RS-232 communications protocol

Version 1.11

M. Jury

Table of Contents

[Introduction 3](#_Toc512159778)

[Hardware connectivity 3](#_Toc512159779)

[Connectivity over Bluetooth 3](#_Toc512159780)

[Connectivity using wrist cabling 3](#_Toc512159781)

[Communications Introduction 4](#_Toc512159782)

[Bluetooth connection commands 5](#_Toc512159783)

[Hand control – Packet commands 6](#_Toc512159784)

[Structure 6](#_Toc512159785)

[Link Packet Example 6](#_Toc512159786)

[Details packet 6](#_Toc512159787)

[Status packet 7](#_Toc512159788)

[Grip pattern select 8](#_Toc512159789)

[Move grip pattern 8](#_Toc512159790)

[Finger move 8](#_Toc512159791)

[Group Finger move 9](#_Toc512159792)

[Accessing motor encoder positions 10](#_Toc512159793)

[Appendix 1 – Sundry commands 12](#_Toc512159794)

[Setting Wrist Baud Rates 12](#_Toc512159795)

[Disabling A/B analogue sensor inputs 12](#_Toc512159796)

[Bluetooth Link reset 12](#_Toc512159797)

[Appendix 2 – Displaying packets in HandCal-Dev 13](#_Toc512159798)

# Introduction

The TASKA hand supports an RS232 communications protocol which provides access to;

1. All configurable features (user and hidden)
2. Diagnostic counters
3. Per digit control.

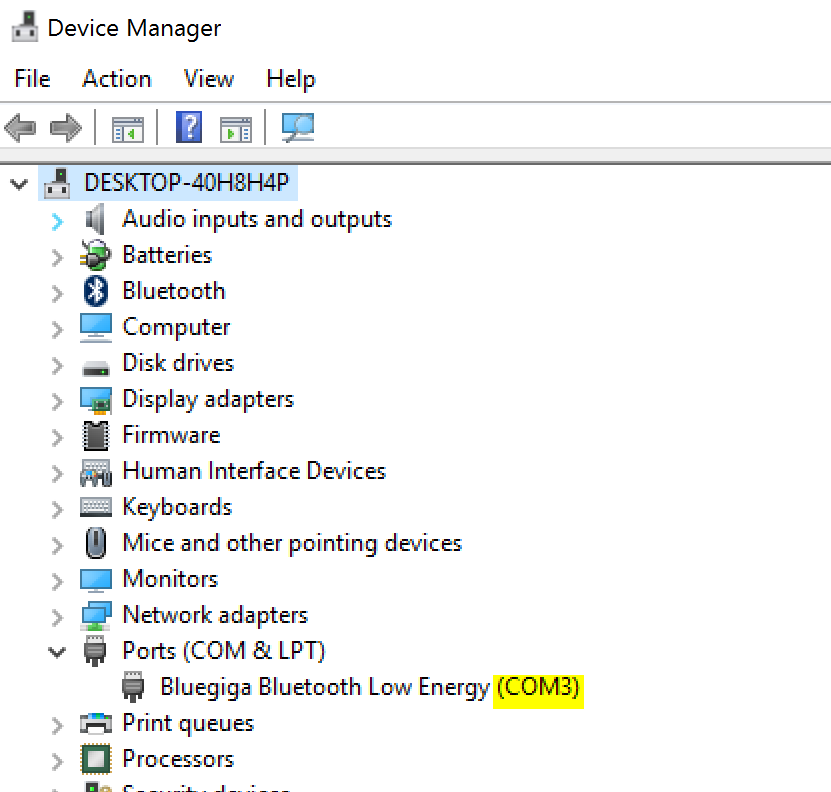
This document is currently an introduction to the most commonly used features used by researchers to control and query the TASKA hand.

