WWVB dpskr gps

09/13/2015

Introduction

Goal to eliminate the new WWVB BPSK signal so that old phase tracking receivers may work.

This system uses predictive message coding to remove the WWVB BPSK signal so that old style phase tracking receivers work again. An additional benefit are the old time clock receivers also work. It also offers the possibility of new approaches to using the new signal.

The system essentially is very simple.

A neo M6 GPS receiver supplies a NEMA only time and date string. Also a precise 1 PPS.

An arduino takes the serial data and converts the time and date to minutes since 2000 according to the NIST specifications. An Arduino NANO was chose to test with for its 32 bit math, wide availability and low cost. It works on all Arduinos actually. This then drives an inverter and finally a H driver that drives a double balanced modulator. The DBM acts as a phase inverter for the analog signal.

The DBM is inserted between the antenna preamp and receiver or receivers. Ther is a 3db loss going through the modulator but all of the tested receivers do not care.

The solution has been tested on the following receivers.

Dymec 5478

2 X HP VLF 117

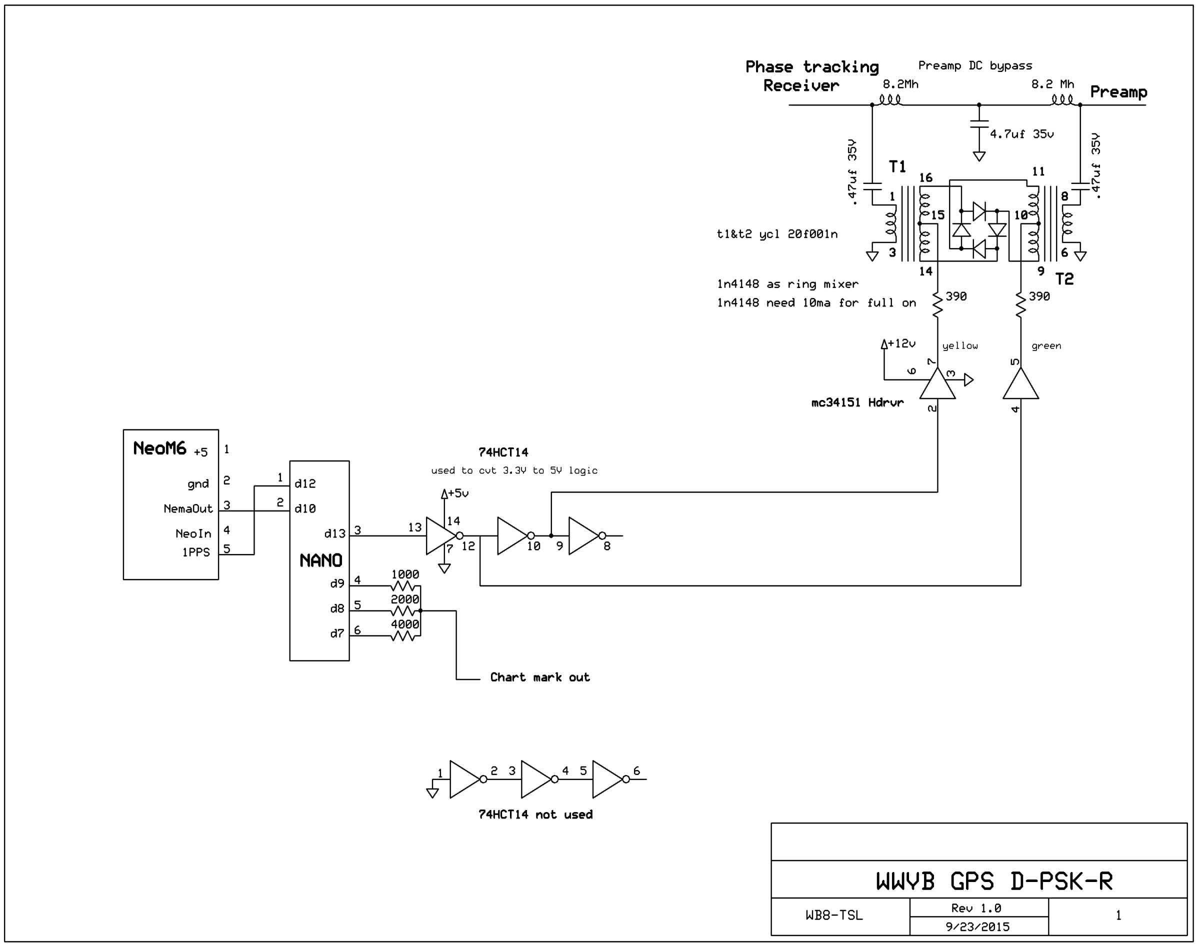
Fluke 207

TrueTime DC 60

Symetricom 8170 and Netclock

It just plain works no magic and uses all simple and readily available components that you can actually solder.

