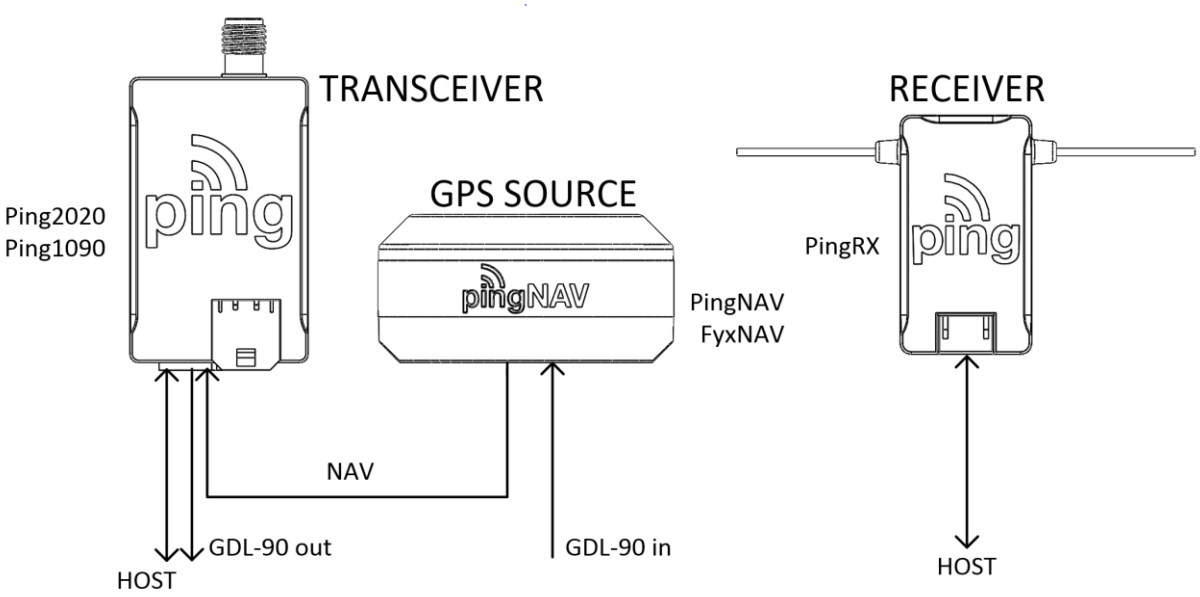
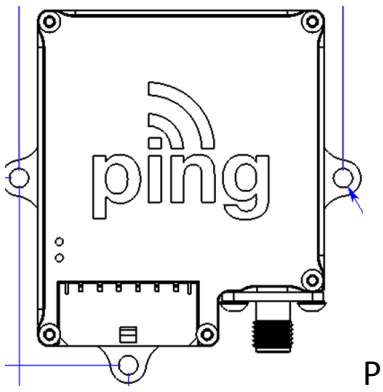
## Interfaces

The following document describes the interfaces and associated protocols on the various uAvoinix products.



PING200S Transponder

Figure 1: Interface Descriptions

| Pin | Type | Function | Definition |
|---|---|---|---|
| 1 | Input | RXD | MAVLink RXD |
| 2 | Output | TXD | MAVLink TXD |
| 3 | Power | 5V | |
| 4 | Ground | Ground | |

Table 1: HOST Interface 4pin Interface 57600bps

| Pin | Type | Function | Definition |
|---|---|---|---|
| 1 | Input | UTC | UTC in |
| 2 | Input | RXD | MAVLink RXD |
| 3 | Output | TXD | GDL90 out |
| 4 | Power | 5V | |
| 5 | Ground | Ground | |

Table 2: Transceiver NAV 5pin Interface 115200bps

| Pin | Type | Function | Definition |
|---|---|---|---|
| 1 | Output | UTC | UTC out |
| 2 | Output | TXD | MAVLink TXD |
| 3 | Input | RXD | GDL90 in |
| 4 | Power | 5V | |
| 5 | Ground | Ground | |

Table 3: GPS Source NAV 5pin Interface 115200bps

| Pin | Type | Function | Definition |
|---|---|---|---|
| 1 | Ground | | |
| 2 | Power In | 30-55V | |
| 3 | Power Out | 5V | |
| 4 | NAV RX | 115200 | MAVlink RXD |
| 5 | HOST RX | 57600 | MAVlink RXD |
| 6 | HOST TX | 57600 | MAVlink TXD |

