A Simple Mobile/Portable WSPR Beacon Controller

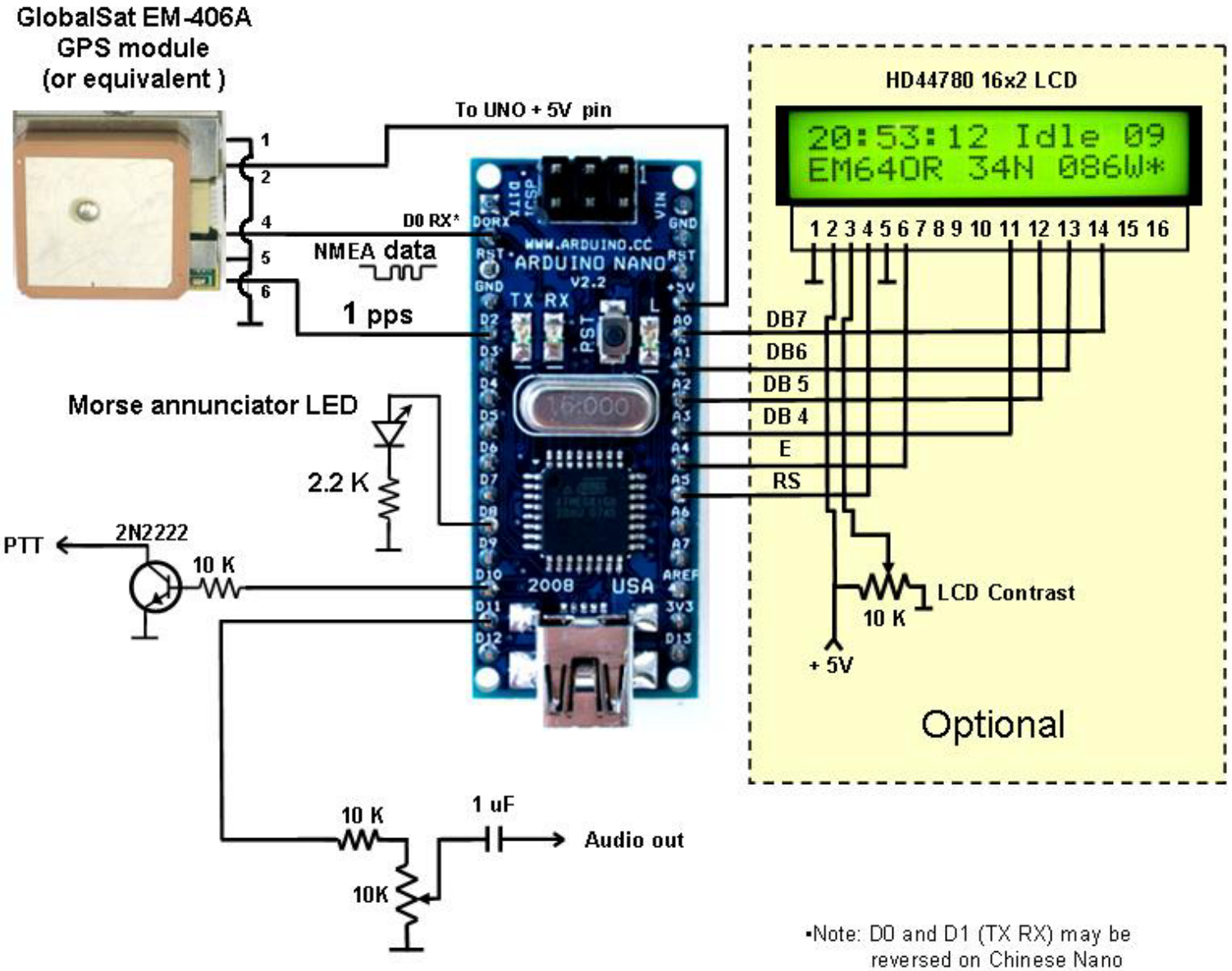
This project provides a simple means of generating an audio WSPR signal using either an Arduino Nano or Arduino Uno to drive a SSB transmitter for mobile or portable WSPR beaconing.

