USER MANUAL

Version 1.1

K-Team S.A.

Lausanne, 10 January 2001

Documentation author

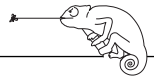
K-Team
Ch. de Vuasset, CP111
1028 Préverenges
Switzerland

info@k-team.com
http://www.k-team.com

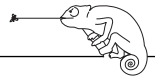
Trademark Acknowledgements

IBM PC: International Business Machines Corp.
Macintosh: Apple Corp.
SUN Sparc-Station: SUN Microsystems Corp.
LabVIEW: National Instruments Corp.
Kameleon, Khepera: K-Team

TABLE OF CONTENT



1 INTRODUCTION



The Kameleon Robotics Extension Board (REB) is a general extension of the Kameleon 376 Single Board Computer. REB manages four DC motors with four high current motor outputs and for each of them two inputs for a corresponding phase quadrature encoder. REB reads up to 11 analogic values with 10-bits ADC converters and generates up to 2 analogic values with 12-bits digital to analog converters. Finally, 16 power digital outputs and 16 digital input/output are available for digital interfaces.

Not only that it gives you new fonctionnalities but it also provides the onboard software extension to interface it. The serial communication protocol (SerCom) and the C Application Programming Interface (CAPI) are very similar with Khepera Standard allowing fast implementation for Khepera user. For new user, the simplicity of the SerCom and the modularity of the CAPI will allow fast learning.

1.1 How to Use this Manual

This manual is organised into five chapters and three appendices. To learn how to make the best use of your REB daughter board you are urged to read all of chapters 1 through 5. Chapter 6 presents the serial communication protocol that makes a remote control from a workstation possible. The appendices can be referred to as necessary.

Chapter 1	gives an introduction to this manual.
Chapter 2	describes the physical layout of the REB (connectors and jumper).
Chapter 3	explains how to set to work and test your REB with the SerCom protocol.
Chapter 4	shows how to use the REB from LabVIEW®.
Chapter 5	shows how to use the REB from MATLAB®.

Appendix A	lists the C functions (CAPI).
Appendix B	lists the serial port commands (SerCom).
Appendix C	details the connectors pinning.

1.2 Safety Precautions

Check the power supply operating voltage before operation.
If you use a power supply not provided by K-Team, check that your power supply has a voltage identical with that of this documentation. **Take care to the size and diameter of your cable for high current motors.**

Don't plug or unplug any connector when the system is switched ON.
All connections (including extension addition or disconnection) must be made when the board and the interface are switched OFF. Otherwise damages can occur.

Switch OFF the board if you will not use it for more than a day.
Disconnect the power supply removing the connector from the board or switching off the power supply.

Be very careful when separating the extensions boards.
Disconnect the CPU to the extension boards in a very careful way, avoiding to bend the pins of the connectors.

Be very careful by adding self-developed boards.
When developing your own extensions to the Kameleon board, be careful in testing them on the Kameleon board. K-Team can assume no responsibility for any damage in your concept, schematics, hardware implementation, manipulations or software. Standard electronic know-how is needed for such operations.

Be very carefully by adding electro-mechanical devices.
Also in this case, be careful in adding them on the REB board. K-Team can assume no responsibility for any damage in your concept, schematics, hardware implementation, manipulations or software.

K-Team can assume no responsibility for the use of the Kameleon in specific applications in any field, including robotics.

If you have any question or problem concerning the board, please contact your Kameleon dealer.

1.3 Recycling

Think about the end of life of your board and of the accessories! Parts of them can be recycled and it is important to do so. It is for instance important to keep Ni-Cd batteries out of the solid waste stream. When you throw away a Ni-Cd battery, it eventually ends up in a landfill or municipal incinerator. These batteries, which contain heavy metals, can contribute to the toxicity levels of landfills or incinerator ash. By recycling the Ni-Cd batteries through recycling programs, you can help to create a cleaner and safer environment for generations to come. For those reasons please take care to the recycling of your board at the end of its life cycle, for instance sending back the board and accessories to the manufacturer or to your local dealer.

Thanks for your contribution to a cleaner environment!

2 THE BOARD AND ITS ACCESSORIES

2.1 Unpacking and Inspection

Open the package and identify the following items:

1. The documentation you are reading now
2. Motors power supply cable
3. Your Kameleon REB board

2.2 Overview

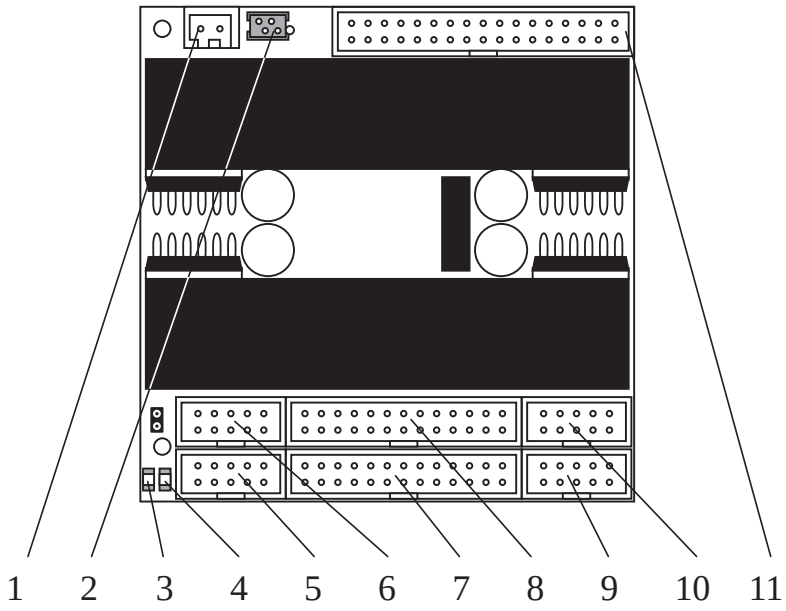


Figure 1: Position of some parts of the board.

Make an external inspection of the board. Note the location of the following parts:

1. Motors power supply connector.
2. I2C connector.
3. User LED.
4. User LED.
5. Motor/encoder connector 0.
6. Motor/encoder connector 1.
7. I/O connector 1.
8. I/O connector 0.
9. Motor/encoder connector 2.
10. Motor/encoder connector 3.
11. Miscellaneous I/O connector (MIO).

2.3 Mechanical support

The REB has to be mounted on the Kameleon board using the CXB and IXB connectors. These connectors are used as mechanical support of the board. No additional fixation is necessary.

2.4 Power supply and motor control

The REB accepts a motor DC power supply having a voltage between 12V and 24V. This voltage has to be applied on the power supply connector between SupplyMOT+ and SupplyMOT- on the connector 1 of figure 1.

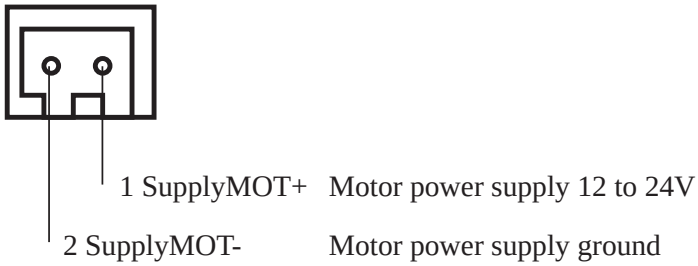


Figure 2: Motor power supply connector.

4 DC motors can be independently controlled. The motors have to be connected to the corresponding connectors, illustrated by items 5, 6, 9 and 10 of figure 1. Each connector correspond to a motor number and include motor power supply and encoder connections.

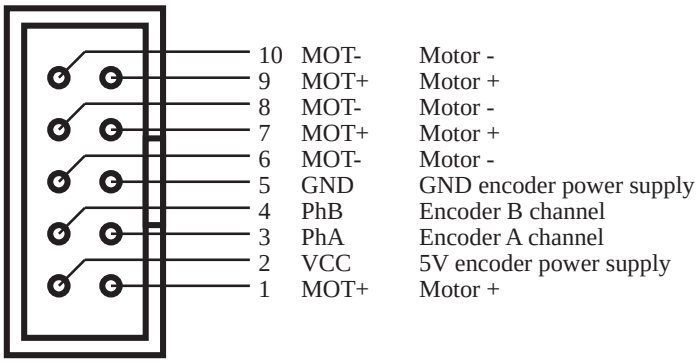


Figure 3: Motor connector pinning.

Every DC motor can be equipped with a position feedback implemented as an incremental encoder or a potentiometer placed on the motor axis. The Kameleon main processor can read the position of the incremental encoder or potentiometer and has the direct control on the motor power supply. The motor power control can be made in PWM or analog. In case of PWM (pulse width modulation) control, the main processor can also read the current used by each motor, which are proportional to the torque on the axis. This value can be read on analog input channels 11 to 14 (for motors0 to 3).

If the motor power supply is controlled by an analog output voltage, please note that the motor cannot be connected directly on the analog output. **The analog output cannot provide sufficient current to control directly a motor. An amplifier has to be placed between the analog output and the motor**.

The motor power supply can be adjusted by the main processor using PWM. This technique consists in switching the motor ON and OFF at a given frequency and for a given period of time. This basic switching frequency is constant and sufficiently high not to let the motor react to the single switching. By this way, the motor react to the time average of the power supply, which can modified by changing the period the motor is switched ON. This means that only the ratio between ON and OFF periods is modified, as illustrated in figure 4. The PWM value is defined as the time the motor is switched ON.

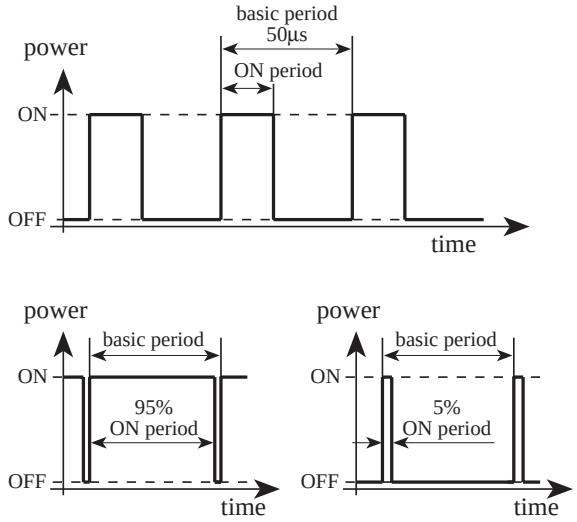


Figure 4: The “pulse width modulation” (PWM) power supply mode is based on a ratio between the ON time and the total time. The basic switching frequency is constant.

