The emProtocol

This document describes the emProtocol, the protocol used to exchange information between devices in the internal IP-based network of iCub.

Approval History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Version | Author | | Date | Approved | | Date |
| 1.0 | Accame | iCub Unit |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 14 Jun 13 | M. Accame | First edit of the revised protocol with material taken from a previous document |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 The emProtocol 1](#_Toc360443557)

[2 Network variables 2](#_Toc360443558)

[2.1 Identification 2](#_Toc360443559)

[2.1.1 IP 2](#_Toc360443560)

[2.1.2 EP 2](#_Toc360443561)

[2.1.3 ID 2](#_Toc360443562)

[2.2 Size of the variable 2](#_Toc360443563)

[2.3 Default value 2](#_Toc360443564)

[3 Remote operation on network variables: ROP 3](#_Toc360443565)

[3.1 Structure of a ROP 3](#_Toc360443566)

[3.1.1 The CTRL field 3](#_Toc360443567)

[3.1.2 The ROPC field 4](#_Toc360443568)

[3.1.3 The EP field 6](#_Toc360443569)

[3.1.4 The ID field 7](#_Toc360443570)

[3.1.5 The SIZE field 7](#_Toc360443571)

[3.1.6 The DATA field 7](#_Toc360443572)

[3.1.7 The SIGNATURE field 8](#_Toc360443573)

[3.1.8 The TIME field 8](#_Toc360443574)

[4 Concatenation of ROPs in a single ROPframe 9](#_Toc360443575)

[4.1 The header of the ROPframe 9](#_Toc360443576)

[4.2 The body of the ROPframe 10](#_Toc360443577)

[4.3 The footer of the ROPframe 10](#_Toc360443578)

[5 Appendix 11](#_Toc360443579)

[5.1 Timing diagrams for ROPs 11](#_Toc360443580)

[5.2 ID assignment in iCub 12](#_Toc360443581)

[5.3 Examples of assignation of IP, EP, ID 13](#_Toc360443582)

[5.3.1 The board EB1 13](#_Toc360443583)

[5.3.2 The IP assignment 13](#_Toc360443584)

[5.3.3 The EP assignment 14](#_Toc360443585)

[5.3.4 The ID assignment for motion control 14](#_Toc360443586)

[5.3.5 The ROP 14](#_Toc360443587)

[5.3.6 Other ROPs 14](#_Toc360443588)

[5.4 Non-mandatory properties of network variable 15](#_Toc360443589)

[5.4.1 RW mode 15](#_Toc360443590)

The emProtocol

This document describes the emProtocol, the protocol used to exchange information between devices in the internal IP-based network of iCub.

The content of the application layer packets transported inside a UDP packet is a frame, or ROPframe containing a variable number of remote operations, or ROPs. A ROP is command aimed at manipulating a network variable, or NV.

The following sections contain a detailed description of how is defined a NV, how the ROPs operate to manipulate NVs and how the ROPs are concatenated to form a ROPframe.

Network variables

A NV is a data structure which is resident in a given device, is identified by a triple of numbers (IP, EP, ID) and has some public properties such as size and default values.

## Identification

The variable is uniquely identified inside the network by the triple (IP, EP, ID).

The 32-bit IP is the address of the device, the 16-bit EP is the endpoint, and the 16-bit ID is the final identifier of the variable.

### IP

The first level of identification is the address of the device. It is represented with 32 bits.

### EP

It is a logical grouping of variables. It is expressed with 16 bits. Its used is to help managing sets of NVs.

### ID

It is the unique identifier of a NV in the space of the device and inside a given endpoint EP. It is expressed with 16 bits.

## Size of the variable

The NV has a size whose knowledge must be public (i.e., shared between the sender and the receiver). However, this information is replicated inside the ROP for easier implementation of a protocol parser.

## Default value

The NV has a default value whose knowledge must be public. Such knowledge is used by the reset command.

Remote operation on network variables: ROP

It is possible for one device to perform a remote operation on the network variable of other device. The device sends a packet containing a ROP to the device with address IP to operate on the NV inside the device which has a given (EP, ID).

REMOTE OPERATIONS ON NETWORK VARIABLES

DEV

ROP<NV>

NV

DEV

ask<>

say<>

set<>

sig<>

rst<>

ALPKT

**Figure 1**: The Application layer packet (ALPKT) contains a remote operation ROP to be performed on a network variable of another device.