Schematic of the system



The NEO M6 must be programmed to 38.4 KB and the ublox sentenances turned off so that the time sentences are available early in the second. Also the 1 pps is brought out for use. The same signal that drives the green LED picture later.

The 1pps and NEMA only are sent to the Arduino NANO that then makes up the new WWVB 60 second sentence.

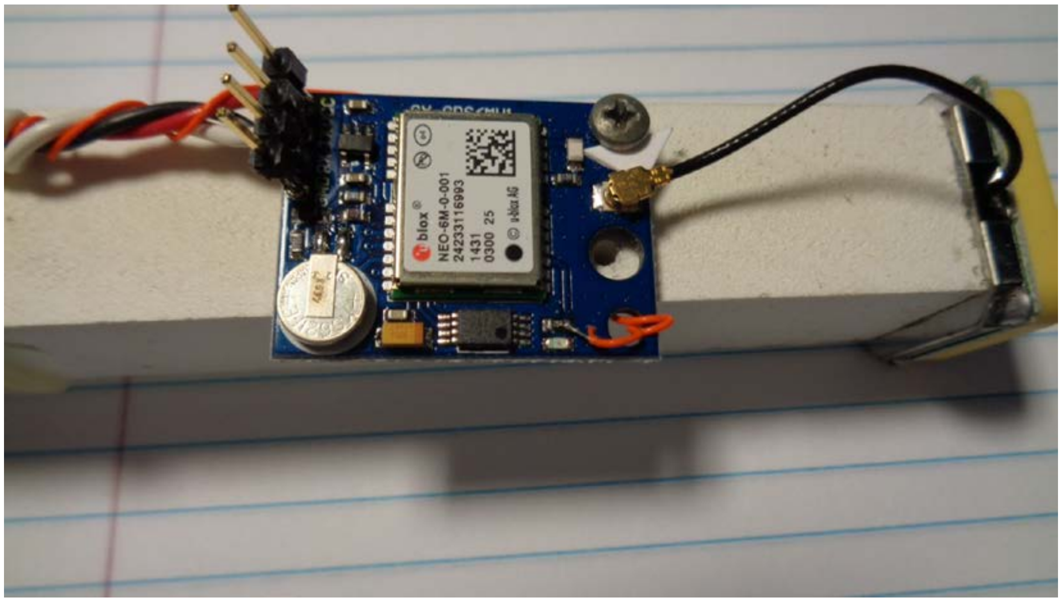
Output of the sentence is voltage level shifted in the 74HCT14 and an inverted version of the signal generated.

These two signals feed a H bridge driver a MC34151 that drives the double balanced mixer in or out of phase with a 10 MA current through 2 X 390 ohm resistors for isolation. The double balanced mixer uses a commonly available dip Ethernet filter available from the Chineese sites 10 for a few dollars. Only 1 is needed as it has 2 transformers.

The pins are precise so that there is no phase shift by the mechanics of the connections. Not that this actually matters even if the phases are wrong they remain always wrong and the receivers are fine with this. The double balanced mixer is home brew as today 60 Khz DBMs simply do not exist at a reasonable cost. There are other ways to approach this process of phase flipping. This is the simplest that can be inserted in the antenna line.

Pictures of the system

The NEO M6 GPS receiver with 1 pps addition. The orange wire.



