**Astronautical Development, LLC**   **Radio Interface Manual**
www.astrodev.com

**Table of Contents**

## Concept of Operations

*Command and Data Interface*

| Header (8 Bytes) | Payload (0 to 255 Bytes) | Payload Check Sum A (1 Byte) | Payload Check Sum B (1 Byte) |
| --- | --- | --- | --- |

**Figure 1--Packet structure for the Command and Data Interface (CDI).**

| Sync Characters (2 Bytes) | Command Type (2 Byte) | Payload Size (2 Bytes) | Header Check Sum A (1 Byte) | Header Check Sum B (1 Byte) |
| --- | --- | --- | --- | --- |

**Figure 2--Description of the packet header used in the CDI.**

Users interface to the Helium radios through the Command and Data Interface (CDI). Through the CDI, users configure the radio, query radio-specific telemetry points, send data to be transmitted, and request data that was received.

Users access the CDI through the UART port at 0-3.3V levels

The packet format for the digital interfaces is pictured above in Figure 1. It consists of an 8 byte, fixed length header, a variable payload segment from 0 to 255 bytes, and 2 check sum bytes. The header is described in Figure 2. The sync characters of the header are a two byte sequence[1]:

Sync Character 0: $0x48$ or 'H'

Sync Character 1: $0x65$ or 'e'

The next two bytes in a message header are the command type. Commands can be divided into two types representing the direction of the communications. Data entering the radio is noted as I-messages. I-messages are command types that begin with 0x10. Data leaving the radio is noted as O-messages. O-messages are messages that begin with 0x20. The full command list is given later in this documentation in section *Transceiver serial communications interface description*.

The "Payload Size" field of the header is a two byte, unsigned short integer containing the total number of bytes in the packet payload. The most significant byte (MSB) is given first. The maximum payload size is 255.

Two checksum bytes are appended to the header for error detection. The 8-bit Fletcher algorithm (see RFC 1145 which describes TCP) is used to calculate the checksums. The algorithm works as follows:

---

[1] Bit stuffing is not needed because checksums and packet lengths are used.

A buffer, $Buffer[N]$, contains data over which the checksum is to be calculated. The two checksum values ($CK\_A$ and $CK\_B$) are 8-bit unsigned integers only. Note, if you implement it with larger sized integers, be sure to mask both $CK\_A$ and $CK\_B$ with $0xFF$ after the calculations complete to ensure they are 8-bit. Psuedo-code for checksum calculation is given below.

```
CK_A = 0, CK_B = 0
For(I=0;I<N;I++)
{
    CK_A = CK_A + Buffer[I]
    CK_B = CK_B + CK_A
}
```

This loop calculates $CK\_A$ and $CK\_B$ which are then appended to the header. Following the header is the packet payload, which has a length as specified in the header. A payload checksum is then used to verify the accuracy of the payload. The checksum is calculated across all pertinent bytes of the message excluding the two sync characters of each message 'He'.

*Radio Configuration*

The default radio configuration is set at factory load during radio acceptance testing. This default configuration is stored in flash and is applied during power up and soft reset. The user can change the radio configuration after radio processor power up by providing a valid configuration command. The configuration message can change multiple settings at once. Changes take effect immediately, however the user should allow a settling time of at least 250 ms for settings to be applied. Default settings are described in section *Transceiver serial communications interface description*.

*Data Protocol Description*

Helium radios support a subset of the AX.25 packet radio protocol as defined by http://www.tapr.org/pub_ax25.html. Only the handling

of UI frames is implemented, not full connected mode. Users are able to configure the source and destination call signs, the packet length, and the TX tail and head parameters (see the command list in Table 1). The radio performs all packetization functions such as bit stuffing and check sum calculations.

*Transmit Overview*

Radio transmissions are performed based on the radio mode setting. All data received from the hardware interface is immediately transmitted unless the radio is busy with a current transmission. Data to be transmitted will be temporarily retained in the transmission buffer based upon the RF baud rate and the buffer length.

During transmission, the radio operates the as described previously in the Command and Data Interface section of this document.

These parameters default to standard AX.25 settings and are defined later in section *Transceiver serial communications interface description*.

