference position (admits ‘+’ or ‘-’. Negative positions are only allowed for motor ‘3’, i.e., index finger)

Byte 9-11 : POS reference position (admits from ‘000’ to ‘255’)

Byte 12 : ignored

Byte 13-15 : HOLDOFF value (admits from ‘000’ to ‘100’)

Byte 16-17 : tail characters (‘*’ and CR [#ord: 0x0d])

To completely define the parameters for a single grasp, this command should be issued three times, each time targeting a different motor.

4.1.8 ‘**g**’: READ GRASP PARAMETERS – read the parameters of the selected automatic grasp

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘1’ ‘2’ ‘3’	**‘g’**	‘C’ ‘P’ ‘L’ ‘S’ ‘T’	IGNORED												‘*’	CR

Byte 0 = ‘@’ : header character

Byte 1 : target motor (admits: ‘1’, ‘2’, ‘3’)

Byte 2 = ‘g’ : command is “READ GRASP PARAMETERS”

Byte 3 : grasp to be read from:

- ‘C’: cylindrical grasp
- ‘P’: pinch grasp
- ‘L’: lateral grasp
- ‘S’: spherical grasp
- ‘T’: tridigital grasp

Byte 4-15 : ignored

Byte 16-17 : tail characters (‘*’ and CR [#ord: 0x0d])

When the “READ GRASP PARAMETERS” command is received, the hand replies with the following message:

Byte	0 … 4	5	6	7 … 9	10 … 13	14 … 16	17 … 20	21 … 23	24 … 27	28
	‘Grasp’	‘1’ ‘2’ ‘3’	‘C’ ‘P’ ‘L’ ‘S’ ‘T’	‘ : ’	REF REST	‘ ; ’	REF POS	‘ ; ‘	HOLDOFF	LF

4.2 Commands with destination ‘A’ – entire hand:

4.2.1 ‘**E**’: ENCODER RESET– resets the relative encoder counts

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘A’	**‘E’**	IGNORED													‘*’	CR

This command sets the relative encoder counters to 0 (i.e., all digits open) for all motors. As a result, all commands based on the PID position controller (“SET TARGET POSITION”, “GRASP” and EMG decoder) could not work as expected before a new COMPLETE CALIBRATION is performed.

4.2.2 ‘**K**’: COMPLETE CALIBRATION – performs a complete calibration routine

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘A’	**‘K’**	IGNORED													‘*’	CR

The command starts a complete calibration routine. A complete calibration is necessary to map the relative encoder counts to absolute finger positions. As described above:

- completely flexed digits are mapped to the absolute position +255.
- completely extended digits are mapped to the absolute position 0.
- completely flexed index finger (with thumb abduced) is mapped to the absolute position -255.

These mappings are stored in the EEPROM after a successful complete calibration. For this reason, once the hand is correctly calibrated, it is not necessary to re-calibrate it at the next power ON. If, instead, any failure occurs during the execution of a complete calibration (e.g., in the case an object impedes the movements of the fingers), a red LED turns on. In this case, commands involving the position PID controller (“SET TARGET POSITION”, “GRASP” and EMG decoder) are disabled until a successful complete calibration is completed.

4.2.3 ‘**k**’: STOP COMPLETE CALIBRATION – stops the complete calibration routine

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘A’	**‘k’**	IGNORED													‘*’	CR

The command immediately stops the complete calibration routine. An issued calibration stop is treated like a failed calibration. Thus, commands involving the position PID controller (“SET TARGET POSITION”, “GRASP” and EMG decoder) are disabled (only commands involving the speed PID controller can be used), and the red LED turns on.

4.2.4 ‘**F**’: FAST CALIBRATION – performs a fast calibration routine

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘A’	**‘F’**	IGNORED													‘*’	CR

This command starts a fast calibration routine. When sending a fast calibration command, if the last complete calibration was not completed successfully, the fast calibration is not executed.

If any failure occurs during the fast calibration, a red LED turns on. In this case, commands involving the position PID controller ("SET TARGET POSITION", "GRASP" and EMG decoder) are disabled until a successful fast calibration is completed.