# Hardware connectivity

RS232 connectivity to the TASKA is available over Bluetooth using the supplied Bluetooth dongle or via fixed wiring using either of the wrist product options. At this time (Since firmware “mainCPU” version 11.3) the RS232 communications rate is at 4800 8N1 (big-endian for Mat-Lab users) to the main hand controller referred to as “mainCPU”. This rate can be increased, see Appendix 1 – Sundry commands.

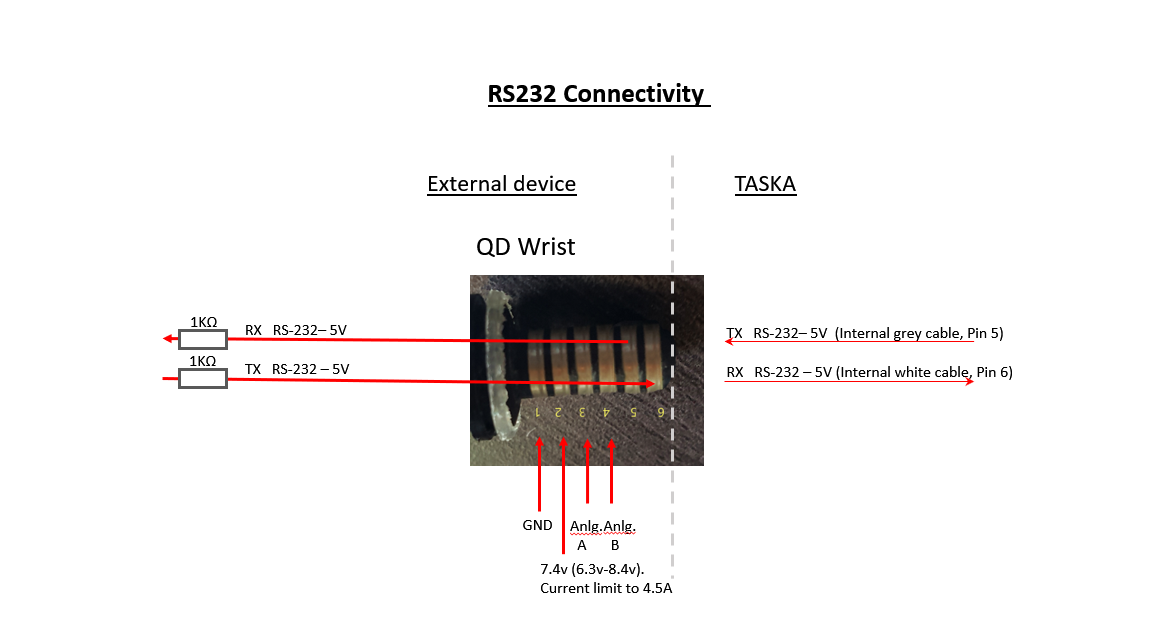
## Connectivity over Bluetooth

With the Bluetooth on the hand enabled, COMs can be achieved using the TASKA Blue Tooth dongle. The COM port of the dongle can be identified by looking at the device manager while the dongle is installed. An example is shown below.



