HAB Tracker Transmitter Program Notes

Stuart Robinson – August 2016

1 Tracker Transmitter Program - Introduction

The HAB tracker transmitter program is designed primarily to be used as a small and lightweight power efficient tracker as would typically be used for a PICO balloon floater.

The software is intended to be used on an Arduino Pro Mini, 3.3V version running at 8Mhz. The LoRa transceiver can be a Dorji DRF1278F or Hope FRM98. The GPS is a UBLOX Max8Q.

The HAB tracker software is written for the Arduino IDE, and compilation was tested against the latest version, 1.68 at the time of writing.

You will need to download and install the following libraries;

TinyGPS++. - http://arduiniana.org/libraries/tinygpsplus/

Low Power, https://github.com/rocketscream/Low-Power

Flash-5 Library - https://github.com/mikalhart/Flash/releases

There is an issue with the Flash-5 library, changes to the Arduino IDE since the Library was published cause a compile error; *'prog_char' does not name a type*. The solution to this is to edit the Flash.h file in the Flash-5 Library. Add the following lines in Flash.h, just after the #include <avr/pgmspace.h>:

```
#if ARDUINO >= 150
typedef char prog_char __attribute__((__progmem__));
#endif
```

There are other Arduino library files used which are part of the standard IDE build which are; Arduino.h, Wire.h, SPI.h, EEPROM.h, avr/pgmspace.h.

The programs are split between a primary program file (the. .ino) file, and several .h files as includes, these are;

Tracker_Definitions.h – Settings and pin definitions used by the programs

LoRa_Registers.h – Definitions for all the LoRa register names in use

LoRa.h – Routines for LoRa both packet and direct modes

FSK_RTTY.h - Generate FSK RTTY from LoRa device in direct mode

GPS.h – GPS routines.

Internal_CPU.h – Internal read of CPU VCC and temperature

Flight_Settings_1.h – Contains settings for the particular flight

1.1 Reading VCC and Temperature

The software uses some routines from the Internal_CPU.h file to take readings of the CPU VCC and temperature. These are not accurate indications but good enough for seeing what's happening with a HAB tracker.

For a HAB tracker set-up with no regulator and powered from 2 xx AAA Lithium batteries, then the CPU VCC is the same as the battery voltage.

To be near accurate there are constants that need to be determined for each individual CPU, the program notes for the LoRA Tracker Test Program has the details on how to calculate these constants. .

Alternatively it would not be difficult to add the code for a BME280 or BMP280 sensor, there are libraries available for these sensors.

2 Software Summary

The software reads the GPS via its I2C interface and builds a UKHAS style payload in a RAM buffer. This buffer is then transmitted as LoRa telemetry and FSK RTTY.

There is a short pause after the transmissions whilst the tracker listens for incoming command packets. Commands are defined for packet and RTTY test, tracker reset and disabling or enabling various options. These are the options can be currently be changed remotely by direct commands.

Disable of Enable FSK RTTY.

Disable or Enable address strip

Disable or Enable GEO fence check

Disable or Enable GPS power off

Adjust frequency offset

Link Budget Request

In addition most all of the program constant settings can be changed by remote command since they are stored in EEPROM and there is a command that can remotely program the EEPROM storage, there are more details of this later.

Hardware (pin) definitions are in the Tracker_Definitions.h file and allow the program to be used with different hardware set-ups.

The software is designed specifically to minimise power consumption and the tracker is put to sleep between GPS Fix and transmission cycles.

The settings for a particular flight, frequencies et, are in the Flight_Settings file, this is designed to be copied across to the matching receiver program so that program picks up the correct settings.

The software is arranged in a manner that I hope makes clear how it operates and what needs to be done to make custom changes. In addition there are a lot of diagnostic messages sent to the Arduino serial monitor. These messages are not needed during flight of course but they are very useful during software testing and proving.

3 Packet Addressing

Most often on a particular frequency there will be one transmitter and one receiver. However, this may not always be the case and there could be several trackers in use as transmitters on the same frequency.