All four PWM motor drivers are constituted of H bridges. Each bridge has the following specifications:

- I_{mot} Output current 3A
- V_{mot} Motors supply voltage +12 V to +24V (SupplyMOT)
- Pd Power dissipation 20 W
- $R_{DS_{on}}$ Switch ON resistance 0.4 Ω

The Pd value is allowed only when one H-Bridge is driving current. In fact H bridge power components are fixed two by two on the same heatsink cooler. Motor 0 and motor 2 H bridges are on the first heatsink cooler. Motor 1 and motor 3 H bridges are on the second heatsink cooler. This means that if you want to drive both motor on the same heatsink cooler (0 and 2 or 1 and 3), the maximum power consumption per H-Bridge is only 10 W.

The PWM and analog output values can be set directly by the user, or can be man-

aged by a motor controller. The motor controller can perform the control of the speed or the position of the motor, setting the correct motor control value according to the real speed or position read on the motor, as illustrated in figure 5.

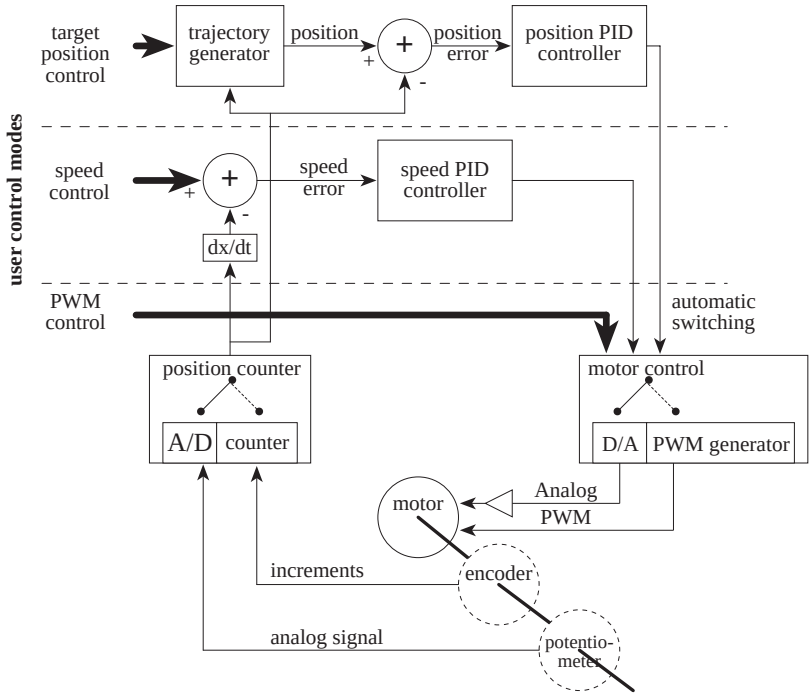


Figure 5: Structure of the motor controllers and levels of user access.

All DC motors can be controlled by a PID controller executed in an interrupt routine of the main processor. Every term of this controller (Proportional, Integral, Derivative) is associated to a constant, setting the weight of the corresponding term: K_p for the proportional, K_i for the integral, K_d for the derivative.

The motor controller can be used in two control modes: The speed and the position modes. The active control mode is set according to the kind of command received. If the controller receives a speed control command, it switches to the speed mode. If the controller receives a position control command, the control mode is automatically switched to the position mode. Different control parameters (K_p, K_i and K_d) can be set for each of the two control modes.

Used in speed mode, the controller has as input a speed value of the motor, and controls the motor to keep this speed. The speed modification is made as quick as possible, in an abrupt way. No limitation in acceleration is considered in this mode.

Used in position mode, the controller has as input a target position of the motor, an acceleration and a maximal speed. Using this values, the controller accelerates the motor until the maximal speed is reached, and decelerates in order to reach the target position. This movement follows a trapezoidal speed profile, as described in figure 6.

The input values and the control mode of this controller can be changed at every moment. The controller will update and execute the new profile in the position mode, or

control the motor speed following the new value in the speed mode. A status of the controller indicates the active control mode, the phase of the speed profile (on target or in movement) and the position error of the controller.

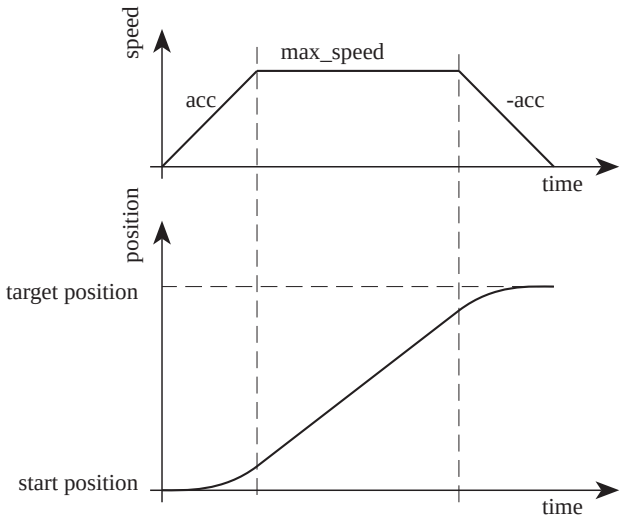


Figure 6: Speed profile used to reach a target position with a fixed acceleration (acc) and a maximal speed (max speed).

2.5 General I/O connectors

Three connectors are devoted to input/output (I/O) ports. The ports can be divided in two categories: analog and digital. Each category has input and output ports. These several groups of ports are presented in this section. Characteristics as well as connectors pinning are given.

2.5.1 Analog inputs

There are 11 A/D converters available on this daughterboard, with 10 bits precision. This correspond to a input value spanning from 0 to 1023 with 4mV/bit resolution, covering a voltage from 0 V to 4,096 V (Vref). The 4,096V (called Vref) voltage is generated by the board and is present on every connector. You can get the best precision in your measurements using, as reference, the analog ground (GNA).

Analog inputs are present on all connectors. Two analog inputs are present on each of the two I/O connector (items 7 and 8 of figure 1), placed as indicated in figure 7.

All other analog inputs are placed on the miscellaneous I/O connector (item 11 of figure 1), placed as indicated in figure 8.

Analog channels 11 to 14 are not present on the I/O connectors and are used to measure the current of the H bridges used for PWM motor control. The unit of the bit in this case correspond to 2.79 mA of current in the corresponding motor.

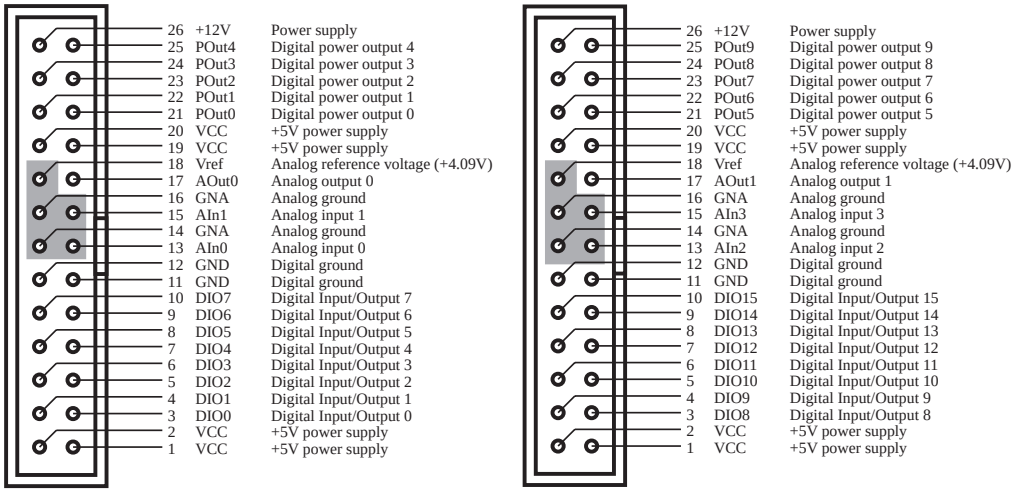


Figure 7: Analog groups on I/O connectors 0 and 1.

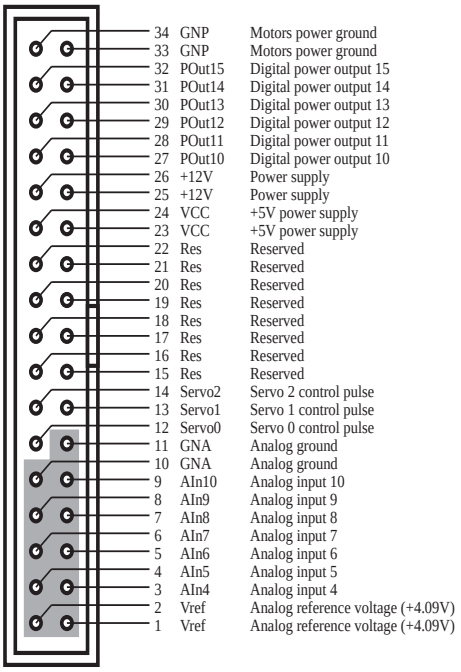


Figure 8: Analog group on miscellaneous I/O connector.

2.5.2 Analog outputs

Two analog outputs are available on this board. D/A converters are able to drive ±10 V analog outputs with a resolution of 5mV/bit. The range value is coded of a signed 12 bits number, i.e. from –2048 to +2047. The maximum current that these outputs can drive is only 20 mA, so you couldn't directly drive motor's phases, for instance.

One analog output is present on each of the two I/O connector (items 7 and 8 of figure 1).

