HORIZON HOBBY, LLC.

Specification for Spektrum® X-Bus Telemetry Sensors

Enabling Use of Non-Spektrum Sensors

Rev L

2016 November 3

Specification for communicating on the X-Bus so that data from non-Spektrum devices can be displayed on the AirWare^{TM} -based transmitters.

Specification for Spektrum® X-Bus Telemetry Sensors

1 INTRODUCTION

With the advent of third-party display (J-Link), annunciation (vSpeak), and data display systems (Robo-Software and TLMViewer.com), we feel it is in everybody's best interests to open the telemetry system by sharing correct implementation data. With that mindset, the purpose of this document is to enable third-party telemetry sensors, both commercial and hobbyists, that can use the Spektrum X-Bus telemetry system as a data transport mechanism for custom sensors including items such as:

- an ESC,
- fuel flow meter,
- high-current battery "fuel gauge" (mAh),
- digital status (for example, landing gear status lights),
- thrust/strain gauge,
- air tank pressure, or
- an individual cell monitor for LiPo batteries.

The intent is that publication of this document will ensure that these third-party devices can inter-operate with one another and with Spektrum products in a non-interfering, cooperative manner. Spektrum will provide an interface to allow generic data display and alarms on certain levels of transmitter products, although they obviously cannot be as thoroughly integrated into the radios as Spektrum products are.

2 AUDIENCE

This document is intended for non-Horizon personnel to be able to develop sensors which function correctly in the Spektrum X-Bus Air Telemetry System. This document includes sufficient information to allow a sensor to be created such that it reports data useful to the users.

This document does not provide information that can be used to access data contained in a Spektrum telemetry file (.TLM). The STi application provides this capability for Apple iPhone and related products. Robo-Software has developed a Windows- and Mac-based shareware product which provides excellent capabilities for post-flight data analysis.

3 RELATED DOCUMENTATION

All necessary technical information is contained within this document, including diagrams and source code guidance.

4 LEGAL INFORMATION

Spektrum X-Bus Telemetry Data Application End User License Agreement

This End User License Agreement **("Agreement")** is a binding agreement between the reader of this publication **("you")** and Horizon Hobby, LLC **("Horizon")**. This Agreement governs your use of the Spektrum X-Bus Telemetry Data as outlined in this publication **("The Telemetry System")**. The Telemetry System is licensed, not sold, to you.

BY USING THE TELEMTRY SYSTEM YOU (A) ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTAND THIS AGREEMENT; (B) REPRESENT THAT YOU ARE OF LEGAL AGE TO ENTER INTO A BINDING AGREEMENT; AND (C) ACCEPT THIS AGREEMENT AND AGREE THAT YOU ARE LEGALLY BOUND BY ITS TERMS. IF YOU DO NOT AGREE TO THESE TERMS, DO NOT USE THE TELEMETRY SYSTEM.

NOTWITHSTANDING ANYTHING TO THE CONTRARY IN THIS AGREEMENT OR YOUR ACCEPTANCE OF THE TERMS AND CONDITIONS OF THIS AGREEMENT, NO LICENSE IS GRANTED (WHETHER EXPRESSLY, BY IMPLICATION OR OTHERWISE) UNDER THIS AGREEMENT.

1. License Grant. Subject to the terms of this Agreement, Horizon grants you a limited, non-exclusive license to use The Telemetry System.

2. Termination. Horizon may terminate this Agreement at any time without notice, which Horizon may do at its sole discretion. In addition, this Agreement will terminate immediately and automatically without any notice if you violate any of the terms and conditions of this Agreement.

Upon termination:

(a) All rights granted to you under this Agreement will also terminate; and

(b) You must cease all use of the Telemetry System

(c) Termination will not limit any of Horizon's rights or remedies at law or in equity.

3. Disclaimer of Warranties. THE APPLICATION IS PROVIDED TO YOU "AS IS" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, HORIZON, ON ITS OWN BEHALF AND ON BEHALF OF ITS AFFILIATES AND ITS AND THEIR RESPECTIVE LICENSORS AND SERVICE PROVIDERS, EXPRESSLY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, WITH RESPECT TO THE APPLICATION, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND WARRANTIES THAT MAY ARISE OUT OF COURSE OF DEALING, COURSE OF PERFORMANCE, USAGE OR TRADE PRACTICE. WITHOUT LIMITATION TO THE FOREGOING, COMPANY PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE APPLICATION WILL MEET YOUR

REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

4. Limitation of Liability. TO THE FULLEST EXTENT PERMITTED UNDER APPLICABLE LAW:

IN NO EVENT WILL HORIZON OR ITS AFFILIATES, OR ANY OF ITS OR THEIR RESPECTIVE LICENSORS OR SERVICE PROVIDERS, BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY USE, INTERRUPTION, DELAY OR INABILITY TO USE THE TELEMETRY SYSTEM OR IMPLEMENTATION DATA, LOST REVENUES OR PROFITS, DELAYS, INTERRUPTION OR LOSS OF SERVICES, BUSINESS OR GOODWILL, LOSS OR CORRUPTION OF DATA, LOSS RESULTING FROM SYSTEM OR SYSTEM SERVICE FAILURE, MALFUNCTION OR SHUTDOWN, FAILURE TO ACCURATELY TRANSFER, READ OR TRANSMIT INFORMATION, FAILURE TO UPDATE OR PROVIDE CORRECT INFORMATION, SYSTEM INCOMPATIBILITY OR PROVISION OF INCORRECT COMPATIBILITY INFORMATION OR BREACHES IN SYSTEM SECURITY, OR FOR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, EXEMPLARY, SPECIAL OR PUNITIVE DAMAGES, WHETHER ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT, BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE) OR OTHERWISE, REGARDLESS OF WHETHER SUCH DAMAGES WERE FORESEEABLE AND WHETHER OR NOT HORIZON WAS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

5. Intellectual Property Rights. You acknowledge and agree that The Telemetry System is provided under license, and not sold, to you. You do not acquire any ownership interest in The Telemetry System or data implementation under this Agreement or any other rights thereto other than to use the same in accordance with the license granted, and subject to all terms, conditions and restrictions, under this Agreement. Horizon reserves and shall retain its entire right, title and interest in and to the Telemetry System and all Intellectual Property Rights arising out of or relating to the technology, except as expressly granted to the Licensee in this Agreement. You shall use best efforts to safeguard The Telemetry System (including all copies thereof) from infringement, misappropriation, theft, misuse or unauthorized access. You shall promptly notify Horizon if you become aware of any infringement of Horizon’s Intellectual Property Rights in The Telemetry System and fully cooperate with Horizon in any legal action taken by Horizon to enforce its Intellectual Property Rights.

6. Indemnification. You agree to indemnify, defend and hold harmless Horizon and its officers, directors, employees, agents, affiliates, successors and assigns from and against any and all losses, damages, liabilities, deficiencies, claims, actions, judgments, settlements, interest, awards, penalties, fines, costs, or expenses of whatever kind, including attorneys' fees, arising from or relating to your use or misuse of the Telemetry System or your breach of this Agreement.

Furthermore, you agree that Horizon assumes no responsibility for the content you submit or make available through this publication.

