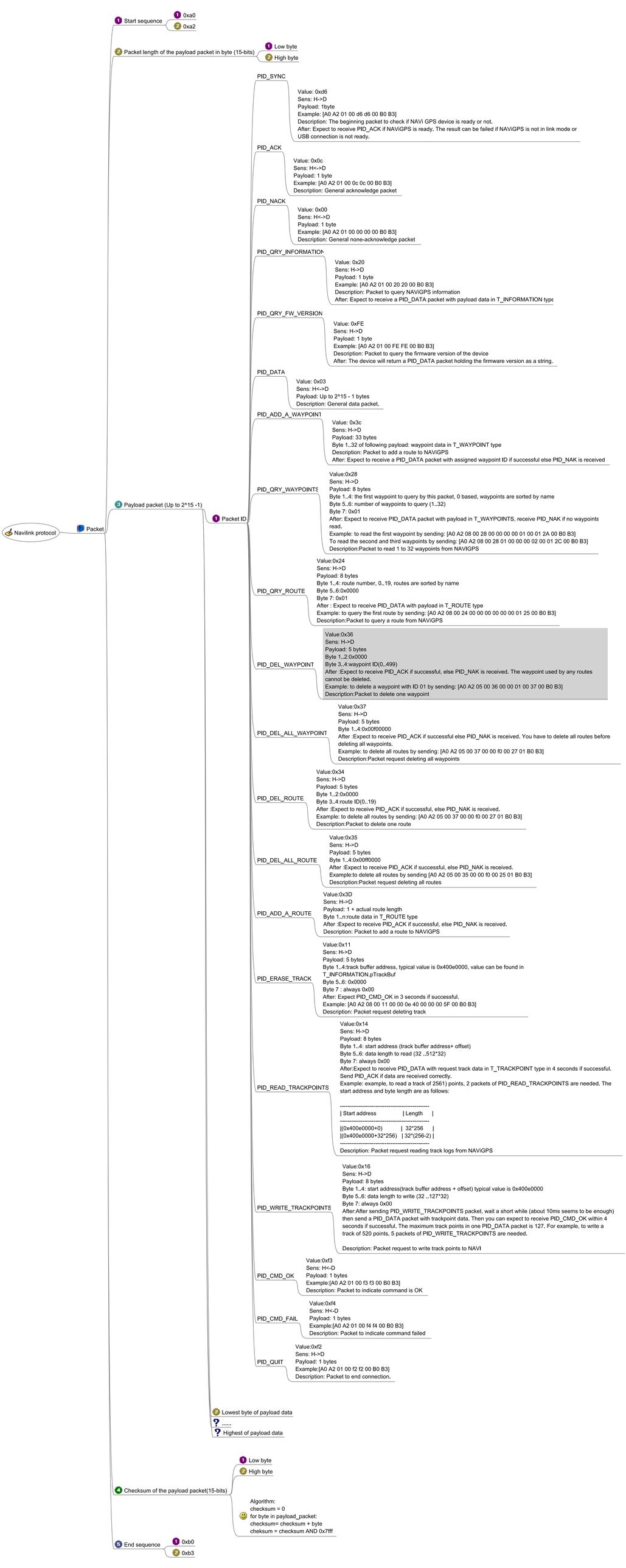
Navilink protocol
Packet
1 Start sequence
1 0xa0
2 0xa2
2 Packet length of the payload packet in byte (15-bits)
1 Low byte
2 High byte
3 Payload packet (Up to 2^15 -1)
1 Packet ID
PID_SYNC
Value: 0xd6
Sens: H->D
Payload: 1byte
Example: [A0 A2 01 00 d6 d6 00 B0 B3]
Description: The beginning packet to check if NAVi GPS device is ready or not.
After: Expect to receive PID_ACK if NAViGPS is ready. The result can be failed if NAViGPS is not in link mode or USB connection is not ready.
PID_ACK
Value: 0x0c
Sens: H<->D
Payload: 1 byte
Example: [A0 A2 01 00 0c 0c 00 B0 B3]
Description: General acknowledge packet
PID_NACK
Value: 0x00
Sens: H<->D
Payload: 1 byte
Example: [A0 A2 01 00 00 00 00 B0 B3]
Description: General none-acknowledge packet
PID_QRY_INFORMATION
Value: 0x20
Sens: H->D
Payload: 1 byte
Example: [A0 A2 01 00 20 20 00 B0 B3]
Description: Packet to query NAViGPS information
After: Expect to receive a PID_DATA packet with payload data in T_INFORMATION type
PID_QRY_FW_VERSION
Value: 0xFE
Sens: H->D
Payload: 1 byte
Example: [A0 A2 01 00 FE FE 00 B0 B3]
Description: Packet to query the firmware version of the device
After: The device will return a PID_DATA packet holding the firmware version as a string.
PID_DATA
Value: 0x03
Sens: H<->D
Payload: Up to 2^15 - 1 bytes
Description: General data packet.
PID_ADD_A_WAYPOINT
Value: 0x3c
Sens: H->D
Payload: 33 bytes
Byte 1..32 of following payload: waypoint data in T_WAYPOINT type
Description: Packet to add a route to NAViGPS
After: Expect to receive a PID_DATA packet with assigned waypoint ID if successful else PID_NAK is received
PID_QRY_WAYPOINTS
Value:0x28
Sens: H->D
Payload: 8 bytes
Byte 1..4: the first waypoint to query by this packet, 0 based, waypoints are sorted by name
Byte 5..6: number of waypoints to query (1..32)
Byte 7: 0x01
After: Expect to receive PID_DATA packet with payload in T_WAYPOINTS, receive PID_NAK if no waypoints read.
Example: to read the first waypoint by sending: [A0 A2 08 00 28 00 00 00 00 01 00 01 2A 00 B0 B3]
To read the second and third waypoints by sending: [A0 A2 08 00 28 01 00 00 00 02 00 01 2C 00 B0 B3]
Description:Packet to read 1 to 32 waypoints from NAVIGPS
PID_QRY_ROUTE
Value:0x24
Sens: H->D
Payload: 8 bytes
Byte 1..4: route number, 0..19, routes are sorted by name
Byte 5..6:0x0000
Byte 7: 0x01
After : Expect to receive PID_DATA with payload in T_ROUTE type
Example: to query the first route by sending: [A0 A2 08 00 24 00 00 00 00 00 00 01 25 00 B0 B3]
Description:Packet to query a route from NAViGPS
PID_DEL_WAYPOINT
Value:0x36
Sens: H->D
Payload: 5 bytes
Byte 1..2:0x0000
Byte 3..4:waypoint ID(0..499)
After :Expect to receive PID_ACK if successful, else PID_NAK is received. The waypoint used by any routes cannot be deleted.
Example: to delete a waypoint with ID 01 by sending: [A0 A2 05 00 36 00 00 01 00 37 00 B0 B3]
Description:Packet to delete one waypoint
PID_DEL_ALL_WAYPOINT
Value:0x37
Sens: H->D
Payload: 5 bytes
Byte 1..4:0x00f00000
After :Expect to receive PID_ACK if successful else PID_NAK is received. You have to delete all routes before deleting all waypoints.
Example: to delete all routes by sending: [A0 A2 05 00 37 00 00 f0 00 27 01 B0 B3]
Description:Packet request deleting all waypoints
PID_DEL_ROUTE
Value:0x34
Sens: H->D
Payload: 5 bytes
Byte 1..2:0x0000
Byte 3..4:route ID(0..19)
After :Expect to receive PID_ACK if successful, else PID_NAK is received.
Example: to delete all routes by sending: [A0 A2 05 00 37 00 00 f0 00 27 01 B0 B3]
Description:Packet to delete one route
PID_DEL_ALL_ROUTE
Value:0x35
Sens: H->D
Payload: 5 bytes
Byte 1..4:0x00f00000
After :Expect to receive PID_ACK if successful, else PID_NAK is received.
Example:to delete all routes by sending [A0 A2 05 00 35 00 00 f0 00 25 01 B0 B3]
Description:Packet request deleting all routes
PID_ADD_A_ROUTE
Value:0x3D
Sens: H->D
Payload: 1 + actual route length
Byte 1..n:route data in T_ROUTE type
After :Expect to receive PID_ACK if successful, else PID_NAK is received.
Description: Packet to add a route to NAViGPS
PID_ERASE_TRACK
Value:0x11
Sens: H->D
Payload: 5 bytes
Byte 1..4:track buffer address, typical value is 0x400e0000, value can be found in T_INFORMATION.pTrackBuf
Byte 5..6: 0x0000
Byte 7 : always 0x00
After: Expect PID_CMD_OK in 3 seconds if successful.
Example: [A0 A2 08 00 11 00 00 0e 40 00 00 00 5F 00 B0 B3]
Description: Packet request deleting track
PID_READ_TRACKPOINTS
Value:0x14
Sens: H->D
Payload: 8 bytes
Byte 1..4: start address (track buffer address+ offset)
Byte 5..6: data length to read (32 ..512*32)
Byte 7: always 0x00
After:Expect to receive PID_DATA with request track data in T_TRACKPOINT type in 4 seconds if successful. Send PID_ACK if data are received correctly.
Example: example, to read a track of 2561) points, 2 packets of PID_READ_TRACKPOINTS are needed. The start address and byte length are as follows:

| Start address | Length |

|(0x400e0000+0) | 32*256 |
|(0x400e0000+32*256) | 32*(256-2) |

Description: Packet request reading track logs from NAViGPS
PID_WRITE_TRACKPOINTS
Value:0x16
Sens: H->D
Payload: 8 bytes
Byte 1..4: start address(track buffer address + offset) typical value is 0x400e0000
Byte 5..6: data length to write (32 ..127*32)
Byte 7: always 0x00
After:After sending PID_WRITE_TRACKPOINTS packet, wait a short while (about 10ms seems to be enough) then send a PID_DATA packet with trackpoint data. Then you can expect to receive PID_CMD_OK within 4 seconds if successful. The maximum track points in one PID_DATA packet is 127. For example, to write a track of 520 points, 5 packets of PID_WRITE_TRACKPOINTS are needed.
Description: Packet request to write track points to NAVI
PID_CMD_OK
Value:0xf3
Sens: H<-D
Payload: 1 bytes
Example:[A0 A2 01 00 f3 f3 00 B0 B3]
Description: Packet to indicate command is OK
PID_CMD_FAIL
Value:0xf4
Sens: H<-D
Payload: 1 bytes
Example:[A0 A2 01 00 f4 f4 00 B0 B3]
Description: Packet to indicate command failed
PID_QUIT
Value:0xf2
Sens: H->D
Payload: 1 bytes
Example:[A0 A2 01 00 f2 f2 00 B0 B3]
Description: Packet to end connection.
2 Lowest byte of payload data
.....
Highest of payload data
4 Checksum of the payload packet(15-bits)
1 Low byte
2 High byte
Algorithm:
checksum = 0
for byte in payload_packet:
checksum= checksum + byte
cheksum = checksum AND 0x7fff
5 End sequence
1 0xb0
2 0xb3