

BASIC Stamp 2p24 GPS Airplane Datalogger

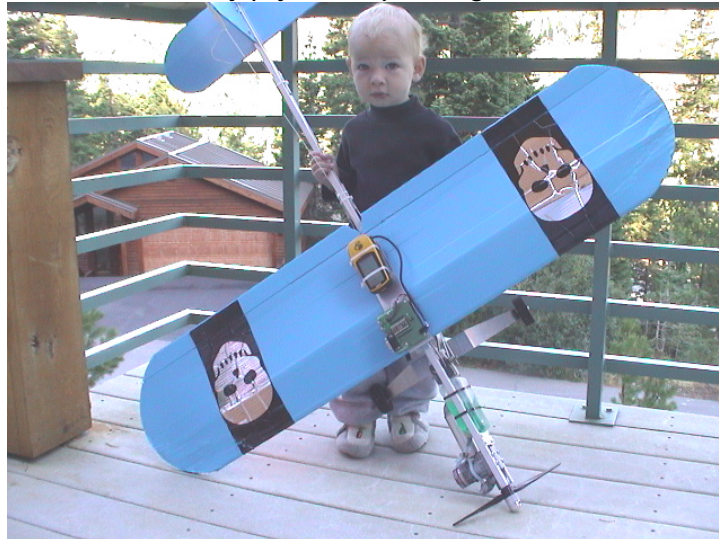
Using the BS2p24's ample RAM and EEPROM program space can store coordinates from an R/C airplane flight for further analysis

by Ken Gracey and Jon Williams

BASIC Stamp users often discuss how to access and log GPS data from a handheld unit. Some of the applications for this include autonomous aircraft, high-altitude balloon flights, trucking logistic mapping and even the parent who wants to record a teenager's evening of travel in the family car. The \$109 Garmin Etrex (www.garmin.com) and a BS2p24 make this project a snap. The Etrex unit is accurate to about 30 feet of the actual location and also provides less accurate.

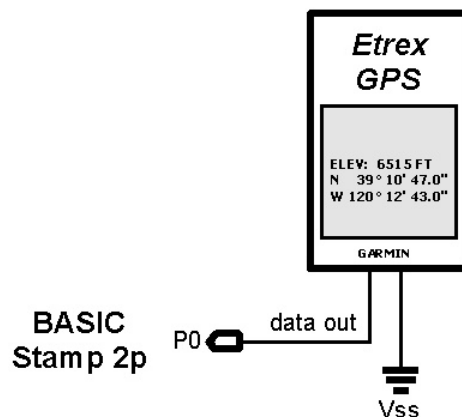
We chose the Airplane Factory's Kombat 40 (www.kombat40.com) with a Tower Hobbies 0.46 cubic inch 2-stroke engine to fly the project. The Kombat 40 is very strong aluminum and corrugated plastic plane that flies quite well and handles our occasional wreck with only minor repairs. We mounted the BS2p24 and Etrex unit on an aluminum frame and bolted it to the top of the wing. This added about 10 ounces but the plane will still fly four more pounds – at sea level. With the engine providing nearly 2 horsepower we didn't even notice the payload.

Carson Gracey (2 years old) holding the Kombat 40.

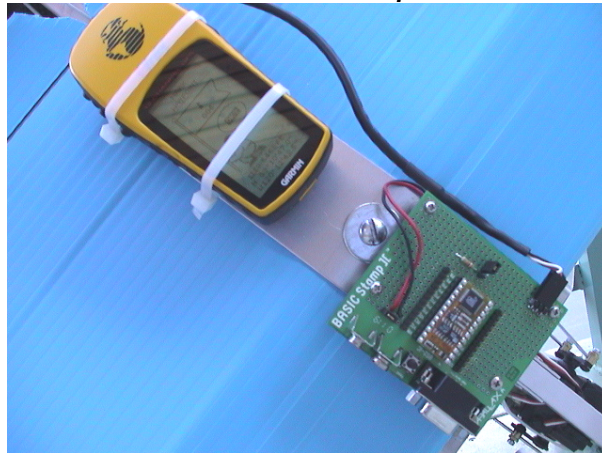


The electronic circuit is simple – one I/O line from the BS2p24 is connected to the Etrex's data out line and the black line is grounded. The Etrex has it's own 3-volt power supply from 2 AA batteries.