In particular, a ROP offers support for asking the value of a NV (ask), to write a value of a NV (set), to reset it to its default value (rst), but also enables a device to spontaneous signal its value to any other device (sig). The write operations can be done only on writeable NVs. Details of the protocol are in section 3.1.

## Structure of a ROP

A ROP contains the fields of the following figure.

ROP

ID

ROPC

DATA

1 BYTE

CTRL

1 BYTE

2 BYTES

0 / N BYTE

TIME

0 / 8 BYTES

SIGNATURE

0 / 4 BYTES

EP

2 BYTES

SIZE

2 BYTES

Optional with presence specified in CTRL field

Its presence depends on the ROPC type

**Figure 2**: Structure of a ROP.

### The CTRL field

It contains flags which define the composition of the ROP or requests it makes.

The USERDEF:7 flag specifies if the following bytes (ROPC and following others) implement a user-defined protocol or if follows the EmProtoc. USERDEF is 0 for emProtocol.

The RQSTCONF:6 flag specifies the request of signalling back a confirmation of the success of ROP using the same ROPC and some info in the CONFINFO field..

The RQSTTIME:5 flag specifies that the reply to the ROP must have the TIME field. This is meaningful only for the ask<> ROP (or for ACK / NAK).

The PLUSSIGN:4 flag specifies that the ROP does have the SIGNATURE field, thus it has 4 bytes extra after DATA.

The PLUSTIME:3 flag specifies that the TIME field is present, thus there are 8 bytes extra after SIGNATURE (if present).

The CONFINFO:2-1 field specifies if the ROP is normal (00b), an ACK (11b) or a NAK (10b). In case of ACK or NAK, the ROPC will be the same as the one being confirmed (with exception of ask<>, for which the ACK is a say<>).

ROP

USERDEF

Bit7 bit6 bit 5 bit4 bit3 bit2 bit 1 bit0

1 BYTE

RQSTCONF

RQSTTIME

PLUSTIME

PLUSSIGN

CONFINFO

FFU2

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

**Figure 3**: OPTIONS field.

### The ROPC field

ROP commands are used to query the value of a NV such as with ask<> and say<>, or also to force the value of a NV, such as with set<>, or also to inform about the value of a NV, such as with sig<>. See table below for a full set of remote operations.

