```
CON
  _CLKMODE      = XTAL1 + PLL16X
  _XINFREQ      = 5_000_000

VAR
' µOLED Variables
  long SAddr
  long MAddr
  word Type
  word HW
  word SW
  word DevX
  word DevY
  word SD_Demo
  byte uSD_Sector[512]

  long  R, G, B
' GPS Variables
  long  gps_Stack[32]
  long  savechar

  long  onesixtyth
  long  knots2mph

  long  current_lat
  long  current_lon

  long  accumulator
  long  fp_minutes
  long  fp_degrees

  long  fp_knots
  long  fp_mph
  long  fp_fph
  long  fp_heading

  long  lat1r                                           'current latitude in radians
  long  lon1r                                           'current longitude in radians

  long  lat2r                                           'destination latitude in radians
  long  lon2r                                           'destination longitude in radians

{
        distance = 2*asin(sqrt((sin((lat1-lat2)/2))^2 + cos(lat1)*cos(lat2)*(sin((lon1-lon2)/2))^2))
}

  long  halfsindlatsqr          'half sin difference latitudes squared; (sin((lat1-lat2)/2))^2 section of
the distance formula.


  long  halfsindlonsqr          'half sin difference longitudes squared; (sin((lon1-lon2)/2))^2 section of
the distance formula.

  long  drad                    'distance in radians between the current and destination positions;  2*asin(
sqrt( halfsindlatsqr + cos(lat1)*cos(lat2)*(halfsindlonsqr)) section of the distance formula.

  long  dkm                     'distance in Km between the current and destination positions

  long  dmiles                  'distance in miles between the current and destination positions
  long  dfeet                   'distance in feet between the current and destination positions

  long  bearing                 'true course in radians to destination
  long  bearing_degrees         'true course in degrees to destination

  long  destination_lat
  long  destination_lon

  long  delta_lat, delta_lon

  long  cogs_Free
  long  targetd

  long keyin, keyin0, keyin1, keyin2, keyin3, msb, lsb


OBJ
  OLED          : "uOLED-128-GMD1"
```

```
  DELAY         : "Clock"

  GFDS          : "FullDuplexSerial"
  f32           : "Float32Full"
  f32string     : "FloatString"

  kb            : "FullDuplexSerial"

PUB start |  baud, tx, rx, i


  f32string.SetPrecision(7)
  f32string.SetPositiveChr("+")

  OLED.INIT

  DELAY.PauseSec(1)

  OLED.INIT_uSD

  f32.start



  baud := 9600
  tx := rx := 1

  kb.start(1,1,%0101,9600)    '%0100

  baud := 4800
  tx := rx := 0
  GFDS.start(rx, tx, %1100, baud)

  cognew(gps_raw, @gps_Stack)

  onesixtyth := f32.fdiv(1.0, 60.0)
  knots2mph := 1.150779

  SETUP

  destination_lat := 32.47171
  destination_lon := -97.28619

  R := 255
  G := 255
  B := 255

  repeat
    if savechar == FALSE

      OLED.ERASE

      if nmeabuffer[14] == "A"
        R := 255
        G := 255
        B := 255
      else
        R := 0
        G := 0
        B := 127

