Raptor interface protocol over RS232.

Standard user commands

Information about Raptor www.raptor-digital.eu

Version 1.1 October 2013.

1.0 GENERAL INFORMATION

The general characteristics of the Raptor interface protocol messages over RS232 is as follows:
- Baudrate 9600, 8 bits, no parity, 2 stopbits
- Full duplex communication
- **No** handshake CTS, RTS, XON, XOFF required
- All command lines in (printable) ASCII format ending with Newline character and/or Linefeed character
- **No** timing requirements

The Raptor interface protocol is easy to implement for any kind of PC software. The fact that there are no timing requirements means that the PC may send characters at any time, with or without delays. Therefor it is also possible to command the interface even through any terminal program(like "Hyperterm") and type in the Raptor "***command***" by hand. (However, when using a terminal program keep in mind that the interface has "full duplex" communication capabilities and therefor Raptor may send "***info***" data at any time to your screen as well.)

Each interface command/message has the following format:

| COMMAND NAME | DATA ARGUMENT SECTION | COMMAND LINE SEPARATOR

The command is separated from the data section with at least one space character. Any data arguments are also separated with at least one space character from each other. The interface starts processing the command after an ending "command line separator" Newline character and/or Linefeed character has been received.

Command name:
Each command is defined by a fixed name (like "BLOck") and *all* are character case **in**sensitive. The first three characters of a command are mandatory while the rest is optional. For example is the input of the string "blo" a valid "BLOck" command. Raptor however always replies with full defined command names.

Data Argument Section:
This section contains the command argument(s), if any, which defines the parsing process of the Data section fields separated with space characters. Depending on the command, data arguments might be character case sensitive. See the individual command descriptions for details. When an argument in the individual command description is depicted between format brackets "<>" then the argument is *required*. When an argument in the individual command description is depicted between format brackets "**[]**" then the argument is *optional* thus may be omitted. A depicted "|" character separates possible choices of exact argument definitions.

2.0 COMMAND DATA MESSAGES

2.0 Command: Go or Halt the Raptor central unit.

POWer command: <<< "**POW** <0|1>"
where <1> = Go.
<0> = Halt.

POWer info: >>> "POWer <0|1>"
where <1> = Raptor(s) have been put in Go state(yellow).
<0> = Raptor(s) have been put in Halt state(red).

2.1 Command: Start or Stop driving the trains.

DRIve command: <<< "**DRI** <0|1>"
where <1> = Start the train drive.(RUN)
<0> = Stop the train drive. (HALT)

DRIve info: >>> "DRIve <0|1>"
where <1> = Train drive started.
<0> = Train drive stopped.

2.2 Command: POWer and DRIve combined into RUNlevel.

RUNlevel command: <<< "**RUN** <0|1|2>"
where <2> = RUN(green).
<1> = Go(yellow).
<0> = Halt(red).

RUNlevel info: >>> "RUNlevel <0|1|2>"
where <2> = Train drive started(green).
<1> = Raptor(s) have been put in Go state(yellow).
<0> = Raptor(s) have been put in Halt state(red).

Note:
Sender gives this (broadcast) message to all Raptor stations to command them toward the Automatic Train Drive status given in the argument. In case a receiving station is in DRIVE(runlevel=2) status then that station stops the Automatic Train Drive and goes to Go(runlevel=1) in case argument=0. In case a receiving station is in GO(runlevel=1) status then that station starts the Automatic Train Drive and goes to DRIVE(runlevel=2) in case argument=1. The receiving station ignores this message in case it's current status is HALT(runlevel=0).

2.3 Command: Request locomotive speed and direction command.

LOK command: <<< "**LOK** <lokid>"
where <lokid> = Lok number registered within Raptor.

LOK info: >>> "LOK <lokid> <speed> <direction>"
where <lokid> = Lok number [1..999] registered within Raptor.
<speed> = Actual lok speed [0..28].
<direction> = Actual lok direction [0=forward|1=reverse].

2.4 Command: Set locomotive speed command.

LOK command: <<< "**LOK** <lokid> <speed>"
where <lokid> = Lok number registered within Raptor.
<speed> = Lok speed [0..28].

LOK info: >>> "LOK <lokid> <speed> <direction>"
where <lokid> = Lok number [1..999] registered within Raptor.
<speed> = Actual lok speed [0..28].
<direction> = Actual lok direction [0=forward|1=reverse].

Note:
With only the “Lok speed” command are the current settings of the locomotive functions and driving direction not effected.

2.5 Command: Set locomotive direction command.

LOK command: <<< "**LOK** <lokid> <speed> <direction>"
where <lokid> = Lok number registered within Raptor.
<speed> = Lok speed [0..28].
<direction> = Lok direction [0=forward|1=reverse].