## Connectivity using wrist cabling

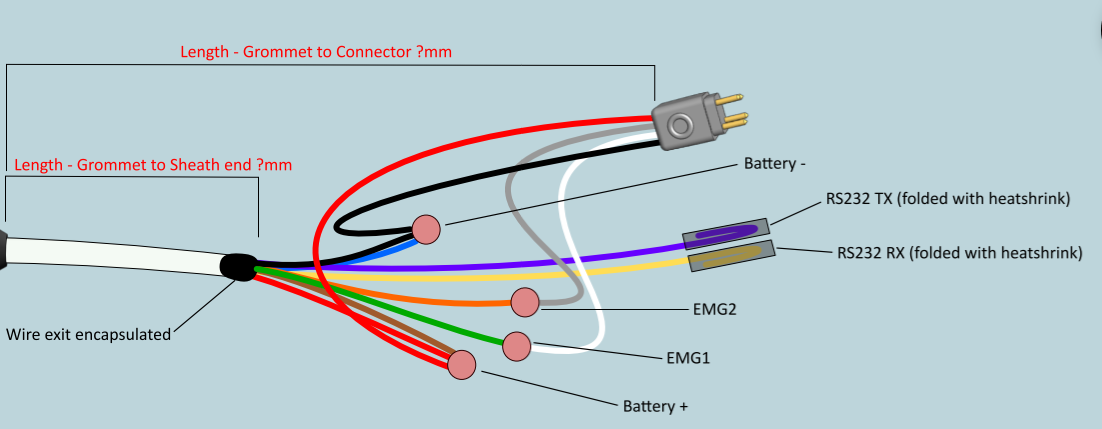
For compatibility with production systems the wrist cable RS232 access is always set to 4800bps 8N1 on power up and is accessed via Quick Disconnect Wrist pins 5 and 6 as shown below. Communications via the wrist cable RS232 lines will only function when the BlueTooth of the Taska hand is disabled. The baud rate of the wrist can be temporarily changed via command (see Appendix 1 – Sundry commands) for higher bauds rates, however the baud rate is always reset to 4800bps on power up.



3K3Ω

3K3Ω

In the case of the LOW PROFILE wrist



The native voltage of the RS232 connectivity is 5v. If an external device uses 3.3V, this can be accommodated by attenuating the received TX signal from the TASKA across a 22K/33K divider to reduce the voltage of the TX to 3.3V. An RX signal of 3.3V will function correctly to the TASKA hand.

# Communications Introduction

The TASKA hand will;

1. Always respond to a correctly formatted packet with a response (this may be requested data or just an acknowledgement depending on the command type)
2. Never respond to a corrupt or malformed packets
3. Never initiate a packet to the externally connected RS232 device.

In a communications sense, this makes the TASKA a “slave” device to commands.

# Bluetooth connection commands

When Bluetooth is used to connect to the TASKA, the additional step of scanning for hand units and selecting the correct unit based on MAC address has to be completed before hand control can begin.

Direct wire connected approaches can ignore this section.

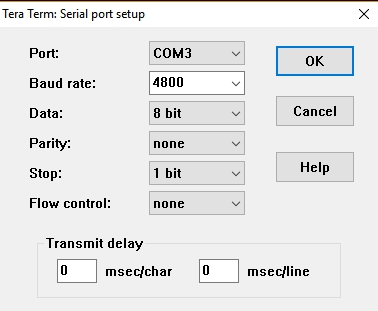
This preliminary connection process is managed with the following ASCII commands by the local dongle command handler attached to the COM port;

|  |  |
| --- | --- |
| Command – Character strings  All commands are followed by a <CR> [0x0d] | Action |
| at | Reset Connection |
| atd | Connect to the first TASKA hand that can be found |
| atd <MAC address> | Connect to a specific TASKA hand (by MAC address) |
| ats | Scan to find and report the MAC address of all TASKA hands |
| ath | Terminate ATS Scan |

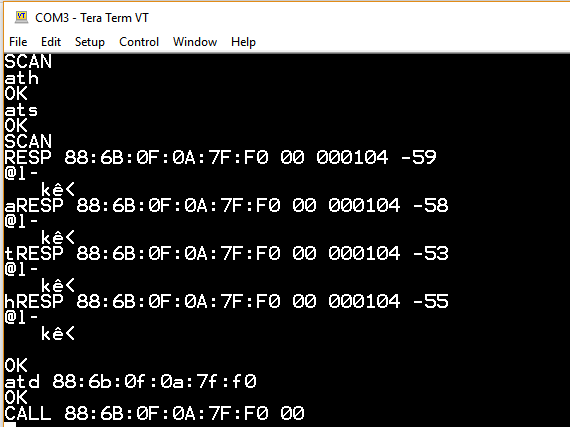
The following should be noted regarding the responses to these commands:

1. The response text to the above AT commands is primarily in a binary format intended for packet processing, however ATS command will show hands discovered in user readable ASCII characters as well as packets.
2. Once a hand connection has been established, these commands are no longer applicable.
3. Once a hand connection has been established, a link reset *packet* (see appendix 1) has to be sent to the handCPU before the Bluetooth link is released by the dongle and the connection process is started again.
4. The connection process can be started again by turning the Bluetooth (or power ) at the hand off and on again.