7. Severability. If any provision of this Agreement is illegal or unenforceable under applicable law, the remainder of the provision will be amended to achieve as closely as possible the effect of the original term and all other provisions of this Agreement will continue in full force and effect.

8. Governing Law. This Agreement is governed by and construed in accordance with the laws of the State of Illinois without giving effect to any choice or conflict of law provision or rule. Any legal suit, action, or proceeding arising out of or related to this Agreement shall be instituted exclusively in the federal courts of the United States or the courts of the State of Illinois in each case located in Champaign County. You waive any and all objections to the exercise of jurisdiction over you by such courts and to venue in such courts.

9. Waiver. No failure to exercise, and no delay in exercising, on the part of either party, any right or any power hereunder shall operate as a waiver thereof, nor shall any single or partial exercise of any right or power hereunder preclude further exercise of that or any other right hereunder. In the event of a conflict between this Agreement and any applicable purchase or other terms, the terms of this Agreement shall govern.

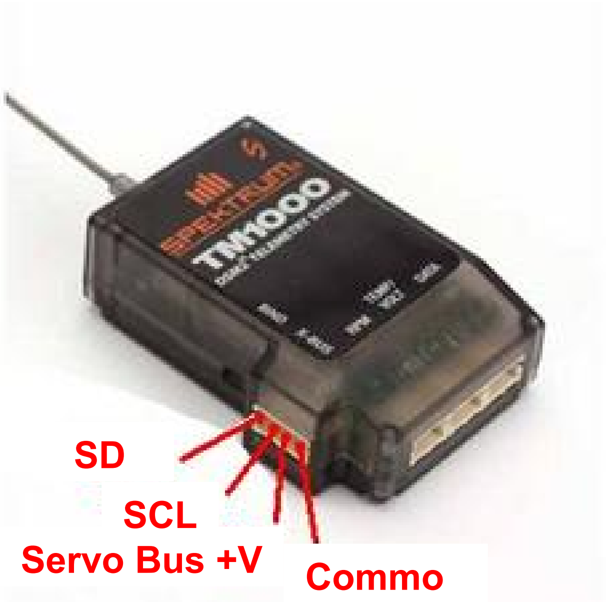
All correspondence regarding this document shall be through the Horizon Hobby legal department:

Horizon Hobby, LLC
Legal Department
4105 Fieldstone Road
Champaign, IL 61822 USA

5 ELECTRICAL DATA

All sensors are powered by the X-Bus. The X-Bus port bus provides the servo bus voltage (3.5 to 9.6V) at a current limited by the JST contact rating (1A). The operational limit in an application may be quite a bit lower, depending upon the method of powering the servo bus.

The X-Bus uses I2C to communicate. Termination resistors are in the TM1000. The pins are defined according to this picture:



Every device shall be responsible to regulate the supply to a level useful for its operation. The I2C signals must be 3.3V logic, and the pins in open drain mode so as to not interfere with the logic levels.

In order to maintain compatibility with other products, it is strongly urged that any sensors include two X-Bus ports to allow them to be daisy-chained in the same manner as Spektrum sensors.

The connector used in the TM1000 and all Spektrum sensors is JST part number S4B-ZR(LF)(SN) or Digikey part 455-1671-ND.

6 HARDWARE-LEVEL PROTOCOL

The TM1000 is an I2C master device talking at 100kHz to the slaves. For best future compatibility, devices should support 400kHz as well.

Every device shall reply to a poll with a 16-byte message, the first byte of which is always the polled I2C address. The remaining bytes are defined in the section on the telemetry header file.

Shortly after the TM1000 starts, it polls all addresses on the bus. During this enumeration phase, the attached devices must reply with their address as the first byte of the reply. The remainder of the first message will be discarded by the TM1000, but the full 16-byte message must be available for the TM1000 to clock in. If a device does not answer the enumeration correctly, the TM1000 will not poll it any more. It is therefore of utmost importance that the first I2C message be answered correctly. The TM1000 allows clock stretching per the I2C specification, which allows slow-to-start devices to enumerate properly in the system. If your device will be slow to start, it is recommended that you first select a higher address, and second that you use the stretched clock.

The TM1000 transmits data to the ground at a rate of one message per 22ms. The time between polls for any single device is dependent upon the number of sensors which enumerated on the bus. Note that the TM1000 reserves two addresses for its internal use, so the maximum rate at which a device is polled will be no less than 44ms. If timing is a critical function for a particular device, it is necessary that the device provide its own clock source and not utilize the X-Bus for timing.

7 ADDRESSING & DEVICE TYPES

The Appendix includes the telemetry header file used by all Spektrum AirWare-based transmitters. It defines the device type codes for all Spektrum products and known reserved values. The device type codes are used as I2C bus addresses by default, but the protocol also provides a means for them to differ.

Spektrum reserves the right to use addresses not listed as we deem necessary. We do not intend to interfere with other products, and therefore urge anybody making a device to provide a mechanism to select different addresses should the need arise. Commercial vendors are urged to contact Spektrum in order to coordinate addresses and prevent interference.

Note that Spektrum is the owner of all address assignments, and does not guarantee that any unused address will be available in the future. Only addresses specifically assigned are guaranteed not to change. Addresses 0x09 and below shall not be used by any third-party devices.

For each of the messages in the header file it should be noted that they begin with the fields *identifier* and *sID*. The *identifier* field is always under all circumstances an exact match to the I2C address, and needs to be the first byte of any reply as noted in the hardware-level section. The second byte, *sID*, serves as a way to allow either multiple devices of the same type to live on the bus, or for a device to retain its type code when there is a conflict of the addresses. At this time, none of the AirWare radios properly display data from multiple instances of the same device type.

Use of the sID field is quite simple:

When sID is zero, then the device type (TELE_DEVICE_xxx) is the same as the bus address *identifier*. This is the norm for all Spektrum products. If *sID* is non-zero, then *sID* is the device type and *identifier* serves only to provide a unique I2C address.

8 DATA FORMATS

All third-party sensors shall report their data in big-endian format (MSB at lower address) if they are to be displayed on the transmitter screens. All data shall binary 8, 16 or 32 bits. Spektrum uses BCD for JetCat and GPS but does not support these formats for third-party products.

The TM1100 module notifies the transmitter that it is in use by setting the high bit of the *identifier* field. This is informational-only to the transmitter and does not affect operation.

The DSMX Ultra Micro receivers provide Flight Log data only, using the standard QoS record structure. The receiver voltage field is fixed at 0xFFFF, indicating “no data” to the transmitter.

The transmitter uses two sentinel values to indicate that there is “no data” for a field. For an unsigned value, a value with all bits set to one (ie, 0xFFFF or 0xFFFFFFFF) indicates this. For a signed value the “no data” value is denoted by all bits set except the sign bit, i.e. 0x7FFF or 0x7FFFFFFF.

These values and standards are also utilized by post-flight systems to properly display logged data.

9 ELECTRONIC SPEED CONTROL

The AirWare-based transmitters include support for a generic Electronic Speed Control (ESC) device. Spektrum does not sell a device which conforms to this telemetry standard, but is instead providing a common interface which may be supported by ESC manufacturers.