4.2.5 '**G**': GRASP – performs a grasp

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'A'	**'G'**	'C'	'M'	STEP / TIME			PWM		IGNORED						'*'	CR
				'L'	'A'													
				'P'	'a'													
				'S'														
				'T'														

Byte 0 = '@' : header character
Byte 1 = 'A' : destination of the command is the entire hand
Byte 2 = 'G' : command is "GRASP"
Byte 3 : grasp to perform:
- 'C': cylindrical grasp
- 'P': pinch grasp
- 'L': lateral grasp
- 'S': spherical grasp
- 'T': tridigital grasp

Byte 4 : grasp mode:
- 'M': manual
- 'A': auto-close
- 'a': auto-open

Byte 5-7 : 'grasp-step' (if in manual mode) or 'grasp-time' (if in auto-close or auto-open mode) expressed in ASCII (from '000' to '099' for 'grasp-step' or from '000' to '999' for 'grasp-time')
Byte 8, 9 : maximum value of the PWM duty cycle applied during the motion (admits from '00' to '99')
Byte 10-15 : ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

Example1 *@AGCA10050/////*\r*
Description1 *Close with a cylindrical grasp in 1 s, with a maximum PWM duty cycle of 50%*
Example2 *@AGPM04045++++++*\r*
Description2 *Reach 40% of pinch grasp, with a maximum PWM duty cycle of 45%*

The "GRASP" command moves the fingers between REST and POS positions, whose values can be set with the "SET GRASP PARAMETERS" command. When the grasp mode is set to manual (byte 4 = 'M'), the motors are moved to an intermediate position between REST and POS, i.e., to grasp step (bytes 5-7), where:

- '000' corresponds to the rest position (REST)
- '099' corresponds to the closed position (POS)

When the grasp mode is set to auto-close (byte 4 = 'A'), each motor moves towards the POS position through 256 steps, automatically. In other words, a single GRASP command allows to perform a complete grasp. The complete grasp is executed in a time equal to grasp-step (expressed in 10 ms units). When the grasp mode is set to auto-open (byte 4 = 'a') each motor moves towards the REST position with the same modality described for "auto-close".

4.2.6 ‘**g**’: SET EMG DECODER – sets the parameters used by the EMG decoder

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘A’	‘**g**’	‘1’	THopen			THclose			PWM		HOLDOFF		K		‘*’	CR
				‘0’														

Byte 0 = ‘@’ : header character
Byte 1 = ‘A’ : destination of the command is the entire hand
Byte 2 = ‘g’ : command is “SET EMG DECODER”
Byte 3 : enable (‘1’) or disable (‘0’) the EMG decoder
Byte 4-6 : Opening threshold, from ‘000’ to ‘999’. It indicates the EMG threshold to be overcome to open Mia Hand.
Byte 7-9 : Closing threshold, from ‘000’ to ‘999’. It indicates the EMG threshold to be overcome to close Mia Hand.
Byte 10, 11 : maximum value of the PWM applied during the movement, expressed in ASCII (admits from ‘00’ to ‘99’)
Byte 12, 13 : HOLDOFF time (in 10 ms units). The decoder starts the execution of the movement only if the input is maintained active for a time longer than the HOLDOFF time.
Byte 14, 15 : gain factor used for calculation of fingers speed from EMG inputs (admits from ‘00’ to ‘99’)
Byte 16-17 : tail characters (‘*’ and CR [#ord: 0x0d])

Example *@Ag1200300600822*\r*
Description *Start EMG decoder with 200 as opening threshold, 300 as closing threshold, 60% maximum PWM duty cycle, 80 ms initial delay (holdoff) and a gain factor of 22.*

NOTE: when the EMG decoder is being disabled (i.e. byte 3 = ‘0’), the values specified for the other parameters are ignored.

4.2.7 ‘**D**’: STREAMING MANAGEMENT – enables/disables the streaming (transmission) of a specific data group

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘A’	‘**D**’	‘P’	‘1’	IGNORED											‘*’	CR
				‘S’	‘0’													
				‘C’														
				‘A’														
				‘I’														
				‘E’														
				‘B’														

Byte 0 = ‘@’ : header character
Byte 1 = ‘A’ : destination of the command is the entire hand
Byte 2 = ‘D’ : command is “STREAMING MANAGEMENT”
Byte 3 : type of streaming to enable/disable:

- ‘P’: finger positions
- ‘S’: motors speed
- ‘C’: motors current
- ‘A’: analog inputs
- ‘I’: general states
- ‘E’: EMG decoder
- ‘B’: finger positions, motors current and force sensors **in binary format**

Byte 4 : enable (‘1’) or disable (‘0’) the selected streaming
Byte 5-15 : ignored
Byte 16-17 : tail characters (‘*’ and CR [#ord: 0x0d])

Enabling a specific streaming type will start a flow of data from the hand. Additional details on the data streaming are provided at the end of this document.

4.2.8 ‘**d**’: STOP STREAMING – stops all streaming messages

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘A’	‘**d**’	IGNORED													‘*’	CR

The command interrupts *all* active streaming messages.

4.3 Commands with destination ‘E’ – EEPROM:

4.3.1 ‘**S**’: SAVE PARAMETERS TO EEPROM – stores all hand parameters in the EEPROM

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘E’	‘**S**’	IGNORED													‘*’	CR

As a result of the command, the following parameters are stored in EEPROM:

- Start-up parameters
- PID gains (position and speed)
- Grasp parameters (for all the grasp types)
- EMG decoder parameters

It should be recalled that this command is necessary to store permanently new values assigned to any of those parameters, which otherwise would be lost when switching off Mia Hand.