Example tera term configuration.



Below is an example of ATS scan with one TASKA hand responding. The ATD command is then used to successfully connect to the hand.



# Hand control – Packet commands

## Structure

Once the Bluetooth connection is established or if communications is being attempted directly, the following binary (unsigned byte not ASCII character) packet structure is used;

Transmit to hand: <‘#’> <command\_type> <sub\_index> <Packet Length> <data1> <data2>…<data30> <checksum>

Response from hand: <‘@’> <command\_type> <sub\_index> <Packet Length> <data1> <data2>…<data30> <checksum>

Notes:

< > : byte size field

<‘x’>: indicates that the binary code for that ASCII character should be used. In this case ‘x’ = 120

The number of Data fields is dependant of the command sent.

<checksum> is the modulus 256 sum of all previous bytes of the current packet.

## Link Packet Example

First example command, “**l**ink packet”. A link packet can be used to confirm that the hand is alive. The command type is an ‘l’, sub\_index is 1 and packet length is 5. The response sub\_index will always be 1 from the hand. This command is useful for initiating communications to confirm the mainCPU of the hand is ready for tasks.

Transmit: <‘#’> <’**l**’> <1> <5> <149>

Checksum calculation: 35+108+1+5 = 149

Response: <‘@’> <**l**> <1> <5> <178>

Checksum calculation: 64+108+1+5 = 178

## Details packet

The **d**etails packet reports firmware version numbers of CPUs, the hand serial number and system alarms. It should be considered an important connection packet after a link packet to confirm CPU firmware versions are at the expected values.

Transmit: <‘#’> <’**d**’> <99> <5> <checksum>

Response: <‘@’> <**’d’**> <99> <34>

**<mainCPU\_Major\_version>**

**<mainCPU\_Minor\_version>**

<Hardware\_Version>

<Product\_ID\_1>

<Product\_ID\_2>

<Product\_ID\_3>

**<motorCPU1\_version>**

**<motorCPU2\_version>**

**<motorCPU3\_version>**

<reserved>

<reserved>

<uptime MSB>

<uptime CSB>

<uptime LSB>

<CPU temperature>

<CPU Humidity>

<Alarm\_log1>

<Alarm\_log2>

<Alarm\_log3>

<Alarm\_log4>

<Alarm\_log5>

<Alarm\_log6>

<serial\_ID\_1>

<serial\_ID\_2>

<serial\_ID\_3>

<serial\_ID\_4>

<serial\_ID\_5>

<serial\_ID\_6>

<configuration\_valid>

<checksum>

## Status packet

The **s**tatus packet is typically used to poll the hand for reporting sensors levels, current grip pattern, alarm flags, battery level, virtual grip positions (from the mainCPU model) several times per second.

There are two variations of the status poll packet, “Normal” and “Training activity”. The Training activity variant is a particularly short packet to allow for high frequency responses but halts digit movement while the Training Activity packets are being sent.