The ESC configuration screen provides the same functions available to other devices, that is, whether the status is actively monitored on the display. Alarms are available for the following conditions:

- Input Voltage too low
- Motor current too high
- FET temperature too high

The units and ranges for each of the fields in the telemetry message are found in the appendix in the definition for the ESC structure. The transmitter does not provide any filtering of data for any ESC fields.

10 FUEL FLOW METER

The AirWare-based transmitters may include support for a generic fuel flow and capacity metering device. Spektrum does not sell a device which conforms to this telemetry standard, but is instead providing a common interface which may be supported by third-party manufacturers.

The “Fuel” configuration screen provides the same functions available to other devices, that is, whether the status is actively monitored on the display. Alarms may be available for some of the following conditions:

- Tank 1 capacity consumed > user-defined value
- Tank 2 capacity consumed > user-defined value
- Fuel flow 1 too low
- Fuel flow 1 too high
- Fuel flow 2 too low
- Fuel flow 2 too high
- Temperature 1 too low
- Temperature 1 too high
- Temperature 2 too low
- Temperature 2 too high

The units and ranges for each of the fields in the telemetry message are found in the appendix in the definition for the FUEL structure. The transmitter does not provide any filtering of data for any fields.

11 HIGH-CURRENT BATTERY CAPACITY

The AirWare-based transmitters include support for a generic battery current and capacity metering device. Spektrum SPMA9605 provides this function, alarming and reporting only the first set of message data (address 0x34) at this time. SPMA9604 provides similar capabilities for low-current applications using address 0x18. PowerSafe receivers use both channels of reporting in the low-current record (0x18 type) for each input power source.

The units and ranges for each of the fields in the telemetry message are found in the appendix in the definition for the MAH structures. The transmitter does not provide any filtering of data for any fields.

12 DIGITAL INPUT AND AIR PRESSURE SENSOR

The AirWare-based transmitters may include support for a generic digital input and air pressure metering device. Spektrum does not sell a device which conforms to this telemetry standard, but is instead providing a common interface which may be supported by third-party manufacturers.

The "Air" configuration screen provides the same functions available to other devices, that is, whether the status is actively monitored on the display. Alarms may be available for the following conditions:

- Digital Bit set (bits 0-16)
- Digital Bit clear (bits 0-16)
- Pressure too low
- Pressure too high

The units and ranges for each of the fields in the telemetry message are found in the appendix in the definition for the DIGITAL_AIR structure. The transmitter does not provide any filtering of data for any fields.

13 THRUST/STRAIN GAUGE

The AirWare-based transmitters may include support for a generic thrust or strain metering device. Spektrum does not sell a device which conforms to this telemetry standard, but is instead providing a common interface which may be supported by third-party manufacturers.

The "Strain" configuration screen provides the same functions available to other devices, that is, whether the status is actively monitored on the display. Alarms may be available for the following conditions:

- Single Strain too high (any input above threshold)
- Sum Strain too high (sum of active strains above threshold)
- Strain Imbalance (delta of min/max strains on active inputs is above threshold)

The units and ranges for each of the fields in the telemetry message are found in the appendix in the definition for the STRAIN structure. The transmitter does not provide any filtering of data for any fields.

14 INDIVIDUAL CELL MONITOR

The AirWare-based transmitters include support for generic multi-tap voltage monitoring devices in both 6S and 14S combinations. Spektrum does not sell a device which conforms to this telemetry standard, but is instead providing a common interface which may be supported by third-party manufacturers.

NOTE: The Common (Ve-) connection of the X-Bus is connected to the receiver, which in an electric model is likely connected directly to the negative terminal in the battery string. It is strongly recommended that the voltage measurements be galvanically isolated from the battery pack being measured so as to prevent short circuits and ground loops. This isolation also permits battery packs of more than 6 cells to be monitored accurately and without concern for wiring problems.

It is recommended that the user familiarize himself with the balance and voltage limit reporting functions within the two cell monitor support screens.

The units and ranges for each of the fields in the telemetry message are found in the appendix in the definition for the LIPOMON structure. The transmitter does not provide any filtering of data for any fields.

15 ATTITUDE & MAGNETIC COMPASS

The AirWare-based transmitters may include a facility to display data from an attitude and magnetic compass. This is currently envisioned as an information-only device which may be of use in certain applications but unable to generate alarms. Data which is unavailable due to limitations of the sensor hardware shall report a value of 0x7FFF to indicate “No data available.”

16 3-AXIS GYRO

The AirWare-based transmitters include a facility to display data from a 3-axis gyro system. This currently is envisioned as an information-only device which may be of use in certain applications, but unable to generate alarms. Data which is unavailable due to limitations of the sensor hardware shall report a value of 0x7FFF to indicate “No data available.”

17 USER-DEFINED DEVICES IN THE TX

The AirWare-based transmitters include a facility to display data from user-defined sensors according to four “user” structures defined in the Appendix. The four structures are associated with four different *identifier* field values.

Transmitters may have generic screens to show the data for each structure type. The transmitters would allow the user to specify a short title for the screen, but not for individual fields, nor would it allow specification of units. Display of individual fields may be only turned on or off using the configuration screen. It is up to the user to know the representation of each field shown on the transmitter for these custom devices.

These devices do not have any alarm capability.

The transmitter does not provide any filtering of data for any fields.

18 USER TEXT DEVICE

The AirWare-based transmitters include a facility to display text data directly on the screen in a formatted manner. This message would typically be used in conjunction with a serial interface telemetry module such as the SPM4649T attached to a flight controller (FC) for purposes of configuring the FC using transmitter stick inputs (aka “Stick Programming”).