Table 4: Ping200S Transponder 6pin Interface

## 1. GDL90 Messages

The GDL90 Data Interface Specification is a public document and can be found at:

https://www.faa.gov/nextgen/programs/adsb/wsa/media/GDL90_Public_ICD_RevA.PDF

### Physical Interface

GDL90 communication is accomplished through a full duplex asynchronous serial interface. The interface should use 3.3v logic levels with a 115200bps baud rate, no parity, 1 stop bits and 8 data bits.

| Message ID | Type | I/O | Length | SubType |
|---|---|---|---|---|
| 0 | Heartbeat | Out | 7 | |
| 7 | Uplink Data | Out | 436 | |
| 10 | Ownship Report | Out | 28 | |
| 20 | Traffic Report | Out | 28 | |
| 117 | Initialization | In | 19 | 1 |

Table 4: GDL90 Message

### 2.1 GDL90 Output Messages.

The GDL90 output message formats can be found in the GDL90 Data Interface Specification. UAT and 1090 traffic are combined in the Traffic Report (ID=20) messages.

### 2.2 GDL Input Messages

Used for programming ICAO, Emitter Type and Callsign in the NAV. These variables are available immediately after receiving the message and are stored in Flash. Data is transferred to the ping2020 at power up and every 10s.

Refer to the GDL90 Data Interface Specification for general message structure, framing, and Frame Check Sequence generation.

### Initialization Message

| | |
|---|---|
| Message ID | 117 |
| Payload Length | 19 |
| SubType | 1 |

| Position | Field | Type | Description |
|---|---|---|---|
| 0 | ID | uint8_t | 117 |
| 1 | signature | uint8_t | 65 |
| 2 | subType | uint8_t | 1 |
| 3 | version | uint8_t | |
| 4 | icao | uint24_t | Vehicle address (24 bits). Byte[2] = msByte. Set to 0x000000 to enable pseudo-random ICAO generation on UAT transmitters |
| 7 | emitter | uint8_t | Transmitting vehicle type. See EMIT_TYPE enum. |
| 8 | callsign | uint8_t[8] | Vehicle identifier (8 characters, valid characters are A-Z, 0-9, " " only, padded with spaces). |
| 16 | stallspeed | uint8_t | Aircraft stall speed in knots. Used for calculating Air/Ground State for certain emitter types. |
| 17 | avLw | uint8_t | Aircraft length and width encoding (table 2-35 of DO-282B). See ADSB_TRANSPONDER_ALW_ENCODE enum. |
| 18 | antOfsLatLon | uint8_t | GPS antenna lateral and longitudinal offset (table 2-36 and Table 2-37 of DO-282B). See ADSB_TRANSPONDER_GPS_LAT_OFFSET and ADSB_TRANSPONDER_GPS_LON_OFFSET enums. Place lateral offset in top 3 bits and longitudinal offset in bottom 5 bits, e.g. antOfsLatLon = (antOfsLat<<5)\|(antOfsLon&0x1F) |

EMIT_TYPE: Type of vehicle being reported

- 0x00: NO_TYPE_INFO
- 0x01: LIGHT_TYPE
- 0x02: SMALL_TYPE
- 0x03: LARGE_TYPE
- 0x04: HIGH_VORTEX_LARGE_TYPE
- 0x05: HEAVY_TYPE
- 0x06: HIGHLY_MANUV_TYPE
- 0x07: ROTOCRAFT_TYPE
- 0x08: UNASSIGNED_TYPE
- 0x09: GLIDER_TYPE
- 0x0A: LIGHTER_AIR_TYPE
- 0x0B: PARACHUTE_TYPE
- 0x0C: ULTRA_LIGHT_TYPE
- 0x0D: UNASSIGNED2_TYPE
- 0x0E: UAV_TYPE
- 0x0F: SPACE_TYPE
- 0x10: UNASSGINED3_TYPE
- 0x11: EMERGENCY_SURFACE_TYPE
- 0x12: SERVICE_SURFACE_TYPE
- 0x13: POINT_OBSTACLE_TYPE

ADSB_TRANSPONDER_ ALW_ENCODE: Aircraft size (upper bound)

- 0: ADSB_TRANSPONDER_AIRCRAFT_SIZE_NO_DATA
- 1: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L15M_W23M
- 2: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L25M_W28P5M
- 3: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L25_W34M
- 4: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L35_W33M
- 5: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L35_W38M
- 6: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L45_W39P5M
- 7: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L45_W45M
- 8: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L55_W45M
- 9: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L55_W52M
- 10: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L65_W59P5M
- 11: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L65_W67M
- 12: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L75_W72P5M
- 13: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L75_W80M
- 14: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L85_W80M
- 15: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L85_W90M