**Normal Status Poll**

Transmit: <‘#’> <’**s**’> <0> <5> <checksum>

Response: <‘@’> <**’s’**> <0> <22>

**<Grip\_id>**

**<virtual\_index\_position>**

**<virtual\_middle\_position>**

**<virtual\_ring\_position>**

**<virtual\_little\_position>**

**<virtual\_Thumb\_position>**

**<virtual\_Thumb\_rotator\_position>**

**<virtual\_grip\_position>**

<button\_index\_position>

<button\_selected>

**<close\_sensor\_value>**

**<open\_sensor\_value>**

<Environment\_alarm\_flags\_index>

<Sundry\_flags\_index>

<Battery\_percentage\_used>

<Sequence\_Number>

<Debug\_byte>

<checksum>

**Training activity Status Poll**

Transmit: <‘#’> <’**s**’> <0> <5> <checksum>

Response: <‘@’> <**’s’**> <0> <22>

**<close\_sensor\_value>**

**<open\_sensor\_value>**

<Sundry\_flags\_index>

<checksum>

## Grip pattern select

To select a specific grip pattern (**G**rip\_id) use the following command with the **G**rip\_id \_id required. The current **G**rip\_id can be obtained from the previously described status packet if required. The response packet contains no data. It is simply a command acknowledgement. The **G**rip\_id must be in the range of 0-23.

Transmit: <‘#’> <’**G**’> <2> <6> <grip\_id> <checksum>

Response: <‘@’> <**’G’**> <2> <5> <checksum>

Grip ID list

#define GRIP\_INTERIM 0 // Not currently used.

#define GRIP\_RELAXED 1

#define GRIP\_OPENPALM 2

#define GRIP\_KEYBOARD 3

#define GRIP\_DONDOFF 4

#define GRIP\_POINTER 5

#define GRIP\_KEY 6

#define GRIP\_OPPOSITION 7

#define GRIP\_HANDSHAKE 8

#define GRIP\_TRIPOD 9

#define GRIP\_MUG\_CLOSE\_ONLY 10

#define GRIP\_PINCER 11

#define GRIP\_ADDUCTION\_TABLET 12

#define GRIP\_FLEX\_TOOL 13

#define GRIP\_OPPOSITION\_PRECISION 14

#define GRIP\_GRAB\_N\_GO 15

#define GRIP\_MOUSE 16

#define GRIP\_ACTIVEINDEX 17

#define GRIP\_ACTIVEINDEX2 18

#define GRIP\_CUSTOM1 19

#define GRIP\_CUSTOM2 20

#define GRIP\_CUSTOM3 21

#define GRIP\_CUSTOM4 22

#define GRIP\_CUSTOM5 23

## Move grip pattern

To move each digit in its respective range of the current **G**rip\_id, use the following command. Range is 0-255, where 0 is fully open and 255 is fully closed.

Transmit: <‘#’> <’**G**’> <3> <6> <grip\_position> <checksum>

Response: <‘@’> <**’G’**> <3> <5> <checksum>

## Finger move

For independent **F**inger (motor) control use the following;

Transmit: <‘#’> <’**F**’> <finger\_index> <10>

<finger\_speed>

<position>

<amperage>

<0>

<stall\_time>

<checksum>

Response: <‘@’> <**’F’**> <finger\_index> <5> <checksum>

Parameter values

finger\_index

INDEX 0

MIDDLE 1

RING 2

PINKY 3

THUMB 4

THUMB ROTATOR 5

finger\_speed

This is in Revolutions per second. Range 0-255

where

* 255 is a special value that overrides speed control and runs as fast as the current power supply will allow.
* 0 is a special value that stops the current move.

position

Range of 0-255 where 0 is fully open and 255 is fully closed.

amperage

Maximum current draw of digit in 10’s of milliamps.

stall\_time

Period of time the digit will maintain a stall position or current draw at the configured amperage in 10s of milliseconds.

**Warning** – Excessive stall time will potentially burn out the motors of the TASKA hand. Normal values are considered to be 500mSec or less.

## Group Finger move

To move all fingers independently at once using a single packet for optimized communication, use the “Group **F**inger Move” command. This will allow for high efficiency in RS232 communications at low baud rates. Current tests show that the group finger move command can be sent 12 times per second at 4800bps. All fingers positions and speeds can be addressed with the command. The stall time (10s of mSec resolution) and Amperage (10s of milli-amp resolution) parameters are global to all digits.

