iPaq-GUI Commnunication Protocol Reference

by Andrés Santiago Pérez-Bergquist <aspb@mapache.org>

The iPaq listens for connections on the following TCP/IP ports, behaving as a server to which the client program running on another computer connects to.

Ports

5000	Primary iPaq Control
5001	DumpFrame Feedback
5002	ActiveTracking Feedback

Port 5000 accepts the following packets:

For all the communications, 255 is a control code.
255,56 is a directive to flush the input buffer and keep listening.
255,55 is a directive to attempt to parse as a command everything received since the last flush , and then flush the input buffer.
255,54 is a command to terminate the iPaq's control program.
The first byte of the actual data packet determines the type of the packet. It is expressed as a single mnemonic ASCII character.
For variable-length packets, the next field expresses the length of the variable section.
After any data in the packet, there comes a checksum, where, `<csum>` is the sum of all the bytes strictly between the starting delimiter and `<csum>`, mod 256.
After formulation by the above rules but prior to transmission, any bytes equal to 255 within the packet (that is, not part of the starting and ending delimiters) are duplicated, to prevent accidental insertion of a stop delimiter. This needs to be undone on the receiving end.

For most commands, the iPaq returns a MiniResponsePacket of the form:

> `<status>` is the status byte, used to indicate if the iPaq understood the previous command packet. 0 is success, anything else is an error, indicating no action was taken.

```
<status>
<csum>
```

For all the response packets, `<status>` is the status byte, used to indicate if the iPaq understood the previous command packet.

0	Success (anything else is an error, indicating no action was taken)
1	Incorrect Checksum
2	Unknown Packet Type
10	Camera timed out
11	Camera spewed out too many response characters
20	BrainStem failed to respond

Set parameters packet

Sets the servo positions, the wheel velocity, and the timeout on the wheel velocity, and may kill the wheel motor. All positions are specified in actual servo settings.

`<mask>` is the command mask.

Bit 0 of `<mask>` corresponds to the `<vel_timeout>` command, and bit 4 corresponds to the `<cam_tilt>` command. If the mask bit of a command value is set, that value is valid, otherwise that particular value is ignored.

If bit 5 of the `<mask>` is set, then the robot will ignore the velocity and kill power to the motor. Note that this is different from setting the speed to zero, which will attempt to hold the wheel still.

If bit 6 of the `<mask>` is set, then the robot will initialize the BrainStem and set the servos to accept the full range of motion.

```
<P = 80>
<vel_timeout><vel><dir><cam_pan><cam_tilt><mask>
<csum>
```

The iPaq returns a FullResponsePacket of the following form, unless it fails to parse the incoming packet, in which case it returns a MiniResponsePacket:

```
<status>
<range>
<wheel_voltage_HI><wheel_voltage_LO>
<wheel_current_HI><wheel_current_LO>
<integrated_X_HI><integrated_X_LO>
<integrated_Y_HI><integrated_Y_LO>
<integrated_TH_HI><integrated_TH_LO>
<csum>
```

Set integration values packet

Sets the current values of the estimated position.

`<mask>` is the command mask.

Bit 0 of `<mask>` corresponds to the `<integrated_X>` command, bit 1 to the `<integrated_Y>` command, and bit 2 corresponds to the `<integrated_TH>` command. If the mask bit of a command value is set, that value is valid, otherwise that particular value is ignored.

```
<I = 73>
<integrated_X_HI><integrated_X_LO>
<integrated_Y_HI><integrated_Y_LO>
<integrated_TH_HI><integrated_TH_LO>
<mask>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveColorTracking Packet

Begins active tracking of the specified color, following the object with the camera. Results are returned on port 5001.

```
<T = 84>
<rmin><rmax><gmin><gmax><bmin><bmax>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveColorTrackingNoTilt Packet

Begins active tracking of the specified color, but leaves the tilt angle fixed. Results are returned on port 5001.

```
<U = 85>
<rmin><rmax><gmin><gmax><bmin><bmax>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveColorTrackingNoPan Packet

Begins active tracking of the specified color, but leaves the pan angle fixed. Results are returned on port 5001.

