Using the All_Configuration File

This document describes how to use the configuration file for the TTN_GPS_TRACKER program file. The program file is for the Arduino integrated development environment (IDE). The configuration file has been created so that all configuration for the program and any changes needed are carried out by editing this file. When you load the program file in the Arduino IDE the 'All_Configuration.h' file should appear on the second tab from the left in the IDE. You should not need to make changes to the program file itself.

The configuration file uses a series of '#defines' and constant definitions that the compiler recognises and uses when building (compiling) the working program. In general the defines are either 'commented in' or 'commented out', an example will explain, for instance look at these lines from the '1) Hardware related definitions and options' section of the settings file;

```
#define Board_Definition "HAB3_Board_Definitions.h"
//#define Board_Definition "LCD_Receiver_Board_Definitions.h"
//#define Board_Definition "Pro_Mini_Mikrobus_Shield_Board_Definitions.h"
//#define Board_Definition "Custom_Board_Definitions.h"
```

The first **#define** ***Board_Definition*** is 'commented in' and within the program itself there is a bit of code that says include the hardware definitions file HAB3_Board_Definitions.h when the program is complied. This hardware definitions file contains the required pin definitions for the HAB3 board in use.

If we want to compile the program for a different board, say the LCD Receiver board, we would comment out the HAB3_Board_Definitions.h file by adding two // characters at the start of its line and then removing the two // characters at the start of the LCD_Receiver_Board_Definitions.h line. The lines would then look like this;

```
//#define Board_Definition "HAB3_Board_Definitions.h"
#define Board_Definition "LCD_Receiver_Board_Definitions.h"
//#define Board_Definition "Pro_Mini_Mikrobus_Shield_Board_Definitions.h"
//#define Board_Definition "Custom_Board_Definitions.h"
```

You can use the TTN_GPS_TRACKER program for other Arduino hardware setups, but you will need to edit the Custom_Board_Definitions.h file to match your pin connections and then #include it.

Some parameters are set with constant definitions. For example from GPS section of the All_Configuration file.

```
static const  uint16_t  GPSBaud = 9600;           //GPS baud rate
```

This constant sets the baud rate for the GPS you are using. Each section of the configuration file is now described in turn.

1) Hardware related definitions and options

There is a definition file for each of the LoRaTracker board types, these definition files specify which pins on the board perform a particular function for instance most all definition files have a line like this;

#define LED1 8

This means the board LED is on pin 8. Thus when the program turns on the LED such as with this line;

digital.Write(LED1, HIGH);

The program knows to set pin 8 high, which turns on the LED.

Using pin definitions in this way makes the programs easier to understand and adapt for different hardware setups..

The TTN_GPS_TRACKER_CayenneLPP program has 4 pre-made definition files for the LoRaTracker boards;

LCD_Receiver_Board_Definitions.h
HAB3_Board_Definitions.h
Pro_Mini_Mikrobus_Shield_Board_Definitions.h
Custom_Board_Definitions.h

The Custom_Board_Definitions.h file allows you to create the pin allocations for a non LoRaTracker board.

Within section 1) to define which board definition file to use, remove the two // characters in front of the # define line for the board you are using. Only ‘comment in’ the define for one of the hardware definition files.

Section 1) of the All_Configuration.h file also contains these two lines;

#define LoRa_Device_in_MB1
#define GPS_in_MB2

Some LoRaTracker boards, such as the LCD_Receiver_Board use Mikrobus modules for the LoRa device and GPS, The two #define lines above allocate the appropriate pin definitions depending on which Mikrobus socket the LoRa module and GPS is in. The LoRa device is normally in MB1 and the GPS in MB2. The HAB3 board does not use Mikrobus sockets.

2) Program Options - Keys for LoRaWAN etc.

This is the section where you enter the keys for the TTN connection, NWKSKEY the network session key,   APPSKEY the application key and DEVADDR the device address.

When entering the keys be sure that the format remains the same, all the 0x00 sections are replaced with the values of your keys and are in the same format.

This section also contains the nominal transmission (into the TTN) interval. The actual interval will be about 5 seconds longer.

5) GPS Options

Define the GPS baud rate here, often it will be 9600 baud, but not always.

static const uint32_t GPSBaud = 9600;

For test purposes the program can use a defined test GPS location rather than the real location obtained from the GPS, this can be useful for testing purposes and if you don't want to reveal your true location for testing purposes.

To use the test location uncomment this define;

//#define Use_Test_Location

And change the test location if you choose to by editing these lines.

//Pen-y-Fan summit, Brecon Beacons
#define TestLatitude 51.88406
#define TestLongitude -3.43644
#define TestAltitude 886

7) Display Settings

The TTN_GPS_TRACKER has an option to attach a I2C display, either a SSD1306 or a 20x4 LCD based using the HD44780 controller and fitted with a PCF8574 I2C driver board.

To use either display option enable this define by un-commenting it;

//#define Use_Display

Then enable the define for one of the appropriate display libraries;

#define Display_Library "Display_I2C_LCD2.h"   //20x4 LCD display with PCF8574 backpack

or

#define Display_Library "Display_SD1306_AVR.h"   //use an SSD1306 I2C display

How the GPS information from the tracker program is displayed on screen is controlled by defining the appropriate display screens library. There are different screen libraries as the two displays used have different numbers of lines and characters.

#define Display_Screens "I2C_LCD_20x4_Screens.h"   //use the 20x4 LCD Screens

or

#define Display_Screens "SD1306_SMALL_TEXT_Screens.h"

The two I2C attached displays need an I2C address defined, they can vary from standard, the addresses are defined here;

const int PCF8574Address = 0x27;   //I2C address of the PCF8574 for 20x4 LCD

or

const int I2C_ADDRESS = 0x3C;   //I2C address of the SD13206

If your unsure of the I2C address of the screen you have attached, run the 'I2C_Scanner_Test' program that is in the \Programs directory on the repository downloaded from GITHUB.

Some temporary screens are used to alert the user to some condition, the period these screens are displayed in mS is defined here;

const unsigned int screen_delaymS = 2000;

Good Luck

Stuart Robinson
www.LoRaTracker.uk
October 2018