*Receive Overview*

Radio receive operations are performed based on the radio mode setting and hardware interface method. The following list describes the methods.

- UART:
  Binary mode, AX.25 packet broadcast

During reception, the radio maintains the interface as described previously in the Command and Data Interface section of this document. When reading data in a polled method the delivery of data consists of the He header, followed by the command type and the payload size. The raw data received is placed into the payload section of the message.

These parameters default to standard AX.25 settings and are defined later in section *Transceiver serial communications interface description*.

**Example communications session, sending a No-Op command:**

First, the user implements a No-Op request. This is performed by loading an array with the proper values and sending them to the radio over a serial connection, below in pseudo code.

> *buffer[0] = SYNC_1; //This is a #define value of 'H'*
> *buffer[1] = SYNC_2; //This is a #define value of 'e'*
>
> *buffer[2] = I_MESSAGE_TYPE; //This is a #define value of 0x10*
> *buffer[3] = NO_OP_COMMAND; //This is a #define value of 0x01*
>
> *buffer[4] = 0x00; //There is no payload size information in a No-Op request*
> *buffer[5] = 0x00;*
>
> *calculate_header_checksum(&buffer[2]); //The first two synch bytes are not included in the checksum*
>
> *serial.Write( &buffer[0], 8 ); //send the information out your serial port*

The radio then responds to the request with either an acknowledge or not-acknowledge. An acknowledge is a response with the value 0x0A0A in the payload bytes. For example:

  Byte[0] = 'H';
  Byte [1] = 'e';
  Byte [2] = 0x20;
  Byte [3] = NO_OP_COMMAND;
  Byte [4] = 0x0A;
  Byte [5] = 0x0A;
  Byte [6] = Checksum A;
  Byte [7] = Checksum B;

A not-acknowledge is a response with 0xFFFF in the payload bytes. For example:
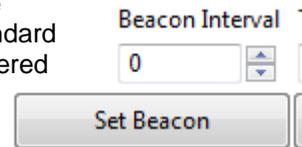
  Byte[0] = 'H';

**10/13/2014**

    Byte [1] = 'e';
    Byte [2] = 0x20;
    Byte [3] = NO_OP_COMMAND;
    Byte [4] = 0xFF;
    Byte [5] = 0xFF;
    Byte [6] = Checksum A;
    Byte [7] = Checksum B;

This is how most communications are performed with the radio. When messages include a payload, the response or command must contain a payload length value and the payload with the message.

## Beacon Use

This section provides an overview of the beacon. The beacon is a message that is transmitted intermittently based on user setting. The beacon data consists of up to 256 bytes and is set using the write beacon data message. The UART message to load the beacon data matches the standard Li-1 interface format. The beacon data can be updated at any time during normal powered operation.

The beacon is enabled by setting the beacon interval in the beacon configuration message to a value greater than zero. Each digit of interval corresponds to (n+1)*2.5 seconds of delay, graphic. For example a beacon configuration set with a interval of five will result in a beacon being transmitted every 15 seconds.

To set the beacon interval in the configuration program first insert a number into the beacon interval box and then press the "Set Beacon" button. The beacon will operate until the beacon interval is set to 0 again. The beacon contents are defined by the user using a Beacon Data command. This command loads up to 256 bytes of data into a fixed buffer for transmission. The contents of the beacon can be modified at any time and begin empty, with 0x00 at power up. If the user does not update the contents the radio will send one byte of 0x00 for the beacon including the standard AX.25 header information configured as default.

Beacon amplifier power level is the configured power level.

Beacon interleveing is automatically performed during a communication session unless the internal transmission buffers are full at the designated time of beacon transmission. If messages are currently in the buffer the beacon transmission will be added to the buffer and will be transmitted upon its turn.

## Receiver Options

This section provides an overview of the receiver options. The receive options are AFSK or CRC enable.

Audio FSK reception is only available as  a hardware ordered option.

Packets are received over the air and sent to the user as fast as possible. Only one option is available for filtering the packets by CRC. The CRC check can be enabled or disabled. If the packet has a invalid CRC the packet is rejected and not sent to the user. If the CRC is not enabled then the packet regardless of its checksum is sent to the user.