Transmit: <‘#’> <’**F**’> <255> <20>

<index\_position>

<middle\_position>

<ring\_position>

<little\_position>

<thumb\_position>

<thumb\_rotator\_position>

<index\_speed>

<middle\_speed>

<ring\_speed>

<little\_speed>

<thumb\_speed>

<thumb\_rotator\_speed>

<max\_amperage\_of\_all\_digits>

<0>

<stall\_time>

<checksum>

Response: <‘@’> <**’F’**> <255> <5> <checksum>

## Accessing motor encoder positions

All previous commands described in this document result in commands being sent from the mainCPU to a motor controller (known as “motorCPUs”) that move digits using encoder indexed feedback for positioning. The target positions are not validated as being reached by the mainCPU after a move command is sent to the motorCPU. In real world applications, a finger stall when an obstruction is encountered is common. The simplex communication method from mainCPU to motorCPU results in the virtual finger model positions reported in the status packet from the mainCPU and the actual achieved position not necessarily being in-sync due to possible digit stalls-outs.

It is possible to read the digit positions directly from the motorCPUs to validate their position in research applications. The first step is to enable the encoder read flag on each **m**otorCPU with the following, one-time commands (saved in TASKA FLASH memory):

Transmit: <‘#’> <’**M**’> <11> <11><1><64> <0> <0> <10> <127> <80>

Response: <’@’> <’**M**’> <11> <6> <11> <169>

Transmit: <‘#’> <’**M**’> <11> <11><2><64> <0> <0> <10> <127> <81>

Response: <’@’> <’**M**’> <11> <6> <11> <169>

Transmit: <‘#’> <’**M**’> <12> <11><1><64> <0> <0> <10> <127> <81>

Response: <’@’> <’**M**’> <12> <6> <11> <170>

Transmit: <‘#’> <’**M**’> <12> <11><2><64> <0> <0> <10> <127> <82>

Response: <’@’> <’**M**’> <12> <6> <11> <170>

Transmit: <‘#’> <’**M**’> <13> <11><1><64> <0> <0> <10> <127> <82>

Response: <’@’> <’**M**’> <13> <6> <11> <171>

Transmit: <‘#’> <’**M**’> <13> <11><2><64> <0> <0> <10> <127> <83>

Response: <’@’> <’**M**’> <13> <6> <11> <171>

Each motorCPU controls a pair of motors (digits). The following command reads status information of the motor pair from a motorCPU.

Transmit: <‘#’> <’**m**’> <I2C\_INDEX> <5><checksum>

Where I2C\_INDEX can be one of the following motorCPUs

11 – (Motor 1) Thumb Rotation and (Motor 2) little finger

12 – (Motor 1) Ring and (Motor 2) Middle finger

13 – (Motor 1) Index and Motor (2) Thumb

The response from the hand will be in two packets as follows. The fields indicating the encoder value (the current position of a digit) and max encoder range for that digit are highlighted below. In some cases a 16 bit value is returned in two byte portions, an “\_LSB” and “\_MSB” which refer to the Most Significant Byte and Least Significant Byte respectively. To combine these bytes for the correct value use the formulae: \_MSB\*256 + \_LSB .