ROP

1 BYTE

**See table**

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

**Figure 4**: ROPCODE field.

|  |  |  |
| --- | --- | --- |
| ROPC(ode) | Value | Description |
| ask<epid> | 0x02 | A device sends this command to request the receiver the value of NV(epid).  The receiving device verifies if the NV(epid) exists on the device. If it exists, it takes its local value of NV(epid).val and transmits it back with a say<epid, NV(epid).val>. If it does not exist then the device sends back a nak-ask<epid> if CTRL.RQSTCONF is set or nothing if not set.  If RQSTTIME is 1, then the reply say<> or nak-ask<epid> must have PLUSTIME at 1.  If PLUSSIGN is 1, then the reply say<> or nak-ask<epid> must have PLUSSIGN at 1 and have the same SIGNATURE as ask<>.  CTRL = RQSTCONF | RQSTTIME | PLUSSIGN | PLUSTIME | CONFINFO  ROPC = 0x02  EP = ep  ID = id  SIZE = 0  DATA = empty  SIGNATURE = present only if CTRL.PLUSSIGN is 1  TIME = present only if CTRL.PLUSTIME is 1. |
| say<epid, val> | 0x03 | A device sends this command to respond to an ask<epid> whenever the EPID exists.  CTRL = PLUSSIGN | PLUSTIME (others are ignored by receiver)  ROPC = 0x03  EP = ep  ID = id  SIZE = size  DATA = data  SIGNATURE = present only if CTRL.PLUSSIGN is 1  TIME = present only if CTRL.PLUSTIME is 1. |
| set<epid, val> | 0x04 | A device sends this command to make the receiver write val into NV(epid).val.  The receiving device verifies if the NV(epid) exists on the device and if it is writeable. If so, then it writes val into V(epid).val. If field CTRL. RQSTCONF is set then the device shall also reply with ack-set<epid> upon effective write or nak-set<epid> if the netvar is not writeable.  If both CTRL. RQSTCONF and CTRL.RQSTTIME are set then the ack-/nak-set<> shall contain a TIME.  If both CTRL. RQSTCONF and CTRL.PLUSSIGN are set then the ack-/nak-set<> shall contain a SIGNATURE.  CTRL = RQSTCONF | RQSTTIME | PLUSSIGN | PLUSTIME | CONFINFO  ROPC = 0x04  EP = ep  ID = id  SIZE = size (or 0 if CONFINFO is ACK or NAK)  DATA = data (or empty if CONFINFO is ACK or NAK)  SIGNATURE = present only if CTRL.PLUSSIGN is 1  TIME = present only if CTRL.PLUSTIME is 1. |
| sig<epid, val> | 0x05 | A device sends this command to spontaneously communicate the value of NV(epid).  If CTRL.RQSTCONF is set, then the receiving node shall reply with a ack-sig<epid> which shall also contain SIGNATURE and/or TIME fields if CTRL. PLUSSIGN and/or CTRL.PLUSSIGN are also set. The receiving node may send a nak-sig<> to tell the sender of the sig<> that the signalled NV is unknown.  OPTIONS = RQSTCONF | RQSTTIME | PLUSSIGN | PLUSTIME  ROPC = 0x05  EP = ep  ID = id  SIZE = size (or 0 if CONFINFO is ACK or NAK)  DATA = data (or empty if CONFINFO is ACK or NAK)  SIGNATURE = present only if CTRL.PLUSSIGN is 1  TIME = present only if CTRL.PLUSTIME is 1. |
| rst<epid> | 0x06 | A device sends this command to force the receiver to revert NV(epid).val to its default value.  The receiving device verifies if the NV(id) exists on the device and if it is writeable. If so, then it writes NV(epid).def into NV(epid).val. If field CTRL. RQSTCONF is set, then the device shall also replies with ack-rst <epid> upon effective write or nak-rst<epid> is the netvar is not writeable.  If both CTRL. RQSTCONF and CTRL.RQSTTIME are set then the ack-/nak-rst shall contain a TIME.  If both CTRL. RQSTCONF and CTRL.PLUSSIGN are set then the ack-/nak-rst shall contain a SIGNATURE.  CTRL = RQSTCONF | RQSTTIME | PLUSSIGN | PLUSTIME | CONFINFO  ROPC = 0x06  EP = ep  ID = id  SIZE = 0  DATA = empty  SIGNATURE = present only if CTRL.PLUSSIGN is 1  TIME = present only if CTRL.PLUSTIME is 1. |

**Table 1** – Remote operations.

### The EP field

It is used as to address the correct ID. It can help to divide NVs in logical sets.

ROP

2 BYTES

NV.ep

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

**Figure 5**: EP field.

### The ID field

It contains the ID of the NV being manipulated.

ROP

2 BYTES

NV.id

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

**Figure 6**: ID field.

### The SIZE field

It contains the size of the DATA field in two bytes in little endian order (LSB first). If DATA is not present SIZE is 0.

ROP

2 BYTES

NV.siz

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

**Figure 7**: SIZE field.

### The DATA field

If present, it contains the value of the NV and contains the proper information encoded in little endian order (LSB first).

The DATA field is padded with zeros to become of total length a multiple of four.

The presence of the DATA field is related to the ROPCODE being used. Only ROPs such as say<>, set<>, rst<>, and sig<> have the DATA field. See table of ROPCODEs for more details.

ROP

SIZE BYTES + 0 padding to reach multiple of 4

NV.val

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

Figure 8: DATA field.

### The SIGNATURE field

It is present if the CTRL.PLUSSIGN is 1. It contains a four-byte signature used by the receiver for any purpose (for instance, for checking validity of message upon a pre-agreed rule).

The receiver of the ROP with a CTRL.PLUSSIGN field being set shall send any reply (say<>, ack-xxx<>, nak-xxx<>) with the same SIGNATURE being present, so that it is used by the receiver for processing the reply information.

ROP

0 / 4 BYTES

Signature defined by the sender

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

**Figure 9**: SIGNATURE field.

### The TIME field

It is present if the CTRL.PLUSTIME field is 1. It contains the time of the preparation of the ROP. The time shall be 8-byte long, report absolute time and be measured in micro-seconds.

ROP

0 / 8 BYTES

Birth of the ROP in micro-seconds

ID

ROPC

DATA

CTRL

TIME

SIGNATURE

EP

SIZE

**Figure 10**: TIME field.

Concatenation of ROPs in a single ROPframe

It is possible, to concatenate multiple ROPs to be transmitted inside a single UDP packet using the following ROPframe, which is formed by a header, a body with the ROPs, and a footer.