2.5.3 Digital I/O

Each of the 16 digital I/O channel can be configured either in input mode or in output mode. For example, you may use 7 channels as inputs and the other 9 as outputs. The port configuration is done by using ext_new_IO_configuration command. This command write a 16 bits word in the direction register and indicates if the channels are in output or input mode. Bits set to 0 indicate that the corresponding channel is set to output. Bits set to 1 indicate that the corresponding channel is set to input.

After reset, all channels are by default in output configuration

Writing a new value when the channel is configured as an input has no effect and reading a channel when it's configured as an input gives an unknown value.

The maximum input voltage is 24 V and the output current can't exceed 1mA because there is a 4,7kW serial resistor for all channels. They are also protected against overvoltage and they have a pull-down resistor, as illustrated in figure 9.

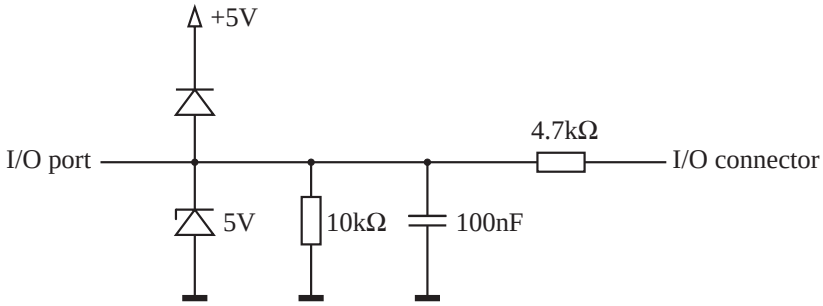


Figure 9: Schematics of digital I/O ports.

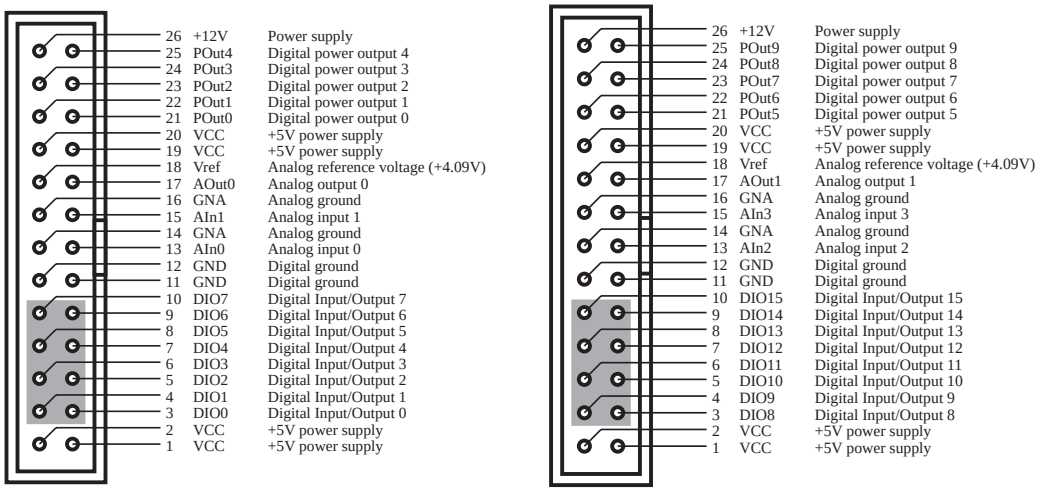


Figure 10: Digital I/O group on I/O connectors 0 and 1.

The digital I/O connections are distributed on the two I/O connectors, as illustrated in figure 10.

2.5.4 Digital servo control outputs

Four outputs are reserved for servo control. These outputs allow to control standard servo-motors used in remote controlled cars or aeroplanes. These servos need 3 electrical signals: power supply ground, power supply 5V, position control signal. This last signal is a digital modulated pulse and is available on the servo control outputs. Power supply can be taken from the power supply pins on the same or other connectors on the board.

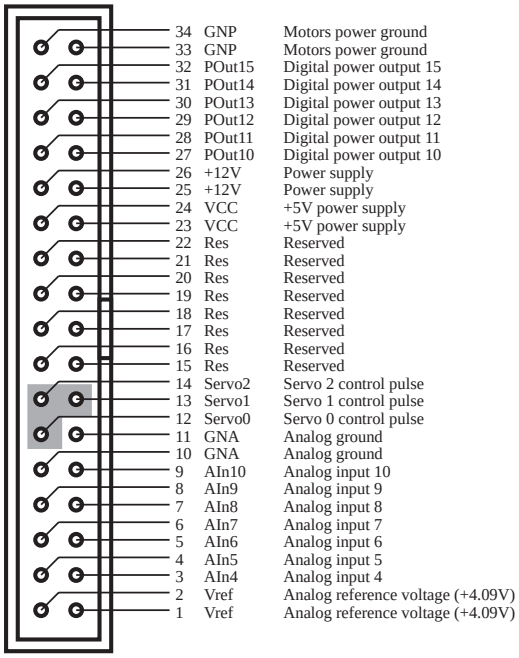


Figure 11: Servo control group on the miscellaneous I/O connector.

WARNING: the servo control functionality is accessible only using BIOS calls and not from the SERCOM serial protocol. Please refer to the appendix “C functions” on page 23 to learn how to use this functionality.

2.5.5 Digital power outputs

You can use up to 16 open drain ports with a maximum current of 1A. These outputs may be used for driving several power components like relay, IR leds, electromagnet, lamps etc. **Be very careful if you connect electro-magnetic components to this output. Remark that these components can generate overvoltages that can generate damages to the control electronics. To avoid problems, please use protection diodes.**

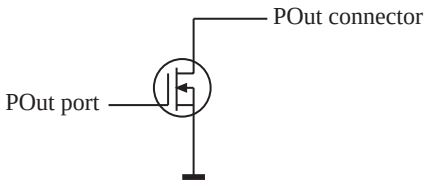


Figure 12: Power output schematics.

The precise output specification is the following:

• $I_{DS_{max}}$	Maximum sink current	1	A
• $U_{DS_{max}}$	Maximum drain voltage	50	V
• $R_{DS_{on}}$	Typical Drain-Source resistance (ON state)	0.1	W

Digital power outputs ports are distributed on the main I/O connectors, as illustrated in figures 13 and figure 14.

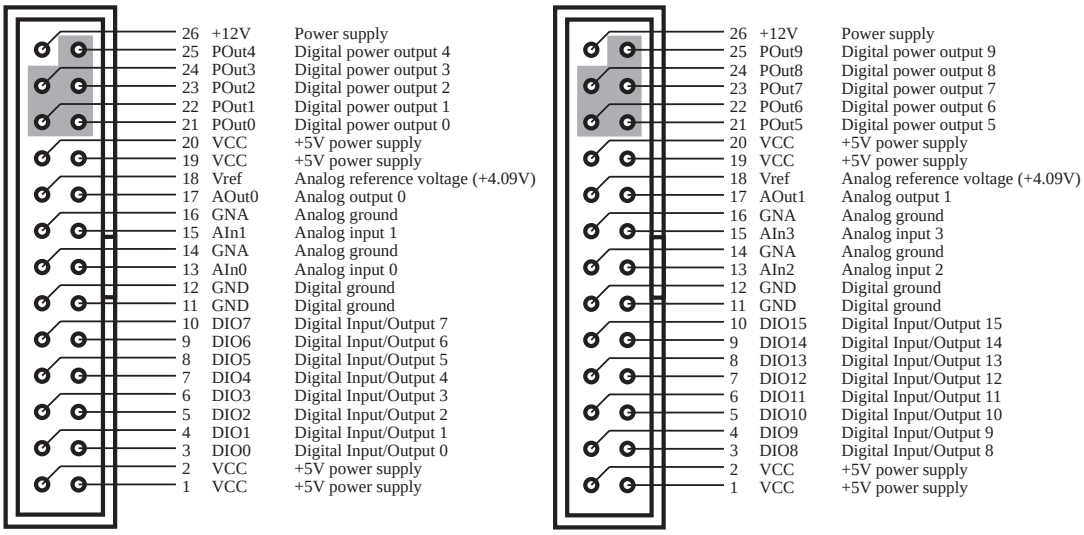


Figure 13: Digital power out group on I/O connectors 0 and 1.

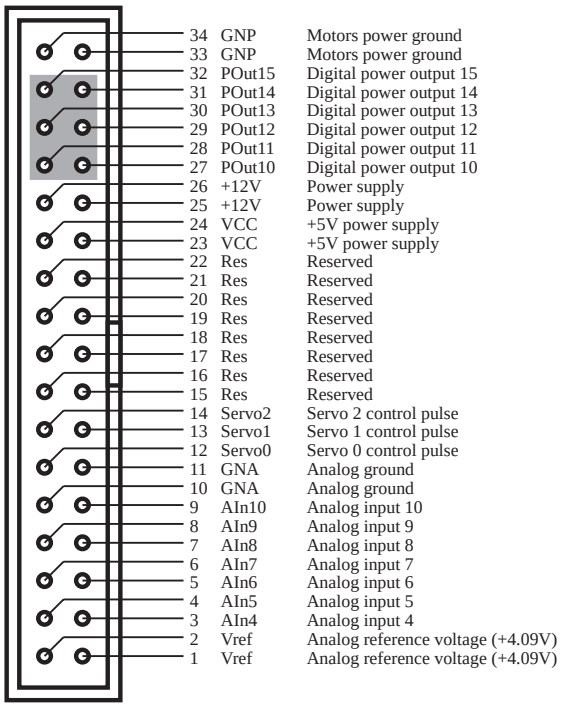


Figure 14: Digital power out group on miscellaneous I/O connector.

2.5.6 I²C connector

The REB includes a connection I²C (item 2 of figure 1). The I²C is a bus for interconnection of chips defined by Philips and used as a standard by many manufacturer. The implementation on the REB follows the standards and is compatible with most standard devices. The connector is a custom one and does not follows a particular format. The pinning is given in figure 15.