      f32string.SetPrecision(7)
      f32string.SetPositiveChr("+")

      oled.wire
      oled.line (68,0,127,0, R, G, B)                   'top
      oled.line (0,68,0,68+52, R, G, B)                 'left
      oled.line (127,74,127,0, R, G, B)                 'right
      oled.line (10,68,10,68+52, R, G, B)               '2nd left
      oled.line (0,68+52,68,68+52, R, G, B)             'under lon
      oled.line (68,68+52,68,0, R, G, B)                'right of lat & lon
      oled.line (0,68+26,68,68+26, R, G, B)             'under lat
      oled.line (69,14,127,14, R, G, B)                 'under heading
      oled.line (69,28,127,28, R, G, B)                 'under bearing
      oled.line (69,52,127,52, R, G, B)                  'under distance
      oled.line (69,74,127,74, R, G, B)                 'under speed
      oled.line (0,68,68,68, R, G, B)                   'under circle
```

```
      oled.UCHAR ("L", 3, 70, R, G, B, 1, 1)
      oled.UCHAR ("A", 3, 78, R, G, B, 1, 1)
      oled.UCHAR ("T", 3, 86, R, G, B, 1, 1)

      oled.UCHAR ("L", 3, 68+28, R, G, B, 1, 1)
      oled.UCHAR ("O", 3, 68+36, R, G, B, 1, 1)
      oled.UCHAR ("N", 3, 68+44, R, G, B, 1, 1)


      Position_text2float

      great_circle

      heading_speed_text2float



      oled.UTEXT (13,68+2, 0, R,G,B, 1, 1, f32string.floattostring(current_lat),0)
      oled.UTEXT (13,68+10, 0, R,G,B, 1, 1, f32string.floattostring(destination_lat),0)

      oled.UTEXT (13,68+28, 0, R,G,B, 1, 1, f32string.floattostring(current_lon),0)
      oled.UTEXT (13,68+36, 0, R,G,B, 1, 1, f32string.floattostring(destination_lon),0)

      delta_lat := f32.fsub(current_lat, destination_lat)
      delta_lon := f32.fsub(current_lon, destination_lon)

      if f32.fcmp(f32.fabs(delta_lat),0.001) == TRUE
        f32string.SetPrecision(2)
      else
        f32string.SetPrecision(5)
      oled.UTEXT (13,68+18, 0, R,G,B, 1, 1, f32string.floattostring(delta_lat),0)

      if f32.fcmp(f32.fabs(delta_lon),0.001) == TRUE
        f32string.SetPrecision(2)
      else
        f32string.SetPrecision(5)
      oled.UTEXT (13,68+44, 0, R,G,B, 1, 1, f32string.floattostring(delta_lon),0)

      oled.uchar("H", 71, 4, R, G, B, 1, 1)
      oled.uchar("B", 71, 18, R, G, B, 1, 1)

      f32string.SetPositiveChr(0)
      f32string.SetPrecision(4)
      oled.UTEXT (79,2, 2, R,G,B, 1, 1, f32string.floattostring(fp_heading),0)

      if f32.fcmp(bearing_degrees, 0.01) == TRUE
        oled.UTEXT (79,16, 2, R,G,B, 1, 1, f32string.floattostring(0.0),0)
      else
        oled.UTEXT (79,16, 2, R,G,B, 1, 1, f32string.floattostring(bearing_degrees),0)

      if f32.fcmp(dmiles, 1.0) ==  TRUE
        oled.UTEXT (73, 30, 0, R,G,B, 1, 1, string("FEET"),0)
        dfeet := f32.fmul(dmiles, 5280.0)
        f32string.SetPrecision(5)
        oled.UTEXT (74,40, 2, R,G,B, 1, 1, f32string.floattostring(dfeet),0)
        targetd := dfeet
      else