WARNING: when issued, this command stores all the current values of the parameters listed above to the EEPROM. Please check whether those parameters contain appropriate values.

4.3.2 ‘**s**’: RESTORE DEFAULT PARAMETERS

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘E’	‘**s**’	IGNORED													‘*’	CR

As a result of this command, all the parameters specified in the SAVE PARAMETERS TO EEPROM command are reset to their default values and saved in the EEPROM. The parameters default values are listed at the end of this document.

4.4 Commands with destination 'S' – General System:

4.4.1 'R': READ FW VERSION – reads firmware version

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'S'	'R'	IGNORED													'*'	CR

The Mia Hand replies with the version of the running firmware, in the following format:

Byte	0 … 2	3 … 7	8	9 … 11	12 … 16	17
	'M: '	<versionMaster>	' '	'S: '	<versionSlave>	LF

Example M: 0.1.2 S: 3.4.5\r
Description Master version is 0.1.2, slave version is 3.4.5

4.4.2 'B': SET START-UP PARAMETERS – manages calibration and EMG decoder at start-up

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'S'	'B'	IGNORED											'0'	'0'	'*'	CR
															'1'	'1'		

Byte 0 = '@' : header character
Byte 1 = 'S' : destination of the command is the general system
Byte 2 = 'B' : command is "SET START-UP PARAMETERS"
Byte 3-13 : ignored
Byte 14 : enable ('1') or disable ('0') the EMG decoder at start-up
Byte 15 : enable ('1') or disable ('0') the complete calibration at start-up
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

If both the EMG decoder and calibration routine are enabled at startup, the calibration is executed first and then the EMG decoder is activated. Please note that the parameters set with this command are discarded when the hand is switched OFF, unless they are stored in the EEPROM with the command "SAVE PARAMETERS TO EEPROM".

4.4.3 'b': READ START-UP PARAMETERS – reads start-up calibration and EMG decoder status

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'@'	'S'	'b'	IGNORED													'*'	CR

Byte 0 = '@' : header character
Byte 1 = 'S' : destination of the command is the general system
Byte 2 = 'b' : command is "READ START-UP PARAMETERS"
Byte 3-15 : ignored
Byte 16-17 : tail characters ('*' and CR [#ord: 0x0d])

After receiving the command, the Mia Hand replies with the following message:

Byte	0 … 12	13	14	15
	‘Boot : 000000’	‘0’	‘0’	LF
		‘1’	‘1’	

Byte 13 : EMG decoder is enabled (‘1’) or disabled (‘0’) at start-up
Byte 14 : complete calibration is enabled (‘1’) or disabled (‘0’) at start-up

4.4.4 **‘C’**: READ GRASP COUNTERS – reads the grasp counters

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	‘@’	‘S’	‘C’	IGNORED													‘*’	CR

Byte 0 = ‘@’ : header character
Byte 1 = ‘S’ : destination of the command is the general system
Byte 2 = ‘C’ : command is “READ GRASP COUNTERS”
Byte 3-15 : ignored
Byte 16-17 : tail characters (‘*’ and CR [#ord: 0x0d])

After receiving the command, the following message is sent back:

Byte	0…10	11…16	17…19	20...25	26…28	29 … 34
	‘EMGCount : ’	CylHigh	‘ ; ’	PinchHigh	‘ ; ’	LatHigh

Byte	35…37	38…43	44…46	47…52	53…55	56…61
	‘ ; ’	CylMed	‘ ; ’	PinchMed	‘ ; ’	LatMed

Byte	62…64	65…70	71…73	74...79	80…82	83...87
	‘ ; ’	CylLow…	‘ ; ’	PinchLow	‘ ; ’	LatLow

Byte	88
	LF

Where:

- CylHigh, CylMed, CylLow: counters of the cylindrical grasps performed with high, medium, and low torque, respectively.
- PinchHigh, PinchMed, PinchLow: counters of the pinch grasps performed with high, medium, and low torque, respectively.
- LatHigh, LatMed, LatLow: counters of the lateral grasps performed with high, medium, and low torque, respectively.

4.4.5 **‘c’**: RESET GRASP COUNTERS – resets the grasp counters

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

	‘@’	‘S’	‘c’	IGNORED	‘*’	CR

Byte 0 = ‘@’ : header character
Byte 1 = ‘S’ : destination of the command is the general system
Byte 2 = ‘c’ : command is “RESET GRASP COUNTERS”
Byte 3-15 : ignored
Byte 16-17 : tail characters (‘*’ and CR [#ord: 0x0d])

As a result of the command, all grasp counters are reset to zero.