Response packet1: <’@’> <’**m**’> <I2C\_INDEX> <21> <1>

<Motor1\_temperature + 20>

<Motor1\_status\_flags>

**<Motor1\_position\_LSB>**

**<Motor1\_position\_MSB>**

<Motor1\_speed\_LSB>

<Motor1\_speed\_MSB>

<Motor1\_amperage\_LSB>

<Motor1\_amperage\_MSB>

**<Motor1\_max\_range\_LSB>**

**<Motor1\_max\_range\_MSB>**

<reserved>

<reserved>

<reserved>

<reserved>

<Motor1\_direction>

<checksum>

Response packet2: <’@’> <’**m**’> <I2C\_INDEX> <26> <2>

<Motor2\_temperature + 20>

<Motor2\_status\_flags>

**<Motor2\_position\_LSB>**

**<Motor2\_position\_MSB>**

<Motor2\_speed\_LSB>

<Motor2\_speed\_MSB>

<Motor2\_amperage\_LSB>

<Motor2\_amperage\_MSB>

**<Motor2\_max\_range\_LSB>**

**<Motor2\_max\_range\_MSB>**

<reserved>

<reserved>

<reserved>

<reserved>

<Motor2\_direction>

<total\_I2C\_bytes1>

<total\_I2C\_bytes2>

<total\_I2C\_bytes3>

<total\_I2C\_bytes4>

<motorCPU\_firmware\_version>

<checksum>

# Appendix 1 – Sundry commands

### Setting Wrist Baud Rates

The wrist cable R232 baud rate can be set with the commands below. The baud rate is always reset to 4800bps at power up. The command is relatively recent; supported from mainCPU version 11.8.

Transmit: <‘#’> <’**D**’> <16> <6><baud\_rate\_value><checksum>

Response: <’@’> <’**D**’> <16> <5> <checksum>

|  |  |
| --- | --- |
| Baud rate value | Set speed |
| 0 (Default/Power up) | 4800 |
| 1 | 9600 |
| 2 | 19200 |
| 3 | 38400 |
| 4 | 57600 |
| 5 | 115200 |
| Any other value | 4800 |

Note that requesting commands from the Taska hand at greater that the default baud rate may not be 100% reliable due to buffering and flow control limitations. Appropriate packet integrity checking (checksum, packet length and packet start byte confirmation) should be performed at all times by 3rd party devices.

### Disabling A/B analogue sensor inputs

The follow commands will disable or enable the A/B analogue sensors. This can be useful when complete control of the hand movements are performed only via RS232 commands.

Disable A/B sensors:

Transmit: <‘#’> <’**G**’> <4> <7><0><0><checksum>

Response: <’@’> <’**G**’> <4> <5> <checksum>

Enable A/B Sensors:

Transmit: <‘#’> <’**G**’> <5> <5><checksum>

Response: <’@’> <’**G**’> <5> <5> <checksum>

### Bluetooth Link reset

To disconnect from a TASKA hand Bluetooth connection, the following reset command is required.

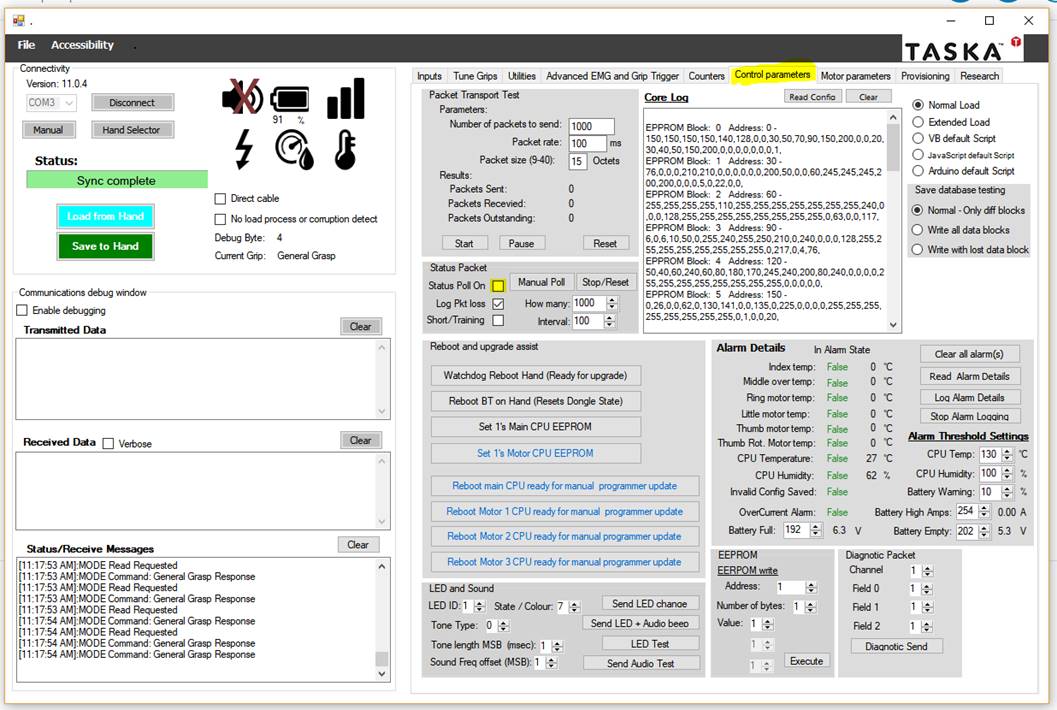
Transmit: <‘#’> <’**R**’> <14> <5><checksum>