APPENDIX – HEADER FILE DATA

Note that some device types cannot be used by third-party devices, in particular voltage (0x01) and temperature (0x02), as these are reserved for internal use within the transmitter. The text below has been re-formatted for tabs that look good on the page. If you copy/paste them into your code, you will probably want to re-tab them.

```
//////////////////////////////////////////////////////////////////
//
//     Copyright 2013 by Horizon Hobby, Inc.
//     All Rights Reserved Worldwide.
//
//     This header file may be incorporated into non-Horizon
//     products.
//
//////////////////////////////////////////////////////////////////

#ifndef TELEMETRY_H
#define    TELEMETRY_H

//////////////////////////////////////////////////////////////////
//
//     Assigned I2C Addresses and Device Types
//
//////////////////////////////////////////////////////////////////

#define    TELE_DEVICE_NODATA            (0x00)    // No data in packet
#define    TELE_DEVICE_VOLTAGE           (0x01)    // High-Voltage sensor (INTERNAL)
#define    TELE_DEVICE_TEMPERATURE       (0x02)    // Temperature Sensor (INTERNAL)
#define    TELE_DEVICE_RSV_03            (0x03)    // Reserved
#define    TELE_DEVICE_RSV_04            (0x04)    // Reserved
#define    TELE_DEVICE_RSV_05            (0x05)    // Reserved
#define    TELE_DEVICE_RSV_06            (0x06)    // Reserved
#define    TELE_DEVICE_RSV_07            (0x07)    // Reserved
#define    TELE_DEVICE_RSV_08            (0x08)    // Reserved
#define    TELE_DEVICE_RSV_09            (0x09)    // Reserved
#define    TELE_DEVICE_PBOX              (0x0A)    // PowerBox
#define    TELE_DEVICE_LAPTIMER          (0x0B)    // Lap Timer
#define    TELE_DEVICE_TEXTGEN           (0x0C)    // Text Generator
#define    TELE_DEVICE_AIRSPEED          (0x11)    // Air Speed
#define    TELE_DEVICE_ALTITUDE          (0x12)    // Altitude
#define    TELE_DEVICE_GMETER            (0x14)    // GForce
#define    TELE_DEVICE_JETCAT            (0x15)    // JetCat interface
#define    TELE_DEVICE_GPS_LOC           (0x16)    // GPS Location Data
#define    TELE_DEVICE_GPS_STATS         (0x17)    // GPS Status
#define    TELE_DEVICE_RX_MAH            (0x18)    // Receiver Pack Capacity (Dual)
#define    TELE_DEVICE_JETCAT_2          (0x19)    // JetCat interface, msg 2
#define    TELE_DEVICE_GYRO              (0x1A)    // 3-axis gyro
#define    TELE_DEVICE_ATTMAG            (0x1B)    // Attitude and Magnetic Compass
#define    TELE_DEVICE_AS3X_LEGACYGAIN   (0x1F)    // Active AS3X Gains for legacy mode
#define    TELE_DEVICE_ESC               (0x20)    // ESC
#define    TELE_DEVICE_FUEL              (0x22)    // Fuel Flow Meter
#define    TELE_DEVICE_ALPHA6            (0x24)    // Alpha6 Stabilizer
//         DO NOT USE                    (0x30)    // Reserved for internal use
//         DO NOT USE                    (0x32)    // Reserved for internal use
#define    TELE_DEVICE_MAH               (0x34)    // Battery Gauge (mAh) (Dual)
#define    TELE_DEVICE_DIGITAL_AIR       (0x36)    // Digital Inputs & Tank Pressure
#define    TELE_DEVICE_STRAIN            (0x38)    // Thrust/Strain Gauge
#define    TELE_DEVICE_LIPOMON           (0x3A)    // 6S Cell Monitor (LiPo taps)
#define    TELE_DEVICE_LIPOMON_14        (0x3F)    // 14S Cell Monitor (LiPo taps)
#define    TELE_DEVICE_VARIO_S           (0x40)    // Vario
#define    TELE_DEVICE_RSV_43            (0x43)    // Reserved
#define    TELE_DEVICE_USER_16SU         (0x50)    // User-Def, STRU_TELE_USER_16SU
#define    TELE_DEVICE_USER_16SU32U      (0x52)    // User-Def, STRU_TELE_USER_16SU32U
#define    TELE_DEVICE_USER_16SU32S      (0x54)    // User-Def, STRU_TELE_USER_16SU32S
```

```
#define    TELE_DEVICE_USER_16U32SU      (0x56)    // User-Def, STRU_TELE_USER_16U32SU
#define    TELE_DEVICE_RSV_60            (0x60)    // Reserved
#define    TELE_DEVICE_RSV_68            (0x68)    // Reserved
#define    TELE_DEVICE_RSV_69            (0x69)    // Reserved
#define    TELE_DEVICE_RSV_6A            (0x6A)    // Reserved
#define    TELE_DEVICE_RSV_6B            (0x6B)    // Reserved
#define    TELE_DEVICE_RSV_6C            (0x6C)    // Reserved
#define    TELE_DEVICE_RSV_6D            (0x6D)    // Reserved
#define    TELE_DEVICE_RSV_6E            (0x6E)    // Reserved
#define    TELE_DEVICE_RSV_6F            (0x6F)    // Reserved
#define    TELE_DEVICE_RSV_70            (0x70)    // Reserved
#define    TELE_DEVICE_RTC               (0x7C)    // Pseudo-device giving timestamp
#define    TELE_DEVICE_FRAMEDATA         (0x7D)    // Transmitter frame data
#define    TELE_DEVICE_RPM               (0x7E)    // RPM sensor
#define    TELE_DEVICE_QOS               (0x7F)    // RxV + flight log data
#define    TELE_DEVICE_MAX               (0x7F)    // Last address available

#define    TELE_DEVICE_SHORTRANGE        (0x80)    // Data is from a TM1100


//////////////////////////////////////////////////////////////////
//
//      Message Data Structures
//
//////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////
//
//     Electronic Speed Control
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;        // Source device = 0x20
   UINT8   sID;               // Secondary ID
   UINT16  RPM;               // RPM, 10RPM (0-655340 RPM).     0xFFFF --> "No data"
   UINT16  voltsInput;        // Volts, 0.01v (0-655.34V).      0xFFFF --> "No data"
   UINT16  tempFET;           // Temperature, 0.1C (0-999.8C)   0xFFFF --> "No data"
   UINT16  currentMotor;      // Current, 10mA (0-655.34A).     0xFFFF --> "No data"
   UINT16  tempBEC;           // Temperature, 0.1C (0-999.8C)   0x7FFF --> "No data"
   UINT8   currentBEC;        // BEC Current, 100mA (0-25.4A).  0xFF ----> "No data"
   UINT8   voltsBEC;          // BEC Volts, 0.05V (0-12.70V).   0xFF ----> "No data"
   UINT8   throttle;          // 0.5% (0-127%).                 0xFF ----> "No data"
   UINT8   powerOut;          // Power Output, 0.5% (0-127%).   0xFF ----> "No data"}
STRU_TELE_ESC;

//////////////////////////////////////////////////////////////////
//
//     LAP TIMER
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;        // Source device = 0x0B
   UINT8   sID;               // Secondary ID
   UINT8   lapNumber;         // Lap last finished
   UINT8   gateNumber;        // Last gate passed
   UINT32  lastLapTime;       // Time of lap in 1ms increments (NOT duration)
   UINT32  gateTime;          // Duration between last 2 gates
   UINT8   unused[4];
} STRU_TELE_LAPTIMER;

//////////////////////////////////////////////////////////////////
//
//     TEXT GENERATOR
//
//////////////////////////////////////////////////////////////////
//
```

```
typedef struct
{
   UINT8   identifier;        // Source device = 0x0C
   UINT8   sID;               // Secondary ID
   UINT8   lineNumber;        // Line number to display (0 = title, 1-8 for general,
                              //     255 = Erase all text on screen)
   char    text[13];          // 0-terminated text
} STRU_TELE_TEXTGEN;

//////////////////////////////////////////////////////////////////
//
//     (Liquid) Fuel Flow/Capacity (Two Tanks/Engines)
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                               // Source device = 0x22
   UINT8   sID;                              // Secondary ID
   UINT16  fuelConsumed_A;                   // Integrated fuel consumption, 0.1mL
   UINT16  flowRate_A;                       // Instantaneous consumption, 0.01mL/min
   UINT16  temp_A;                           // Temperature, 0.1C (0-655.34C)
   UINT16  fuelConsumed_B;                   // Integrated fuel consumption, 0.1mL
   UINT16  flowRate_B;                       // Instantaneous consumption, 0.01mL/min
   UINT16  temp_B;                           // Temperature, 0.1C (0-655.34C)
   UINT16  spare;                            // Not used
} STRU_TELE_FUEL;

//////////////////////////////////////////////////////////////////
//
//     Battery Current/Capacity (Dual Batteries)
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                               // Source device = 0x34
   UINT8   sID;                              // Secondary ID
   INT16   current_A;                        // Instantaneous current, 0.1A (0-3276.8A)
   INT16   chargeUsed_A;                     // Integrated mAh used, 1mAh (0-32.766Ah)
   UINT16  temp_A;                           // Temperature, 0.1C (0-150.0C,
                                             // 0x7FFF indicates not populated)