5 Data streaming

The command "STREAMING MANAGEMENT" (described above) can enable the streaming of up to 7 data groups. These are:

- 'P': Finger positions
- 'S': Motor speeds
- 'C': Motor currents
- 'A': Analog signals
- 'I': General states
- 'E': EMG decoder state
- 'B': Finger positions, motor currents and finger forces (note: this is a binary streaming)

For all streaming types except 'B', the streaming period is 10 ms (100 Hz). More than one type can be enabled simultaneously, but this will affect the streaming period of each data group. In particular, if *n* streaming types are enabled, a single data group will be streamed every 10 ms. This means that, while the Mia Hand will stream information every 10 ms, a specific data group will be streamed every $(n * 10)$ ms.

The 'B' streaming type has to be considered separately. This streaming type differs from the others as it has a different streaming period and different format. The streaming period is 35 ms, and the format is binary (as opposed to ASCII of the other types).

NOTE: the 'B' streaming type is not compatible with the other streaming types. Thus, it should be enabled when all other streaming types are disabled.

5.1 'P' – Finger positions

Byte	0 ... 5	6 … 11	12 … 14	15 … 20	21 … 23	24 ... 29	30 … 32	33 … 38	39
	'enc : '	pos[0]	' ; '	pos[1]	' ; '	pos[2]	' ; '	stream_count	LF

pos[0] : position of the thumb, between 0 (thumb extended) and 255 (thumb flexed)
pos[1] : position of the MRL fingers, between 0 (fingers extended) and 255 (fingers flexed)
pos[2] : position of the index, between -255 (index flexed, thumb abducted) and 255 (index flexed, thumb adducted)
stream_count : number of data groups previously streamed

Example 1 *enc : +00255 ; +00000 ; -00127 ; +00005\n*
Description 1 *Thumb completely flexed, MRL completely extended, index half-flexed with thumb abducted, 5th package*
Example 2 *enc : +00255 ; +00000 ; +00127 ; +00020\n*
Description 2 *Thumb completely flexed, MRL completely extended, index half-flexed with thumb adducted, 20th package*

5.2 'S' - Motor speed

Byte	0 ... 5	6 … 11	12 … 14	15 … 20	21 … 23	24 ... 29	30 … 32	33 … 38	39
	'spe : '	speed[0]	' ; '	speed[1]	' ; '	speed[2]	' ; '	stream count	LF

speed[0] : thumb motor speed

speed[1] : MRL motor speed
speed[2] : index motor speed
stream_count : number of data groups previously streamed

Example 1 *spe : -00020 ; -00045 ; -00012 ; +00128\n*
Description 1 *All fingers extending, 128th package*
Example 2 *spe : +00020 ; +00045 ; +00012 ; +00058\n*
Description 2 *All fingers flexing, 58 th package*

5.3 'C' - Motor current

Byte	0 ... 5	6 … 11	12 … 14	15 … 20	21 … 23	24 ... 29	30 … 32	33 … 38	39
	'cur : '	current[0]	' ; '	current[1]	' ; '	current[2]	' ; '	stream count	LF

current[0] : thumb motor current
current[1] : MRL motor current
current[2] : index motor current
stream_count : number of data groups previously streamed

Example 1 *cur : +00583 ; +00021 ; +00075 ; +00042\n*

The actual motor current (I_{mot}, in A) provided to the motors can be calculated as:

$$I_{mot}[i] = \frac{current[i]}{750}$$

5.4 'A' – Analog signals

Byte	0 ... 5	6 … 11	12 …14	15 ... 20	21 … 23	24 … 29
	'adc : '	Force_sensor[0]	' ; '	Force_sensor[1]	' ; '	Force_sensor[2]

Byte	30 … 32	33 ... 38	39 … 41	42 … 47	48 … 50	51 … 56
	' ; '	Force_sensor[3]	' ; '	Force_sensor[4]	' ; '	Force_sensor[5]

Byte	57 … 59	60 ... 65	66 … 68	66 … 71	72 … 74	75 … 80	81
	' ; '	HV	' ; '	Vin_level	' ; '	stream_count	LF

Force_sensor[i] : output voltage from the *i*-th force sensor. The mapping between output signals and physical forces is as follows:

- Force_sensor[0] → Middle tangential force
- Force_sensor[1] → Index normal force
- Force_sensor[2] → Index tangential force
- Force_sensor[3] → Thumb tangential force
- Force_sensor[4] → Thumb normal force
- Force_sensor[5] → Middle normal force

HV : voltage provided to the motors
Vin_level : Power supply voltage
stream_count : number of data groups previously streamed

Example *adc : +00824 ; +00235 ; +00128 ; +00459 ; +00500 ; +00920 ; +00924 ; +00539 ; +00023\n*

The actual motor voltage (V_{mot}, in V) can be calculated as:

$$V_{mot} \approx \frac{HV}{77}$$

In normal operating conditions, $V_{mot} = 12\,V$, so $HV \approx 924$.