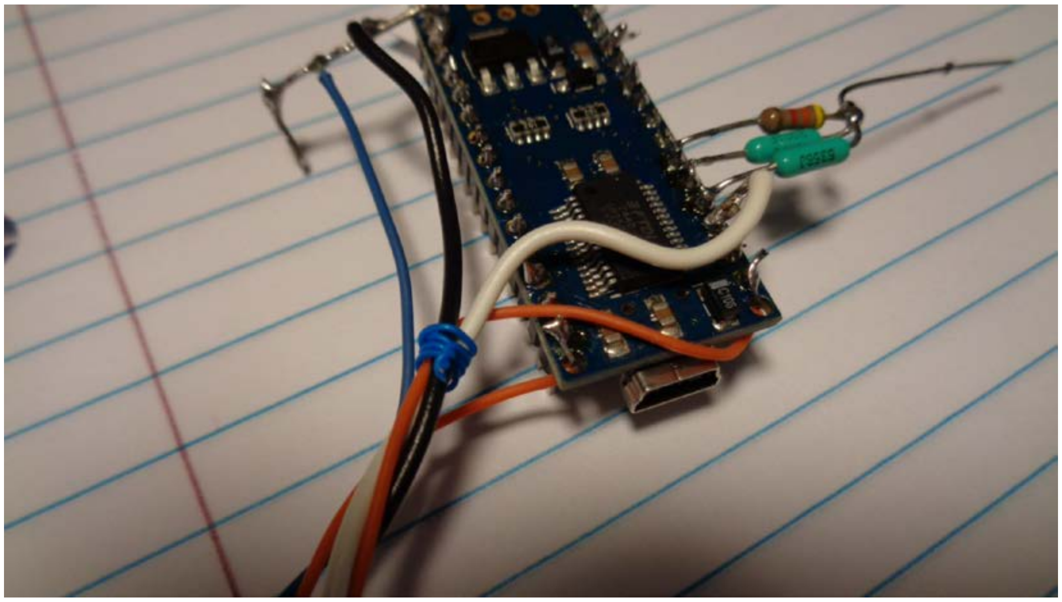
The 4 pins were used to reprogram the NEO to 38,400 Kb/s and only NEMA sentences.

This assembly happens to live on a stick and base so that it can standup anyplace on its own.

The arduino with the GPS d-psk-r code. This requires a 1 pps pulse positive going at the tick and NEMA code at 38,400 Kb/s.

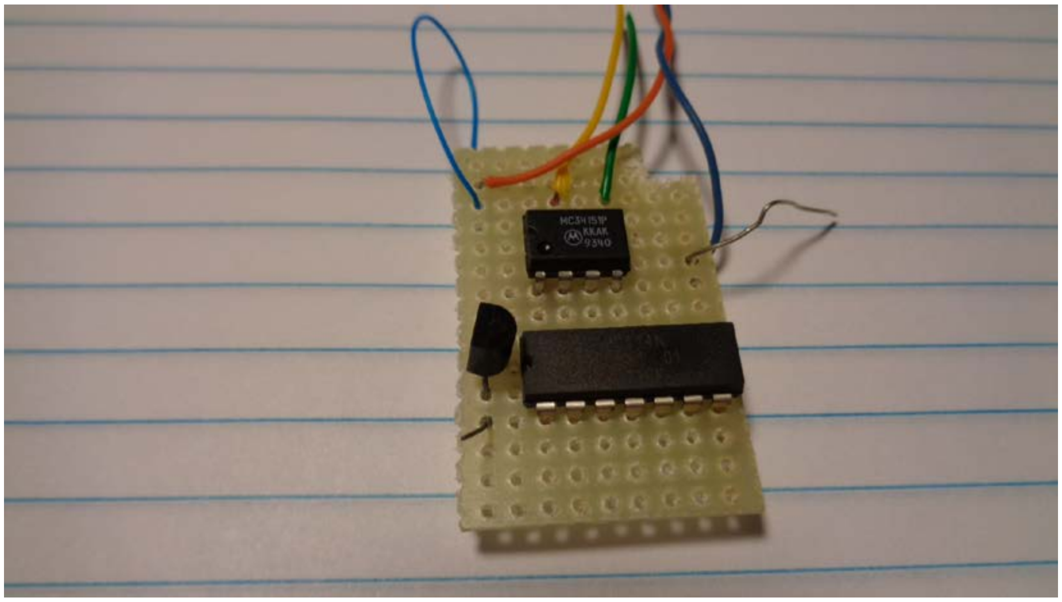
It spends much of its time waiting.