Gene Marcus W3PM GM4YRE
21 May, 2013

Features

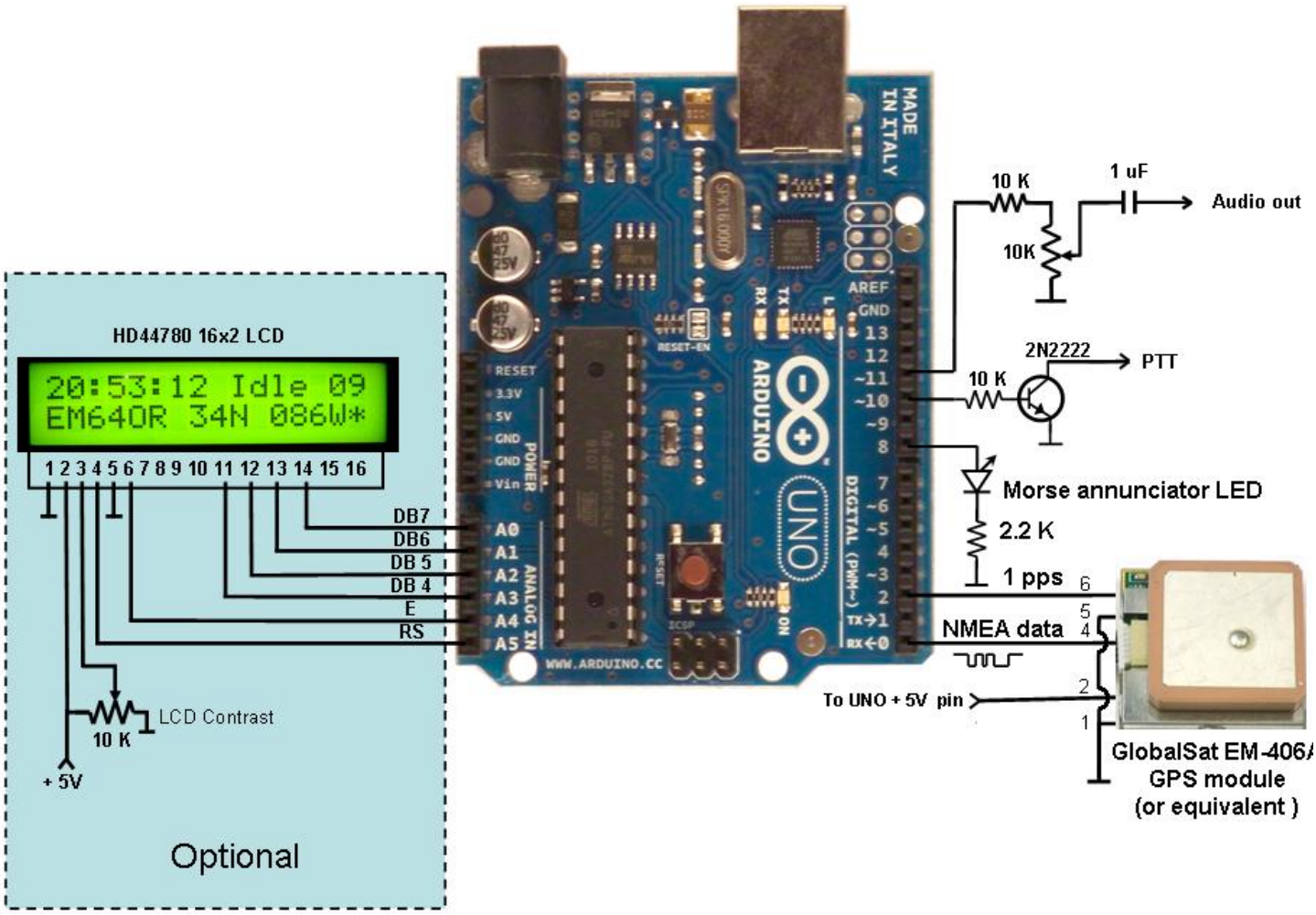
- NMEA GPS timing for UTC synchronization of WSPR transmissions.
- Generation of compound callsigns with 6 digit locator WSPR message or conventional callsigns with 4 digit locator.
- Automatic GPS derived 6 digit grid locator for mobile or portable operation.
- Automatic switching between home and mobile/portable compound callsigns.
- Morse encoded LED annunciator for UTC time, 6 digit grid locator, and satellites in view.
- Optional 16x2 LCD display of UTC time, 6 digit grid locator, latitude, longitude, transmit mode, GPS status, and satellites in view.