GPS circuit with the BS2p24



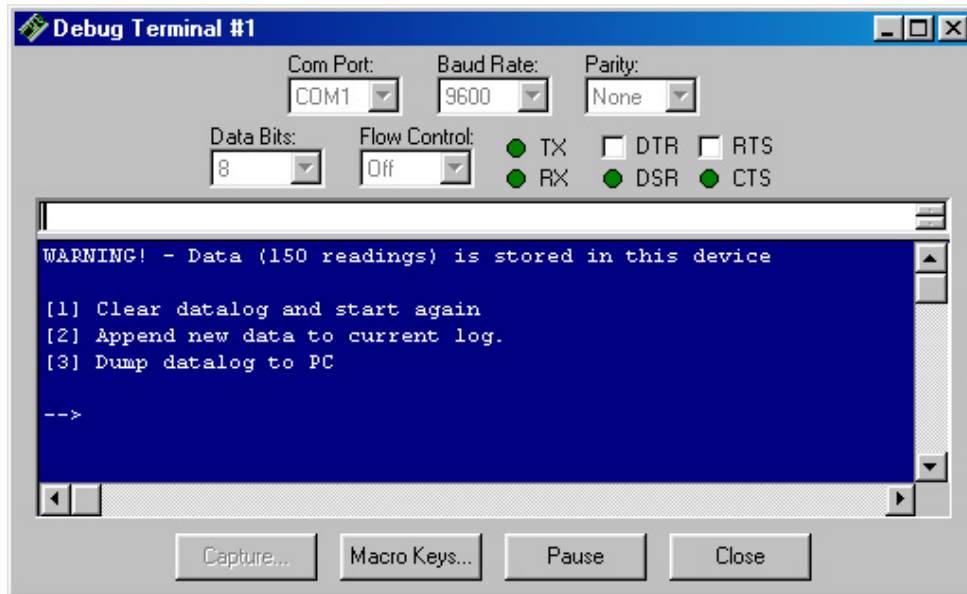
Hardware closeup



The BS2p24 source code `aerogps.bsp` is available for download from www.parallaxinc.com under the "Resources" page.

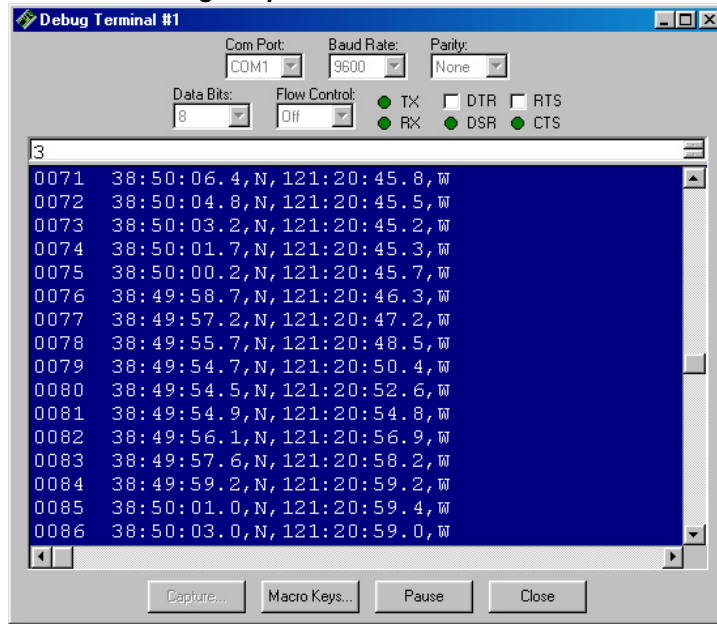
The BASIC Stamp program first launches a user interface with a DEBUG screen prompting you to select your option. If you select [1] the BS2p24 starts to receive 65 bytes of latitude and longitude data into RAM. The `Parse_GPS` routine extracts the latitude and longitude data for storage in EEPROM. This part is a bit tricky as some of the data falls into different positions and the commas must be counted before deciding where the next number begins. The BS2p24 is well suited towards datalogging with its 16K EEPROM and 93 bytes of RAM. Though the EEPROM space is spread across 8 x 2K program banks, the entire space can be accessed from one program bank using the `READ`, `WRITE` and `STORE` commands.