In order to keep the software simple and allow for the receipt of signals from multiple receivers or directed commands to a particular tracker, a basic addressing scheme has been implemented. The key to this is that regardless of the data content of the actual payload each payload is preceded in the transmitted packet with 3 control bytes. In general the control bytes have been restricted to ASCII printable characters so that they can be shown directly on a terminal\monitor, the 3 bytes are;

Packet type. This either describes the content of the packet, which could be a GPS location payload or is a command to do something and there is no payload.

Packet Destination. The node number that the packet is destined for.

Packet Source. The node number that the packet was sent from.

The destination and source packet bytes mean that node ‘2’ (could be your base station receiver) can send a command that only station ‘3’ can be programmed to respond to. This command could be a reset command, a request to turn on and off certain transmission types etc.

In my tracker software the 3 control bytes are automatically added to each outgoing packet and stripped from each received packet.

An example of the 3 control bytes from a tracker would be;

G*2

Which means there is a GPS error (G) its been sent as a broadcast (*) and its from node 2.

This simple protocol can be used to queue requests to intermittent receivers. For instance the high altitude balloon version of my tracker software assumes that the in flight tracker does not listen for commands continuously; it only listens for incoming packets for short periods in order to minimise power consumption. At the beginning of a listen period (which may only be a few seconds) the tracker sends a ready (clear to send) packet. The receiver is normally not so power constrained so is left in permanent listen mode. The receiver has been given a queued request (could be to release a payload command for instance) and is waiting for the ready packet from the tracker. When the ready packet is received, the receiver sends the queued

command and listens for an acknowledge packet. If the receiver does not receive an acknowledge it will keep queuing the request.

3.1 Defined Packet Types

ACK = 'A'
LinkReport = 'B'
ClearToSend = 'C'
ClearToSendCommand = 'c'
Error = 'E'
NoFix = 'F'
NoGPS = 'G'
GLONASSDetected = 'g'
Memory = 'M'
NACK = 'N'
NACKCommand = 'n'
PowerUp = 'P'
Repeated = 'R'
ShortPayload = 'S'
Test = 'T'
Wakeup = 'W'
ResetTracker = 'X'
Config1 = 'Y'
Config0 = 'Z'
WritePacketEEPROM = '0'
LongPayload = '$'
Bind = '#'

3.2 Use of Packet Addressing

The high altitude balloon tracking software uses the packet addressing to implement a command and control interface between the ground receiver and the remote balloon. This program structure has been used previously and is basically unchanged since its first flight outing in January 2015 when a PICAXE processor was used.

The HAB tracker software has a constant back chatter of short command packets, these are used for;

A power up ‘Alive’ packet sent at restart\power up that allows the receiver to directly display the tracker supply voltage.

Packets sent during initial first GPS fix to indicate to the receiver that there is no fix.

Packet sent when GPS first fix is acquired.

Standard UKHAS style payload packet

A Clear to send (CTS) packet at the beginning of the tracker listen period.

Sending test requests to the tracker.

Changing various configuration options, FSK RTTY, packet output address strip, GPS power saving etc.

A generic routine that will change specific location in EEPROM. This enables the receiver to make adjustments to the tracker parameters stored in EEPROM, such as frequency, offset, gaps between transmissions etc.

3.3 Program Constants in EEPROM

Most of the tracker transmitter program constants and definitions, such as frequency, LoRa settings, sleep periods, GPS updates, payload ID etc are contained in the file ‘Flight_Settings.h’.

When the tracker program is programmed into the Pro Mini this option should be selected;

#define WriteEEPROM //Write initial values of changeable constants into EEPROM

When the program then runs these constants are copied into EEPROM. Once that has been done then at tracker start or reset these ‘constants’ are read from EEPROM into RAM variables and thereafter treated as constants, the . ‘**#define WriteEEPROM’** option can then be commented out.

Initially this may seem perverse, but once the program constants are in stored in EEPROM they can be changed remotely without the need to directly re-program the tracker.

Thus once a HAB tracker is in flight commands can be sent to it to compensate for frequency drift or change RTTY baud rates and these changes will survive reset or power downs.

Clearly you can get into a big mess doing changes remotely, you could for instance move the frequency to one where reception (by the remote tracker) is not possible. To allow recovery from this problem, the software has an option to enable a bind mode, see this line;

#define EnableBind //at startup tracker will listen for a bind packet

This bind mode is on a defined frequency and with a fixed set of LoRa parameters that cannot be changed remotely. The bind mode when activated will send out a bind request at start-up and during every GPS read and transmission loop. The receiver could therefore take the role of uploading all of the definitions in ‘Flight_Settings.h’ for a particular flight. At present the receiver does not implement this fuctionality.