Figure 1. Arduino Nano Audio WSPR Schematic



•Note: D0 and D1 (TX RX) may be reversed on Chinese Nano boards.

Figure 2. Arduino Uno Audio WSPR Schematic



Introduction

The objective of this project was to create a small, simple, stand-alone, battery operated WSPR audio generator using common modules and minimal hardware. Other criteria included world wide operation and automatic selection of home or mobile/portable compound callsign. The objectives were met through the use of direct digital synthesis of audio sine waves using an Arduino board. Direct digital synthesis provided an audio signal using very little external hardware. A small GPS receiver with a built in antenna allowed world wide coverage for transmit timing and on-the-fly generation of on-the-fly generation of the 6 digit maidenhead grid location.

Acknowledgements

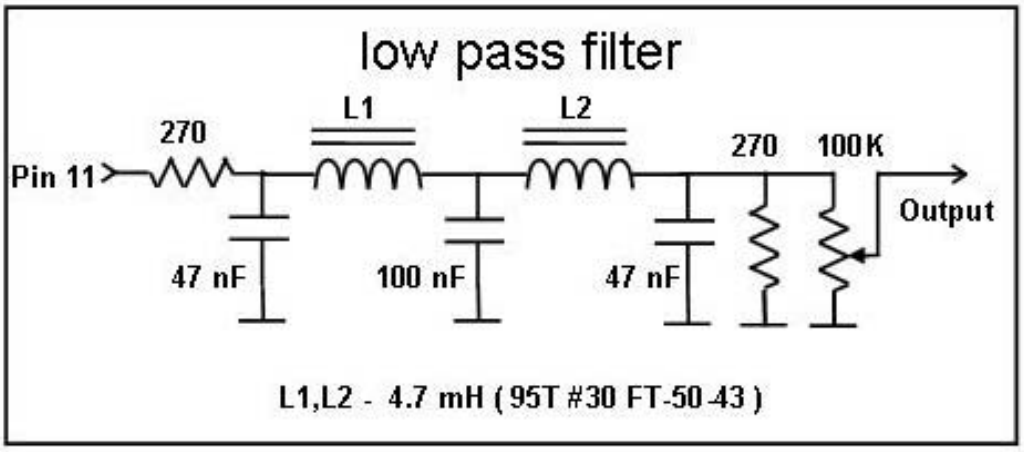
The direct digital synthesis method of sine wave generation was found in a paper authored by Martin Nawrath, DH3JO. The compound callsign/6 digit locator algorithms were derived from FORTRAN and C++ programs found in K1JT's WSPR source code. Portions of the WSPR message algorithm were influenced by the work of Andy Talbot, G4JNT. Portions of the GPS receive code were derived from Igor Gonzalez Martin's Arduino tutorial.

Audio Low Pass Filter?

One of the limitations of digitally synthesizing an audio signal using pulse width modulation is that a 32KHz sampling frequency is present at the audio output on pin 11 of the Arduino. For additional information, refer to: http://interface.khm.de/index.php/lab/experiments/arduino-dds-sinewave-generator/.

The first controller prototype tests included an audio low pass filter (Figure 3) connected between Arduino pin 11 and the data input jack of a Yeasu FT-817 transceiver.

Figure 3.



Both bench and on-the-air tests confirmed that the controller worked well.

Upon examining the schematic diagram of the FT-817, I noticed that an active low pass filter was part of the transceiver's audio chain. To determine if the internal audio filtering was adequate, the low pass filter was removed and a second series of tests were conducted. Identical test results were obtained as shown in the spectrum analyzer images below. The images show close-in and harmonic response at 14 MHz (10 dB/vertical division).