DEBUG Screen User Interface

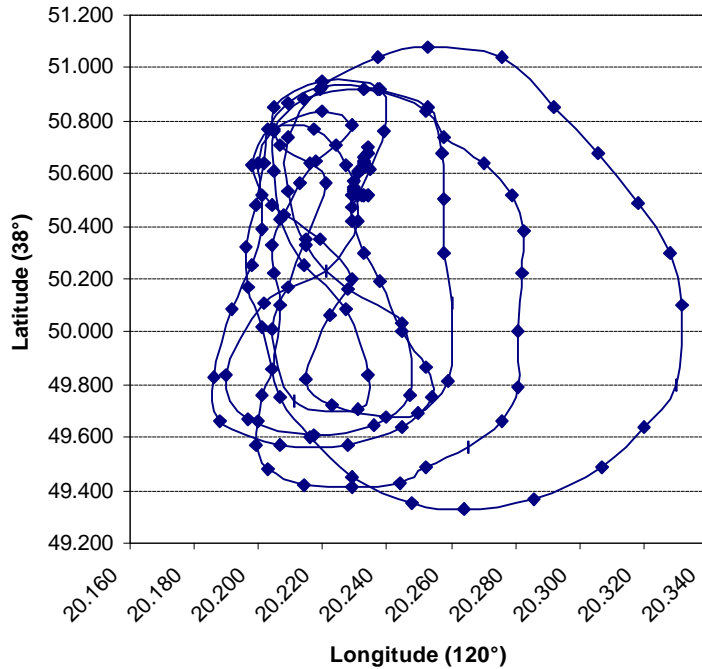


The default source code logs longitude and latitude approximately every two seconds. With seven 2048-byte EEPROM cells and nine bytes per record, the BS2p24 can store 1592 records over a 52-minute period. Of course, you can add a `PAUSE` command to lengthen the data logging period substantially. If you download the data, replace the `DEBUG` command with `SEROUT 16` and run a PC's terminal program with 8-N-1 settings since data cannot be cut and paste from the Stamp's DEBUG window.

Downloading BS2p24 GPS data in the DEBUG screen



**Kombat 40 Flight Path, September 9, 2001
Rocklin, California**



Graphing the data in Excel is easiest if you convert the minutes and seconds to a single decimal number and plot it as a "scatter" diagram. If you want to work with the formatted GPS data, the web has a variety of shareware GPS plotting programs but they require some serious data formatting before importing and graphing.

After making the first plot we noticed that in two sets of 20-30 records the longitude was shifted west a few hundred yards. After searching the PBASIC code for places where we could have dropped a byte, we later concluded that this error is intentionally introduced in civilian GPS units as a government requirement. We fixed the problem by adding 0.250 seconds to these records. The BS2p could also handle this error correction, but some things are easily done with a spreadsheet.