        oled.UTEXT (73, 30, 0, R,G,B, 1, 1, string("MILES"),0)
        f32string.SetPrecision(5)
        oled.UTEXT (74,40, 2, R,G,B, 1, 1, f32string.floattostring(dmiles),0)
        targetd := dmiles

      if f32.fcmp(fp_mph, 3.0) == TRUE
        oled.UTEXT (73, 54, 0, R,G,B, 1, 1, string("Ft/Sec"),0)
        fp_fph := f32.fmul(fp_mph, 1.466667)
        f32string.SetPrecision(5)
        oled.UTEXT (74,62, 2, R,G,B, 1, 1, f32string.floattostring(fp_fph),0)
      else
        oled.UTEXT (73, 54, 0, R,G,B, 1, 1, string("Miles/Hr"),0)
        f32string.SetPrecision(5)
        oled.UTEXT (74,62, 2, R,G,B, 1, 1, f32string.floattostring(fp_mph),0)

      cogs_free := f32.ffloat(free_cogs)

      oled.UTEXT (74, 77, 0, R,G,B, 1, 1, f32string.floattostring(cogs_free),0)
      oled.UTEXT (74, 77, 0, R,G,B, 1, 1, string("  Free"),0)
```

```
      oled.circle(34,34,34, R,G,B)
      oled.PUT_PIXEL (34,34,R,G,B)

      headingfp( fp_heading)
      anglesfp( bearing_degrees)

      north

      keyin := kb.rxcheck
      msb:=lsb := keyin
      oled.uchar(lookupz((msb>>4) & $F : "0".."9", "A"..."F"), 71, 86, R, G, B, 1, 1)
      oled.uchar(lookupz((lsb) & $F : "0".."9", "A"..."F"), 77, 86, R, G, B, 1, 1)

      keyin0 := kb.rxcheck
      if keyin0 <> -1

        keyin1 := kb.rxcheck
        keyin2 := kb.rxcheck
        keyin3 := kb.rxcheck

        msb:=lsb := keyin0
        oled.uchar(lookupz((msb>>4) & $F : "0".."9", "A"..."F"), 71, 95, R, G, B, 1, 1)
        oled.uchar(lookupz((lsb) & $F : "0".."9", "A"..."F"), 77, 95, R, G, B, 1, 1)

        msb:=lsb := keyin1
        oled.uchar(lookupz((msb>>4) & $F : "0".."9", "A"..."F"), 85, 95, R, G, B, 1, 1)
        oled.uchar(lookupz((lsb) & $F : "0".."9", "A"..."F"), 91, 95, R, G, B, 1, 1)

        msb:=lsb := keyin2
        oled.uchar(lookupz((msb>>4) & $F : "0".."9", "A"..."F"), 99, 95, R, G, B, 1, 1)
        oled.uchar(lookupz((lsb) & $F : "0".."9", "A"..."F"), 105, 95, R, G, B, 1, 1)

        msb:=lsb := keyin3
        oled.uchar(lookupz((msb>>4) & $F : "0".."9", "A"..."F"), 113,95, R, G, B, 1, 1)
        oled.uchar(lookupz((lsb) & $F : "0".."9", "A"..."F"), 119, 95, R, G, B, 1, 1)

      waitcnt(25_000_000 + cnt)


PUB heading_speed_text2float


      accumulator :=  (nmeabuffer[47] - 48) * 1_000
      accumulator := accumulator + ((nmeabuffer[48] - 48) * 100)
      accumulator := accumulator + ((nmeabuffer[49] - 48) * 10)
      accumulator := accumulator + ((nmeabuffer[51] - 48))

      fp_heading := f32.fdiv(f32.ffloat(accumulator), 10.0)

      accumulator :=  (nmeabuffer[41] - 48) * 1_000
      accumulator := accumulator + ((nmeabuffer[42] - 48) * 100)
      accumulator := accumulator + ((nmeabuffer[43] - 48) * 10)
      accumulator := accumulator + ((nmeabuffer[45] - 48))

      fp_knots := f32.fdiv(f32.ffloat(accumulator), 10.0)
      fp_mph := f32.fmul(fp_knots, knots2mph)


PUB position_text2float