Figure 4. Controller with low pass filter

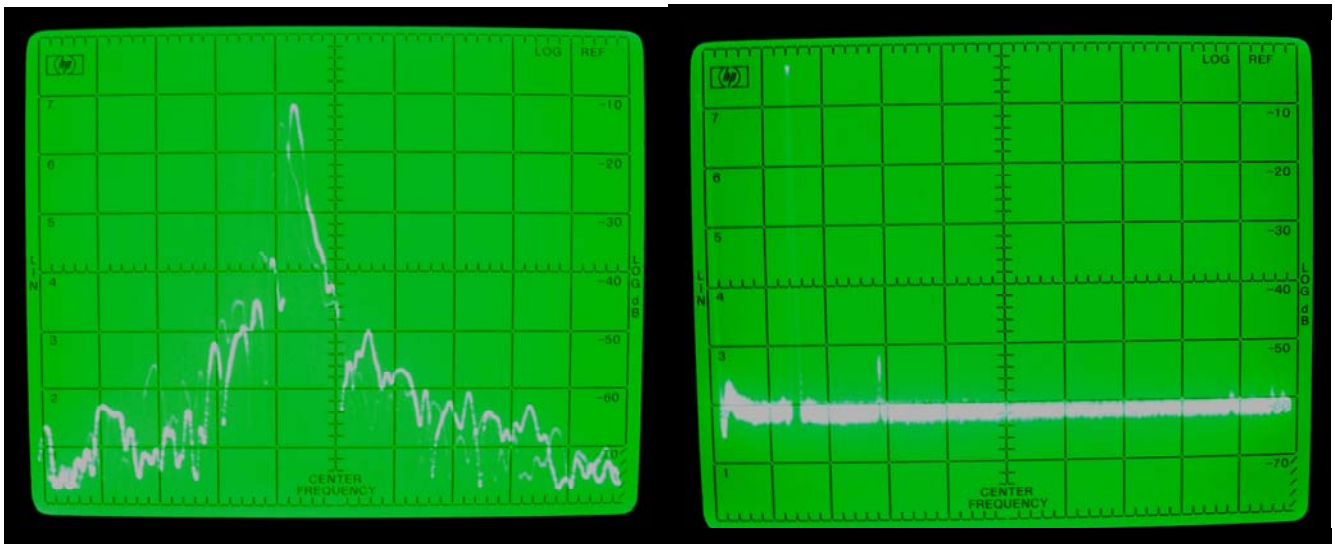
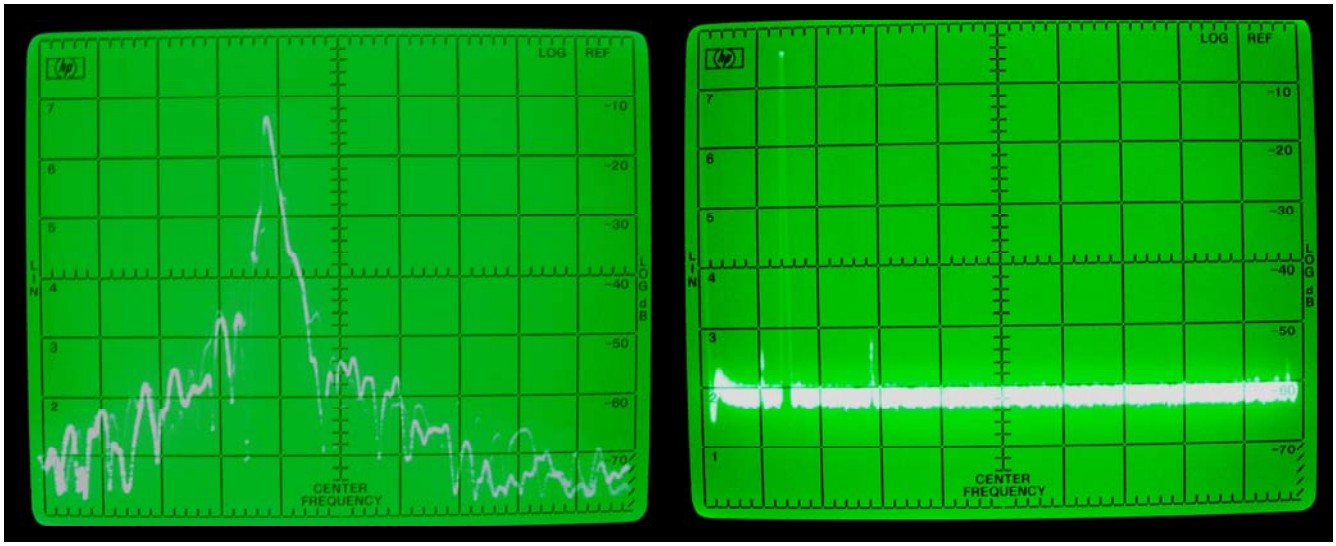