```
<V = 86>
<rmin><rmax><gmin><gmax><bmin><bmax>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveColorTrackingNoMove Packet

Begins active tracking of the specified color, but does not move the camera at all. Results are returned on port 5001.

```
<A = 65>
<rmin><rmax><gmin><gmax><bmin><bmax>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveWindowTracking Packet

Begins active tracking of the currently set window, following the object with the camera. Results are returned on port 5001.

```
<W = 87>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveWindowTrackingNoTilt Packet

Begins active tracking of the currently set window, but leaves the tilt angle fixed. Results are returned on port 5001.

```
<X = 88>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveWindowTrackingNoPan Packet

Begins active tracking of the currently set window, but leaves the pan angle fixed Results are returned on port 5001.

```
<Y = 89>
<csum>
```

The iPaq returns a MiniResponsePacket.

ActiveWindowTrackingNoMove Packet

Begins active tracking of the currently set window, but does not move the camera at all. Results are returned on port 5001.

```
<Z = 90>
<csum>
```

StopActiveTracking Packet

Cancels any previously set active tracking commands.

```
<N = 78>
<csum>
```

The iPaq returns a MiniResponsePacket.

Safety Packet

Sets the safety such that if the iPaq notes a range-finder reading less than the specified value, it immediately set the velocity to zero.

If distance==0, disables any such safety.

```
<S = 83>
<distance>
<csum>
```

The iPaq returns a MiniResponsePacket.

DumpFrame Packet

Tells the iPaq to grab the current camera image and send it back on port 5002.

```
<D = 68>
<csum>
```

The iPaq returns a MiniResponsePacket immediately stating whether or not it understood the command. All further commmunication regarding the frame dump comes back on port 5002.

CMUcam Packet

`<<command>>` is a variable-length field of length `<length>`.

The iPaq will set the serial switch to communicate with the CMUcam, then pass along `<<command>>` verbatim. Does not include the required terminating carriage return.

```
<C = 67>
<length>
<<command>>
<csum>
```

The iPaq returns a StringResponsePacket of the following form, unless it fails to parse the incoming packet, in which case it returns a MiniResponsePacket:

`<status>` is the status byte, used to indicate if the iPaq understood the previous command packet. 0 is success, anything else is an error, indicating no action was taken.

`<<response>>` is a variable-length field of length `<length>`, which includes the CMUcam's response up to the first colon read.

```
<status>
<length>
<<response>>
<csum>
```

BrainStem Packet

`<<command>>` is a variable-length field of length `<length>`.

The iPaq will set the serial switch to communicate with the BrainStem, then pass along `<<command>>` verbatim.

```
<B = 66>
<length>
<<command>>
<csum>
```

The iPaq returns a StringResponsePacket of the following form, unless it fails to parse the incoming packet, in which case it returns a MiniResponsePacket:

`<status>` is the status byte, used to indicate if the iPaq understood the previous command packet. 0 is success, anything else is an error, indicating no action was taken.

`<<response>>` is a variable-length field of length `<length>`, which includes the BrainStem's response.

```
<status>
<length>
<<response>>
<csum>
```

5001 does not accept input. It returns framegrabs from the camera, when asked for via port 5000. The framegrab should start with the “ACK” and contain all camera responses through the terminating 3 and then the colon. Channel control commands (255,56 and 255,55) are NOT used on this port.

5002 does not accept input. It continuously returns data from active tracking, when enabled via port 5000. When line mode is not enabled, packets are of the form:

```
<pan>
<tilt>
<trackX>
<trackY>
<trackWidth>
<trackHeight>
<trackPixels>
<trackConf>
<csum>
```

When line mode is enabled, packets returned alternate between containing tracking results of the form above, and bitmap data of the form below.

The bitmap packet consists of a single variable-length field containing the bitmap itself; for a standard full-size window, it will be 480 bytes. Note that this is a full-fledged packet, delimited by \255\56 and \255\55, and with all message bytes equal to 255 duplexed. (The iPaq should just take all bytes returned by the camera between the xAA and the xAAxAA (but not including those) and treat them as as any other packet. Note that we are not bothering with a checksum.)

```
<<bitmap>>
```