With the Flight_Settings in EEPROM it is also possible (with suitable receiver software) to easily make last minute adjustments to operating frequency or LoRa parameters without having to unpack or take apart a pre-programmed tracker.

I hope to be able to improve the receiver software in future so that tracker flight settings can be set by a series of terminal menus and\or stored as text files on SD Flash cards.

4 Config Byte

The program uses a single config byte where individual bits act as indicator flags or control what the tracker program does. This config byte goes out as a number (0-255) with the standard location payload, so basic information of what the tracker is up to can be seen remotely.

The bits of the config byte are defined as follows;

Bit 0, flag bit when set indicate GPS has a fix

Bit 1, flag bit when set indicates a GLONASS NMEA sentence was detected from GPS

Bit 2, when set enables FSK RTTY

Bit 3, when set enables the GEO fence check

Bit 4, when set enables short payload (not implemented at this time)

Bit 5, when set enables packet repeat mode (not implemented at this time)

Bit 6, when set will enable strip of packet addressing from UKHAS payload

Bit 7, when set bit when set enables GPS on\off power save.

There is a command to change the config byte so the receiver can change the program operations above remotely.

5 Flight LoRa Settings and Command LoRa settings

The program allows for the standard tracker transmissions (LoRa and FSK RTTY location payloads) to be sent on separate frequencies and\or LoRa modem settings to the command and control interaction.

Whilst flight and command can be set-up with the same frequencies and LoRa settings there are circumstances when this setup encourages illegal operation, so the programs compensate for this.

In most cases the receiver base station will be connected to an omni directional gain antenna, typically this will have 5dB forward gain. This is a benefit for reception of course, you can receive at greater distances, but it causes difficulties when the receiver transmits commands through the gain antenna. For licence exempt use restrictions apply to the effective radiate power (ERP) so if you were to set the LoRa device for 10dBm power output (the normal UK limit) then the gain antenna would be transmitting 15dBm ERP, which is too much. You could of course simply reduce the power output by 5dBm but now you have a mismatch. You can receive signals from the tracker but may not be able to send commands since 5dB is now missing from the link budget.

The answer to this dilemma, and what is necessary to make the setup legally compliant, is to reduce the receivers transmit power (by 5dBm say) but use a set of LoRa parameters that have a 5dB link gain over the flight settings. This does mean that the packets sent to the tracker will take longer to transmit, but they are usually short packets in any case.

If you choose the command frequency can be different to the location payload transmissions so that all the average listener only sees the standard location payload, the command and control interaction is ‘hidden’ on another frequency.

The tracker and receiver software is designed so that both use a copy of the same ‘Flight_Settings.h’ text file. This ensures that the frequency and LoRa settings are co-ordinated between the two programs.

So when setting up frequencies and LoRa settings for a flight, make the changes to the ‘Flight_Settings.h’ in one program (transmitter or receiver) and then copy the file to the other program.

6 Power Saving

For HAB trackers to be used for PICO floater balloons the tracker needs to be light, and this implies a small and light battery, assuming you want to avoid the complexity of implementing solar charging.

6.1 Processor Power Saving

The ATMEGA328 will use approximately 4mA when running at 8Mhz and 3V. This may not seem a lot, but if the power source was a 200mA Lithium polymer battery then it would be flat in 50 hours just keeping the processor running.

In a typical 1 minute transmission and update cycle the processor is only needed for 15 seconds or so. Therefore if the processor is put to sleep for 45 seconds, battery life in the above example is increased to circa 200 hours.

Using the low power library the processor can be put to sleep for defined periods (which are approximate) and the supply current falls to 30uA or so.

6.2 GPS Power Saving – Hot Start

If the UBLOX GPS is put into cyclic power save mode, assuming the GPS antenna is good enough, reduce its average power consumption to below 10mA. This is far lower than previous generations of GPS, but still a high power consumption for a small battery.