Figure 5. Controller without low pass filter



In conclusion, the low pass filter is not required for the FT-817 and is probably not needed when the controller is used with most modern transceivers. The builder is advised to use his or her own judgment when determining if an audio low pass filter is required for their transmitter.

Initial Software Setup

The Arduino Nano and Arduino Uno use the same serial port for GPS data input and programming. The GPS serial data line must be disconnected or turned off before the program sketch is uploaded

Open “WSPR_audio_v1.ino” into the Arduino development environment and customize the script to personalize your station information. The station information is found near the beginning of the script. Change the data highlighted in red. Do not change any other data. Be sure to remember to save your changes and upload the file into the Uno or Nano.

```
//_____________________________Enter home callsign and grid square below:_____________________
char call3[13] = "W3PM"; //e.g. "W3PM" or "GM4YRE"
char locator2[7] = "EM64";  // Use 4 character locator e.g. "EM64"

//_____________________________Enter portable/mobile callsign below:_____________________________
char call4[13] = "W3PM/M"; //e.g. "W3PM/M" or "W4/GM4YRE"
```

Note:
- Upper or lower case characters are acceptable.
- Compound callsigns may use up to a three letter/number combination prefix followed by a “/”.
  A one letter or two number suffix may be used preceded by a “/”.

```
//_____________________________Enter power level below:_____________________________________
byte ndbm = 27; // Min = 0 dBm, Max = 43 dBm, steps 0,3,7,10,13,17,20,23,27,30,33,37,40,43

//_____________________________Enter audio output frequency in Hz below:_____________________
int audioFreq = 1520;  // Use any audio frequency between 1400 and 1600 Hz

//_____________________________Enter even minute to begin transmission:_____________________________
byte txTime = 0;  // Select even minute to transmit. Enter either 0,2,4,6, or 8.
.
```

Operation

Set the 10K output potentiometer to minimum before connecting the controller to the SSB transmitter's data input or microphone jack. Upon connecting the controller to the transmitter, select the desired band and WSPR USB transmit frequency. The audio frequency is added to the transmitters display frequency, so use the "Dial" frequency displayed in K1JT's WSPR program.

After the controller is first turned on, there will be a delay of up to a few minutes before the GPS receiver obtains a valid fix. The controller will no go into transmit until a valid fix is obtained. The optional LCD display will display a "*" in the lower right corner upon obtaining a valid fix. The Morse LED annunciator will blink once every few seconds until a fix is obtained. When a valid fix is obtained, the LCD display will indicate UTC time, 6 digit grid square, latitude, longitude, satellites in view, transmit status, and GPS status. The Morse annunciator will send at a slow rate the UTC time (hhmm), satellites in view, and 6 digit grid locator. A typical Morse output is " 2147 s7 em64or". The satellite number is limited to one digit for brevity. A satellite number of 0, 1, or 2 corresponds 10, 11, or 12 satellites in view