The next step with this project is to add a pitot-tube based airspeed indicator. It's also possible to have the BS2p fly the entire airplane, but unfortunately we can't get paid to do just research work at Parallax.

```
' -----[ Title ]-----
'
' File..... GPSLOG.BSP
' Purpose... GPS Datalogger
' Author.... Jon Williams
' E-mail.... jwilliams@parallaxinc.com
' Started... 01 AUG 2001
' Updated... 12 AUG 2001
'
' {$STAMP BS2p, GPSDATA.BSP}
'
' -----[ Program Description ]-----
'
' This program runs in Slot 0 of a BS2p24. When started, it checks to see
' if it can log and starts immediately if possible. If log data already
' exists, it will prompt the user before overwriting.
'
' -----[ Revision History ]-----
'
' -----[ I/O Definitions ]-----
'
' -----[ Constants ]-----
'
GPSpin CON      0

RxD          CON      16          ' serial input
TxD          CON      16          ' serial output
T9600        CON      240        ' 9600-8-N-1 (matches DEBUG)
N4800        CON      16884
LF           CON      10          ' linefeed

FirstSlot    CON      1
MaxAddr CON    8-FirstSlot*2048-1
FirstAddr    CON      2          ' make room for readings

LogTm        CON      2          ' four seconds between readings

EST          CON      -5          ' Eastern Standard Time
CST          CON      -6          ' Central Standard Time
MST          CON      -7          ' Mountain Standard Time
PST          CON      -8          ' Pacific Standard Time

EDT          CON      -4          ' Eastern Daylight Time
CDT          CON      -5          ' Central Daylight Time
MDT          CON      -6          ' Mountain Daylight Time
PDT          CON      -7          ' Pacific Daylight Time
```

```

UTCfix CON    PDT

Comma          CON    ","

DegSym CON    176          ' degrees symbol
MinSym CON    39          ' minutes symbol
SecSym CON    34          ' seconds symbol

' -----[ Variables ]-----
'
readings      VAR    Word
response      VAR    Byte

idx           VAR    Byte          ' index into GPS data in SPRAM

flags         VAR    Byte
valid        VAR    flags.Bit7

laDeg        VAR    Byte
laMin        VAR    Byte
laSec        VAR    Word          ' tenths of seconds
laSLo        VAR    laSec.LowByte
laSHi        VAR    laSec.HighByte
laNS         VAR    flags.Bit0    ' 0 = N

loDeg        VAR    Byte
loMin        VAR    Byte
loSec        VAR    Word          ' tenths of seconds
loSLo        VAR    loSec.LowByte
loSHi        VAR    loSec.HighByte
loEW         VAR    flags.Bit1    ' 0 = E

eeBase VAR    Word
eeAddr VAR    Word          ' flat EE address
eeData VAR    Byte
slot         VAR    Nib          ' pgm slot for storage
addr         VAR    Word          ' address for storage

temp         VAR    Byte
temp2        VAR    Word
rdngNum VAR    Word

' -----[ EEPROM Data ]-----
'

' -----[ Initialization ]-----
'
Initialize:
  STORE 1
  READ 0,readings.LowByte          ' retrieve readings from EEPROM
  READ 1,readings.HighByte
  IF (readings = 0) THEN Main

Has_Data:
  PAUSE 250

  DEBUG CLS,"WARNING! - Data (",DEC readings," readings) "
  DEBUG "is stored in this device",CR,CR
  DEBUG "[1] Clear datalog and start again",CR
  DEBUG "[2] Append new data to current log.",CR
  DEBUG "[3] Dump datalog to PC",CR,CR,"--> "
  SERIN  RxD,T9600,[ response]
  IF (response < "1") OR (response > "3") THEN Has_Data

  BRANCH (response - "1"),[Clear_Log,Main,Dump_Data]
  GOTO Has_Data

```

```

Clear_Log:
  readings = 0
  STORE FirstSlot
  WRITE 0,0
  WRITE 1,0

' -----[ Main Code ]-----
,
Main:
  eeAddr = readings * 9
  DEBUG CLS, "Logging...",CR,CR

DataLog:
  SERIN GPSpin,N4800,[WAIT("$GPRMC"),SPSTR 65]
  GOSUB Parse_GPS

  ' show current reading

  DEBUG DEC5 (readings+1)," "
  DEBUG DEC2 laDeg,":",DEC2 laMin,": "
  DEBUG DEC2 laSec / 10, ".",DEC1 laSec // 10,Comma
  DEBUG "N" + (laNS * 5),Comma
  DEBUG DEC3 loDeg,":",DEC2 loMin,": "
  DEBUG DEC2 loSec / 10, ".",DEC1 loSec // 10,Comma
  DEBUG "E" + (loEW * 18)
  DEBUG CR

  GOSUB Save_Reading

  ' put delay code here if required

  GOTO DataLog
END

' -----[ Subroutines ]-----
,
' *****
' Extract Latitude and Longitude from GPS string
' *****

Parse_GPS:

Get_Lat:
  GET 10,temp
  laDeg = (temp - "0") * 10
  GET 11,temp
  laDeg = (temp - "0") + laDeg

  GET 12,temp
  laMin = (temp - "0") * 10
  GET 13,temp
  laMin = (temp - "0") + laMin

  GET 15,temp
  temp2 = (temp - "0") * 1000
  GET 16,temp
  temp2 = (temp - "0") * 100 + temp2
  GET 17,temp
  temp2 = (temp - "0") * 10 + temp2
  GET 18,temp
  temp2 = (temp - "0") + temp2
  laSec = temp2 * 6 / 10 + 5 / 10      ' temp2 * 0.06 (with rounding)

  laNS = 0
  GET 20,temp
  IF (temp = "N") THEN Get_Long
  laNS = 1