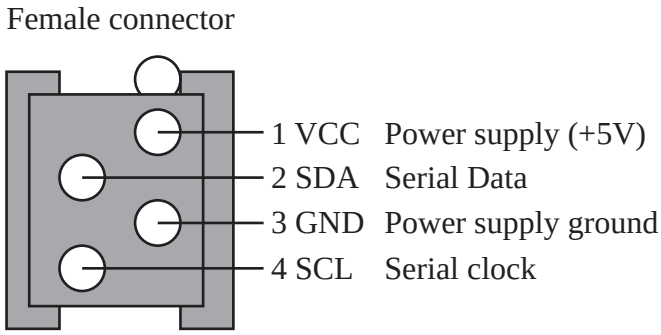


Figure 15: I²C connector pinning.

WARNING: the I²C functionality is accessible only using BIOS calls and not from the SERCOM serial protocol. Please refer to the appendix “C functions” on page 23 to learn how to use this bus.

2.6 Software

The BIOS software on your Kameleon board has to support the REB features. If you have acquired your Kameleon board WITH the REB extension, an upgrade is done by the distributor. If you have acquired Kameleon and REB separately, please do the upgrade yourself.

To check if the REB is supported, please connect the Kameleon board to a terminal, select a SerCom mode as indicated on the Kameleon user manual and check the welcome message displayed by the board. If this message includes the indication “REB supported”, you can start using the REB immediately. If the message does not include this indication, please donwload the BIOS upgrade and upgrade instructions from the K-Team WEB site:

http://www.k-team.com/download/

Software for your host computer is also available from the same donwnload page.

Please check this page for LabVIEW, MATLAB modules as well as compilers. This page also include also documentation updates.

3 THE SERIAL COMMUNICATION PROTOCOL

The serial communication protocol allows the complete control of the functionnalities of the REB through an RS232 serial line. It correspond to running mode 5 (see “Running modes” of the Kameleon board). The connection configuration necessary to use these functionnalities is presented in the section 4.2 of the Kameleon user manual. The configuration (baudrate as well as data, start, stop and parity bits) of the serial line of your host computer must correspond to the one set on the Kameleon board with the jumpers (running mode 5, 38400 Baud, 8 bit, 1 start bit, 2 stop bit, no parity).

The communication between the host computer and the Kameleon board is made sending and receiving ASCII messages. Every interaction is composed by:

- A command, sent by the host computer to the Kameleon board and followed by a carriage return or a line feed.
- When needed, a response, sent by the Kameleon to the host computer.

In all communications the host computer plays the role of master and the Kameleon board the role of slave. All communications are initiated by the master.

The communication is based on two types of interactions: one type of interaction for the set-up of the board (for instance to set the running modes, the transmission baudrate...), and one type of interaction for the control of the functionality of the board (for instance to set LEDs state, read analog inputs...). The interactions allowing the set-up of the board are based on commands called *tools*. The interactions for the control of the board functionality uses *protocol* commands and responses.

3.1 The tools

Here is the description of some basic tools:

“run”	Starts a function stored in the FLASH. It has to be followed by the function name. The functions available in the FLASH can be listed with the *list* tool and have an identification string beginning with “FU”. Using the *run* tool it is possible, for instance, to start the user FLASH mode typing “run user-flash” and return.
“serial”	Sets the serial channel to a baud-rate, given as parameter. For instance, typing “serial 19200” and return you set the baudrate to 19200 Baud.
“help”	Shows the help message of the modules available in the FLASH. A parameter can be used to indicate a need of help on a particular item. “help list” gives an help message for the *list* tool.
“list”	Gives the list of all the tools, functions, protocol commands and other modules available in FLASH. For every item listed you get an ID, a name, a description and a version. The ID is composed by four letters. The first two letters define the fam-

	ily of the module: IDs starting by “TA” define tasks running on the Kameleon, “FU” functions that can be executed with the *run* command, “PR” protocol commands, “TO” tools like this one and “BI” BIOS components.
“k-team”	Gives a short description of the K-Team active members.
“net”	Gives an information about the intelligent extension turrets installed on the K-Net connector. For each item listed you get a name, an ID (to be used to address the turret, see also the command ‘T’ in appendix A), a description and a revision.
“memory”	Gives an information about the memory used by the system.
“restart”	Resets the Kameleon board. This action is equivalent to a hardware reset.
“process”	Gives the list of all processes running on the Kameleon board in parallel to the serial communication protocol management.
“sfill”	Starts a Motorola S format downloader. This downloader does not start the execution of the code at the end of the download. To download and execute a code, please use the sloader function, started by the command “run sloader”.

3.2 The control protocol

To control the functionnalities of the Kameleon board and the REB, a set of command are implemented in the control protocol. Also in this case, the communication with the Kameleon board is made sending and receiving ASCII messages. Every interaction between host computer and Kameleon is composed by:

- A command, beginning with one or two ASCII capital letters and followed, if necessary, by numerical or literal parameters separated by a comma and terminated by a carriage return or a line feed, sent by the host computer to the Kameleon board.
- A response, beginning with the same one or two ASCII letters of the command but in lower-case and followed, if necessary, by numerical or literal parameters separated by a comma and terminated by a carriage return and a line feed, sent by the Kameleon board to the host computer.

In all communications the host computer plays the role of master and the Kameleon the role of slave. All communications are initiated by the master.

The protocol commands is described in Appendix A.

3.3 Testing a simple interaction

To better understand both tools and protocol commands, we propose to do a very simple test as following:

- Verify that the Kameleon board is set in the running mode 5 (see “Running modes” on the Kameleon user manual).

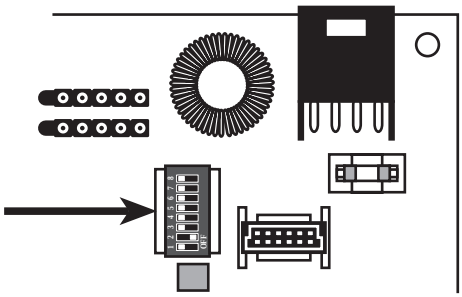


Figure 16: Settings of the jumpers for the running mode 5.

- Set the connection configuration presented in section 4.2 of the Kameleon user manual.
- Connect a motor equipped with incremental encoder to the motor connector number 0 (item 6 in figure 1, pinning as in figure 3).
- Start on your host computer a terminal emulator (for instance Hyper Terminal on PCs, Z-Term on MACs, tip on UNIX or minicom on linux) with the serial line set to 38400 Baud, 8 bit data, 1 start bit, 2 stop bit, no parity.

We start testing some tools:

- Type the command **help** followed by a carriage return or a line feed.
- The Kameleon board must respond with the list of all tools available.
- Type the command **help serial** followed by a carriage return or a line feed.
- The Kameleon board must respond with the description of the “serial” tool.
- Type the command **help I** followed by a carriage return or a line feed.
- The Kameleon board must respond with the description of the “I” protocol command.
- Type the command **list** followed by a carriage return or a line feed.
- The Kameleon board must respond with the list of all software modules present in the FLASH. In addition to the “tools” (characterised by a “TOXX” ID) and the “protocol” commands (characterised by a “PRXX” ID) you can find on the list “functions” (characterised by a “FUXX” ID) and BIOS modules (characterised by a “BIXX” ID). On every module you can have an help message.

We continue with some protocol commands:

- Type the string **H,0** followed by a carriage return or a line feed.
- The Kameleon board must respond with **H** followed by an indication of the

position of the motor. If the Kameleon board does not recognise the H command, check the software installed on your Kameleon board.

- Turn the motor axis manually and type again the string **H,0** followed by a carriage return or a line feed.
- The Kameleon board must respond with **H** followed by an indication of the new position of the motor.
- Try other commands following the description given in Appendix A.

4 USING LABVIEW®

The goal of this chapter is to familiarise you with the LabVIEW® environment in the context of REB use. LabVIEW® is a product of National Instruments (http://www.natinst.com). To this end, the examples are presented in an increasing order of complexity. Our advice is to follow the chronological order of presentation. Please refer to the LabVIEW manuals for more information about this software.

The following examples and the files available on the K-Team WEB site are based on LabVIEW® version 5.

LabVIEW® runs on your PC, Macintosh® or SUN®, and can control the functionality of the REB board using the serial communication protocol described in section 3.2.

4.1 Hardware configuration

Set your environment as given in the Kameleon board user manual. The jumpers of the Kameleon board must be set as showed in figure 17, to obtain the running mode 5.

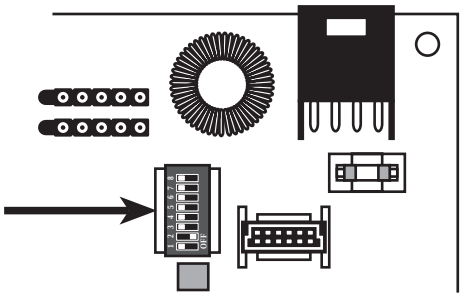


Figure 17: Settings of the jumpers for using LabVIEW ® and a serial connection at 38400 baud.

4.2 Set up of the serial link

To enable the exchange of information between your computer and the board, you have to configure the serial link of your host computer, according to the one you settled on the Kameleon board.

Be sure that the connection cable is connected at both ends (Kameleon and host computer), that the board is powered (DC power supply), then start LabVIEW® and open the Set-up.vi virtual instrument (called “VI”) present in the package.

The panel illustrated in figure 18 should appear.

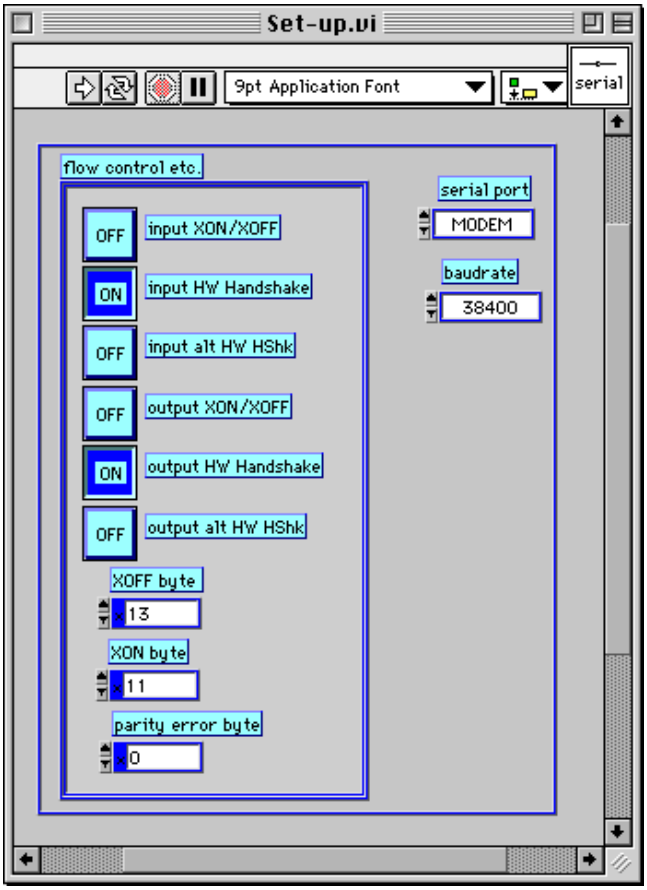


Figure 18: Set up panel for serial link initialisation.