The power supply voltage (V_{in}, in V) can be calculated in the same way as V_{mot}:

$$V_{in} \approx \frac{Vin\_level}{77}$$

5.5 'I' – General states

Byte	0 … 7	8	9	10
	'Sta : 00'	Stat[0]	FC[0]Open	FC[0]Close

Byte	11 … 16	17	18	19
	'0 ; 00'	Stat[1]	FC[1]Open	FC[1]Close

Byte	20 … 24	25	26	27
	'0 ; 00'	Stat[2]	FC[2]Open	FC[2]Close

Byte	28 … 31	32 … 34	35 … 41	42	43 … 45	46..51	52
	'0 ; '	HandStatus	' ; O ; '	CalibStatus	' ; '	stream_count	LF

Stat[i] : type of control currently active for motor *i*

- 'P': Position control
- 'S': Speed control
- 'H': Motor stopped

FC[*i*]Open : open limit switch status for motor *i*

- '0': limit switch reached
- '1': limit switch not reached

FC[*i*]Close : closed limit switch status for motor *i*

- '0': limit switch reached
- '1': limit switch not reached

HandStatus : status of Mia Hand:

- '+00': standard working conditions
- '+10': calibration currently running
- '+20': EMG decoder enabled

CalibStatus : status of the calibration while it is ongoing. At the end it indicates the outcome of the calibration as:

- '+00': when correctly calibrated
- '-01': if the calibration was stopped
- '-02': if the calibration failed

stream_count : number of data groups previously streamed

Example *Sta : 00H01 ; 00H10 ; 00S110 ; +00 ; O ; +00 ; +00348\n*

Description *Thumb completely extended, index completely flexed, MRL in movement under speed control, standard working conditions, last calibration successful and 348th package*

5.6 'E' – EMG decoder status

Byte	0 … 5	6 ... 11	12 … 14	15 … 20	21 … 23	24
	'emg : '	emgInOpen	' ; '	emgInClose	' ; '	grasp

Byte	25 … 27	28 … 31	32 … 34	35 … 40	41 … 43	44 ... 49
	' ; '	graspStep	' ; '	thOpen	' ; '	thClose

Byte	50 … 52	53 … 58	59
	' ; '	stream_count	LF

emgInOpen : EMG2, analog input used to open the hand

emgInClose : EMG1, analog input used to close the hand

grasp : selected grasp:

- 'C': cylindrical
- 'P': pinch
- 'L': lateral
- 'X': inactive grasp

graspStep : - current "grasp step"

thOpen : - threshold on the opening input (EMG2)

thClose : threshold on the closing input (EMG1)

stream_count : number of data groups previously streamed

Example *emg : +00125 ; +00350 ; C ; +150 ; +00200 ; +00300 ; +00001\n*

Description *EMG2 level at 125, EMG1 level at 350, cylindrical grasp currently performed, grasp step 150, open threshold of 125, close threshold of 350, 1st package*

Please refer to the "SET EMG DECODER" command for more details on the variables of this streaming type.

5.7 'B' – Binary

Byte	0 … 5	6, 7	8, 9	10, 11
	'bin : '	pos[0]	pos[1]	pos[2]

Byte	12, 13	14, 15	16, 17
	current[0]	current[1]	current[2]

Byte	18, 19	20, 21	22, 23	24, 25	26, 27	28, 29
	Force_sensor[0]	Force_sensor[1]	Force_sensor[2]	Force_sensor[3]	Force_sensor[4]	Force_sensor[5]

Byte	30, 31	32

stream_count	LF

pos[*i*] : 16-bit position of the *i*-th motor
current [*i*] : 16-bit motor current of the *i*-th motor
Force_sensor[*i*] : - 16-bit averaged input from the *i*-th force sensor
stream_count : number of data groups previously streamed

6 Demo application

The demo application (available from Prensilia website) is able to issue all the commands Mia Hand is capable of executing, and which are presented in this user guide. The application is compatible with 32 and 64 bit machines with MS Windows installed as the operative system (Windows 7, 8, 10). To install the software, simply double click on the **MiaHandApp Installer.msi** file and follow the installation instructions. Once installed launch the MiaHandApp.exe application from Program Files (x86) ► Prensilia srl ► MiaHandApp or from your Desktop.