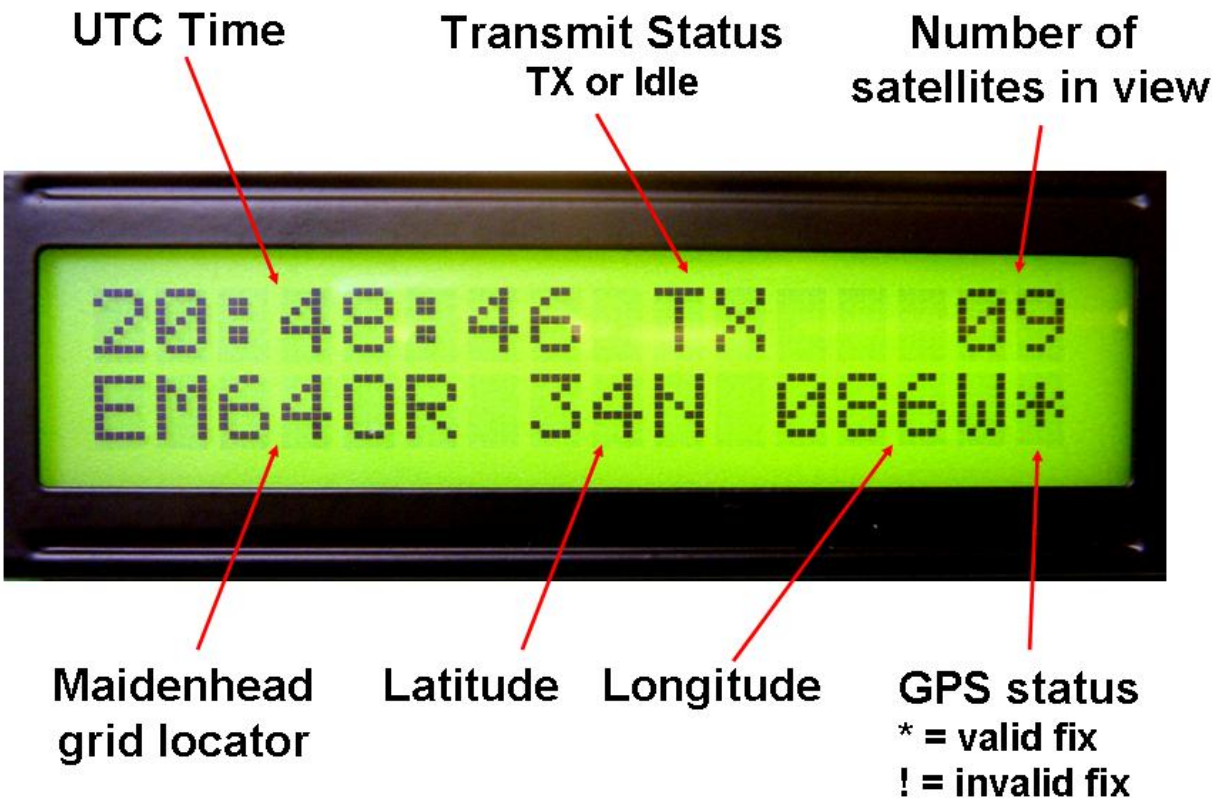
The controller will go into transmit at the selected even minute time slot. At this time the 10K potentiometer is used to adjust the transmitters ALC per manufacturer's specification. The Morse annunciator LED will stay lit when during transmit times and the LCD display will display "TX".

When the controller detects that it is outside the 4 digit home grid square, it will default to the mobile/portable callsign and use GPS coordinates for WSPR message reporting. If a compound callsign is used, the unit will default to the 6 digit location/compound callsign WSPR protocol that requires two transmissions. The second transmission will immediately follow the first.

Note to Yeasu FT-817 owners:

The Ft-817 provides a 5 VDC output at the microphone jack and a 13.8 VDC output at the accessory jack. Do NOT be tempted to use any of these ports to power this controller. The FT-817 output voltages are supplied through 1/16 watt 10 ohm chip resistors that serve as fuses. The controller draws approximately 60 mA when using the Arduino Nano and GlobalSat-406A GPS receiver and slightly more with the LCD display.

Figure 6. Optional LCD Display



I may be contacted at w3pm at amsat dot org for any questions or comments.