ADSB_TRANSPONDER_GPS_LAT_OFFSET: Lateral GPS offset
- 0: ADSB_TRANSPONDER_GPS_LAT_OFFSET_NO_DATA
- 1: ADSB_TRANSPONDER_GPS_LAT_OFFSET_LEFT_2M
- 2: ADSB_TRANSPONDER_GPS_LAT_OFFSET_LEFT_4M
- 3: ADSB_TRANSPONDER_GPS_LAT_OFFSET_LEFT_6M
- 4: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_0M
- 5: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_2M
- 6: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_4M
- 7: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_6M

ADSB_TRANSPONDER_GPS_LON_OFFSET: Longitudinal GPS offset
- 0: ADSB_TRANSPONDER_GPS_LON_OFFSET_NO_DATA
- 1: ADSB_TRANSPONDER_GPS_LON_OFFSET_APPLIED_BY_SENSOR
- 2-31: Compute: (meters/2 + 1)

## 3. MAVLink Messsages

MAVLink documentation can be found at: http://qgroundcontrol.org/mavlink/start

### Physical Interface

Communication to a uAvionix transponder is accomplished through a full duplex asynchronous serial interface. The interface should use 3.3v logic levels. The baud rate on the NAV interface is 115200bps and on the HOST interface 57600bps. Both interfaces, no parity, 1 stop bits and 8 data bits. Any multi-byte data is formatted and transmitted as little-endian.

| Message ID | Description | I/O | Length | CRC Extra |
|---|---|---|---|---|
| 66 | DataStream Request | Out | 6 | 148 |
| 246 | Traffic Report | Out | 38 | 184 |
| 203 | Status | Out | 1 | 85 |
| 202 | Ownship | Out | 42 | 7 |
| 202 | Dynamic | In | 42 | 7 |
| 201 | Static | In | 19 | 126 |

Table 5: MAVLink Messages

### Interface Priority

The two interfaces on the ping share a priority system based upon their intended purpose. This system provides prioritization and redundancy since the same information could be provided over either interface. Here is a list of required messages and timeout's for each interface:

| Messages Required | Description | Timeout | Preferred/Backup Interface |
|---|---|---|---|
| MAVLink Static (ID: 201) | 0.1Hz | 30s | Host/NAV |
| MAVLink Dynamic (ID: 202) | 5Hz | 5s | NAV/Host |

Table 6: Interface Priority

When a particular message has not been received through the preferred interface for the specified timeout, the data being received from that message switches from the preferred interface to the backup interface.

**DataStream Request Message**

| | |
|---|---|
| Message ID | 66 |
| Payload Length | 6 |
| CRC_EXTRA | 148 |

This is a legacy MAVLink message to request telemetry position messages from Ardupilot. Prior to v3.7(arduplane) and 3.4(arducopter)

**Traffic Report Message**

| | |
|---|---|
| Message ID | 246 |
| Payload Length | 38 |
| CRC_EXTRA | 184 |

| Position | Field | Type | Description |
|---|---|---|---|
| 0 | ICAO_address | uint32_t | ICAO Address |
| 4 | lat | int32_t | The reported latitude in degrees * 1E7 |
| 8 | lon | int32_t | The reported longitude in degrees * 1E7 |
| 12 | altitude | int32_t | Altitude in Meters * 1E3 (up is +ve) - Check ALT_TYPE for reference datum |
| 16 | heading | uint16_t | Course over ground in degrees * 10^2 |
| 18 | hor_velocity | uint16_t | The horizontal velocity in (m/s * 1E2) |
| 20 | ver_velocity | int16_t | The vertical velocity in (m/s * 1E2) |
| 22 | validFlags | uint16_t | Valid data fields from FLAGS enum |
| 24 | squawk | uint16_t | Mode A Squawk code (0xFFFF = no code) |
| 26 | altitude_type | uint8_t | Altitude Type from ALT_TYPE enum |
| 27 | callsign | char[9] | The callsign |
| 36 | emitter_type | uint8_t | Emitter Category from EMIT_TYPE enum |
| 37 | tslc | uint8_t | Time since last communication in seconds |

ALT_TYPE: The type of altitude source used to measure the altitude being reported.
- 0x00: PRESSURE_ALTITUDE (AMSL, QNH)
- 0x01: GEOMETRIC (GNSS, WGS84)