The receive CRC check is toggled by radio box on the windows configuration program.

**Telemetry and Logging (In Progress)**

This section provides an overview of the telemetry capabilities. The telemetry system is accessable from both UART interface and over the air. The telemetry system can internally log up to 1425 data points. The telemetry points are the standard telemetry packet. The telemetry can be accessed over the air or by the UART interface.

The telemetry storage operates as a FIFO circular buffer. Each telemetry packet has a radio CPU time count associated with its collection time allowing for alignment with other time stamps. Each time stamp corresponds to a specific interval set in the configuration message of 10, 1, 0.5, or 0.25 seconds. Based on the time interval setting telemetry can be collected for up to approximately 4, 0.4, 0.2, or 0.1 hrs.

Over the air telemetry access is enabled by the UART interface and users transmitting the correct access packet to download a telemetry packet.

This functionality is in progress as of release V3.11.

**Digital IO and Over the Air Digital IO**

This section provides an overview of the digital input and output pins available on the radio. The digital IO lines can be used for any functionality desired. Over the Air digital IO is restricted to only Config Pin#1 at this time. Digital IO allows users to use activity signals generated by the radio to operate system redundancies in the spacecraft. The digital IO can be triggered by a 2.5 Hz timer event, received packet, or transmitted packet events. For example, the user can configure a 30 second beacon which upon transmission will toggle the digital IO line to verify radio operation.

Over the air digital IO allows users to access the low level pin functions to perform system restore activities.

This feature is completed as of release V3.11.

Digital IO pins can be configured in the Windows program using the three radio boxes. Each radio box configures the possible behavior for each pin. Users should select the desired behavior then press the write configuration button. Note, to ensure that current default values in the radio are not over written, press the "read configuration" button first.

**10/13/2014**

OA Key

Users may use a 16 byte key to access the digital IO commands from a ground station. To set the 16 byte key the user should type in the HEX value in the OA Key window in the configuration screen, then press the "Write OA Key" button. Next the user should enable over the air access to the radio configuration. This enables these functions to be interpreted by the radio.

Step 1: Enter a OA Key: Example "Hi          " = 0x48692020202020202020202020202020
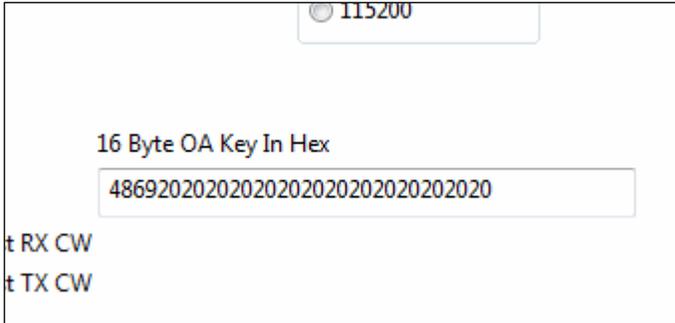


Step 2: Enable OA Functions: Note: "Enable Ping" is currently mis-labeled.
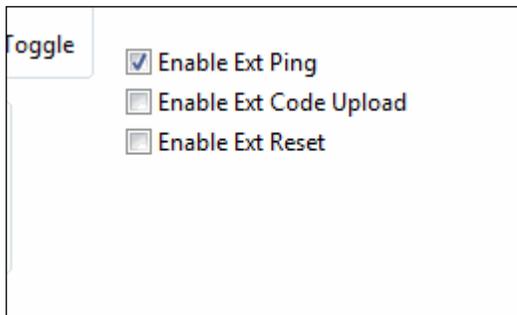


Step 3: Enable desired reset pin behavior:
Pattern A: Pin Logic Low to Logic High, Latches High
Pattern B: Pin Logic Low to Logic High to Logic Low. Pulse approx. 72 milliseconds
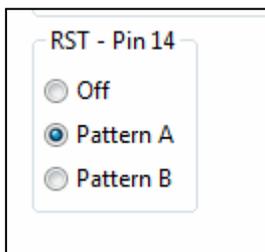


Step 4: Sending a digital command from a ground station:

There are currently 4 commands available for users in OA functions. They are:
#define TELEMETRY_DUMP_COMMAND 0x30 //'0' (functionality pending)
#define PING_RETURN_COMMAND      0x31 //'1'
#define CODE_UPLOAD_COMMAND      0x32 //'2' (functionality pending)
#define RADIO_RESET_COMMAND      0x33 //'3'

#define PIN_TOGGLE_COMMAND        0x34 //'4'
The command is located in a packet immediately proceeding the OA Key.

