

## Packet Descriptions:

### Ping

<i>Header</i>	A	[data]	...														
<i>Header</i>	a	[data]	...														

Ping provides a simple confirmation of device operation and communication. Responds with a copy of the data sent.

### Version

<i>Header</i>	B																
<i>Header</i>	b	local_addr	device_type	major_ver	minor_ver	reset_type											

Version responds with the current software version number.

- local\_addr = RDB device address
- device\_type = RDB device type
- major\_ver = Major version number
- minor\_ver = Minor version number
- reset\_type = Reason for sending Version packet
  - 0: No reset, response to Version packet request
  - 'P': Power-on reset
  - 'S': Software (warm) reset
  - 'W': Watchdog reset

### Shutup

<i>Header</i>	C																
<i>Header</i>	c																

Shutup stops all scheduled RDB transmission.

### Schedule

<i>Header</i>	E	pkt_time	pkt_dest														
<i>Header</i>	e																

Schedule defines the interval and destination for sending scheduled 'I' and 'J' packets.

- pkt\_time = Time interval (1 second steps) between sending 'I' packets. Interval for 'J' packets = 5\*pkt\_time
- pkt\_dest = RDB destination address for sending scheduled 'I' and 'J' packets

### Horn

<i>Header</i>	H	on_time															
<i>Header</i>	h																

Horn sounds the horn for the specified time interval.

- on\_time = Time interval for sounding horn (0.1 second steps)

## Info

Header	I															
Header	i	speed_H	speed_L	cruise	status	volt_H	volt_L	I_H	I_L	motor_T	heatsink_T	fault_1	fault_2	fault_3	fault_4	Input_I

Info sends the primary set of data obtained from/about the motor, controller, and car.

speed\_H:L = Speed reported by motor controller (RAM 0x0C), **deci-MPH**  
Returns 0xFF:0xFF if motor controller communication is lost

cruise = Set speed for cruise control (MPH)

status = bit.7: current limit  
bit.6: brake active  
bit.5: cruise active  
bit.4: hazards active  
bit.3: right turn active  
bit.2: left turn active  
bit.1: direction (1=forward, 0=reverse)  
bit.0: enable (1=enabled, 0=disabled)

volt\_H:L = System voltage reported by motor controller (RAM 0x64), **deci-Volts**

I\_H<sub>0-3</sub>:L = Phase current reported by motor controller (RAM 0x10), 12-bit value, **deci-Amps**

motor\_T = Motor temperature reported by motor controller (RAM 0x65), °C

heatsink\_T = Heatsink temperature reported by motor controller (RAM 0x66), °C

fault\_1:2:3:4 = Fault register data reported by motor controller (RAM 0x98, 0x9A, 0x9B, 0x9C)

I\_H<sub>4-7</sub>:Input\_I = Input current to motor controller from A/D converter, 12-bit value, **Amps**; Conversion equation:  $\frac{[(I_{H4-7} \cdot 256) + \text{input}_1] \cdot 250}{4096} - 125$

## Info2

Header	J															
Header	j	timeout	badchar	query_cycle	loop_cycle	odo_H	odo_M	odo_L	coef_spd_H	coef_spd_L						

Info2 sends the secondary set of data obtained from/about the motor, controller, and car.

timeout = Number of timeouts with motor controller in previous second

badchar = Number of bad characters received from motor controller in previous second

query\_cycle = Number of motor controller queries in previous second

loop\_cycle = Number of complete loops through motor controller queries in previous second

odo\_H:M:L = Odometer reading, see equation below

coef\_spd\_H:L = Coefficient for converting revolutions to miles, see equation below

$$\text{Odometer}(\text{miles}) = \left( \frac{\text{odo} \cdot 256}{9} (\text{rev}) \right) \left( \frac{\text{coef\_spd}}{65536 \cdot 60 \cdot 10} \left( \frac{\text{miles}}{\text{rev}} \right) \right)$$

## Configuration

<i>Header</i>	L	type	'='	[data]	...											
<i>Header</i>	l	type	'='	[data]	...											

Configuration redefines and sets certain variables in the system. Responds with a copy of the data sent.

type = ASCII character representing the variable to change

- 'D': Display intensity [data+0] = {0x00 ≤ val ≤ 0x0F}
- 'S': RPM to MPH coefficient [data+0] = high byte, [data+1] = low byte  $MPH = RPM \cdot \frac{coef}{65536}$
- 'H': Cruise override hysteresis current [data+0] = val, **deci-Amps**
- 'U': Cruise underspeed lockout speed [data+0] = val, **deci-MPH**
- 'O': Odometer value [data+0] = high byte, [data+1] = middle byte, [data+2] = low byte

## Motor

<i>Header</i>	M	[data]	...	13 (0x0D)												
<i>Header</i>	m	[data]	...													

Motor sends serial information to the motor controller and responds with the motor controller's response.

Responds with "ERROR" if motor controller communication fails.

## Query

<i>Header</i>	Q	addr														
<i>Header</i>	q	addr	val													

Query returns the specified RAM location in the PIC.

addr = RAM location address  
 val = Value at RAM location <addr>

## Reset

<i>Header</i>	R	type														
<i>Header</i>	r															

Reset resets various components including CCB itself.

type = Signifies type of reset

- 0: Warm reset, reinitializes then restarts PIC
- 1: Watchdog, forces a watchdog reset on the PIC
- 'M': Motor, reinitializes then resets the motor controller

## Turns

<i>Header</i>	T	command														
<i>Header</i>	t	command														

Turns controls the status of the turn signals. Responds with a copy of the command sent.

command = Specifies action to be taken with turn signals

‘L’: Turn on left turn

‘R’: Turn on right turn

‘H’: Turn on hazards

‘O’: Turn off all lights