ROP CONCATENATION INSIDE A UPD PACKET

UDP PAYLOAD

ROPFRAME

HEADER

BODY

24 BYTES

FOOTER

0 / n BYTES

4 BYTES

**Figure 11**: The ROPframe.

## The header of the ROPframe

It contains the following fields. The values of SIZEOFBODY, NUMBEROFROPS, SEQUENCENUM, and AGEOFFRAME are represented in little endian mode (LSB first).

ROPFRAME

HEADER

BODY

28 BYTES

FOOTER

STARTCODE

SIZEOFBODY

NUMBEROFROPS

AGEOFFRAME

4 BYTES

2 BYTES

2 BYTES

8 BYTES

In sec as measured at the time of transmission

Used to recognize a valid ROPframe: 0x12345678

SEQUENCENUM

8 BYTES

**Figure 12**: The header of the ROPframe.

## The body of the ROPframe

It contains the ROPS concatenated each after the other.

ROPFRAME

HEADER

0 / n BYTES

FOOTER

ROP(i)

BODY

i = 1 .. HEADER.NUMBEROFROPS

**Figure 13**: The body of the ROPframe.

## The footer of the ROPframe

It contains a marker informing of the end of the ROPframe.

ROPFRAME

HEADER

4 BYTES

BODY

FOOTER

Used to recognize a valid ROPframe: 0x87654321

**Figure 14**: The footer of the ROPframe.

Appendix

## Timing diagrams for ROPs

Timing diagrams for ROPs which contain also actions in the receiver are shown in the following.

SEQUENCE DIAGRAM FOR ROPs in case CTRL.RQSTCONF = 0

DEV0

set<id, val>

DEV1

if NV(id) exists and is writeable:

write val into NV(id).val

DEV0

ask<id>

DEV1

say<id, val>

if NV(id) exists:

send back NV(id).val to DEV0

DEV0

sig<id, val>

DEV1

send NV(id).val following some event inside DEV1

DEV0

rst<id>

DEV1

if NV(id) exists and is writeable:

write NV(id).def into NV(id).val

If the ROP from DEV0 has CTRL.RQSTTIME = 1, any reply from DEV1 shall have CTRL.PLUSTIME = 1 and the field TIME filled with the absolute time of preparation or the reply

If the ROP from DEV0 has CTRL.PLUSSIGN = 1, any reply from DEV1 shall have CTRL.PLUSSIGN = 1 and the field SIGN filled with the same received signature.

**Figure 15**: Sequence diagrams for the main ROPs when there is no confirmation request

SEQUENCE DIAGRAM FOR ROPs in case CTRL.RQSTCONF = 1

DEV0

set<id, val>

DEV1

if NV(id) exists and is writeable:

write val into NV(id).val

send ack

else

send nak

DEV0

ask<id>

DEV1

say<id, val>

or nak-ask<id>

if NV(id) exists:

send back NV(id).val to DEV0

else

send nak

DEV0

sig<id, val>

DEV1

send NV(id).val following some event inside DEV1

the receiver sends an ack (or nak if it cannot process it)

DEV0

rst<id>

DEV1

if NV(id) exists and is writeable:

write NV(id).def into NV(id).val

send ack

else

send nak

ack-set<id>

or nak-set<id>

ack-rst<id>

or nak-rst<id>

ack-sig<id>

or nak-sig<id>

The reply ROP (ack-\*, nak-\*, say) SHALL HAVE CTRL.RQSTCONF = 0 to avoid Ping-Pong effect.

If the ROP from DEV0/1 has CTRL.RQSTTIME = 1, any reply from DEV1/0 shall have CTRL.PLUSTIME = 1 and the field TIME filled with the absolute time of preparation or the reply.

If the ROP from DEV0/1 has CTRL.PLUSSIGN = 1, any reply from DEV1/0 shall have CTRL.PLUSSIGN = 1 and the field SIGN filled with the same received signature.

**Figure 16**: Sequence diagrams for the main ROPs when there is a request of confirmation

## ID assignment in iCub

The IDs inside an EP can be freely assigned, however for iCub we use the following convention. The most significant 4 bits identify what is called an ENTITY, the following 4 bits tells what is the INDEX of the specified entity, whereas the less significant 8 bits represents a TAG of what we want to address inside that particular entity.