To send a pin toggle to reset a spacecraft, using the OA Key described previously, the user should thus send"
"Hi        4" or 0x48692020202020202020202020202034

## Firmware Upload (In Progress)

This section provides an overview of the firmware upload capability. The firmware upload capability allows users to upgrade flight software without having immediate access to the radio. The firmware is accumulated at any time when the option is enabled. Upon completion of a firmware image the radio will perform an update by command.

A firmware image is uploaded to the radio using the available images from the AD website. Users must transmit each valid packet to the system then send the firmware update command with the corresponding MD5 checksum which matches the firmware image. At the time of command the radio will enter the flash update interval. Power must not be removed from the radio during this flash update time.

This functionality is in progress as of release V3.11.

## Fast Commands

This section provides an overview of the fast commands capability.  A set of reduced size commands are used for fast adjustment of the radio during pass operations.

> Power Amplifier Setting:
> To allow users to quickly adjust the power output level to changes in system voltage, dynamic link adjustment, or efficiency optimization the power amplifier command can be adjusting using a 1 byte payload packet. This is done with the 0x1020 command which has a one byte payload that over writes the current power amplifier setting in the full radio configuration. Within the configuration program this can be found on the "Telemetry and Misc" page, graphic.

This functionality is in progress as of release V3.11.

## Data Rates

This section provides an overview of the multi data rate capabilities. The data rate settings of the radio are specific to the hardware type purchased. Increasing data rate uses different filter, modulation, and power settings than traditional AX.25 HAM communications. Each option is described below.

> Audio FSK: Audio FSK operates at 1200 baud. When configuring for Audio FSK the user needs to ensure that the data rate setting matches the selection of modulation. If the settings do not match the radio will not receive over the air information and will not notify the user has entered an undesirable operation mode. Audio FSK receive functions are only available in the Li-2 model radio.

> 9.6 kbps GFSK: Standard HAM AX.25 data rate.

> 19.2 kbps GFSK: Higher speed GFSK modulation.

> 38.4 kbps GFSK: Higher speed GFSK modulation.

This functionality is in progress as of release V3.11.

## Low Level RF Configuration (In Progress)

This section provides an overview of the low level RF configuration. The low level RF configuration allows users to adjust for on the fly efficiency, dopplar, and performance.

This functionality is in progress as of release V3.11.

**Transceiver Serial Communications Interface Description**

This section provides an overview of commands sent from the host to the radio over the CDI and the response from the radio.

Table 1  -- Summary of Input (I) Commands and Output (O) Radio Output.