EMIT_TYPE: Type of vehicle being reported
- 0x00: NO_TYPE_INFO
- 0x01: LIGHT_TYPE
- 0x02: SMALL_TYPE
- 0x03: LARGE_TYPE
- 0x04: HIGH_VORTEX_LARGE_TYPE
- 0x05: HEAVY_TYPE
- 0x06: HIGHLY_MANUV_TYPE
- 0x07: ROTOCRAFT_TYPE
- 0x08: UNASSIGNED_TYPE
- 0x09: GLIDER_TYPE
- 0x0A: LIGHTER_AIR_TYPE
- 0x0B: PARACHUTE_TYPE
- 0x0C: ULTRA_LIGHT_TYPE
- 0x0D: UNASSIGNED2_TYPE
- 0x0E: UAV_TYPE
- 0x0F: SPACE_TYPE
- 0x10: UNASSGINED3_TYPE
- 0x11: EMERGENCY_SURFACE_TYPE
- 0x12: SERVICE_SURFACE_TYPE
- 0x13: POINT_OBSTACLE_TYPE

FLAGS: The 'flags' field shows the status of the response payload fields as follows:

0x0001: LATLON_VALID
0x0002: ALTITUDE_VALID
0x0004: HEADING_VALID
0x0008: VELOCITY_VALID
0x0010: CALLSIGN_VALID
0x0020: IDENT_VALID
0x0040 SIMULATED_REPORT
0x0080 VERTICAL_VELOCITY_VALID
0x0100 BARO_VALID
0x8000: SOURCE UAT

### Status Message

| | |
|---|---|
| Message ID | 203 |
| Payload Length | 1 |
| CRC_EXTRA | 85 |

| Position | Field | Type | Description |
|---|---|---|---|
| 0 | status | uint8_t | Self test status. See ADSB_TRANSPONDER_ STATUS_FLAGS enum. |

ADSB_TRANSPONDER_STATUS_FLAGS: Device self-test status

0x00: ASDB_TRANSPONDER_INITIALIZING
0x01: ADSB_TRANSPONDER_OK
0x02: ADSB_TRANSPONDER_TX_FAIL_1090ES
0x04: ADSB_TRANSPONDER_RX_FAIL_1090ES
0x08: ADSB_TRANSPONDER_TX_FAIL_UAT
0x16: ADSB_TRANSPONDER_RX_FAIL_UAT

### Dynamic and Ownship Message

| | |
|---|---|
| Message ID | 202 |
| Payload Length | 42 |
| CRC_EXTRA | 7 |

| Position | Field | Type | Description |
|---|---|---|---|
| 0 | utcTime | uint32_t | UTC time in since GPS epoch (in s since Jan 6, 1980). If unknown set to UINT32_MAX. |
| 4 | latitude | int32_t | Latitude WGS84 (deg * 1E7). If unknown set to INT32_MAX |
| 8 | longitude | int32_t | Longitude WGS84 (deg * 1E7). If unknown set to INT32_MAX. |
| 12 | altPres | int32_t | Barometric pressure altitude relative to a standard atmosphere of 1013.2 mBar and NOT bar corrected altitude |

| | | | |
|---|---|---|---|
| | | | (meters * 1E3) UP +ve. If unknown set to INT32_MAX. |
| 16 | altGNSS | int32_t | Altitude (meters * 1E3). (up +ve). WGS84 altitude. If unknown set to INT32_MAX. |
| 20 | accHoriz | uint32_t | Horizontal accuracy(HFOM) (mm). If unknown set to UINT32_MAX. |
| 24 | accVert | uint16_t | Vertical accuracy(VFOM) (cm). If unknown set to UINT16_MAX. |
| 26 | accVel | uint16_t | Velocity accuracy (m/s * 1E3). If unknown set to UINT16_MAX. |
| 28 | velVert | int16_t | GPS vertical speed (m/s * 1E2). If unknown set to INT16_MAX. |
| 30 | nsVog | int16_t | North-South velocity over ground (m/s * 10) North +ve. If unknown set to INT16_MAX. |
| 32 | ewVog | int16_t | East-West velocity over ground (m/s * 10) East +ve. If unknown set to INT16_MAX. |
| 34 | state | uint16_t | ADS-B input flags. See ADSB_TRANSPONDER_STATE_FLAGS enum. |
| 36 | squawk | uint16_t | Mode A code (typically 1200 [0x04B0] for VFR). |
| 38 | fixType | uint8_t | GPS Fix. See ADSB_TRANSPONDER_FIX_TYPE_FLAGS enum. |
| 39 | numSats | uint8_t | Number of satellites visible. If unknown set to UINT8_MAX. |
| 40 | emStatus | uint8_t | Emergency status (table 2-78 of DO-260B). |
| 41 | control | uint8_t | ADS-B transponder dynamic input control flags. See ADSB TRANSPONDER_CONTROL _FLAGS enum. |