Once launched, the main form will open (Figure 6). Input the correct COM port number (follow the instructions in Chapter 3 – Installation to know which COM port is dedicated to Mia Hand) and click “OPEN com”. This will enable all “controls” icons in the centre of the form (excluding DEMO), unlocking all its functionalities.

The main form is also used to display the last command sent to Mia Hand from the application (“out” label) and the latest commands received from Mia Hand (“in” label).

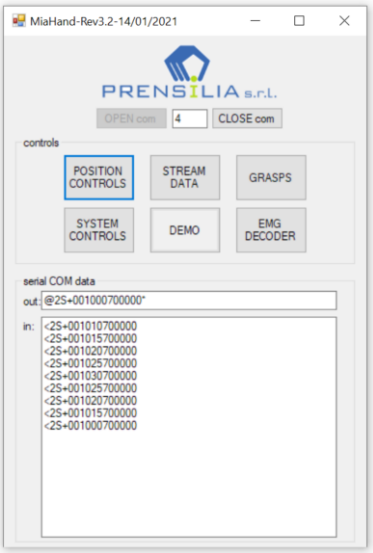


Figure 6 Main form of the Mia Hand demo application.

The “position controls” form can be used to control all fingers of Mia Hand, exploiting one of the available PID controllers. In addition, on the right side it summarizes the information coming from the hand and related to the motors. These are the status of the limit switches and position, current and speed of each motor. For these to work, the related streaming types have to be activated (see below).

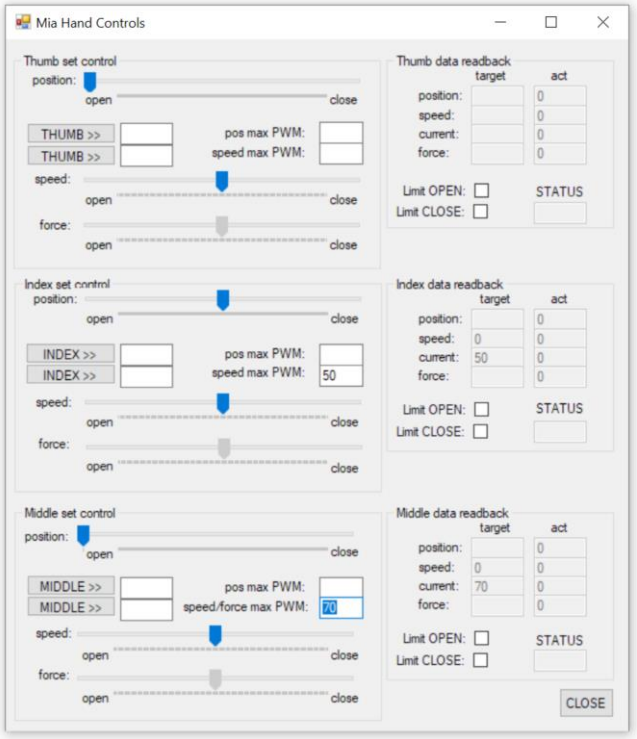


Figure 7 The position control form of the Mia Hand demo application.

The “stream data” form visualizes in an intuitive format all of the available streaming modes (excluding the binary streaming). Each streaming type can be activated on the upper left corner of the form, by clicking on the corresponding ON button. When the application detects that the data streaming is being received, it turns the ON button green. The received data can be saved in a text file by acting on the “write stream to file” section. All graphs can be customized by changing the axes limits to better appreciate all signals being received.

Figure 8 The stream controls form of the Mia Hand demo application.

The automatic grasps can be managed through the “grasps” form. It allows to set the reference positions for all the five grasps available, as well as to perform them in both the auto-grasp and manual mode.