On the left blue and black are ground. It gets power through the usb connector. The 3 resistor chart dack is on the right with D10 being the NEMA code in at 38,400Kb/s. The orange wire next is the 1 pps and the orange wire on the left is the flip control to the H driver.



Below is the DBM driver it uses a 74hct14 to adapt 3.3V logic to 5 V logic then drives a mc34151 mos driver chip. There are many chips out there. But the MC H driver is a very nice 8 pin dip.

The small transistor like TO92 is actually a LM7805 5V regulator.



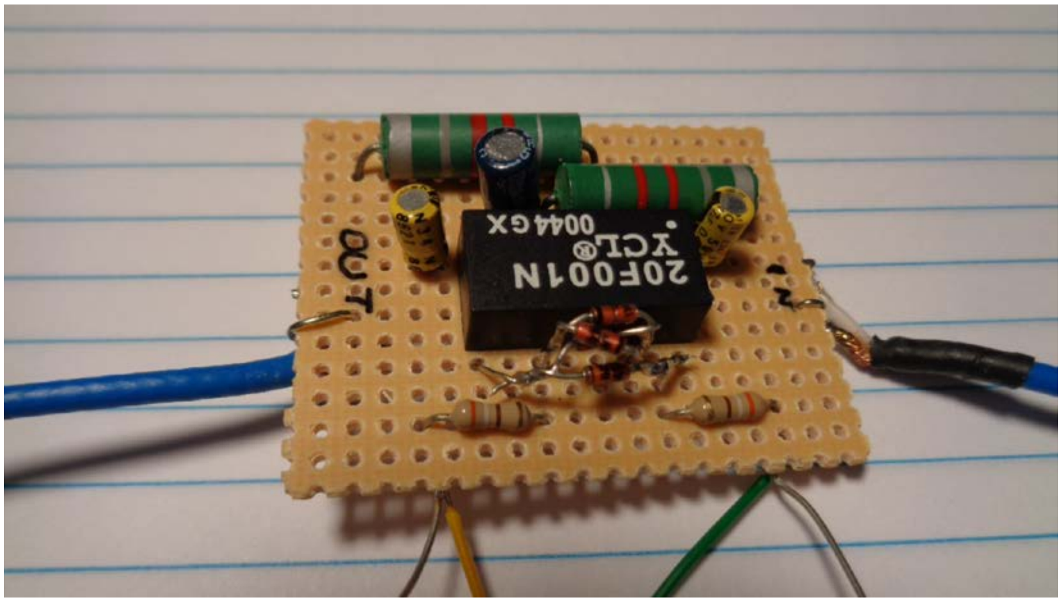
Next the modulator/Double Balanced Mixer/ bit flipper.This is made of two transformers in 1 dip package and a ring of 1n4148 diodes. The 1n4148 diodes need 10 ma of current so a 12V supply is used and 390 ohm resistors are in each modulator control leg to isolate the RF from the switching.

The modulator is completely floating. So as an example it could live in a small metal box 20 feet from the controller. Some capacitors like a .47 uf might be added to the control lines for noise suppression in that case.