ADSB_TRANSPONDER_STATE_FLAGS: State flags for ADS-B transponder dynamic input

1: ADSB_TRANSPONDER_INTENT_CHANGE
2: ADSB_TRANSPONDER_AUTOPILOT_ENABLED
4: ADSB_TRANSPONDER_NICBARO_CROSSCHECKED
8: ADSB_TRANSPONDER_ON_GROUND
16:ADSB_TRANSPONDER_IDENT

ADSB_TRANSPONDER_CONTROL_FLAGS: Control flags for ADS-B transponder dynamic input

0: ADSB_TRANSPONDER_STANDBY
1: ADSB_TRANSPONDER_RECEIVE
2: ADSB_TRANSPONDER_TX_ENABLE_UAT
4: ADSB_TRANSPONDER_TX_ENABLE_1090ES

ADSB_TRANSPONDER_FIX_TYPE_FLAGS: Status flags for ADS-B transponder GPS fix type flags

0: ADSB_TRANSPONDER_GPS_NO_FIX_0
1: ADSB_TRANSPONDER_GPS_NO_FIX_1
2: ADSB_TRANSPONDER_GPS_2D_FIX
3: ADSB_TRANSPONDER_GPS_3D_FIX
4: ADSB_TRANSPONDER_GPS_DGPS
5: ADSB_TRANSPONDER_GPS_RTK

### Static Message

| | |
|---|---|
| Message ID | 201 |
| Payload Length | 19 |
| CRC_EXTRA | 126 |

| Position | Field | Type | Description |
|---|---|---|---|
| 0 | CSID_EN:SIL:SDA | uint8_t | [7:5] RFU, CSID_EN[4] 1 or 0, [3:2] SIL 0 to 3, [1:0] SDA 0 to 3 |
| 1 | ICAO | uint8_t[3] | Vehicle address (24 bits). Byte[2] = msByte |
| 4 | stallSpeed | uint16_t | Aircraft stall speed in cm/s. |
| 6 | callsign | char[9] | Vehicle identifier (8 characters, null terminated, valid characters are A-Z, 0-9, " " only). |
| 15 | emitter | uint8_t | Transmitting vehicle type. See EMIT_TYPE enum. |
| 16 | alwEncode | uint8_t | Aircraft length and width encoding (table 2-35 of DO-282B). See ADSB_TRANSPONDER_ALW_ENCODE enum. |
| 17 | gpsLatOffs | uint8_t | GPS antenna lateral offset (table 2-36 of DO-282B). See ADSB_TRANSPONDER_GPS_LAT_OFFSET enum. |
| 18 | gpsLonOffs | uint8_t | GPS antenna longitudinal offset from nose [if non-zero, take position (in meters) divide by 2 and add one with max 60m] (table 2-37 DO-282B). See ADSB_TRANSPONDER_GPS_LON_OFFSET enum. |

ADSB_TRANSPONDER_ ALW_ENCODE: Definitions for aircraft size (upper bound)

0: ADSB_TRANSPONDER_AIRCRAFT_SIZE_NO_DATA
1: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L15M_W23M
2: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L25M_W28P5M
3: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L25_W34M
4: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L35_W33M
5: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L35_W38M
6: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L45_W39P5M
7: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L45_W45M
8: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L55_W45M
9: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L55_W52M
10: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L65_W59P5M
11: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L65_W67M
12: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L75_W72P5M
13: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L75_W80M
14: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L85_W80M
15: ADSB_TRANSPONDER_AIRCRAFT_SIZE_L85_W90M

ADSB_TRANSPONDER_GPS_LAT_OFFSET: Definitions for lateral GPS offset

0: ADSB_TRANSPONDER_GPS_LAT_OFFSET_NO_DATA
1: ADSB_TRANSPONDER_GPS_LAT_OFFSET_LEFT_2M
2: ADSB_TRANSPONDER_GPS_LAT_OFFSET_LEFT_4M
3: ADSB_TRANSPONDER_GPS_LAT_OFFSET_LEFT_6M
4: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_0M
5: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_2M
6: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_4M
7: ADSB_TRANSPONDER_GPS_LAT_OFFSET_RIGHT_6M