ID

INDEX

ENTITY

BITS: 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

TAG

There are up to 256 possible tags for a given index of an entity: the config, the status, the command1, the command2, etc..

Four bits allow up to 16 instances of the same kind of entity (e.g., 16 joints).

Four bits allow up to 16 different entities inside an endpoint (e.g., in endpoint motion-control: a joint, a motor, a .. ).

**Figure 17**: ID assignment for a NV. The 16 bits are organised to contain specific information on the kind of data to be manipulated.

## Examples of assignation of IP, EP, ID

In the following it is reported an example of how the triple (IP, EP, ID) has been assigned in the case of board EB1 in iCub.

We shall also describe the binary of a ROP of kind set related to motion control.

### The board EB1

The board EB1 is representative of the left upper arm of iCub and as such it performs motion control on four joints with four motors and works as a gateway of one strain analog sensor. Moreover, the board can be configured to periodically signal some values and to enter or exit a 1 ms control mode loop.

Those functionalities are organised with ROPs related to a number of network variables which can be grouped into three endpoints: one for motion control, one for analog sensors, and one for board management.

The ROP we describe is set<”position PID of joint 2”, value>, where value is a 16 bytes struct of value {0x01, 0x02, …, 0x10}.

### The IP assignment

Every device which wants to communicate with the EB1 board shall send a ROP related to a variable with IP equal to the IP address of the board EB1: 100.0.1.1.

The IP value is not contained inside the ROP. It is just used to send the UDP packet to the EB1 board.

### The EP assignment

The variables related to motion control shall have EP equal to eoprot\_endpoint\_motioncontrol (0x0011), those related to analog sensors shall have EP equal to eoprot\_endpoint\_analogsensors (0x0021), and those related to management shall have EP equal to eoprot\_endpoint\_management (0x0001).

In board EB1, the motion control endpoint contains four joints, four motors and one controller, the analog sensors endpoint contains one strain, and the management endpoint contains configuration of communication and of the application.

### The ID assignment for motion control

The 16 bits ID is partitioned to contain the type of entity (joint, motor, or controller), its number, and the tag of that particular entity.

For example, the ID of a variable related to position PID of joint 2 has the following field values:

* ID.ENTITY = eomc\_entity\_joint (0x0),
* ID.INDEX = 0x2 (first is index 0),
* ID.TAG = eoprot\_ep\_mc\_joint\_tag\_config\_\_pidposition (0x01).

### The ROP

It is the following.

ROP

01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10

10

DATA

SIZE

00

04

set

CTRL

00 11

EP

02 01

ID

ENTITY = eomc\_entity\_joint = 0x0

INDEX = 0x2

TAG = eoprot\_ep\_mc\_joint\_tag\_jconfig\_\_pidposition =0x01

**Figure 18**: Binary for a set command (in hexadecimal values).

### Other ROPs

Here are some other ROPs.

ROP

01 02 03 04 05 06 07 08 09 0A 0B 0C

0C

DATA

SIZE

00

05

sig

CTRL

00 11

EP

10 05

ID

ENTITY = eomc\_entity\_motor = 0x1

INDEX = 0x0

TAG = eoprot\_ep\_mc\_motor\_tag\_mstatus\_\_basic = 0x05

**Figure 19**: Binary for sig<”basic status of motor 0”, data[12]>.

ROP

10 00 00 00

01

DATA + 3 byte padding

SIZE

08

04

set

CTRL

has

PLUSTIME

00 21

EP

00 02

ID

ENTITY = eoas\_entity\_strain = 0x0

INDEX = 0x0

TAG = eoprot\_ep\_as\_strain\_tag\_config\_\_datarate = 0x02

12 34 46 78 87 65 43 21

TIME

**Figure 20**: Binary for set<”datarate of strain 0”, data[1]>.

## Non-mandatory properties of network variable

Other properties which are not contained inside the protocol but that are useful are the following.

### RW mode

It tells if ROPs can or cannot write variables.

|  |  |
| --- | --- |
| RW mode | Description |
| RO | The NV can be read by not written by a ROP. It is used for inputs to the board, such as a ADC, or a status. |
| RW | The NV can be read and written by a ROP. It is used for outputs of the board, such as a PWM or for a configuration. |
| WO | The NV keeps a value which is meaningful only if written but not meaningful to read back. It can be used for commands. |

**Table 2** – RW modes.