The modulators purpose is to invert or not the incoming signal to remove BPSK

Logic 0 = 000 degrees

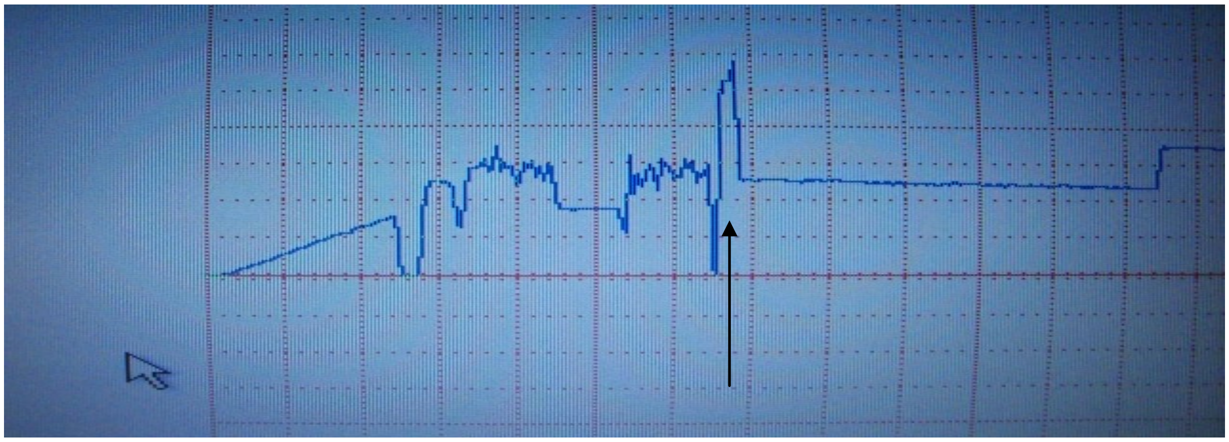
Logic 1 = 180 degrees



Actual first start

The initial startup. Much of the jumping due to issues with the 207 that had not been on in 3 years. Essentially since WWVB changed format.

I had also changed the local reference and everything settled down when I added a terminator to the 100 KHz that drives the fluke 207. Without the terminator severe ringing upset the fluke.



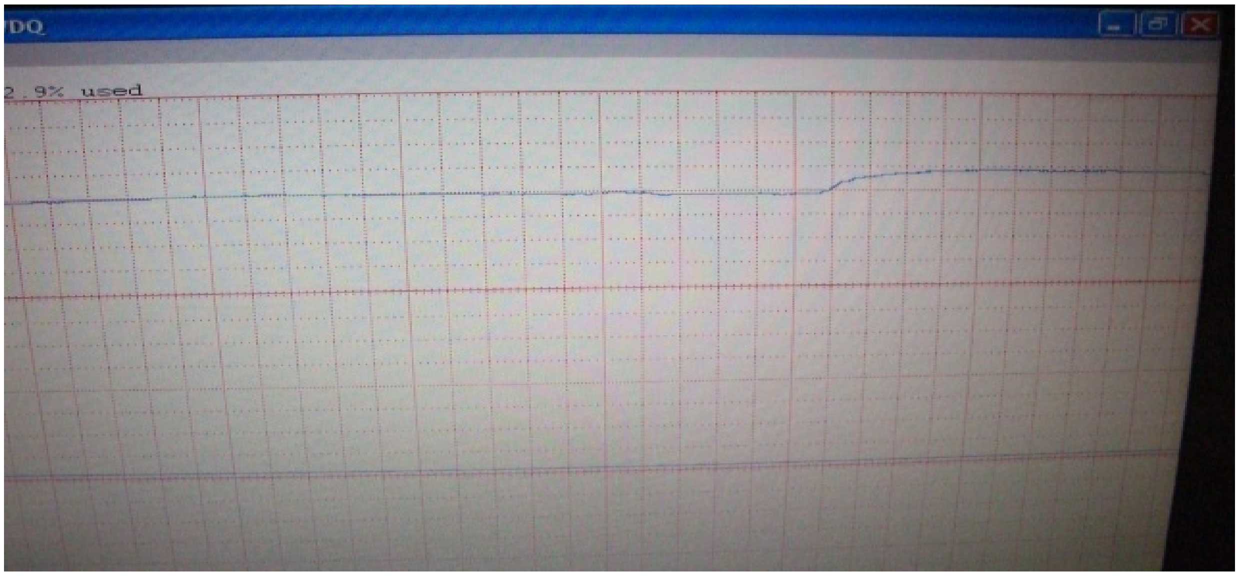
Left side of chart startup. First gap tinkering with chart program

First arrowstraightened out the 100 Khz ref termination on the Fluke 207

There is a third phase jump not shown due to I believe the over driven signal. There is no AGC from the Dymec receiver that's used as the current frontend.