ADSB_TRANSPONDER_GPS_LON_OFFSET: Definitions for longitudinal GPS offset

0: ADSB_TRANSPONDER_GPS_LON_OFFSET_NO_DATA
1: ADSB_TRANSPONDER_GPS_LON_OFFSET_APPLIED_BY_SENSOR
2-31: Compute: (meters/2 + 1)

### MAVLink Protocol Frame Format

Each message will be formatted into a frame for transmission across the physical interface. Each frame must be transmitted in its entirety with the start of the frame being inferred by the beginning of any transfer initiated after the end of a previous frame. The end of a Frame will be delineated with an idle condition on the interface.

The uAvionix transponder is 'stateless' MAVLink packets do not have to be coordinated.

Ping OEM Frame 8 - 263 bytes


| Byte Index | Content | Value | Description |
|---|---|---|---|
| 0 | Packet Start Flag | 0xFE | Indicates the start of a new packet |
| 1 | Payload Length | 0 - 255 | Indicates the length of the payload |
| 2 | Packet Sequence | 0 - 255 | Send sequence. Allows to detect packet loss |
| 3 | System ID | 0 - 255 | ID of the Sending system. Allows to differentiate different PINGs on the same network |
| 4 | Component | 0 - 255 | ID of the Sending Component. |
| 5 | Message ID | 0 - 255 | ID of the Message - the 'id' defines what the payload "means" and how it should be correctly decoded. |
| 6 to (n+6) | Data | 0 - 255 Bytes | Message Data |
| (n+7) to (n+8) | Checksum (low Byte, high Byte) | ITU X.25/SAE AS-4 hash, **excluding packet start flag, so bytes 1..(n+6)** Note: The checksum also includes CRC_EXTRA | |

CRC Code:
Code to validate the packet CRC.

```
#define X25_INIT_CRC 0xffff
#define X25_VALIDATE_CRC 0xf0b8

/**
 * @brief Accumulate the X.25 CRC by adding one char at a time.
 *
 * The checksum function adds the hash of one char at a time to the
 * 16 bit checksum (uint16_t).
 *
 * @param data - New char to hash
 * @param crcAccum - Already accumulated checksum
 **/
void crc_accumulate(uint8_t data, uint16_t *crcAccum)
{
   // Accumulate one byte of data into the CRC
   uint8_ttmp;

   tmp = data ^ (uint8_t)(*crcAccum&0xff);
   tmp ^= (tmp<<4);
   *crcAccum = (*crcAccum>>8) ^ (tmp<<8) ^ (tmp<<3) ^ (tmp>>4);
}
#endif


/**
 * @brief Initialize the buffer for the X.25 CRC
 *
 * @param crcAccum - 16 bit X.25 CRC
 */
void crc_init(uint16_t *crcAccum)
{
   *crcAccum = X25_INIT_CRC;
}

/**
 * @brief Calculates the X.25 checksum on a byte buffer
 *
 * @param  pBuffer - buffer containing the byte array to hash
 * @param  length - length of the byte array
 * @return the checksum over the buffer bytes
 **/
uint16_t crc_calculate(const uint8_t *pBuffer, uint16_t length)
{
   uint16_tcrcTmp;
   crc_init(&crcTmp);
   while (length--) crc_accumulate(*pBuffer++, &crcTmp);
   return crcTmp;
}

/**
 * @brief Accumulate the X.25 CRC by adding an array of bytes
 *
 * The checksum function adds the hash of one char at a time to the
 * 16 bit checksum (uint16_t).
```

```
 *
 * @param data - New bytes to hash
 * @param crcAccum - Already accumulated checksum
 **/
void crc_accumulate_buffer(uint16_t *crcAccum, const char *pBuffer, uint16_t length)
{
   const uint8_t *p = (const uint8_t *)pBuffer;
   while (length--) crc_accumulate(*p++, crcAccum);
}


// Note CRC_EXTRA is defined for each individual packet in the document.
crc_accumulate_buffer(&msg->checksum, _PAYLOAD(msg), msg->len);
crc_accumulate(CRC_EXTRA, &msg->checksum);
ck_a(msg) = (uint8_t)(msg->checksum & 0xFF);
ck_b(msg) = (uint8_t)(msg->checksum >> 8);
```