   INT16   current_B;                        // Instantaneous current, 0.1A (0-6553.4A)
   INT16   chargeUsed_B;                     // Integrated mAh used, 1mAh (0-65.534Ah)
   UINT16  temp_B;                           // Temperature, 0.1C (0-150.0C,
                                             // 0x7FFF indicates not populated)
   UINT16  spare;                            // Not used
} STRU_TELE_MAH;

//////////////////////////////////////////////////////////////////
//
//     Digital Input Status (Retract Status) and Tank Pressure
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                               // Source device = 0x36
   UINT8   sID;                              // Secondary ID
   UINT16  digital;                          // Digital inputs (bit per input)
   UINT16  pressure;                         // Tank pressure, 0.1PSI (0-6553.4PSI)
} STRU_TELE_DIGITAL_AIR;

//////////////////////////////////////////////////////////////////
//
//     Thrust/Strain Gauge
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                              // Source device = 0x38
   UINT8   sID;                             // Secondary ID
   UINT16  strain_A,                        // Strain sensor A
           strain_B,                        // Strain sensor B
           strain_C,                        // Strain sensor D
           strain_D;                        // Strain sensor C
} STRU_TELE_STRAIN;

//////////////////////////////////////////////////////////////////
//
//     THIRD-PARTY 16-BIT DATA SIGNED/UNSIGNED
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                              // Source device = 0x50
   UINT8   sID;                             // Secondary ID
   INT16   sField1,                         // Signed 16-bit data fields
           sField2,
           sField3;
   UINT16  uField1,                         // Unsigned 16-bit data fields
           uField2,
           uField3,
           uField4;
} STRU_TELE_USER_16SU;

//////////////////////////////////////////////////////////////////
//
//     THIRD-PARTY 16-BIT SIGNED/UNSIGNED AND 32-BIT UNSIGNED
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                              // Source device = 0x52
   UINT8   sID;                             // Secondary ID
   INT16   sField1,                         // Signed 16-bit data fields
           sField2;
   UINT16  uField1,                         // Unsigned 16-bit data fields
           uField2,
           uField3;
   UINT32  u32Field;                        // Unsigned 32-bit data field
} STRU_TELE_USER_16SU32U;

//////////////////////////////////////////////////////////////////
//
//     THIRD-PARTY 16-BIT SIGNED/UNSIGNED AND 32-BIT SIGNED
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                              // Source device = 0x54
   UINT8   sID;                             // Secondary ID
   INT16   sField1,                         // Signed 16-bit data fields
           sField2;
   UINT16  uField1,                         // Unsigned 16-bit data fields
           uField2,
           uField3;
   INT32   u32Field;                        // Signed 32-bit data field
} STRU_TELE_USER_16U32SU;

//////////////////////////////////////////////////////////////////
//
//     THIRD-PARTY 16-BIT UNSIGNED AND 32-BIT SIGNED/UNSIGNED
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                              // Source device = 0x56
   UINT8   sID;                             // Secondary ID
   UINT16  uField1;                         // Unsigned 16-bit data field
   INT32   u32Field;                        // Signed 32-bit data field
   INT32   u32Field1,                       // Signed 32-bit data fields
           u32Field2;
 } STRU_TELE_USER_16U32SU;

//////////////////////////////////////////////////////////////////
//
//     POWERBOX
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;                      // Source device = 0x0A
   UINT8   sID;                             // Secondary ID
   UINT16  volt1;                           // Volts, 0v01v
   UINT16  volt2;                           // Volts, 0.01v
   UINT16  capacity1;                       // mAh, 1mAh
   UINT16  capacity2;                       // mAh, 1mAh
   UINT16  spare16_1;
   UINT16  spare16_2;
   UINT8   spare;
   UINT8   alarms;                          // Alarm bitmask (see below)
} STRU_TELE_POWERBOX;

#define    TELE_PBOX_ALARM_VOLTAGE_1        (0x01)
#define    TELE_PBOX_ALARM_VOLTAGE_2        (0x02)
#define    TELE_PBOX_ALARM_CAPACITY_1       (0x04)
#define    TELE_PBOX_ALARM_CAPACITY_2       (0x08)
//#define  TELE_PBOX_ALARM_RPM              (0x10)
//#define  TELE_PBOX_ALARM_TEMPERATURE      (0x20)
#define    TELE_PBOX_ALARM_RESERVED_1       (0x40)
#define    TELE_PBOX_ALARM_RESERVED_2       (0x80)

//////////////////////////////////////////////////////////////////
//
//     DUAL ENERGY
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   id;                              // Source device = 0x18
   UINT8   sID;                             // Secondary ID
   INT16   current_A;                       // Instantaneous current, 0.01A (0-328.7A)
   INT16   chargeUsed_A;                    // Integrated mAh used, 0.1mAh (0-3276.6mAh)
   UINT16  volts_A;                         // Voltage, 0.01VC (0-16.00V)
   INT16   current_B;                       // Instantaneous current, 0.1A (0-3276.8A)
   INT16   chargeUsed_B;                    // Integrated mAh used, 1mAh (0-32.766Ah)
   UINT16  volts_B;                         // Voltage, 0.01VC (0-16.00V)
   UINT16  spare;                           // Not used
} STRU_TELE_ENERGY_DUAL;

//////////////////////////////////////////////////////////////////
//
//     HIGH-CURRENT
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;                      // Source device = 0x03
   UINT8   sID;                             // Secondary ID
   INT16   current,                         // Range: +/- 150A
                                            // Resolution: 300A/2048 = 0.196791 A/tick
           dummy;                           // TBD
} STRU_TELE_IHIGH;
```

```
#define    IHIGH_RESOLUTION_FACTOR          ((FP32)(0.196791))

//////////////////////////////////////////////////////////////////
//
//     VARIO-S
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;                      // Source device = 0x40
   UINT8   sID;                             // Secondary ID
   INT16   altitude;                        // .1m increments
   INT16   delta_0250ms,                    // delta last 250ms, 0.1m/s increments
           delta_0500ms,                    // delta last 500ms, 0.1m/s increments
           delta_1000ms,                    // delta last 1.0 seconds
           delta_1500ms,                    // delta last 1.5 seconds
           delta_2000ms,                    // delta last 2.0 seconds
           delta_3000ms;                    // delta last 3.0 seconds
} STRU_TELE_VARIO_S;

//////////////////////////////////////////////////////////////////
//
//     ALTIMETER
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;
   UINT8   sID;                             // Secondary ID
   INT16   altitude;                        // .1m increments
   INT16   maxAltitude;                     // .1m increments
} STRU_TELE_ALT;

//////////////////////////////////////////////////////////////////
//
//     AIRSPEED
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;
   UINT8   sID;                             // Secondary ID
   UINT16  airspeed;                        // 1 km/h increments
   UINT16  maxAirspeed;                     // 1 km/h increments
} STRU_TELE_SPEED;

//////////////////////////////////////////////////////////////////
//
//     GFORCE
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;                      // Source device = 0x14
   UINT8   sID;                             // Secondary ID
   INT16   GForceX;                         // force is reported as .01G increments
   INT16   GForceY;                         // Range = +/-4000 (+/- 40G) in Pro model
   INT16   GForceZ;                         // Range = +/-800 (+/- 8G) in Standard model
   INT16   maxGForceX;                      // abs(max G X-axis)   FORE/AFT
   INT16   maxGForceY;                      // abs (max G Y-axis)  LEFT/RIGHT
   INT16   maxGForceZ;                      // max G Z-axis        WING SPAR LOAD
   INT16   minGForceZ;                      // min G Z-axis        WING SPAR LOAD