The overall drift down is the local reference to WWVB. The local reference is a Jackson Labs LTE reference.

8 hour period with diurnal shift and everything stable. Chart program is WinDAQ $29 with usb 4 channel A>D module.



Program

Below is the Arduino program.

This is the original.

// Released to the public domain no warranty nor support implied.

// Uses GPS parser and converts months days hours to minutes.

// This begins the process of converting Date and time to minutes

// Now correctly converts years and months to minutes.

// Has normal and leap offsets. Month and year aggregates. Leap and yr agg logic.

```
// 0910 parity generated par0-4 using bitread. All bits now exist

// XXX Do not advance 1 minute. Xmission is over the staed minute Advance 1 minute added

// Get GPS runs at 38400 and only NEMA drops from 832 to 220ms.

// 09122015 start of actual timing wait to second 00 run loop then bit send 1 per second xx This actually
works

// look for 1 pps to be used to start the bit march Working

// Sync marches out

// AC works correctly several days testing.

// AD adds a D>A converter for chart marking of hours MSB D9 D8 D7 LSB

// D9 = 00 hour D8 =1200 D7 toggle by hour lsb = hour


#include <SoftwareSerial.h>

#include <TinyGPS.h>


// Create an instance of the TinyGPS object

TinyGPS gps;

// Initialize the NewSoftSerial library to the pins you defined above

SoftwareSerial uart_gps(10, 11);


// This is where you declare prototypes for the functions that will be

// using the TinyGPS library.

void getgps(TinyGPS &gps);


// In the setup function, you need to initialize two serial ports; the

// standard hardware serial port (Serial()) to communicate with your

// terminal program an another serial port (NewSoftSerial()) for your
// GPS.
void setup()
{
  // setup display and gps communications
  Serial.begin(115200);
  //Sets baud rate of your GPS
  uart_gps.begin(38400);
  Serial.println("");
  pinMode(13, OUTPUT); pinMode(9, OUTPUT); pinMode(8, OUTPUT); pinMode(7, OUTPUT);
  pinMode(12, INPUT);

}
// global variables
unsigned long int yrtotal;
unsigned long int yrtotala;
int dstnext = 0x011011;          // fixed DST next to start
int dstls = 0x00011;             // DST and leap second
int delaytic = 107;              // gps tick to phase shift 100ms after ick delay 7ms to east coast
                                 // calculate your delay from wwvb to nearest ms rance 100-110ms for US



// Next section agrregate year accumulation for 10 years leap years etc
// years aggregate next data

unsigned long yearagg[]= {
```

```
0x786360, 0x806880, 0x887340, 0x907860, 0x987D80,

0xA082A0, 0xA88D60, 0xB09280, 0xB897A0, 0xC09CC0,

0xC8A780, 0xD0ACA0, 0xD8B1C0, 0xE0B6E0, 0xE8C1A0,

0xF0C6C0 };


// Leap years 2016 2020 2024 2028 2032


// years offset subtract 2014 from year to get aggregate add this

// 2014 used so that 2015 equals 1 offset in table


// Month non leap year aggregation

unsigned long monthagg[]= {

0x00000, 0x00000, 0x0ae60, 0x14be0, 0x1fa40,

0x2a300, 0x35160, 0x3fa20, 0x4a880, 0x556e0,

0x5ffa0, 0x6ae00, 0x756c0 };


// Month leap year aggregation

unsigned long monthleap[]= {

0x00000, 0x00000, 0x0ae60, 0x15180, 0x1ffe0,

0x2a8a0, 0x35700, 0x3ffc0, 0x4ae20, 0x55c80,

0x60540, 0x6b3a0, 0x75c60 };



// This is the main loop of the code. All it does is check for data on

// the RX pin of the ardiuno, makes sure the data is valid NMEA sentences,
// then jumps to the getgps() function.

void loop()

{


  while(uart_gps.available())     // While there is data on the RX pin...

  {

      int c = uart_gps.read();    // load the data into a variable...

      if(gps.encode(c))           // if there is a new valid sentence...

        {

          getgps(gps);            // then grab the data.

        }

  }

}


// The getgps function will get and print the values.

void getgps(TinyGPS &gps)

{


  unsigned int hrmin;

  unsigned int daymin;


  int year;

  byte month, day, hour, minute, second;

  gps.crack_datetime(&year,&month,&day,&hour,&minute,&second);

  //Print data and time
  Serial.print(month, DEC); Serial.print(day, DEC); Serial.print(year);

  Serial.print(" ");

  Serial.print(hour, DEC); Serial.print(minute, DEC); Serial.println(second, DEC);