Now, select the serial port on which the board is connected. This selection depend on which port you use and the type of computer you have. This choice must be made for every module that you will use.

Then click once on the run arrow  at the top of the window.

A stop icon  appears for a few seconds, after what the front panel returns to its initial state.

That's all! The serial link with Kameleon is set to 38400 baud. It will remain so until you quit LabVIEW® or you execute the Set-up.vi again.

4.3 Motor control

We will now check the motor control functionality of the board. Connect a motor equipped with incremental encoder to the motor connector number 0 (item 6 in figure 1, pinning as in figure 3). Be sure that the serial link has been correctly installed, then open the Get_position.vi present on the package. Now your screen displays the following panel:

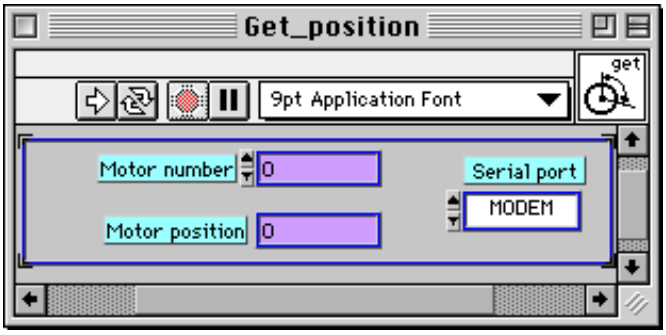


Figure 19: Get_position: one indicator shows the voltage of your power supply.

To read the position, just click once on the arrow. You can turn the motor axis and click on the arrow again to read the new position value.

If you are getting bored with clicking on the arrow, try one click on the double arrow  . Click on the stop icon to stop the execution.

The REB has 11 analog inputs (see “Analog inputs” on page 7). The VI “Get_ana_value.vi” enable to display the analog value present of each analog channel.

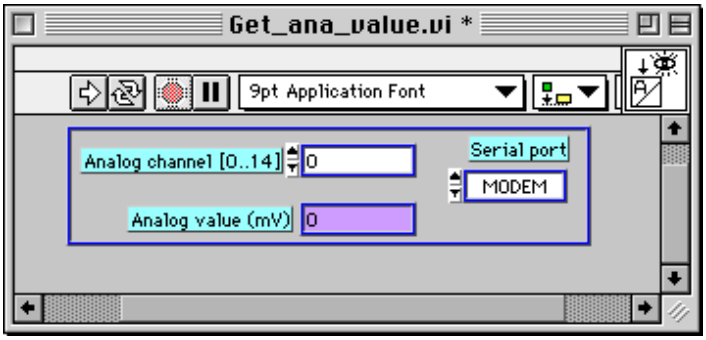
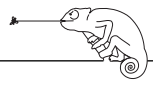


Figure 20: Get_ana_value panel: an indicator shows the voltage on the given channel.

Check the other VIs present in the package.

5 USING MATLAB®



The goal of this chapter is to familiarise you with the MATLAB® environment in the context of REB use. MATLAB® is a product of The Mathworks Inc. (http://www.mathworks.com/). To this end, the examples are presented in an increasing order of complexity. Our advice is to follow the chronological order of presentation. Please refer to the MATLAB manuals for more information about this software.

The following examples and the files available on the K-Team WEB site are based on MATLAB® version 5.0.

MATLAB® runs on several platforms but K-Team modules are available only for MATLAB® under Windows® and Macintosh®. These modules can control the functionality of the Kameleon board using the serial communication protocol described in section 3.2.

5.1 Hardware configuration

Set your environment as illustrated in section 4.2 of the Kameleon user manual. The jumpers of the Kameleon board must be set as showed in figure 21, to obtain the running mode 5.

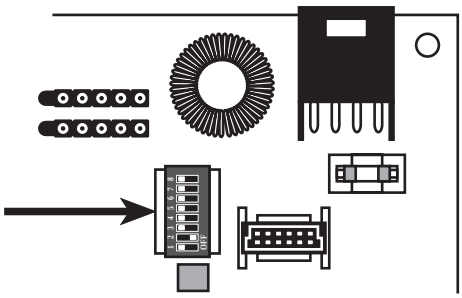


Figure 21: Settings of the jumpers for using MATLAB® and a serial connection at 38400 baud.

5.2 Set up of the serial link

To enable the exchange of information between your computer and the board, you have to configure the serial link of your host computer, according to the one you settled on the Kameleon board.

To install the MATLAB® modules to control the Kameleon board, please download the package from the K-Team WEB site (http://www.k-team.com/download) and follow the README instructions.

Be sure that the connection cable is connected at both ends (Kameleon and host computer), that the board is powered (DC power supply), then start MATLAB® and set the accesses to the K-Team package as indicated in the README file.

To open a serial connection with your Kameleon board you have to use the kopen command as follow:

```
» ref=kopen([0,38400,1])
ref =
1.0e+009 *
0.0000    2.0133    0.0009    0.0386
```

The first parameter of kopen is the serial port. The serial port number 0 correspond to the modem port on MAC and to COM1 on PC. The serial port number 1 correspond to the printer port on MAC and to COM2 on PC. The second parameter is the baudrate. The third parameter is the timeout in seconds. This function returns a ref matrix which will be

used for all other communication with your Kameleon board.

Before leaving MATLAB, always close the connection using:

```
» kclose(ref)
```

To send a command string and get a single line answer string from your Kameleon board, please use the “kcmd” command. To get the position of motor 0, for instance, do as following:

```
» kcmd(ref,'H,0')
ans =
h,67
```

The first parameter of the “kcmd” command is the reference matrix obtained from the kopen command. The second parameter is the string which has to be sent to the Kameleon board. The answer string (line finished by carriage return and line feed) received before the timeout is returned. In this example, 67 represent the position of the motor in pulses of the encoder or voltage of the potentiometer.

For commands which have multi-line answers, “kcmd” has a third parameter. If set to one, this parameter enable to get all characters until the timeout. Please adjust the timeout in order to ensure a correct transmission. For instance you can try the following command:

```
» kcmd(ref,'list',1)
```

A given command, like **H** seen before, can be implemented in a specific function, like “kgetposition”. The same example seen before becomes

```
» kgetposition(ref,0)
ans =
67
```

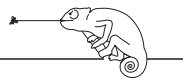
This function is implemented as following:

```
function r = kGetPosition(ref,motor)
% KGETPOSITION Read the motor position

if nargin ~= 2
	disp('Usage:')
	disp('value = kGetPosition(ref,motor)')
	disp('Returns a vector of positions corresponding to a vector of motors.')
	disp('Returns -1 if the motor is not found.')
	disp('Use the reference obtained with kopen.')
	disp('Skye Legon, K-Team SA, 10/99')
	return
end

r = ones(size(motor));
for i = 1:length(motor)
	reply = kcmd(ref,sprintf('H,%d',round(motor(i))));
	[value,count,errmsg] = sscanf(reply,'h,%d');
	if isempty(errmsg)
```

```
                r(i) = value;
        else
                r(i) = -1;
        end
end
```

Other similar functions can cover all the protocol commands available and presented in Appendix A.

6 REFERENCES



[National98] National Instruments, LabVIEW manuals for the release 5, january 1998.

[Motorola98] Motorola, A Background Debugging Mode Driver Package, AN1230/D.

http://mcu.motsps.com/lit/app_notes/an1230.pdf

[K-Team99] K-Team, High Performance 68376 CPU board, 1999.

http://www.k-team.com/download/kameleon/documentation/Data-Kameleon376SBC.pdf

void ext_reset(void)

This function starts all services related to the REB extension board. This function has to be called once before use of the extension functionalities.

Input (stacking order) :

-

Output:

-

DA converter functions:

int32 ext_put_analog_value(uint32 DACnum, int32 DACvalue)

This function send a new value to the selected converter.

Input (stacking order):

DACvalue	New converter's value (-2048..+2047).
DACnum	Number of the converter you want to use. (0..1)

Output:

0	OK
-1	The converter does not exist

IO port functions:

void ext_new_IO_configuration(uint32 IOconfiguration)

This system call sets the IO port configuration. Each bit indicates the direction of the selected channel (0 indicates that the channel is output mode and 1 indicates that the channel is an input).

Input (stacking order):

IOconfiguration	new IO configuration (0..65535)

Output:

-

int32 ext_new_IO_pin_configuration(uint32 pinNumber, uint32 PinConfiguration)

This system call sets the IO configuration of a single channel. PinConfiguration indicates the direction of the selected channel (0 indicates that the channel is output mode and 1 indicates that the channel is an input).

Input (stacking order):

PinConfiguration	new IO channel configuration (0 or 1)
PinNumber	channel number (0...15)

Output:

0	OK
-1	The channel number does not exist
-2	The channel configuration does not exist

void ext_new_IO_value(uint32 value)

This system call sends a new value to the IO port. Writing a bit on an input channel has no effect.

Input (stacking order):

Value	new IO value (0..65535).

Output:

-

int32 ext_new_IO_pin_value(uint32 pinNumber, uint32 action)

This system call makes an action on the IO channel selected. Acting on an input channel has no effect. Possible actions are: 0: turn OFF, 1: turn ON, 2: change status.