Response: <’@’> <’**R**’> <14> <5> <checksum>

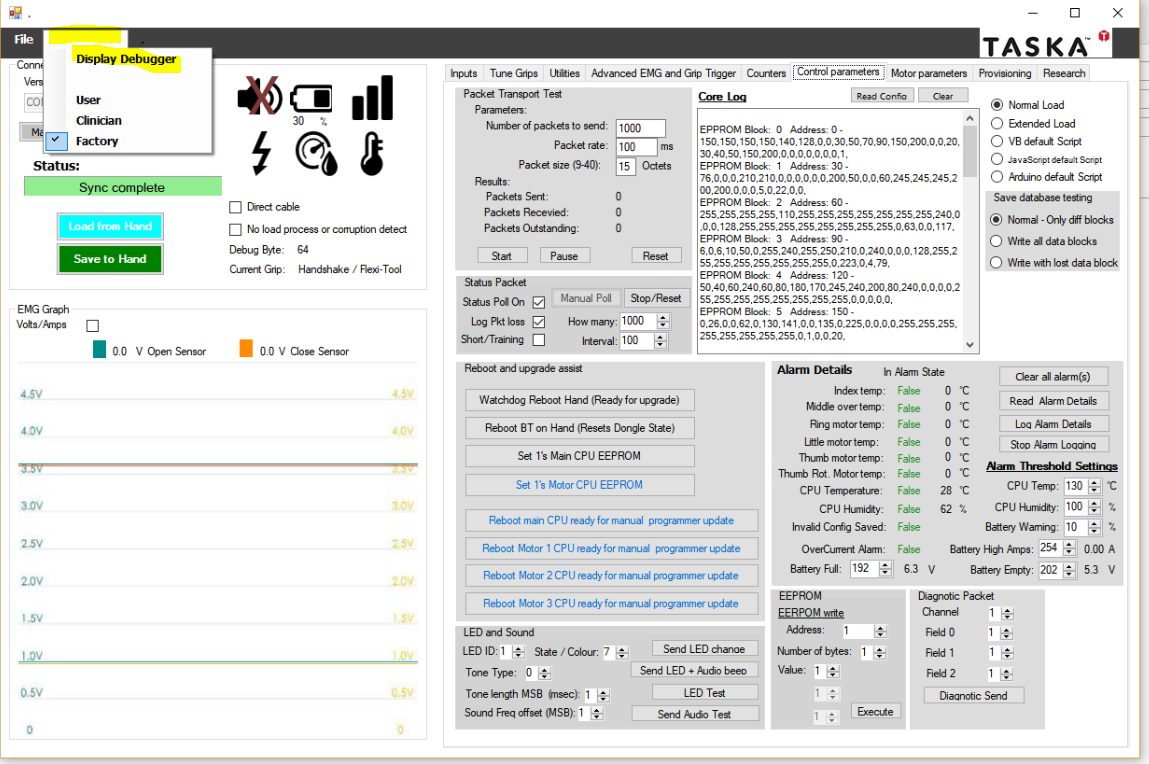
Note: Bluetooth is still enabled after this command.

# Appendix 2 – Displaying packets in HandCal-Dev

1. After connection, turn off status poll.



2) Enable the debug window



3) Turn on packet debugging.