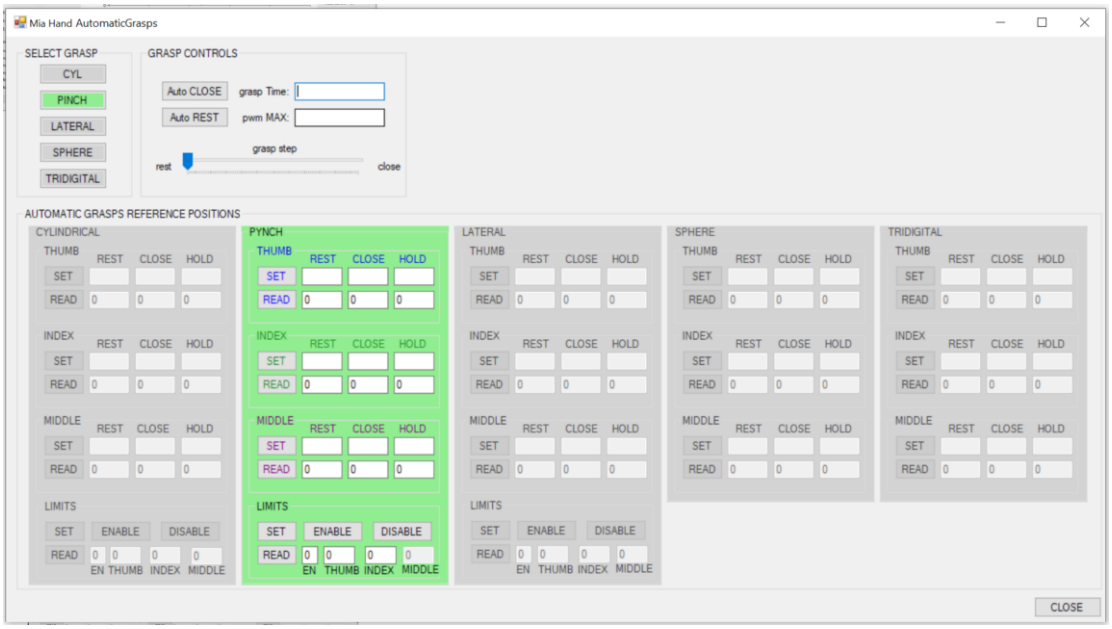


Figure 9 The automatic grasps form of the Mia Hand demo application.

Similarly to the previous one, the “EMG decoder” form provides access to all the functionalities associated with the EMG decoder. These are: setting up all of its parameters, activating the EMG decoder, activating the EMG decoder status streaming and plotting it.

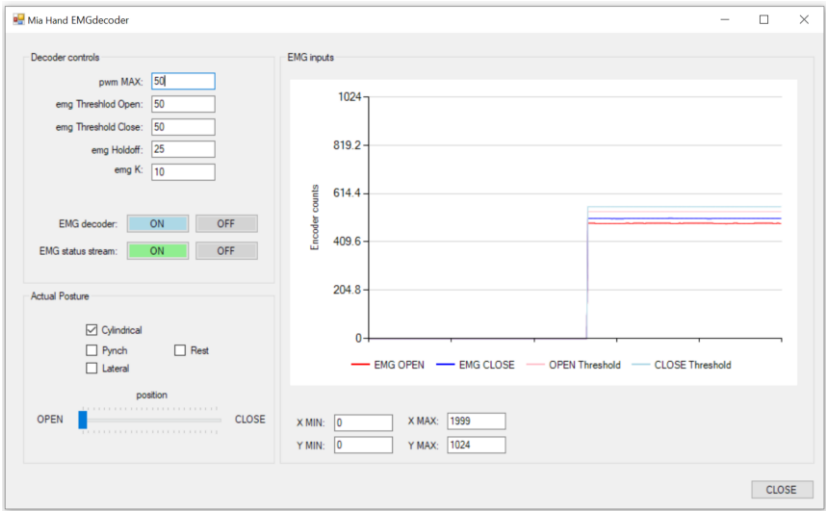


Figure 10 The EMG decoder form of the Mia Hand demo application.

Finally, the “system controls” form allows to manage all the EEPROM parameters, including the possibility to simply set them for the current session or store them permanently in the EEPROM. Additionally, it can be used to trigger a complete calibration routine and to read the firmware version installed in the main board of your prototype.

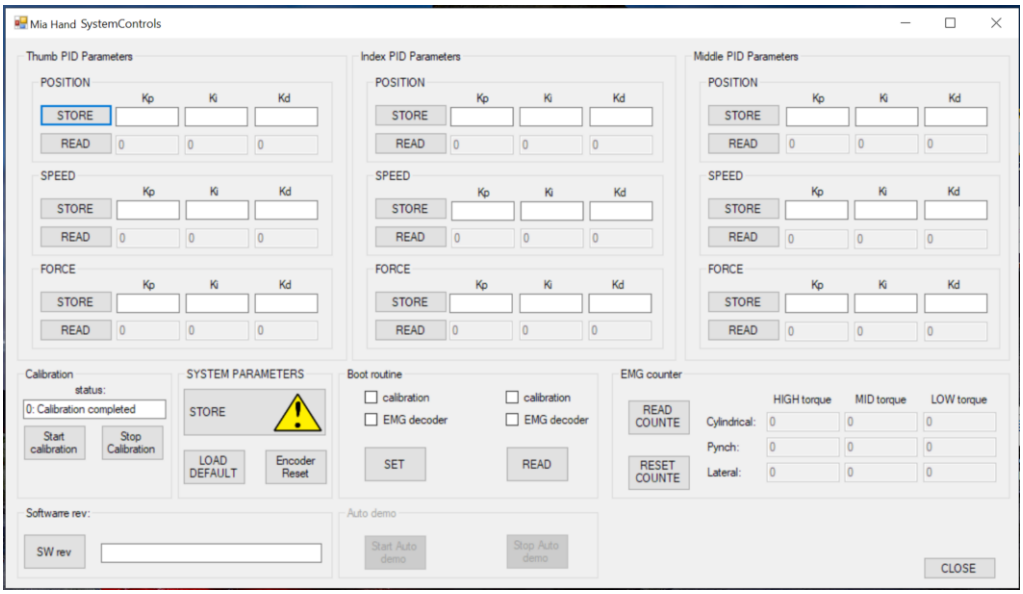


Figure 11 The system controls form of the Mia Hand demo application.