Input (stacking order):

Action	action value (0... 2).
PinNumber	channel number (0-15)

Output:

0	OK
-1	The pin number does not exist
-2	The action does not exist

int32 ext_get_IO_value(void)

This function returns the IO port value.

Input (stacking order):

-

Output:

The IO port value

Power port functions:

void ext_new_power_out(uint32 powerValue);

This function send a new value to the power_out port. This port is always in output configuration.

Input (stacking order):

PowerValue	new power_out port value (0..65535)

Output:

-

int32 ext_new_power_pin_value(uint32 pinNumber, uint32 action)

This system call makes an action on the digital power channel selected. Possible actions are: 0: turn OFF, 1: turn ON, 2: change status.

Input (stacking order):

Action	action value (0... 2).
PinNumber	power channel number (0-15)

Output:

0	OK
-1	The pin number does not exist
-2	The action does not exist

AD converter functions:

int32 sens_get_ana_value(uint32 inputNb)

These function read an analog input. The returned value is coded on a 32-bits value but only the lowest 10-bits are use.

There are 16 analog inputs available on the Kameleon but only the 11 first is usable. In fact, the channel 15 is used for the Kameleon supply value, and the channel 11, 12, 13 and 14 is used for reading the current on each four motors.

Input (stacking order):

InputNb	AD converter input (0..10).

Output:

0..1023	AD value (10-bits value)
-1	The number of channel does not exist

Motor controller functions:

int32 mot_config_speed_1m(uint32 motorNb, uint32 kp, uint32 ki, uint32 kd)

This system call configures the PID speed controller. As usual, the values stacked are coded on 32-bits, but only the LSW is used for the PID. An error is return if the motor does not exist.

Input (stacking order):

Kd	derivative coefficient
Ki	integral coefficient
Kp	proportional coefficient
MotorNb	number of the motor (0..3)

Output:

0	OK
-1	The motor does not exist

void mot_reset(void)

This system call initialises the hardware used for the motion and starts the sampling for the PID computation. The pid is intitialised in the position control mode.

Input (stacking) order):

-

Output:

-

int32 mot_config_speed_1m(uint32 motorNb, uint32 kp, uint32 ki, uint32 kd);

This system call initialises the speed PID coefficients for one motor. As usual, the value stacked are coded on 32-bits, but only the LSW is used for the PID. An error is returned if the motor does not exist.

Input (stacking order):

Kd	derivative coefficient
Ki	integral coefficient
Kp	proportional coefficient
MotorNb	number of the motor (0..3).

Output:

0	OK
-1	the motor does not exist

int32 mot_new_speed_1m(uint32 motorNb, int32 speed);

This system call change the speed for one motor. An error is returned if the motor does not exist. The speed is measured in increment per 10 ms. So, the real speed of the robot (speed in mm/sec) depends of the mechanic part (reductor).

Input (stacking order):

Speed	Speed value.
MotorNb	Number of the motor (0..3)

Output:

0	OK
-1	The motor does not exist

int32 mot_get_position(uint32 motorNb)

This system call returns the absolute position of one motor. The position provides directly of the incremental sensor or of one of the AD channel (it depend of the feedback mode you're using, see function mot_config_system).

Input (stacking order):

MotorNb	Numberof the motor (0..3)

Output:

Position	Absolute position value

int32 mot_get_speed(uint32 motorNb);

This system call returns the speed (difference of two absolute positions in one sample time) of one motor.

Input (stacking order):

MotorNb	Number of the motor (0..3)

Output:

Speed	instantaneous speed

int32 mot_put_sensors_1m(uint32 motorNb, int32 position)

This system call sets the value of the incremental sensor of one motor. This function have no effect if the feedback is different of Encoder (see function

mot_config_system). An error is returned if the motor does not exist.

Input (stacking order):

Position	Position value
MotorNb	Number of the motor (0..3)

Output:

0	OK
-1	The motor does not exist

void mot_stop(void)

This system call immediately stops the four motors and sets the PID coefficients to zero.

Input (stacking order):

-

Output:

-

int32 mot_new_position_1m(uint32 motorNb, int32 position)

This system call changes the position of one motor. An error is returned if the motor does not exist.

Maximum forward position+2^{31}-1

Maximum backward position-2^{31}

Input (stacking order):

Position	Position value
MotorNb	Number of the motor

Output:

0	OK
-1	The number of the motor does not exist

int32 mot_new_pwm_1m(uint32 motorNb, int32 pwm)

This system call changes the PWM of one motor. An error is return if the motor does not exist.

Maximum forward PWM+2^{8}-1

Maximum backward PWM-2^{8}

Input (stacking order):

pwm	PWMvalue

MotorNb	Number of the motor (0..3)

Output:

0	OK
-1	The number of the motor does not exist

void mot_new_speed_2m(int32 speed1, int32 speed0)

This system call change the speed of the two first motor (motor0 and motor1).

Input (stacking order):

Speed0	speed value (motor 0)
Speed1	speed value (motor1)

Output:

-

int32 mot_config_position_1m(uint32 motorNb, uint32 kp, uint32 ki, uint32 kd)

This system call initialises the position PID coefficient for one motor. As usual, the values are coded on 32-bits, but only the LSW is used for PID. An error is return if the motor does not exist.

Input (stacking order):

Kd	derivative coefficient
Ki	integral coefficient
Kp	proportional coefficient
MotorNb	number of the motor (0..3).

Output:

0	OK
-1	the motor does not exist

void mot_put_sensors_2m(int32 position1, int32 position0)

This system call changes the value of the incremental sensor of the two first motor (motor0 and motor1).

Input (stacking order):

Position0	Position of the sensor 0
Position1	Position of the sensor 1

Output:

-

void mot_new_position_2m(int32 position1, int32 position0)

This system call changes the position of the two first motors (motor0 and motor1)

Input (stacking order):

Position0	Position of the motor 0
Position1	Position of the motor 1

Output:

-

int32 mot_config_profil_1m(uint32 motorNb, uint32 maxSpeed, uint32 maxAcceleration)

This system call initialise s the speed and the acceleration coefficient for the profile controller for one motor. As usual, the value stacked are coded on 32-bits. The acceleration representation is coded in fixed point 24.8 (24-bits integer and 8-bits fractionnary). An error is return if the motor does not exist.

Input (stacking order):

MaxAcceleration	Acceleration coefficient
MaxSpeed	Speed coefficient
MotorNb	The number of the motor

Output:

0	OK
-1	The number of the motor does not exist

int32 mot_get_status(uint32 motorNb)

This system call gives the status of the motion controller (speed and position) for one motor

Input (stacking order):

MotorNb	Number of the motor (0..3)

Output:

Status	status of the motion
	2^0 .. 2^{15}error of the controller
	2^{16} .. 2^{17}mode (0=speed, 1=position, 2=PWM)
	2^{18}on target if mode = position

void mot_new_pwm_2m(int32 pwm1, int32 pwm0)

This system call changes the PWM of the two first motors (motor0 and motor1).

Maximum forward PWM+2^8-1

Maximum backward PWM-2^8

Input (stacking order):

Pwm0	PWM of the motor0
Pwm1	PWM of the motor1

Output:

-

int32 mot_config_system(int32 motorNb, int32 ControlValueMode, int32 PowerControlMode, int32 FeedBackMode)

This system call sets the configuration of system. The second parameter indicates the type of the control value (0..10 for A/D channel and 11 is used for a numeric value). If the control value mode is given to an A/D channel, all functions for motor position control are disabled and the only mode you can use is “position”. The third parameter indicates the type of the power control (0 and 1 are used for the two D/A channels and 2 is used for PWM). The last parameter defines the type of feedback (like ControlValueMode, values 0..10 indicates that the motor position feedback is done by A/D channel and value 11 is used for encoder feedback). An error is returned if the motor does not exist.

Input (stacking order):

FeedbackMode	The feedback mode (0..11)
CommandMode	The command mode (0..2)
ControlValueMode	The control value mode(0..11)
MotorNb	Number of the motor (0..3)

Output:

0	OK
-1	The number of the motor does not exist

Servo control functions:

int32 ext_config_servo(uint32 servoNb, uint32 minTime, uint32 maxTime, uint32 polarity)

This system call configures the servo controller. This call for every servo has to be performed before using the corresponding servo. The servos are connected to the REB following the description given in section “Digital servo control outputs” on page 10. An

error is return if the servo does not exist or if the polarity is impossible (other than 0 or 1). MinTime and maxTime gives the shortest and longest period of the pulse in microseconds. Polarity indicates the type of pulse following figure 22.

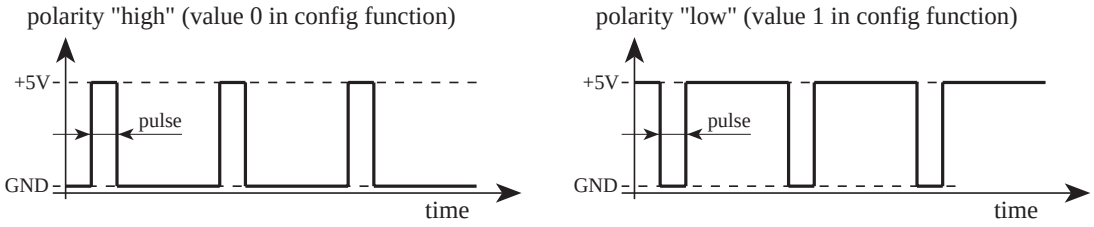


Figure 22: Polarity of the servo pulse.

Input (stacking order):

polarity	polarity of the servo control pulse
maxTime	length (in µs) of the pulse when active at 100%
minTime	length (in µs) of the pulse when active at 0%
servoNb	number of the servo (0..3)

Output:

0	OK
-1	The servo does not exist (must be 0,1,2 or 3)
-2	The polarity does not exist (must be 0 or 1)

int32 ext_new_servo_position(uint32 servoNb, uint32 servoPosition)

This system call set the length of the servo control pulse. ServoPosition is defined in percent (between 0 and 100) and allow to change the pulse (corresponding to the position of the servo) between minTime (0%) and maxTime (100%).