      accumulator := ((nmeabuffer[16] - 48) * 10)                                                                    '
BEGIN Text to current latitude fp value
      accumulator := accumulator + (nmeabuffer[17] - 48)

      fp_degrees := f32.ffloat(accumulator)

      accumulator :=  (nmeabuffer[18] - 48) * 100_000
      accumulator := accumulator + ((nmeabuffer[19] - 48) * 10_000)
      accumulator := accumulator + ((nmeabuffer[21] - 48) * 1_000)
      accumulator := accumulator + ((nmeabuffer[22] - 48) * 100)
      accumulator := accumulator + ((nmeabuffer[23] - 48) * 10)
      accumulator := accumulator + (nmeabuffer[24] - 48)

      fp_minutes := f32.fdiv(f32.ffloat(accumulator),10_000.0)

      if nmeabuffer[26] == "N"
```

```
        current_lat := f32.fadd(fp_degrees, (f32.fmul(fp_minutes,onesixtyth)))
      else
        current_lat := f32.fneg(f32.fadd(fp_degrees, (f32.fmul(fp_minutes,onesixtyth))))                  '
END Text to current latitude fp value


      accumulator := ((nmeabuffer[28] - 48) * 100)                                                       '
BEGIN Text to current longitude fp value
      accumulator := accumulator + ((nmeabuffer[29] - 48) * 10)
      accumulator := accumulator + (nmeabuffer[30] - 48)

      fp_degrees := f32.ffloat(accumulator)

      accumulator :=  (nmeabuffer[31] - 48) * 100_000
      accumulator := accumulator + ((nmeabuffer[32] - 48) * 10_000)
      accumulator := accumulator + ((nmeabuffer[34] - 48) * 1_000)
      accumulator := accumulator + ((nmeabuffer[35] - 48) * 100)
      accumulator := accumulator + ((nmeabuffer[36] - 48) * 10)
      accumulator := accumulator + (nmeabuffer[37] - 48)

      fp_minutes := f32.fdiv(f32.ffloat(accumulator),10_000.0)

      if nmeabuffer[26] == "E"
        current_lon := f32.fadd(fp_degrees, (f32.fmul(fp_minutes,onesixtyth)))
      else
        current_lon := f32.fneg(f32.fadd(fp_degrees, (f32.fmul(fp_minutes,onesixtyth))))                  'END
Text to current longitude fp value


PUB north | radius, cx, cy, tdegrees, trad, xfp, yfp, xi, yi, treal, tc
  radius := 30.0
  cx := 34.0
  cy := 34.0

  tc := f32.fsub(270.0, fp_heading)
  trad := f32.radians(tc)
  xfp :=    f32.fadd(cx, f32.fmul(radius, f32.cos(trad)))
  yfp :=    f32.fadd(cy, f32.fmul(radius, f32.sin(trad)))

  xi := f32.fround(xfp)
  yi := f32.fround(yfp)
  oled.uchar("N", xi, yi, 255, 0, 0, 1, 1)
  oled.line (34,34,xi,yi, R, G, B)

  tc := f32.fsub(360.0, fp_heading)
  trad := f32.radians(tc)
  xfp :=    f32.fadd(cx, f32.fmul(radius, f32.cos(trad)))
  yfp :=    f32.fadd(cy, f32.fmul(radius, f32.sin(trad)))

  xi := f32.fround(xfp)
  yi := f32.fround(yfp)
  oled.uchar("E", xi, yi, 255, 0, 0, 1, 1)
  oled.line (34,34,xi,yi, R, G, B)

  tc := f32.fsub(180.0, fp_heading)
  trad := f32.radians(tc)
  xfp :=    f32.fadd(cx, f32.fmul(radius, f32.cos(trad)))
  yfp :=    f32.fadd(cy, f32.fmul(radius, f32.sin(trad)))

  xi := f32.fround(xfp)
  yi := f32.fround(yfp)