The GPS can be powered down after the first fix acquisition and used in hot start mode. This is a universal method that can be used with most GPSs that provide for battery backed up last position and almanac. The backup supply can either be a small lithium battery, supercap or simple a voltage source provided by the controlling microcontroller.

The tracker board design has a MOSFET and associated components to switch of the power supply going to the GPS. It is then restarted some time later, several minutes maybe, and should with a good GPS antenna acquire a new fix in 5 seconds or less. Using hot fix in this way can allow for very extended tracker operation from small batteries, if very regular position updates are not required.

6.3 GPS Power Saving – Software Backup Mode

UBLOX GPS provide methods by which particular parts of the GPS receiver can be turned off via software in order to reduce power consumption. One of the easiest mechanisms to use with the UBLOX GPS is the software backup mode. By sending an appropriate UBX command, the GPS powers off (mostly) within a couple of seconds of receiving the command. The GPS can be woken from sleep by any activity on its serial input pin, a single pulse is sufficient.

In software backup mode the power consumption of the GPS falls to circa 500uA, so its not really suitable method if the GPS off times are significantly extended in order to get multi month or year endurance from small batteries. Powering the GPS off completely using a MOSFET as described above is a better method in such circumstances.

Having said that, the Software Backup method is suitable for high altitude balloon trackers and it removes the need for the additional components to switch off the GPS.

6.4 GLONASS Mode

To use the power saving modes the UBLOX GPS needs to have GLONASS mode disabled. This is straight forward and carried out at power up. On occasion I noticed during testing that the GPS would occasionally slip back into GLONASS mode, which unfortunately means the TinyGPS++ software library does not recognise the sentences output. To get around this potential problem the software uses a custom TinyGPS++ sentence definition to check if the GPS is putting out GLONASS NMEA and reconfigure the GPS back.

7 Tracker in Listen Mode

To (significantly) reduce power consumption the tracker is only in listen mode for commands for a short period, typically 3 seconds. At the start of the listen period the CTS packet is sent which the receiver detects and begins the desired interaction.

8 Program Flow following processor reset

Copies program constants to EEPROM, need to be run once only.

Copy constants from EEPROM into RAM.

Initialise hardware

Send start up messages to console

Transmit a FM to for frequency counter check (optional)

Send a power up packet

Check for bind mode.

Set-up GPS for flight mode

Wait for GPS fix

Set GPS power save and cyclic mode

Start main loop

Check for new GPS fix

Power down GPS (if enabled)

Increment sequence number

Set-up LoRa for transmission

Build long payload

Send FSK RTTY (if enabled)

Send LoRa payload (always enabled)

Listen for command

Update mAhr used calculation

Go into sleep mode

Wake up

Read CPU temperature (when cold!)

Resume program loop

9 Calculation of milliamp hours used.

This is based on actual measurements taken on a working board the software calculates based on running times how much power the tracker has used. The running mAhr is sent out with the tracker payload so you can see how well (or not!) the tracker power saving is performing.

If the software has power save turned off (implying that the GPS is in cyclic mode) the mAhr used does not reflect and power used by the GPS.

10 Tracker test functions

The tracker can be commanded to send a some communications tests, these are useful for checking how well a link is working and how close to failure it is.

10.1 FSK RTTY test

The tracker sends very short FSK RTTY ‘packets’ These start at 10dBm transmitted power and fall to 2dBm power. Thus if the receiver reports all packets down to 7dbm, you have 3 dB of link margin remaining.

10.2 LoRa packet test

The tracker sends very short LoRa packets. These start at 10dBm transmitted power and fall to 2dBm power. Thus if the receiver reports all packets down to 7dbm, you have 3 dB of link margin remaining.

10.3 Link budget report

The tracker responds with the RSSI and SNR of the last LoRa packet it received.

11 Problems and Issues

11.1 GPS Fix

If your having problems getting a first GPS fix, load and run the appropriate GPS Echo program, there is one for Serial GPSs another for I2C GPSs. These programs Echo all the GPS output to the serial monitor so you can see the progress of the GPS towards fix.

If the GPS is a Ublox Max8 type and it takes a very long time to get a fix, or never does, then its antenna input may be damaged, this pin is rather sensitive to static damage.

Stuart Robinson

GW7HPW

June 2016