  //Serial.print(" ");

  digitalWrite(13, LOW);            // set phase to 0


  if (second == 0) {                // this waits for the top of the minute Everything drives from time 00


  // Insert chart D2A Purpose to mark a second channel on a stripchart software recorder with time
pulses

  // Levels 0 and 1 hour, 12 hour higher, and 0000 highest voltage


  digitalWrite(9, LOW); digitalWrite(8, LOW); digitalWrite(7, LOW);

   if (hour == 0){

      digitalWrite(9, HIGH);

      }

   else if (hour == 12){

      digitalWrite(8, HIGH);

      }

   else {

      digitalWrite(8, bitRead(hour,0));

       }


 // end of chart D2A


  // Convert year month day hour to minutes begin
  unsigned int yearag = year - 2015;  // Get aggregated minutes to correct for any error buildup

  // hrmin = (hour*60) + minute;

  daymin = ((day * 1440)-1440) + (hour*60) + minute; //Have to cleanup day which is day 0 so subtract
1440

 // must add +1 minute to yrtotal for correct encode out as its the time it will be

 // Leap year logic guess we will know if it works in 2016

  if (year == 2016 || year == 2020 || year == 2024 || year == 2028 || year == 2032)

   {

     yrtotal = yearagg[yearag] + monthleap[month] + daymin;  //yearag is the offset in the year table

   }

  else

   {

     yrtotal = yearagg[yearag] + monthagg[month] + daymin;

   }

// Convert year month day hour to minutes end

//Gather 19th bit and calculate FEC parity start

  int bit19 = bitRead(yrtotal,0); // gets bit 19/second 19

  int par0;
```

```
    par0=bitRead(yrtotal,0) + bitRead(yrtotal,2) + bitRead(yrtotal,4) + bitRead(yrtotal,5) +
bitRead(yrtotal,6)

      + bitRead(yrtotal,8) + bitRead(yrtotal,9) + bitRead(yrtotal,13) + bitRead(yrtotal,14) +
bitRead(yrtotal,15)

      + bitRead(yrtotal,16) + bitRead(yrtotal,17) + bitRead(yrtotal,20) + bitRead(yrtotal,21) +
bitRead(yrtotal,23);

    par0=bitRead(par0,0);


  int par1;

    par1=bitRead(yrtotal,1) + bitRead(yrtotal,3) + bitRead(yrtotal,5) + bitRead(yrtotal,6) +
bitRead(yrtotal,7)

      + bitRead(yrtotal,9) + bitRead(yrtotal,10) + bitRead(yrtotal,14) + bitRead(yrtotal,15) +
bitRead(yrtotal,16)

      + bitRead(yrtotal,17) + bitRead(yrtotal,18) + bitRead(yrtotal,21) + bitRead(yrtotal,22) +
bitRead(yrtotal,24);

    par1=bitRead(par1,0);


  int par2;

    par2=bitRead(yrtotal,2) + bitRead(yrtotal,4) + bitRead(yrtotal,6)+ bitRead(yrtotal,7) +
bitRead(yrtotal,8)

      + bitRead(yrtotal,10)+ bitRead(yrtotal,11) + bitRead(yrtotal,15) + bitRead(yrtotal,16) +
bitRead(yrtotal,17)

      + bitRead(yrtotal,18)+ bitRead(yrtotal,19) + bitRead(yrtotal,22) + bitRead(yrtotal,23) +
bitRead(yrtotal,25);

    par2=bitRead(par2,0);


  int par3;

    par3=bitRead(yrtotal,0) + bitRead(yrtotal,2) + bitRead(yrtotal,3) + bitRead(yrtotal,4) +
bitRead(yrtotal,6)
```

```
      + bitRead(yrtotal,7) + bitRead(yrtotal,11) + bitRead(yrtotal,12) + bitRead(yrtotal,13) +
bitRead(yrtotal,14)

      + bitRead(yrtotal,15) + bitRead(yrtotal,18) + bitRead(yrtotal,19) + bitRead(yrtotal,21) +
bitRead(yrtotal,24);

    par3=bitRead(par3,0);


  int par4;

   par4=bitRead(yrtotal,1) + bitRead(yrtotal,3) + bitRead(yrtotal,4) + bitRead(yrtotal,5) +
bitRead(yrtotal,7)

      + bitRead(yrtotal,8) + bitRead(yrtotal,12) + bitRead(yrtotal,13) + bitRead(yrtotal,14) +
bitRead(yrtotal,15)

      + bitRead(yrtotal,16) + bitRead(yrtotal,19) + bitRead(yrtotal,20) + bitRead(yrtotal,22) +
bitRead(yrtotal,25);

    par4=bitRead(par4,0);


 //Parity calculation end


 digitalWrite(13, LOW);

 Serial.print(yrtotal, HEX), Serial.print (" "), Serial.print(yrtotal, DEC); Serial.print (" ");

 Serial.print(par4, DEC); Serial.print(par3, DEC); Serial.print(par2, DEC); Serial.print(par1, DEC);
Serial.println(par0, DEC);

 delay (100);


// begin 60 second bit transmission toggles d13 on arduino to BPSK modulator. "bit flipper"


 // march syncword    syncword = 0x0001011011100

 while(digitalRead(12) == 0) { }  // first sync 0 is 00 data proc

 delay (delaytic);
```

```
digitalWrite(13, LOW);          // bit1

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, HIGH);          // bit2