LOK info: >>> "LOK <lokid> <speed> <direction>"
where <lokid> = Lok number [1..999] registered within Raptor.
<speed> = Actual lok speed [0..28].
<direction> = Actual lok direction [0=forward|1=reverse].

Note:
With the “Lok direction and speed” command are the current settings of the locomotive functions **not** effected.

2.6 Command: Set locomotive function command.

FNC command: <<< "**FNC** <lokid> <function number> <command>"
where <lokid> = Lok number registered within Raptor.
<function number> = Lok function number [0..28].
<command> = Function command [0=Off|1=On].

Note:
With the “FNC function” command are the current settings of the locomotive direction and speed **not** effected.

2.7 Command: Request LOK occupation of given block id.

BLOck command: <<< "**BLO** <blockid>"
where <blokid> = Block name registered within Raptor.

BLOck info: >>> "BLOck <blockid> <lokid>"
where <blockid> = Block name registered within Raptor.
<lokid> = Lok number occupying Block name.

2.8 Command: Switch article number/Set output decoder.

OUTput command: <<< "**OUT** <address> <requested position>"
where <address> = Decoder address [1..255] of article.
<position> = Position[R|G] to set.

OUTput info: >>> "OUT <address> <actual position>"
where <address> = Decoder address [1..255] of article set.
<position> = Position[R|G] that have been set.

Note:
Each station receiving this message shall output the article switch command settings. This command is inhibited in case the article is locked by a runway during the Automatic Train Drive. The receiving station replies with a OUTput message , indicating the new position, *after* the article has been switched completely by the receiving station.

2.9 Command: Switch runway turnout-street.

STReet command: <<< "**STR** <blockid_from> <blockid_to>"
where <blockid_van> = Block name registered within Raptor of begin/departure.
<blockid_naar> = Block name registered within Raptor of end/destination.

STReet info: >>> "By means of OUTput info data"

Note:
The receiving station executes the turnouts of runway Departure->Destination which will result in the corresponding number of OUTput messages.

2.10 Command: Relocate trains command.

MOVe command: <<< "**MOV** <lokid> <blockname>"
where <lokid> = Lok number [1..999] registered within Raptor.
<blockid> = Block name where lokid have been relocated.

MOVe info: >>> "By means of BLOck info data"

Note:
The receiving station deploys/register the Lok_id into the the block name. Replies are send back with information from which block name(s) the Lok_id has been taken(if any) and the confirmation of the new block location.

2.11 Command: Input data request command.

INPut command: <<< "**INP** <S88id> <Bitno> <Node>"
where <S88id> = S88 module number registered on the Raptor network.
<Bitno> = S88 connection number [1..16].
<Node> = Raptor (ID)Node to address [0=all].

INput info: >>> "INPut <S88id> <Bitno> <Node> <Databit>"
where <S88id> = S88 module number registered on the Raptor network.
<Bitno> = S88 connection number [1..16].
<Node> = *Responding* Raptor (ID)Node.
<Databit> = S88 connection number data [0,1].

Note:
With the "INPut" command are the current collections of Raptor used automatic S88 data **not** effected. The Bit number is automatically "reset" after reading with this command without affecting the data status of the other (interface reserved) bits.

This "INPut" command is ***not*** designed and ***not*** intended to be used as a part of train controlling by a computer. However, its purpose is intended to provide the status of "switches" and other graphical computer designed input from the layout to fully integrate the computer software with the automatic functions of Raptor and its controlling layout. In this the computer has fully access to "grab" S88 data straight through the Raptor without interfering with the Raptor automatic control.

2.12 Command: Set node id of interface

NODe command: <<< "**NOD** [node id]"
where [node id] = new node ID to registered within this interface.

NODe info: >>> "NODe <current or set node id>"
where [node id] = node ID as is registered within this interface.

Note:
At powering up the interface the interface Node-ID is always set to zero. Meaning that all messages will be processes(send to the PC) as received from all Raptor stations. Setting the node id allows the individual reception of one Raptor station (or a group of stations) only.

2.13 Command: Request firmware version and ID of the interface.

VERsion command: <<< "**VER**"

VERsion info: >>> "VERsion 1.x IDyyyyyy"
where the "x" is a sub-version sequence number and y the Raptor ID of the Raptor with the interface connected.

Note:
This document describes the command set of the version "1" interface firmware. It is advised always let the PC to check the version number in order to use the available functionality correctly. The number of the "x" can always be ignored while this indicates just an updated version of the exact described command format of ***this*** document.