  oled.uchar("W", xi, yi, 255, 0, 0, 1, 1)
  oled.line (34,34,xi,yi, R, G, B)

  tc := f32.fsub(90.0, fp_heading)
  trad := f32.radians(tc)
  xfp :=    f32.fadd(cx, f32.fmul(radius, f32.cos(trad)))
  yfp :=    f32.fadd(cy, f32.fmul(radius, f32.sin(trad)))

  xi := f32.fround(xfp)
  yi := f32.fround(yfp)
  oled.uchar("S", xi, yi, 255, 0, 0, 1, 1)
  oled.line (34,34,xi,yi, R, G, B)
```

```
PUB headingfp(t) | radius, cx, cy, tdegrees, trad, xfp, yfp, xi, yi, treal, tc

  radius := 34.0
  cx := 34.0
  cy := 34.0

  treal := f32.fadd(270.0, t)

  tc := f32.fsub(treal, t)



  trad := f32.radians(tc)
  xfp :=    f32.fadd(cx, f32.fmul(radius, f32.cos(trad)))
  yfp :=    f32.fadd(cy, f32.fmul(radius, f32.sin(trad)))

  xi := f32.fround(xfp)
  yi := f32.fround(yfp)

  oled.line (34, 34, xi, yi, 255, 255, 0)

PUB anglesfp(t) | radius, cx, cy, tdegrees, trad, xfp, yfp, xi, yi, tc

  if f32.fcmp(targetd, 33.0) == TRUE
    radius := targetd
  else
    radius := 34.0

  cx := 34.0
  cy := 34.0

  tc := f32.fsub(f32.fadd(270.0, t), fp_heading)



  trad := f32.radians(tc)
  xfp :=    f32.fadd(cx, f32.fmul(radius, f32.cos(trad)))
  yfp :=    f32.fadd(cy, f32.fmul(radius, f32.sin(trad)))

  xi := f32.fround(xfp)
  yi := f32.fround(yfp)

  oled.PUT_PIXEL (xi, yi ,0,255,0)
  oled.PUT_PIXEL (xi+1, yi+1 ,0,255,0)
  oled.PUT_PIXEL (xi-1, yi-1 ,0,255,0)

  oled.PUT_PIXEL (xi+1, yi-1 ,0,255,0)
  oled.PUT_PIXEL (xi-1, yi+1 ,0,255,0)

PUB SETUP
  OLED.ERASE
  DELAY.PauseMSec(20)
  OLED.BACKGROUND(0,0,0)

PUB CLEAR_SECTORS | Temp
  'Clears a range of sectors on the uSD card
  REPEAT Temp from 0 to 511                               'Clear uSD buffer
    uSD_Sector[Temp] := $00

  OLED.OPAQUE
  OLED.FTEXT(4,2,2, 255,255,255, string("Clearing "), 0)
  OLED.FTEXT(4,3,2, 255,255,255, string(" Sector  "), 0)

  REPEAT Temp from 0 to 255
    SAddr := Temp
    OLED.WRITE_SECTOR(@SAddr, @uSD_Sector)
    OLED.UTEXT(30,55,2, 0,240,0, 3,3, SAddr, 1)

  DELAY.PauseSec(1)
  SETUP



PUB SHUTDOWN | Temp
  OLED.ERASE
```

```
  OLED.BACKGROUND(0,0,0)
  OLED.OPAQUE

  OLED.FTEXT(2,2, 0, 255,255,255, string("Shutdown in:"),0)
  'OLED.FONT_SIZE(2)
  OLED.UCHAR("5", 90,8, 250,250,0, 2,2)

  OLED.FTEXT(3,8, 0, 250,250,0, string("Restart in 5 Sec"),0)
  DELAY.pauseSec(1)
  OLED.UCHAR("4", 90,8, 0,250,0, 2,2)
  DELAY.pauseSec(1)
  OLED.UCHAR("3", 90,8, 250,250,0, 2,2)
  DELAY.pauseSec(1)
  OLED.UCHAR("2", 90,8, 250,250,0, 2,2)
  DELAY.pauseSec(1)
  OLED.UCHAR("1", 90,8, 250,0,0, 2,2)
  DELAY.PauseSec(1)
  OLED.UCHAR("0", 90,8, 250,0,0, 2,2)
  DELAY.PauseSec(1)

  OLED.ERASE
  DELAY.PauseMSec(20)

  OLED.POWER(0)
  DELAY.PauseSec(4)
  OLED.POWER(1)
  DELAY.PauseMSec(500)
  'OLED.RESET
  OLED.AUTO_BAUD
  SETUP

PUB great_circle

  lat1r := f32.radians(current_lat)
  lon1r := f32.radians(current_lon)

  lat2r := f32.radians(destination_lat)
  lon2r := f32.radians(destination_lon)

{