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, HIGH);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, HIGH);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, LOW);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, HIGH);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, HIGH);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, LOW);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, HIGH);
```

```
while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, LOW);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, LOW);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, LOW);


// end sync


// Parity 4-0


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, par4);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, par3);

while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, par2);

while(digitalRead(12) == 0) { }

delay (delaytic);
```

```
 digitalWrite(13, par1);

 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13, par0);


// end parity


// start time in minutes


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13,(bitRead( yrtotal, 25)));  // first bit 25 out


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13, bit19);                  //Marker at 19


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13,(bitRead( yrtotal, 24)));


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13,(bitRead( yrtotal, 23)));
```

```
while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 22)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 21)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 20)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 19)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 18)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 17)));


while(digitalRead(12) == 0) { }
```

```
delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 16)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13, LOW);                // Marker at 29


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 15)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 14)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 13)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 12)));


while(digitalRead(12) == 0) { }

delay (delaytic);
```

```
digitalWrite(13,(bitRead( yrtotal, 11)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 10)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 9)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 8)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 7)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,LOW);                  // Marker at 39


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 6)));
```

```
while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 5)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 4)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 3)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 2)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 1)));


while(digitalRead(12) == 0) { }

delay (delaytic);

digitalWrite(13,(bitRead( yrtotal, 0)));
```

```
// end of time

// DST_LS 0x00011                Though correct and may stay correct is a question

 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, LOW);

 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, LOW);

 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, LOW);          // Marker 49 notice

 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, LOW);

 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, HIGH);
 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13, HIGH);


// End of DST_LS


// DST Next dstnext = 0x011011      remains the same till government changes


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13, LOW);


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13, HIGH);


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13, HIGH);


 while(digitalRead(12) == 0) { }

 delay (delaytic);

 digitalWrite(13, LOW);


 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, HIGH);


 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, HIGH);


 while(digitalRead(12) == 0) { }
 delay (delaytic);
 digitalWrite(13, LOW);         // Marker at 59 always low


delay (300);                    // insures system waits till next second


  }
}
```