} STRU_TELE_G_METER;

//////////////////////////////////////////////////////////////////
//
//     JETCAT/TURBINE
```

```
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;                        // Source device = 0x15
   UINT8   sID;                               // Secondary ID
   UINT8   status;                            // See table below
   UINT8   throttle;                          // (BCD) xx Percent
   UINT16  packVoltage;                       // (BCD) xx.yy
   UINT16  pumpVoltage;                       // (BCD) xx.yy
   UINT32  RPM;                               // (BCD)
   UINT16  EGT;                               // (BCD) Temperature, Celsius
   UINT8   offCondition;                      // (BCD) See table below
   UINT8   spare;
} STRU_TELE_JETCAT;

enum JETCAT_ECU_TURBINE_STATE {              // ECU Status definitions
       JETCAT_ECU_STATE_OFF = 0x00,
       JETCAT_ECU_STATE_WAIT_for_RPM = 0x01, // (Stby/Start)
       JETCAT_ECU_STATE_Ignite = 0x02,
       JETCAT_ECU_STATE_Accelerate = 0x03,
       JETCAT_ECU_STATE_Stabilise = 0x04,
       JETCAT_ECU_STATE_Learn_HI = 0x05,
       JETCAT_ECU_STATE_Learn_LO = 0x06,
       JETCAT_ECU_STATE_UNDEFINED = 0x07,
       JETCAT_ECU_STATE_Slow_Down = 0x08,
       JETCAT_ECU_STATE_Manual = 0x09,
       JETCAT_ECU_STATE_AutoOff = 0x10,
       JETCAT_ECU_STATE_Run = 0x11, // (reg.)
       JETCAT_ECU_STATE_Accleleration_delay = 0x12,
       JETCAT_ECU_STATE_SpeedReg = 0x13, // (Speed Ctrl)
       JETCAT_ECU_STATE_Two_Shaft_Regulate = 0x14, // (only for secondary shaft)
       JETCAT_ECU_STATE_PreHeat1 = 0x15,
       JETCAT_ECU_STATE_PreHeat2 = 0x16,
       JETCAT_ECU_STATE_MainFStart = 0x17,
       JETCAT_ECU_STATE_NotUsed = 0x18,
       JETCAT_ECU_STATE_KeroFullOn = 0x19,
       // undefined states 0x1A-0x1F
       EVOJET_ECU_STATE_off = 0x20,
       EVOJET_ECU_STATE_ignt = 0x21,
       EVOJET_ECU_STATE_acce = 0x22,
       EVOJET_ECU_STATE_run = 0x23,
       EVOJET_ECU_STATE_cal = 0x24,
       EVOJET_ECU_STATE_cool = 0x25,
       EVOJET_ECU_STATE_fire = 0x26,
       EVOJET_ECU_STATE_glow = 0x27,
       EVOJET_ECU_STATE_heat = 0x28,
       EVOJET_ECU_STATE_idle = 0x29,
       EVOJET_ECU_STATE_lock = 0x2A,
       EVOJET_ECU_STATE_rel = 0x2B,
       EVOJET_ECU_STATE_spin = 0x2C,
       EVOJET_ECU_STATE_stop = 0x2D,
       // undefined states 0x2E-0x2F
       HORNET_ECU_STATE_OFF = 0x30,
       HORNET_ECU_STATE_SLOWDOWN = 0x31,
       HORNET_ECU_STATE_COOL_DOWN = 0x32,
       HORNET_ECU_STATE_AUTO = 0x33,
       HORNET_ECU_STATE_AUTO_HC = 0x34,
       HORNET_ECU_STATE_BURNER_ON = 0x35,
       HORNET_ECU_STATE_CAL_IDLE = 0x36,
       HORNET_ECU_STATE_CALIBRATE = 0x37,
       HORNET_ECU_STATE_DEV_DELAY = 0x38,
       HORNET_ECU_STATE_EMERGENCY = 0x39,
       HORNET_ECU_STATE_FUEL_HEAT = 0x3A,
       HORNET_ECU_STATE_FUEL_IGNITE = 0x3B,
       HORNET_ECU_STATE_GO_IDLE = 0x3C,
       HORNET_ECU_STATE_PROP_IGNITE = 0x3D,
       HORNET_ECU_STATE_RAMP_DELAY = 0x3E,
       HORNET_ECU_STATE_RAMP_UP = 0x3F,
       HORNET_ECU_STATE_STANDBY = 0x40,
       HORNET_ECU_STATE_STEADY = 0x41,
       HORNET_ECU_STATE_WAIT_ACC = 0x42,
       HORNET_ECU_STATE_ERROR = 0x43,
       // undefined states 0x44-0x4F
       XICOY_ECU_STATE_Temp_High = 0x50,
       XICOY_ECU_STATE_Trim_Low = 0x51,
       XICOY_ECU_STATE_Set_Idle = 0x52,
       XICOY_ECU_STATE_Ready = 0x53,
       XICOY_ECU_STATE_Ignition = 0x54,
       XICOY_ECU_STATE_Fuel_Ramp = 0x55,
       XICOY_ECU_STATE_Glow_Test = 0x56,
       XICOY_ECU_STATE_Running = 0x57,
       XICOY_ECU_STATE_Stop = 0x58,
       XICOY_ECU_STATE_Flameout = 0x59,
       XICOY_ECU_STATE_Speed_Low = 0x5A,
       XICOY_ECU_STATE_Cooling = 0x5B,
       XICOY_ECU_STATE_Igniter_Bad = 0x5C,
       XICOY_ECU_STATE_Starter_F = 0x5D,
       XICOY_ECU_STATE_Weak_Fuel = 0x5E,
       XICOY_ECU_STATE_Start_On = 0x5F,
       XICOY_ECU_STATE_Pre_Heat = 0x60,
       XICOY_ECU_STATE_Battery = 0x61,
       XICOY_ECU_STATE_Time_Out = 0x62,
       XICOY_ECU_STATE_Overload = 0x63,
       XICOY_ECU_STATE_Igniter_Fail = 0x64,
       XICOY_ECU_STATE_Burner_On = 0x65,
       XICOY_ECU_STATE_Starting = 0x66,
       XICOY_ECU_STATE_SwitchOver = 0x67,
       XICOY_ECU_STATE_Cal_Pump = 0x68,
       XICOY_ECU_STATE_Pump_Limit = 0x69,
       XICOY_ECU_STATE_No_Engine = 0x6A,
       XICOY_ECU_STATE_Pwr_Boost = 0x6B,
       XICOY_ECU_STATE_Run_Idle = 0x6C,
       XICOY_ECU_STATE_Run_Max = 0x6D,

       TURBINE_ECU_MAX_STATE = 0x74
};

enum JETCAT_ECU_OFF_CONDITIONS {             // ECU off conditions. Valid only when the
ECUStatus = JETCAT_ECU_STATE_OFF
       JETCAT_ECU_OFF_No_Off_Condition_defined = 0,
       JETCAT_ECU_OFF_Shut_down_via_RC,
       JETCAT_ECU_OFF_Overtemperature,
       JETCAT_ECU_OFF_Ignition_timeout,
       JETCAT_ECU_OFF_Acceleration_time_out,
       JETCAT_ECU_OFF_Acceleration_too_slow,
       JETCAT_ECU_OFF_Over_RPM,
       JETCAT_ECU_OFF_Low_Rpm_Off,
       JETCAT_ECU_OFF_Low_Battery,
       JETCAT_ECU_OFF_Auto_Off,
       JETCAT_ECU_OFF_Low_temperature_Off,
       JETCAT_ECU_OFF_Hi_Temp_Off,
       JETCAT_ECU_OFF_Glow_Plug_defective,
       JETCAT_ECU_OFF_Watch_Dog_Timer,
       JETCAT_ECU_OFF_Fail_Safe_Off,
       JETCAT_ECU_OFF_Manual_Off, // (via GSU)
       JETCAT_ECU_OFF_Power_fail, // (Battery fail)
       JETCAT_ECU_OFF_Temp_Sensor_fail, // (only during startup)
       JETCAT_ECU_OFF_Fuel_fail,
       JETCAT_ECU_OFF_Prop_fail,
       JETCAT_ECU_OFF_2nd_Engine_fail,
       JETCAT_ECU_OFF_2nd_Engine_Diff_Too_High,
       JETCAT_ECU_OFF_2nd_Engine_No_Comm,
       JETCAT_ECU_MAX_OFF_COND
};

typedef struct
{
   UINT8   identifier;                       // Source device = 0x19
   UINT8   sID;                              // Secondary ID
   UINT16  FuelFlowRateMLMin;                // (BCD) mL per Minute
   UINT32  RestFuelVolumeInTankML;          // (BCD) mL remaining in tank
   // 8 bytes left
} STRU_TELE_JETCAT2;

//////////////////////////////////////////////////////////////////
//
//     GPS
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;                      // Source device = 0x16
   UINT8   sID;                             // Secondary ID
   UINT16  altitudeLow;                     // BCD, meters, format 3.1 (Low bits of alt)
   UINT32  latitude;                        // BCD, format 4.4,
                                            // Degrees * 100 + minutes, < 100 degrees
   UINT32  longitude;                       // BCD, format 4.4,
                                            // Degrees * 100 + minutes, flag --> > 99deg
   UINT16  course;                          // BCD, 3.1
   UINT8   HDOP;                            // BCD, format 1.1
   UINT8   GPSflags;                        // see definitions below
} STRU_TELE_GPS_LOC;

typedef struct
{
   UINT8   identifier;                      // Source device = 0x17
   UINT8   sID;                             // Secondary ID
   UINT16  speed;                           // BCD, knots, format 3.1
   UINT32  UTC;                             // BCD, format HH:MM:SS.S, format 6.1
   UINT8   numSats;                         // BCD, 0-99
   UINT8   altitudeHigh;                    // BCD, meters, format 2.0 (High bits alt)
} STRU_TELE_GPS_STAT;

// GPS flags definitions:
#define    GPS_INFO_FLAGS_IS_NORTH_BIT      (0)
#define    GPS_INFO_FLAGS_IS_NORTH          (1 << GPS_INFO_FLAGS_IS_NORTH_BIT)
#define    GPS_INFO_FLAGS_IS_EAST_BIT       (1)
#define    GPS_INFO_FLAGS_IS_EAST           (1 << GPS_INFO_FLAGS_IS_EAST_BIT)
#define    GPS_INFO_FLAGS_LONG_GREATER_99_BIT       (2)
#define    GPS_INFO_FLAGS_LONG_GREATER_99   (1 << GPS_INFO_FLAGS_LONG_GREATER_99_BIT)
#define    GPS_INFO_FLAGS_GPS_FIX_VALID_BIT (3)
#define    GPS_INFO_FLAGS_GPS_FIX_VALID     (1 << GPS_INFO_FLAGS_GPS_FIX_VALID_BIT)
#define    GPS_INFO_FLAGS_GPS_DATA_RECEIVED_BIT     (4)
#define    GPS_INFO_FLAGS_GPS_DATA_RECEIVED (1 << GPS_INFO_FLAGS_GPS_DATA_RECEIVED_BIT)
#define    GPS_INFO_FLAGS_3D_FIX_BIT        (5)
#define    GPS_INFO_FLAGS_3D_FIX            (1 << GPS_INFO_FLAGS_3D_FIX_BIT)
#define    GPS_INFO_FLAGS_NEGATIVE_ALT_BIT  (7)
#define    GPS_INFO_FLAGS_NEGATIVE_ALT      (1 << GPS_INFO_FLAGS_NEGATIVE_ALT_BIT)

//////////////////////////////////////////////////////////////////
//
//     GYRO
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8                      identifier;           // Source device = 0x1A
   UINT8                      sID;                  // Secondary ID
   INT16                      gyroX;                // Rotation rates of the body - Rate