```

```

Get_Long:
  GET 22,temp
  loDeg = (temp - "0") * 100
  GET 23,temp
  loDeg = (temp - "0") * 10 + loDeg
  GET 24,temp
  loDeg = (temp - "0") + loDeg

  GET 25,temp
  loMin = (temp - "0") * 10
  GET 26,temp
  loMin = (temp - "0") + loMin

  GET 28,temp
  temp2 = (temp - "0") * 1000
  GET 29,temp
  temp2 = (temp - "0") * 100 + temp2
  GET 30,temp
  temp2 = (temp - "0") * 10 + temp2
  GET 31,temp
  temp2 = (temp - "0") + temp2
  loSec = temp2 * 6 / 10 + 5 / 10      ' temp2 * 0.06 (with rounding)

  loEW = 0
  GET 33,temp
  IF (temp = "E") THEN Parse_GPS_Done
  loEW = 1

Parse_GPS_Done:
  RETURN

' *****
' Save current GPS reading to EEPROM
' *****

Save_Reading:

  eeBase = (readings - 1) * 9 + FirstAddr

  eeAddr = eeBase + 0 : eeData = flags : GOSUB WriteBigEE
  eeAddr = eeBase + 1 : eeData = laDeg : GOSUB WriteBigEE
  eeAddr = eeBase + 2 : eeData = laMin : GOSUB WriteBigEE
  eeAddr = eeBase + 3 : eeData = laSLo : GOSUB WriteBigEE
  eeAddr = eeBase + 4 : eeData = laSHi : GOSUB WriteBigEE
  eeAddr = eeBase + 5 : eeData = loDeg : GOSUB WriteBigEE
  eeAddr = eeBase + 6 : eeData = loMin : GOSUB WriteBigEE
  eeAddr = eeBase + 7 : eeData = loSLo : GOSUB WriteBigEE
  eeAddr = eeBase + 8 : eeData = loSHi : GOSUB WriteBigEE

  readings = readings + 1

  STORE FirstSlot      ' save readings in slot 1
  WRITE 0,readings.LowByte
  WRITE 1,readings.HighByte

  RETURN

' *****
' Dump Acquired Data To PC
' *****

Dump_Data:
  DEBUG CLS,"Press any key when ready to dump datalog..."
  SERIN RxD,T9600,[response]
  DEBUG CLS
  PAUSE 1000

  STORE FirstSlot

```

```

READ 0,readings.LowByte
READ 1,readings.HighByte

FOR rdngNum = 1 TO readings

  ' retrieve readings from EEPROM

  eeBase = (rdngNum - 1) * 9 + FirstAddr

  eeAddr = eeBase + 0 : GOSUB ReadBigEE : flags = eeData
  eeAddr = eeBase + 1 : GOSUB ReadBigEE : laDeg = eeData
  eeAddr = eeBase + 2 : GOSUB ReadBigEE : laMin = eeData
  eeAddr = eeBase + 3 : GOSUB ReadBigEE : laSLo = eeData
  eeAddr = eeBase + 4 : GOSUB ReadBigEE : laSHi = eeData
  eeAddr = eeBase + 5 : GOSUB ReadBigEE : loDeg = eeData
  eeAddr = eeBase + 6 : GOSUB ReadBigEE : loMin = eeData
  eeAddr = eeBase + 7 : GOSUB ReadBigEE : loSLo = eeData
  eeAddr = eeBase + 8 : GOSUB ReadBigEE : loSHi = eeData

  ' send to PC in comma-delimited format

  DEBUG DEC4 rdngNum, " "
  DEBUG DEC2 laDeg,":",DEC2 laMin,": "
  DEBUG DEC2 laSec / 10,".",DEC1 laSec // 10,Comma
  DEBUG "N" + (laNS * 5),Comma
  DEBUG DEC3 loDeg,":",DEC2 loMin,": "
  DEBUG DEC2 loSec / 10,".",DEC1 loSec // 10,Comma
  DEBUG "E" + (loEW * 18),CR

NEXT
END

' *****
' Big EEPROM Management
' -- uses available slots as big EEPROM
' *****

WriteBigEE:
  IF (eeAddr > MaxAddr) THEN NoWrite          ' check for bad eeAddr
  slot = (eeAddr / 2048) + FirstSlot          ' calc pgm slot
  addr = eeAddr // 2048                       ' calc address in slot
  STORE slot
  WRITE addr,eeData
NoWrite:
  RETURN

ReadBigEE:
  IF (eeAddr > MaxAddr) THEN NoRead          ' check for bad eeAddr
  slot = (eeAddr / 2048) + FirstSlot          ' calc pgm slot
  addr = eeAddr // 2048                       ' calc address in slot
  STORE slot
  READ addr,eeData
NoRead:
  RETURN

```