Input (stacking) order):

servoPosition	position of the servo in % of the displacement
servoNb	number of the servo (0..3)

Output:

0	OK
-1	The servo does not exist (must be 0,1,2 or 3)
-2	The position is bigger than 100 (%)

I^2C access functions:

int32 ext_write_I2C(uint32 I2CAddress, uint32 messageSize, uint8 *ptrData)

This system call allows to write to the address I2CAddress a string of data of length messageSize pointed by ptrData.

Input (stacking order):

ptrData	pointer on the data
messageSize	length of the data in bytes
I2CAddress	address on the I2C bus where the data should be written

Output:

0	OK
-1	General error (for instance I2C not initialized)
-2	destination device does not respond
-3	transmission error

int32 ext_read_I2C(uint32 I2CAddress, uint32 messageSize, uint8 *ptrData)

This system call allows to read at the address I2CAddress a string of data of length messageSize and store it in a memory location pointed by ptrData.

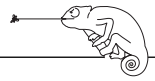
Input (stacking order):

ptrData	pointer on the place where data will be saved
messageSize	length of the data in bytes
I2CAddress	address on the I2C bus where the data should be read

Output:

0	OK
-1	General error (for instance I2C not initialized)
-2	source device does not respond
-3	transmission error

APPENDIX B COMMUNICATION PROTOCOL TO CONTROL THE BOARD



This communication protocol allows the complete control of the functionality of the REB trough a RS232 serial line. The connection configuration needed is presented in section 4.2 of the Kameleon user manual. The set-up of the serial line of your host computer must correspond to the one set on the Kameleon board with the jumpers (normally running mode 5). The protocol is constituted by commands and responses, all in standard ASCII codes. A command goes from the host computer to the Kameleon board: it is constituted by a capital letter followed, if necessary, by numerical or literal parameters separated by a comma and terminated by a line feed. The response goes from the Kameleon board to the host computer: it is constituted by the same letter of the command but in lower case, followed, if necessary, by numerical or literal parameters separated by a comma and terminated by a line feed.

In this appendix we present only the commands specific to the REB. Refer to the Kameleon user manual for the basic commands.

To better understand this protocol we propose to do a very simple test as following:

- Set the switch of the Kameleon board for the running mode number 5 (See figure 17).
- Set the connection configuration presented in section 4.2 of the Kameleon user manual.
- Start on your host computer a terminal emulator with the serial line set to 38400 Baud, 8 bit data, 1 start bit, 2 stop bit, no parity.
- Type the capital letter **B** followed by a carriage return or a line feed.
- The board must respond with **b** followed by an indication of the version of software running on the board and terminated by a line feed.
- Try other commands:

List of Available Commands

(∏ indicates CR (carriage return) **or** LF (line feed). ¶ indicates CR **and** LF.)

A Configure PID speed controller

Format of the command: A, motor_number, Kp, Ki, Kd∏

Format of the response: a¶

Effect: Set the proportional (Kp), integral (Ki) and derivative (Kd) parameters of the speed controller of the selected motor.

C Set a position to be reached

Format of the command: C,motor_number,position∏

Format of the response: c¶

Effect: Indicate to the motor position controller an absolute position to be reached. The motion control perform the movement using the three control phases of a trapezoidal speed shape: an acceleration, a constant speed and a deceleration period. These phases are performed according to the parameters selected for the trapezoidal speed controller (command J). The maximum distance that can be given by this command is (2**23)-2 pulses. The movement is done immediately after the command is sent. In the case another command is under execution (speed or position control) the last command replaces the precedent one. Any replacement transition follows acceleration and maximal speed constraints.

D Set speed

Format of the command: D, motor_number, speed_motor∏

Format of the response: d¶

Effect: Set the speed of the selected motor. The unit is the pulse/10 ms.

E Read speed

Format of the command: E, motor_number∏

Format of the response: e, speed_motor¶

Effect: Read the instantaneous speed of the selected motor. The unit is the pulse/ 10 ms.

F Configure the position PID controller

Format of the command: F,motor_number,Kp,Ki,Kd∏

Format of the response: f¶

Effect: Set the proportional (Kp), the integral (Ki) and the derivative (Kd) parameters of the position regulator of the selected motor.

G Set position to the position counter

Format of the command: G, motor_number, position_motor∏

Format of the response: g¶

Effect: Set the 32 bit position counter of the selected motor. The unit is the pulse or the A/D bit.

H Read position

Format of the command: H,motor_number∏

Format of the response: h, position_motor¶

Effect: Read the 32 bit position counter of the selected motors. The unit is the pulse or the A/D bit.

I Read A/D input

Format of the command: I, channel_number∏

Format of the response: i, analog_value¶

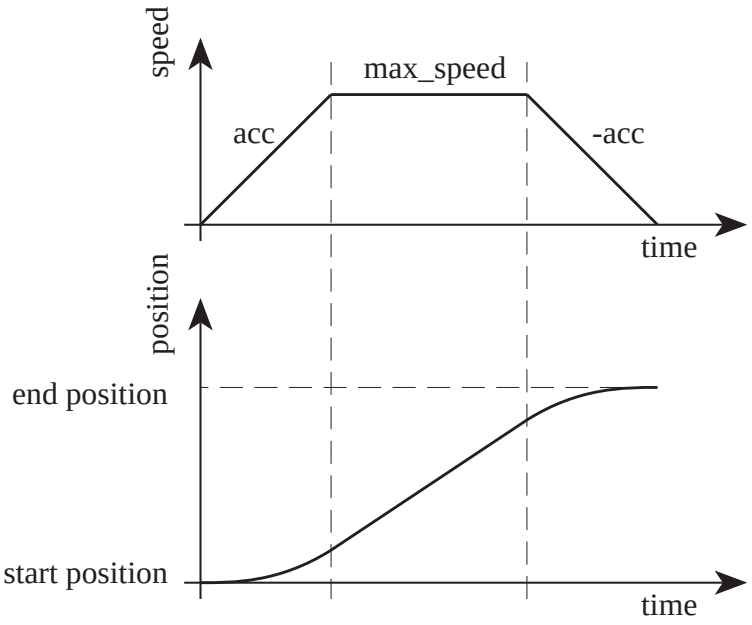
Effect: Read the 10 bit value corresponding to the channel_number analog input. The value 1024 corresponds to an analog value of 4,09 Volts. Channels 0 to 10 are the channels having a connection on the I/O connectors. Channels 11 to 14 give the current of the H-bridges of motors 0 to 3. The unit in this case is 2.79mA.

J Configure the speed profile controller

Format of the command: J, motor_number,max_speed, acceleration∏

Format of the response: j¶

Effect: Set the speed and the acceleration for the trapezoidal speed shape of the position controller. The max_speed parameter indicates the maximal speed reached during the displacement. The unit for the speed is the pulse(or A/D bit)/10ms. The unit for the acceleration is the ((pulse/256)/ 10 ms)/10 ms.



K Read the status of the motion controller

Format of the command: K, motor_number∏

Format of the response: k, T_status, M_status, E_status¶

Effect: Read the status of the motion controller. The status is given by three numbers for every motor: T_status (target), M_status (mode) and E_status (error). T_status=0 means that the robot is still on movement. T_status=1 means that the robot is on the target position. M_status=0 means that the current displacement is controlled in the position mode. M_status=1 means that the current displacement is controlled in the speed mode. E_status indicates controller position or speed error.

M Read robot management sensors

Format of the command: M, motor_number, command_value_mode, power_control_mode, feedback_mode∏

Format of the response: m¶

Effect: This command sets the configuration of system. “command_value_mode” indicates the type of the control value (0..10 for A/D channel and 11 is used for a numeric value). If the control value mode is given to an A/D channel, all functions for motor position control are disabled and the only mode you can use is "position". “power_control_mode” indicates the type of the power control (0 and 1 are used for the two D/A channels and 2 is used for PWM). “feedback_mode” defines the type of feedback (like “command_value_mode”, values 0..10 indicates that the motor position feedback is done by A/D channel and value 11 is used for encoder feedback).

P Set PWM (pulse with modulation)

Format of the command: P, motor_number, pwm_motor∏

Format of the response: p¶

Effect: Set the desired PWM amplitude (see “Motor control” on page 18 for more details) on the selected motor. The minimum PWM ratio is 0 (0%). The maximal forward ratio (100%) correspond to a value of 255. The maximal backwards ratio (100%) correspond to a value of -255.

Q Set digital power output state

Format of the command: Q, output_number, action_number∏

Format of the response: q¶

Effect: Perform an action on one of the power output lines of the REB. Possible actions are: 0: turn OFF, 1: turn ON, 2: change status. The ON status correspond to the transistor closed.

S Set the direction of I/O port

Format of the command: S, I/O_number, direction∏

Format of the response: s¶

Effect: Set the direction of the I/O lines. direction = 0 means that this line is set to output. direction = 1 means that this line is set to input.

U Write I/O digital port

Format of the command: U, I/O_number, action_number∏

Format of the response: u¶

Effect: Set the status of the corresponding output line. If the line is set as input, the command has no sense. Possible actions are: 0: turn OFF, 1: turn ON, 2: change status.

V Write analog output

Format of the command: V,channel_number,value∏

Format of the response: v¶

Effect: Set an analog value to the corresponding D/A channel. The unit is 5mV.

Y Read I/O digital port

Format of the command: Y, I/O_number∏

Format of the response: y, input_value¶

Effect: Read the status of the corresponding input line. If the line is set as output, the value has no sense.