                                                    // is about the X Axis which is
                                                    // defined out the nose of the
                                                    // vehicle.
   INT16                      gyroY;                // Units are 0.1 deg/sec  - Rate is
                                                    // about the Y Axis which is defined
                                                    // out the right wing of the vehicle.
   INT16                      gyroZ;                // Rate is about the Z axis which is
                                                    // defined down from the vehicle.
   INT16                      maxGyroX;             // Max rates (absolute value)
   INT16                      maxGyroY;
   INT16                      maxGyroZ;
} STRU_TELE_GYRO;

//////////////////////////////////////////////////////////////////
//
//     Alpha6 Stabilizer
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8                      identifier;           // Source device = 0x24
   UINT8                      sID;                  // Secondary ID
   UINT16                     volts;                // 0.01V increments
   UINT8                      state_FM;             // Flight Mode and System State
                                                    // (see below)
   UINT8                      gainRoll,             // Roll Gain,  high bit -->
                                                    //    Heading Hold
                              gainPitch,            // Pitch Gain
                              gainYaw;              // Yaw Gain
   INT16                      attRoll,              // Roll Attitude, 0.1degree, RHR
                              attPitch,             // Pitch Attitude
                              attYaw;               // Yaw Attitude
   UINT16                     spare;
} STRU_TELE_ALPHA6;

#define    GBOX_STATE_BOOT           (0x00)         // Alpha6 State - Boot
#define    GBOX_STATE_INIT           (0x01)         // Init
#define    GBOX_STATE_READY          (0x02)         // Ready
#define    GBOX_STATE_SENSORFAULT    (0x03)         // Sensor Fault
#define    GBOX_STATE_POWERFAULT     (0x04)         // Power Fault
#define    GBOX_STATE_MASK           (0x0F)

#define    GBOX_FMODE_FM0            (0x00)         // FM0 through FM4
#define    GBOX_FMODE_FM1            (0x10)
#define    GBOX_FMODE_FM2            (0x20)
#define    GBOX_FMODE_FM3            (0x30)
#define    GBOX_FMODE_FM4            (0x40)
#define    GBOX_FMODE_PANIC          (0x50)
#define    GBOX_FMODE_MASK           (0xF0)

//////////////////////////////////////////////////////////////////
//
//     6S LiPo Cell Monitor
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8                      identifier;           // Source device = 0x3A
   UINT8                      sID;                  // Secondary ID
   UINT16                     cell[6];              // Voltage across cell 1, .01V steps
                                                    // 0x7FFF --> cell not present
   UINT16                     temp;                 // Temperature, 0.1C (0-655.34C)
} STRU_TELE_LIPOMON;

//////////////////////////////////////////////////////////////////
//
//     14S LiPo Cell Monitor
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8                      identifier;           // Source device = 0x3F
   UINT8                      sID;                  // Secondary ID
   UINT8                      cell[14];             // Voltage across cell 1, .01V steps,
                                                    // excess of 2.56V (ie, 3.00V would
                                                    // report 300-256 = 44)
                                                    // 0xFF --> cell not present
} STRU_TELE_LIPOMON_14;
```

```
//////////////////////////////////////////////////////////////////
//
//     ATTITUDE & MAG COMPASS
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8                      identifier;           // Source device = 0x1B
   UINT8                      sID;                  // Secondary ID
   INT16                      attRoll;              // Attitude, 3 axes.  Roll is a
                                                    // rotation about the X Axis of
                                                    // the vehicle using the RHR.
   INT16                      attPitch;             // Units are 0.1 deg - Pitch is a
                                                    // rotation about the Y Axis of the
                                                    // vehicle using the RHR.
   INT16                      attYaw;               // Yaw is a rotation about the Z
                                                    // Axis of the vehicle using the RHR.
   INT16                      magX;                 // Magnetic Compass, 3 axes
   INT16                      magY;                 // Units are TBD
   INT16                      magZ;                 //
} STRU_TELE_ATTMAG;

//////////////////////////////////////////////////////////////////
//
//                            Real-Time Clock
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8                      identifier;           // Source device = 0x7C
   UINT8                      sID;                  // Secondary ID
   UINT8                      spare[6];
   UINT64                     UTC64;                // Linux 64-bit time_t for
                                                    // post-2038 date compatibility
} STRU_TELE_RTC;

//////////////////////////////////////////////////////////////////
//
//     RPM/Volts/Temperature
//
//////////////////////////////////////////////////////////////////
//
typedef struct
{
   UINT8   identifier;                      // Source device = 0x7E
   UINT8   sID;                             // Secondary ID
   UINT16  microseconds;                    // microseconds between pulse leading edges
   UINT16  volts;                           // 0.01V increments
   INT16   temperature;                     // degrees F
   INT8    dBm_A,                           // Average signal for A antenna in dBm
           dBm_B;                           // Average signal for B antenna in dBm.
                                            // If only 1 antenna, set B = A
} STRU_TELE_RPM;

//////////////////////////////////////////////////////////////////
//
//     QoS DATA
//
//////////////////////////////////////////////////////////////////
//
// NOTE:  AR6410-series send:
//         id = 7F
//         sID = 0
//         A = 0
//         B = 0
//         L = 0
//         R = 0
//         F = fades
//         H = holds
//         rxV = 0xFFFF
//
typedef struct
{
   UINT8   identifier;                      // Source device = 0x7F
   UINT8   sID;                             // Secondary ID
   UINT16  A;
   UINT16  B;
   UINT16  L;
   UINT16  R;
   UINT16  F;
   UINT16  H;
   UINT16  rxVoltage;                       // Volts, 0.01V increments
} STRU_TELE_QOS;

//////////////////////////////////////////////////////////////////
//
//     UNION OF ALL DEVICE MESSAGES
//
//////////////////////////////////////////////////////////////////
//
typedef union
{
   UINT16                     raw[8];
   STRU_TELE_RTC              rtc;
   STRU_TELE_QOS              qos;
   STRU_TELE_RPM              rpm;
   STRU_TELE_FRAMEDATA        frame;
   STRU_TELE_ALT              alt;
   STRU_TELE_SPEED            speed;
   STRU_TELE_ENERGY_DUAL      eDual;
   STRU_TELE_VARIO_S          varioS;
   STRU_TELE_G_METER          accel;
   STRU_TELE_JETCAT           jetcat;
   STRU_TELE_JETCAT2          jetcat2;
   STRU_TELE_GPS_LOC          gpsloc;
   STRU_TELE_GPS_STAT         gpsstat;
   STRU_TELE_GYRO             gyro;
   STRU_TELE_ATTMAG           attMag;
   STRU_TELE_POWERBOX         powerBox;
   STRU_TELE_ESC              escGeneric;
   STRU_TELE_LAPTIMER         lapTimer;
   STRU_TELE_TEXTGEN          textgen;
   STRU_TELE_FUEL             fuel;
   STRU_TELE_MAH              mAh;
   STRU_TELE_DIGITAL_AIR      digAir;
   STRU_TELE_STRAIN           strain;
   STRU_TELE_LIPOMON          lipomon;
   STRU_TELE_LIPOMON_14       lipomon14;
   STRU_TELE_USER_16SU        user_16SU;
   STRU_TELE_USER_16SU32U     user_16SU32U;
   STRU_TELE_USER_16SU32S     user_16SU32S;
   STRU_TELE_USER_16U32SU     user_16U32SU;
} UN_TELEMETRY;                             // All telemetry messages
```