| Op Code | Command/Response Name | Arguments | Total Bytes | Summary |
|---|---|---|---|---|
| 0x1001 | No-Op | - | 0 | No-op command.  Increments command processing counter. |
| 0x2001 | No-Op Ack | - | 0 | No-op Acknowledge. |
| 0x1002 | Reset | - | 0 | Reset radio processors and systems. |
| 0x2002 | Reset Ack | - | 0 | Reset Acknowledge. |
| 0x1003 | Transmit | Bytes | N | Send n number of bytes to radio board. |
| 0x2003 | Transmit Ack | - | 0 | Transmit Acknowledge. |
| 0x2004 | Received Data | Bytes | N | Received n number of bytes AX.25 packet |
| 0x1005 | Get Transceiver Configuration | - | 0 | Read radio configuration. |
| 0x2005 | Transceiver Configuration | Configuration Structure [2] | N | Radio configuration structure. |
| 0x1006 | Set Transceiver Configuration | Configuration Structure | N | Set radio configuration. |
| 0x2006 | Set Transceiver Configuration Ack | - | 0 | Set radio configuration Acknowledge. |
| 0x1007 | Telemetry | - | 0 | Query a telemetry frame. |
| 0x2007 | Telemetry | Telemetry Structure | N | Telemetry frame.[3] |
| 0x1008 | Write Flash | 16 Byte MD5 | 16 | Write Flash with MD5 Checksum |
| 0x2008 | Write Flash Ack | - | 0 | Write Flash Acknowledge |
| 0x1009 | RF Configure | RF Structure | N | Low Level RF Configuration |
| 0x2009 | RF Configure Ack | - | 0 | RF Configuration Acknowledge |
| 0x1010 | Beacon Data | Bytes | N | Set Beacon Contents |
| 0x2010 | Beacon Data Ack | - | 0 | Ack Set Beacon Contents |
| 0x1011 | Beacon Configure | Beacon Structure | N | Set beacon configuration |
| 0x2011 | Beacon Conf. Ack | - | 0 | Ack Beacon Configuration |
| 0x1012 | Read Firmware Rev | - | 0 | Read radio firmware revision. |
| 0x2012 | Firmware Rev | Bytes | 4 | Firmware number, float 4 byte |
| 0x1013 | DIO Key Write | Bytes | 16 | DIO Key Write |
| 0x2013 | DIO Key Write Ack | - | 0 | Ack DIO Key Write |
| 0x1014 | Firmware Update | 16 Byte MD5 | 16 | Firmware Update Command |
| 0x2014 | Firmware Update Ack | - | 0 | Firmware Update Ack |

---

[2] Refer to transceiver configuration message description.
[3] Telemetry points and their code are described in the next section.

| 0x1015 | Firmware Packet | Bytes | TBD | Firmware Packet Write |
|--------|-----------------|-------|-----|-----------------------|
| 0x2015 | Firmware Packet Ack | - | 0 | Firmware Packet Ack |
| 0x1020 | Fast Set PA | Byte | 1 | Power Amplifier Level Set High Speed |
| 0x2020 | Fast Set PA Ack | - | 0 | Power Amplifier Set Ack |

*No-Op Message: 0x01*
The user sends:

| Hex | 0x48 | 0x65 | 0x10 | 0x01 | 0x00 | 0x00 | Ck_A | Ck_B |
|---|---|---|---|---|---|---|---|---|
| ASCII/Dec | 'H' | 'e' | 16 | 1 | 0 | 0 | Calculated | Calculated |

On Success the radio replys with and Acknowledge:

| Hex | 0x48 | 0x65 | 0x20 | 0x01 | 0x0A | 0x0A | Ck_A | Ck_B |
|---|---|---|---|---|---|---|---|---|
| ASCII/Dec | 'H' | 'e' | 32 | 1 | 11 | 11 | Calculated | Calculated |

On Failure the radio replys with Not-Acknowledge:

| Hex | 0x48 | 0x65 | 0x20 | 0x01 | 0xFF | 0xFF | Ck_A | Ck_B |
|---|---|---|---|---|---|---|---|---|
| ASCII/Dec | 'H' | 'e' | 32 | 1 | N/A | N/A | Calculated | Calculated |


Other messages using the same format:
*Reset Message: 0x02*

Performs soft reset of the internal processor.

*Transmit Message: 0x03*

Transmits the message up to 256 bytes. If currently transmitting the message is stored in the 6 deep buffer. If the buffer is full a NACK is issued.

*Received Data Message: 0x04*

When the radio receives a packet it is forwarded across the UART interface.

*Get Transceiver Configuration Message: 0x05*

Requests the configuration message.

*Set Transceiver Configuration Message: 0x06*

Writes the RF front end configuration of the radio, 34 bytes

*Telemetry Query Message: 0x07*

Requests the telemetry packet.

*Write Flash Message: 0x08*

Stores the current configuration in radio flash.

*Write RF Configuration Message: 0x09*

Writes the RF front end configuration of the radio, 10 bytes

*Write Beacon Data Message: 0x10*

Sets the beacon packet data, up to 256 bytes

*Write Beacon Configuration Message: 0x11*

Writes the rate at which the beacon is transmitted

---

*Read Firmware Revision: 0x12*

Returns 4 byte float value of firmware revision, ie 3.06

*Write Digital IO Over Air Key: 0x13*

Writes the 16 byte sequence used to trigger external events.