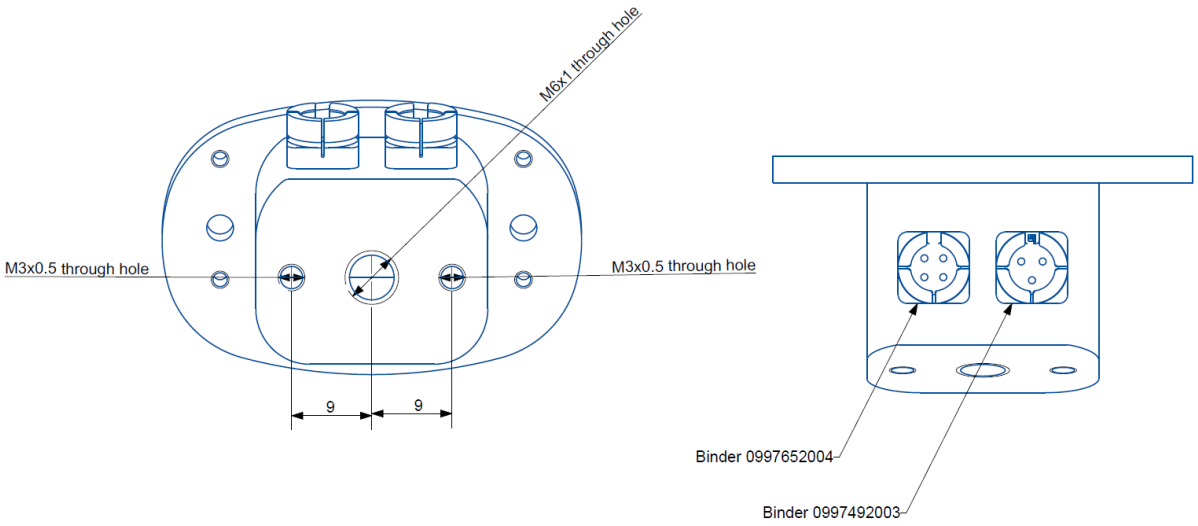
7 Default EEPROM values

The “RESTORE DEFAULT PARAMETERS” command resets all the position/speed PID parameters and the auto-grasp reference positions:

Position PID parameters	**Thumb flexion (Motor 1)**	**Index finger flexion and thumb abduction (Motor 3)**	**MRL fingers flexion (Motor 2)**
Kp	30	40	30
Ki	5	10	10
Kd	80	80	80
Speed PID parameters			
Kp	10	10	10
Ki	1	1	1
Kd	0	0	0
Cylindrical grasp references			
REST	0	50	20
POS	140	240	255
HOLDOFF	30	0	0
Pinch grasp references			
REST	20	140	0
POS	150	250	0
HOLDOFF	40	0	0
Lateral Grasp references			
REST	50	-230	255
POS	210	-230	255
HOLDOFF	0	0	0
Spherical grasp references			
REST	20	20	0
POS	220	240	240
HOLDOFF	0	0	0
Tridigital grasp references			
REST	20	20	0
POS	220	240	240
HOLDOFF	0	0	0

EMG decoder parameters	
Th OPEN	100
Th CLOSE	100
HOLDOFF	10
PWM MAX	70
K	3

8 Mechanical interface



All threaded holes:
- max depth 6 mm

	DATA/DATE (YY-MMM-DD) 21-May-02	PROGETTATO DA/DESIGNED BY		APPROVATO DA/APPROVED BY	
NOME COMPLESSIVO/ASSEMBLY NAME ARM_IN		MATERIALE/MATERIAL		TRATTAMENTI/TREATMENTS	
NOME FILE/FILE NAME MIARH-1-140000-G_00-ARMIN		DESCRIZIONE/DESCRIPTION Mia hand interface for robot arms			
NUM. FOGLIO/SHEET NO. 1/1	DIM. FOGLIO/SHEET SIZE A4	SCALA/SCALE 2.000	QUANTITA'/QUANTITY	TOLLERANZE/TOLERANCES ISO 2768-mK	PRENSILIA s.r.l.

9 ASCII code chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Prensilia SRL
Viale R. Piaggio 32
56025 Pontedera (PI)
ITALY