REVISION HISTORY

Rev	Date	Author	Description
P0	2013-03-28	AK	For initial review.
P1	2013-04-04	AK	Fix address in JetCat_2 struct definition.
P2	2013-07-08	AK	Add RF data type/struct (Bug MD 1000).
P3	2013-07-10	AK	Add Gyro and Attitude/Compass info.
P4	2013-07-16	AK	Add 0x43 as reserved address. Correct text on TM1000.
P5	2013-11-19	AK	Change Dual Energy and MAH structs. Reserved addresses 0x30 and 0x32 for internal sensors, reassigned devices to 0x20 and 0x22.
P6	2014-03-31	AK	Correct ESC struct .currentBEC units to 100mA
P7	2014-05-05	AK	Revise ESC struct for powerOut and No Data sentinels
A	2015-01-16	AK	Release to the public.
B	2015-01-23	AK	Update temp resolution for ESC.
B’	2015-11-24	TB	Legal Information added for public release
C	2015-12-28	AK	Annotate Turbine fields as BCD per code.
D	2015-12-30	AK	Expand/Correct Turbine Status code values for more ECUs.
E	2016-02-16	AK	Integrate B’ into published document.
F	2016-03-06	AK	Correct Pbox ID in struct area, add Alpha6, add reference to SPMA9604/5.
G	2016-03-26	AK	Revise 6S, Add 14S LiPo Monitors
H	2016-07-28	AK	Add Lap Timer, Text Generator
I	2016-08-26	AK	Add dBm fields to RPM record
J	2016-08-30	AK	Update Lap Timer
K	2016-10-28	AK	Add RTC report device
L	2016-11-03	AK	Add Text description, update status of some sensors per Tx.