*Firmware Update: 0x14*

*Firmware Packet: 0x15*

*Write Fast Power Amplifier Level: 0x20*

**The Configuration Structure**

*Compiler Directives:*

```
//Configuration Commands
#define NO_OP_COMMAND            0x01
#define RESET_SYSTEM             0x02
#define TRANSMIT_DATA            0x03
#define RECEIVE_DATA             0x04
#define GET_TRANSCEIVER_CONFIG 0x05
#define SET_TRANSCEIVER_CONFIG 0x06
#define TELEMETRY_QUERY          0x07
#define WRITE_FLASH              0x08
#define RF_CONFIG                0x09
#define BEACON_DATA              0x10
#define BEACON_CONFIG            0x11
#define READ_FIRMWARE_REVISION 0x12
#define WRITE_OVER_AIR_KEY       0x13
#define FIRMWARE_UPDATE          0x14
#define FIRMWARE_PACKET          0x15
#define FAST_PA_SET              0x20

#define BAUD_RATE_9600    0
#define BAUD_RATE_19200   1
#define BAUD_RATE_38400   2
#define BAUD_RATE_76800   3
#define BAUD_RATE_115200  4

#define RF_BAUD_RATE_1200    0
#define RF_BAUD_RATE_9600    1
#define RF_BAUD_RATE_19200   2
#define RF_BAUD_RATE_38400   3
#define RF_BAUD_RATE_57600   4
#define RF_BAUD_RATE_115200  5

#define RF_MODULATION_GFSK   0
#define RF_MODULATION_AFSK   1
#define RF_MODULATION_BPSK   2

#define TELEMETRY_DUMP_COMMAND 0x30
#define PING_RETURN_COMMAND      0x31
#define CODE_UPLOAD_COMMAND      0x32
```

```
#define RADIO_RESET_COMMAND         0x33
#define PIN_TOGGLE_COMMAND          0x34
```

*Structure:*

```
typedef struct
{
  uint_1 interface_baud_rate;  //Radio Interface Baud Rate (9600=0x00)
  uint_1 tx_power_amp_level;    //Tx Power Amp level (min = 0x00 max = 0xFF)
  uint_1 rx_rf_baud_rate;       //Radio RX RF Baud Rate (9600=0x00)
  uint_1 tx_rf_baud_rate;       //Radio TX RF Baud Rate (9600=0x00)
  uint_1 rx_modulation;         //(0x00 = GFSK);
  uint_1 tx_modulation;         //(0x00 = GFSK);
  uint_4 rx_freq;               //Channel Rx Frequency (ex: 45000000)
  uint_4 tx_freq;               //Channel Tx Frequency (ex: 45000000)
  unsigned char source[6];      //AX25 Mode Source Call Sign (default NOCALL)
  unsigned char destination[6]; //AX25 Mode Destination Call Sign (default CQ)
  uint_2 tx_preamble;           //AX25 Mode Tx Preamble Byte Length (0x00 = 20 flags)
  uint_2 tx_postamble;          //AX25 Mode Tx Postamble Byte Length (0x00 = 20 flags)
  uint_2 function_config;       //Radio Configuration Discrete Behaviors
  uint_2 function_config2;      //Radio Configuration Discrete Behaviors #2
} RADIO_CONFIGURATION_TYPE;
```

**function_config bit values**

Li-1 Pin #12 External Event Functions, He-100 LED
```
0000 0000 0000 xx00   Off Logic Low
0000 0000 0000 xx01   2.5 second Toggle
0000 0000 0000 xx10   TX Packet Toggle (12 ms toggle)
0000 0000 0000 xx11   Rx Packet Toggle  (1.25 ms toggle)
```

Li-1 Pin #13 Config #2  Functions, He-100 Ant_P Pin 4
```
0000 0000 0000 00xx   Off Logic Low
0000 0000 0000 01xx   Tx/Rx Switch (approx 0.35 seconds high, depends on pre/postamble)
0000 0000 0000 10xx   2.5 hz WDT
0000 0000 0000 11xx   Rx Packet Toggle (1.25 ms toggle)
```