  d = 2*asin(sqrt((sin((lat1-lat2)/2))^2 + cos(lat1)*cos(lat2)*(sin((lon1-lon2)/2))^2))
}

  halfsindlatsqr := f32.fmul(f32.sin(f32.fdiv(f32.fsub(lat1r,lat2r), 2.0)), f32.sin(f32.fdiv(f32.fsub(lat1r,
lat2r), 2.0)))
  halfsindlonsqr := f32.fmul(f32.sin(f32.fdiv(f32.fsub(lon1r,lon2r), 2.0)), f32.sin(f32.fdiv(f32.fsub(lon1r,
lon2r), 2.0)))

  drad :=  f32.fmul(2.0,f32.asin(f32.fsqr(f32.fadd(halfsindlatsqr,f32.fmul( f32.cos(lat1r), f32.fmul(f32.cos(
lat2r),halfsindlonsqr))))))
  dkm :=  f32.fmul(f32.fmul(2.0,f32.asin(f32.fsqr(f32.fadd(halfsindlatsqr,f32.fmul( f32.cos(lat1r), f32.fmul(
f32.cos(lat2r),halfsindlonsqr))))),6371.0)
  dmiles := f32.fdiv(dkm,1.609344 )

{
IF sin(lon2-lon1)<0
   tc1=acos((sin(lat2)-sin(lat1)*cos(d))/(sin(d)*cos(lat1)))
ELSE
   tc1=2*pi-acos((sin(lat2)-sin(lat1)*cos(d))/(sin(d)*cos(lat1)))
ENDIF
}

  if f32.fcmp(0.0,f32.sin(f32.fsub(lon2r,lon1r))) == -1
{
    bearing = acos((sin(lat2)-sin(lat1)*cos(d))/(sin(d)*cos(lat1)))
}
    bearing :=   f32.acos(f32.fdiv((f32.fsub(f32.sin(lat2r),f32.fmul(f32.sin(lat1r),f32.cos(drad)))),(f32.
fmul(f32.sin(drad),f32.cos(lat1r)))))

  else
{
    bearing = 2*pi-acos((sin(lat2)-sin(lat1)*cos(d))/(sin(d)*cos(lat1)))
}
    bearing := f32.fsub(f32.fmul(2.0,pi), f32.acos(f32.fdiv((f32.fsub(f32.sin(lat2r),f32.fmul(f32.sin(lat1r),
f32.cos(drad)))),(f32.fmul(f32.sin(drad),f32.cos(lat1r))))))

  bearing_degrees :=  f32.degrees(bearing)
```

```
PUB gps_raw |  nmeachar, idx

  dira[16]~~
  repeat
    !outa[16]
    nmeachar :=- "x"
    repeat while nmeachar <> "$"
      nmeachar := GFDS.rx

    idx := 0                                                                                '
    savechar := TRUE
    repeat while nmeachar <> ","

      if testgprmc[idx] <> nmeachar
        savechar := FALSE

      if savechar
        nmeabuffer[idx] := nmeachar
        idx++

      nmeachar := GFDS.rx

    repeat while nmeachar <> 13

      if savechar
        nmeabuffer[idx] := nmeachar
        idx++
        nmeabuffer[idx] := 0


      nmeachar := GFDS.rx
PUB free_cogs : free | i, cog[8], jmp_0                   'thanks deSilva

  jmp_0 := %010111_0001_1111_000000000_000000000
  repeat while (cog[free] := cognew(@jmp_0, 0)) => 0
    free++
  if free
    repeat i from 0 to free - 1
      cogstop(cog[ i ])
  return free

DAT

        testgprmc       byte    "$GPRMC,"
        testgpgsa       byte    "$GPGSA,"
        testgpgga       byte    "$GPGSA,"

        nmeabuffer      byte    "0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",0

        keys            byte    $AB,$CD,$EF,$50,0
```

```
  if free
    repeat i from 0 to free - 1
      cogstop(cog[ i ])
  return free

DAT

        testgprmc       byte    "$GPRMC,"
        testgpgsa       byte    "$GPGSA,"
        testgpgga       byte    "$GPGSA,"

        nmeabuffer      byte    "0000000000000000000000000000000000000000000000000000000000000000",0

        keys            byte    $AB,$CD,$EF,$50,0
```