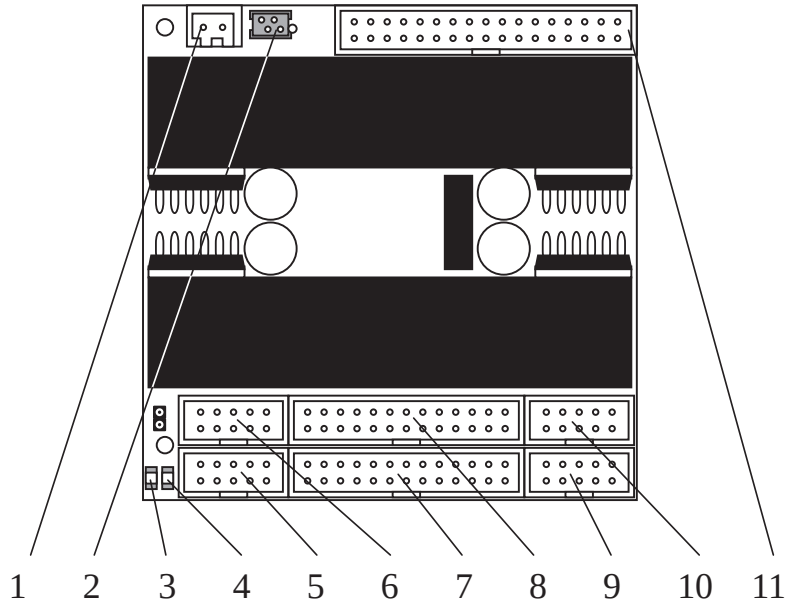
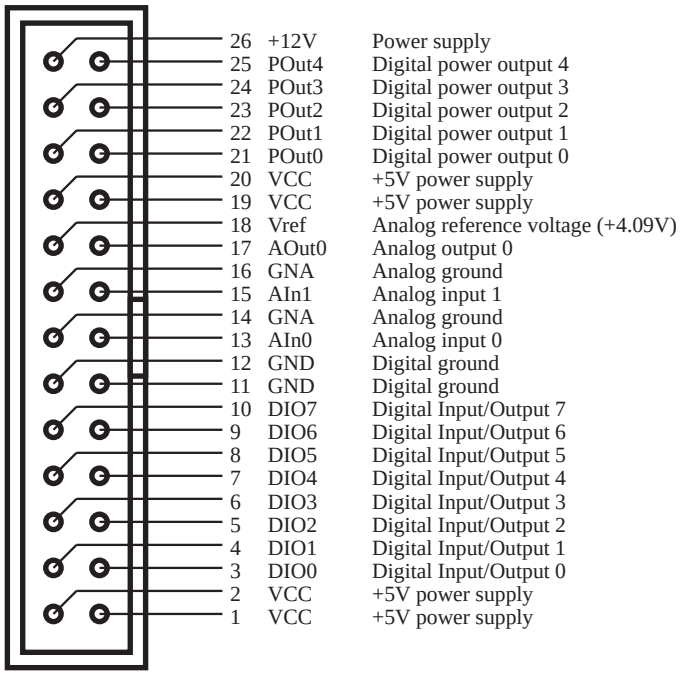


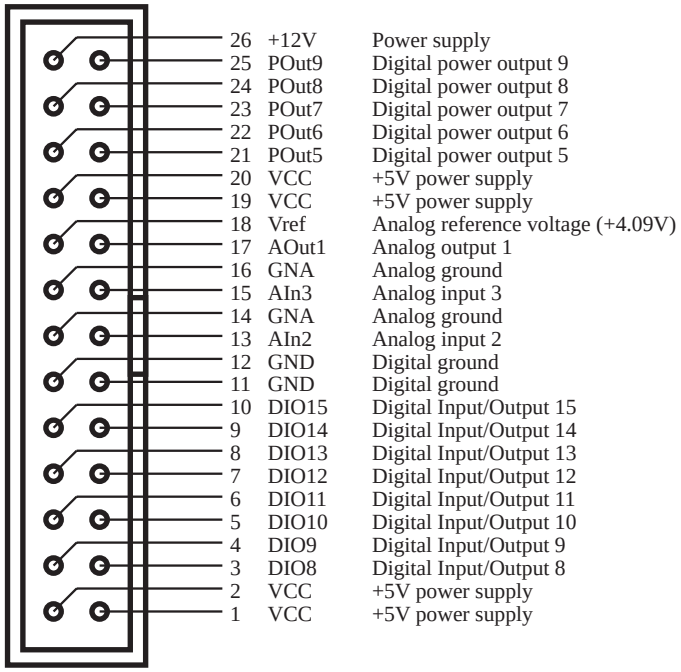
Figure 23: Position of some parts of the board.

1. Motors power supply connector.
2. I2C connector.
3. User LED.
4. User LED.
5. Motor/encoder connector 0.
6. Motor/encoder connector 1.
7. I/O connector 1.
8. I/O connector 0.
9. Motor/encoder connector 2.
10. Motor/encoder connector 3.
11. Miscellaneous I/O connector (MIO).



26	+12V	Power supply
25	POut4	Digital power output 4
24	POut3	Digital power output 3
23	POut2	Digital power output 2
22	POut1	Digital power output 1
21	POut0	Digital power output 0
20	VCC	+5V power supply
19	VCC	+5V power supply
18	Vref	Analog reference voltage (+4.09V)
17	AOut0	Analog output 0
16	GNA	Analog ground
15	AIn1	Analog input 1
14	GNA	Analog ground
13	AIn0	Analog input 0
12	GND	Digital ground
11	GND	Digital ground
10	DIO7	Digital Input/Output 7
9	DIO6	Digital Input/Output 6
8	DIO5	Digital Input/Output 5
7	DIO4	Digital Input/Output 4
6	DIO3	Digital Input/Output 3
5	DIO2	Digital Input/Output 2
4	DIO1	Digital Input/Output 1
3	DIO0	Digital Input/Output 0
2	VCC	+5V power supply
1	VCC	+5V power supply

Figure 24: I/O connector 0 (item 9 of figure 23).



26	+12V	Power supply
25	POut9	Digital power output 9
24	POut8	Digital power output 8
23	POut7	Digital power output 7
22	POut6	Digital power output 6
21	POut5	Digital power output 5
20	VCC	+5V power supply
19	VCC	+5V power supply
18	Vref	Analog reference voltage (+4.09V)
17	AOut1	Analog output 1
16	GNA	Analog ground
15	AIn3	Analog input 3
14	GNA	Analog ground
13	AIn2	Analog input 2
12	GND	Digital ground
11	GND	Digital ground
10	DIO15	Digital Input/Output 15
9	DIO14	Digital Input/Output 14
8	DIO13	Digital Input/Output 13
7	DIO12	Digital Input/Output 12
6	DIO11	Digital Input/Output 11
5	DIO10	Digital Input/Output 10
4	DIO9	Digital Input/Output 9
3	DIO8	Digital Input/Output 8
2	VCC	+5V power supply
1	VCC	+5V power supply

Figure 25: I/O connector 1 (item 8 of figure 23).

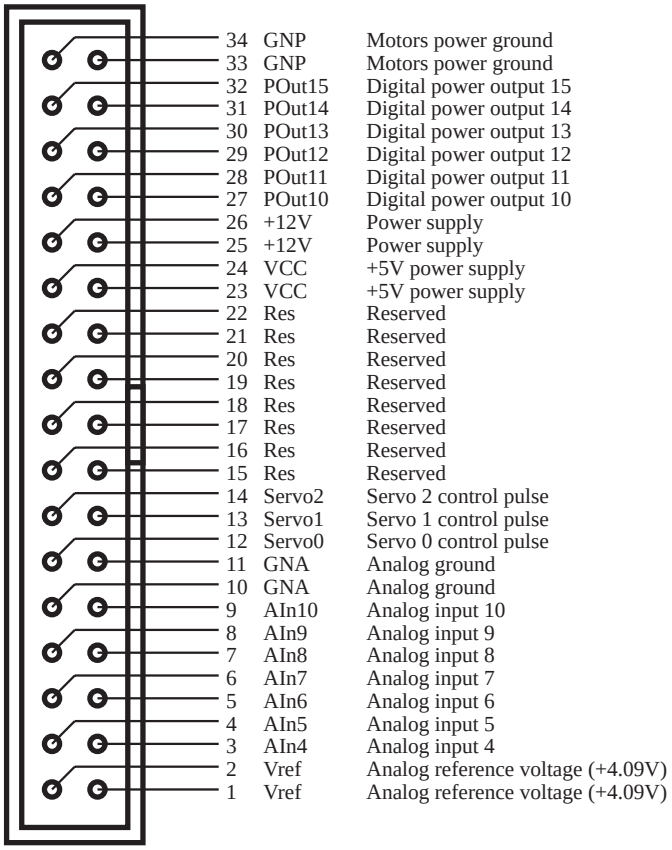


Figure 26: miscellaneous I/O connector (item 2 of figure 23).

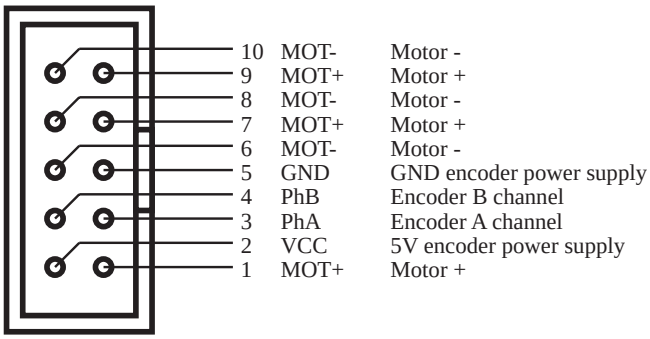


Figure 27: motor connector (items 6, 7, 10 and 11 of figure 23).

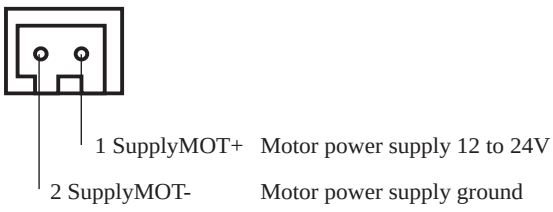


Figure 28: motor power supply connector (item 1 of figure 23).

Female connector

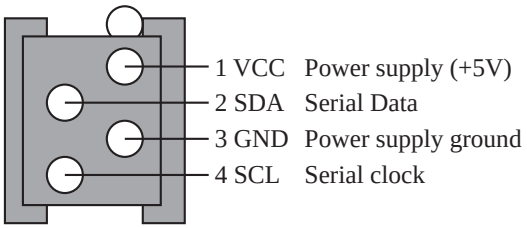


Figure 29: I^2C connector.