Li-1 Pin #14 Config #1 Functions, He-100 Ant_P Pin 3
```
0000 0000 xx00 xxxx   Off Logic Low
0000 0000 xx01 xxxx   Digital IO Over the Air Key Enable
0000 0000 xx0X xxxx   DIO Over Air Pattern A (Latching High)
0000 0000 xx1X xxxx   DIO Over Air Pattern B (Toggle, 72 ms high)
```

CRC Functions
```
0000 0000 x1xx xxxx   RX CRC Enable 1/Disable 0
0000 0000 1xxx xxxx   TBD // TX CRC Enable 1/Disable 0
```

Telemetry Functions
```
0000 xxx1 xxxx xxxx   Telemetry Packet Logging Enable 1/Disable 0
0000 x01x xxxx xxxx   Logging Rate 0 1/10 Hz, 1 1 Hz, 2 2 Hz,3 4 Hz
0000 1xxx xxxx xxxx   Telemetry Dump Enable 1/Disable 0
```

Beacon Functions
```
0xx1 xxxx xxxx xxxx   Enable OA Commands Enable 1/Disable 0
0x1x xxxx xxxx xxxx   Code Upload Enable 1/Disable 0
01xx xxxx xxxx xxxx   Radio Reset Enable 1/Disable 0
```

**10/13/2014**

Factory Defaults Restored
1xxx xxxx xxxx xxxx     Factory settings restore complete flag

**function_config2 bit values**

Li-1 Functions
0000 0000 0000 xxx1    Receiver Automatic Frequency Control On/Off
0000 0000 0000 xx0x    RX CW (CAUTION TEST MODE)
0000 0000 0000 x0xx    TX CW (CAUTION TEST MODE)
0000 0000 0000 0xxx    TBD (CAUTION TEST MODE)

**The Telemetry Structure**

```
typedef struct
{
  uint_1 front_end_level; //0 to 63 Value
  uint_1 tx_power_amp_level; //0 to 255 value, non-linear
  uint_4 tx_frequency_offset; //Up to 20 kHz
  uint_4 rx_frequency_offset; //Up to 20 kHz
  uint_1 tx_frequency_deviation; //Set for your baud rate options: 0 (2.7 kHz),1 (5.4 kHz),2 (10.8 kHz),3 (21.6 kHz),4
(43.2 kHz) CAUTION
  uint_1 rx_frequency_deviation; //N/A for release 3.10
  uint_1 pre_transmit_delay;//Delay in tens of milliseconds. Default 1 second = 100;
  uint_1 post_transmit_delay;//Delay in tens of milliseconds. Default 0  (NOT IMPLEMENTED USE POSTAMBLE)
} RADIO_RF_CONFIGURATION_TYPE;
```

**The Low Level RF Structure**

```
typedef struct
{
  uint_1 front_end_level; //0 to 63 Value
  uint_1 tx_power_amp_level; //0 to 255 value, non-linear
  uint_4 tx_frequency_offset; //Up to 20 kHz
  uint_4 rx_frequency_offset; //Up to 20 kHz
  uint_1 tx_frequency_deviation; //Set for your baud rate options: 0 (2.7 kHz),1 (5.4 kHz),2 (10.8 kHz),3 (21.6 kHz),4
(43.2 kHz) CAUTION
  uint_1 rx_frequency_deviation; //N/A for release 3.10
} RADIO_RF_CONFIGURATION_TYPE;
```

**The Beacon Structure**

```
typedef struct
{
  uint_1 beacon_interval;  //value of 0 is off, 2.5 sec delay per LSB
} RADIO_BEACON_CONFIGURATION_TYPE;
```

**The MD5 Checksum**

Please refer to the standard MD5 algorithm:
http://en.wikipedia.org/wiki/MD5

**Trademarks**
In progress.

**Disclaimer**
All information in this document is subject to change at anytime. Look for continued updates at:
http://www.astrodev.com/

**Notes**