' =========================================================================

'

' File...... GPSDemoV1.1.BS2

' Purpose... Demonstrates features of the Parallax GPS Receiver Module

' Author.... (c) Grand Idea Studio, Inc. [www.grandideastudio.com]

' E-mail.... support@parallax.com

' Updated... 04 Oct 2006 (by Parallax Tech Support)

'

' {$STAMP BS2}

' {$PBASIC 2.5}

'

' =========================================================================

' -----[ Program Description ]---------------------------------------------

'

' This program demonstrates the capabilities of the Parallax GPS Receiver

' Module.

'

' Before running this demo, ensure that the /RAW pin is left unconnected

' or pulled HIGH to enable "smart" mode, in which the GPS Receiver Module

' will accept commands and return the requested GPS data.

'

' For an application that requires constant monitoring of multiple GPS

' data components, it is recommended to use the "raw" mode of the GPS

' Receiver Module by pulling the /RAW Pin LOW. In this mode, the module

' will transmit a constant stream of raw NMEA0183 data strings, which can

' then be parsed by the host application.

' -----[ I/O Definitions ]-------------------------------------------------

Sio PIN 15 ' connects to GPS Module SIO pin

' -----[ Constants ]-------------------------------------------------------

T4800 CON 188

Open CON $8000

Baud CON Open | T4800 ' Open mode to allow daisy chaining

MoveTo CON 2 ' DEBUG positioning command

ClrRt CON 11 ' clear line right of cursor

FieldLen CON 22 ' length of debug text

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 PST ' for San Diego, California

DegSym CON 176 ' degrees symbol for report

MinSym CON 39 ' minutes symbol

SecSym CON 34 ' seconds symbol

' GPS Module Commands

GetInfo CON $00

GetValid CON $01

GetSats CON $02

GetTime CON $03

GetDate CON $04

GetLat CON $05

GetLong CON $06

GetAlt CON $07

GetSpeed CON $08

GetHead CON $09

' -----[ Variables ]-------------------------------------------------------

char VAR Byte

workVal VAR Word ' for numeric conversions

eeAddr VAR workVal ' pointer to EE data

ver\_hw VAR Byte

ver\_fw VAR Byte

valid VAR Byte ' signal valid? 0 = not valid, 1 = valid

sats VAR Byte ' number of satellites used in positioning calculations

tmHrs VAR Byte ' time fields

tmMins VAR Byte

tmSecs VAR Byte

day VAR Byte ' day of month, 1-31

month VAR Byte ' month, 1-12

year VAR Byte ' year, 00-99

degrees VAR Byte ' latitude/longitude degrees

minutes VAR Byte ' latitude/longitude minutes

minutesD VAR Word ' latitude/longitude decimal minutes

dir VAR Byte ' direction (latitude: 0 = N, 1 = S, longitude: 0 = E, 1 = W)

heading VAR Word ' heading in 0.1 degrees

alt VAR Word ' altitude in 0.1 meters

speed VAR Word ' speed in 0.1 knots

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

NotValid DATA "No", 0

IsValid DATA "Yes", 0

DaysInMon DATA 31,28,31,30,31,30,31,31,30,31,30,31

MonNames DATA "JAN",0,"FEB",0,"MAR",0,"APR",0,"MAY",0,"JUN",0

 DATA "JUL",0,"AUG",0,"SEP",0,"OCT",0,"NOV",0,"DEC",0

' -----[ Initialization ]--------------------------------------------------

Initialize:

 PAUSE 250 ' let DEBUG open

 DEBUG CLS ' clear the screen

 DEBUG "Parallax GPS Receiver Module Test Application", CR,

 "---------------------------------------------"

Draw\_Data\_Labels:

 DEBUG MoveTo, 0, 3, " Hardware Version: "

 DEBUG MoveTo, 0, 4, " Firmware Version: "

 DEBUG MoveTo, 0, 6, " Signal Valid: "

 DEBUG MoveTo, 0, 7, " Acquired Satellites: "

 DEBUG MoveTo, 0, 9, " Local Time: "

 DEBUG MoveTo, 0, 10, " Local Date: "

 DEBUG MoveTo, 0, 12, " Latitude: "

 DEBUG MoveTo, 0, 13, " Longitude: "

 DEBUG MoveTo, 0, 14, " Altitude: "

 DEBUG MoveTo, 0, 15, " Speed: "

 DEBUG MoveTo, 0, 16, " Direction of Travel: "

' -----[ Program Code ]----------------------------------------------------

Main:

 GOSUB Get\_Info

 GOSUB Get\_Valid

 GOSUB Get\_Sats

 GOSUB Get\_TimeDate

 GOSUB Get\_Lat

 GOSUB Get\_Long

 GOSUB Get\_Alt

 GOSUB Get\_Speed

 GOSUB Get\_Head

 GOTO Main

' -----[ Subroutines ]-----------------------------------------------------

' ----------------------------------------------------

Get\_Info:

 SEROUT Sio, Baud, ["!GPS", GetInfo]

 SERIN Sio, Baud, 3000, No\_Response, [ver\_hw, ver\_fw]

 DEBUG MoveTo, FieldLen, 3, HEX ver\_hw.HIGHNIB, ".", HEX ver\_hw.LOWNIB

 DEBUG MoveTo, FieldLen, 4, HEX ver\_fw.HIGHNIB, ".", HEX ver\_fw.LOWNIB

 RETURN

' ----------------------------------------------------

Get\_Valid:

 SEROUT Sio, Baud, ["!GPS", GetValid]

 SERIN Sio, Baud, 3000, No\_Response, [valid]

 DEBUG MoveTo, FieldLen, 6 ' was the signal valid?

 LOOKUP valid, [NotValid, IsValid], eeAddr ' get answer from EE

 GOSUB Print\_Z\_String ' print it

 DEBUG ClrRt ' clear end of line

 IF (valid = 0) THEN Signal\_Not\_Valid

 RETURN

' ----------------------------------------------------

Get\_Sats:

 SEROUT Sio, Baud, ["!GPS", GetSats]

 SERIN Sio, Baud, 3000, No\_Response, [sats]

 DEBUG MoveTo, FieldLen, 7, DEC sats

 RETURN

' ----------------------------------------------------

Get\_TimeDate:

 SEROUT Sio, Baud, ["!GPS", GetTime]

 SERIN Sio, Baud, 3000, No\_Response, [tmHrs, tmMins, tmSecs]

 SEROUT Sio, Baud, ["!GPS", GetDate]

 SERIN Sio, Baud, 3000, No\_Response, [day, month, year]

 GOSUB Correct\_Local\_Time\_Date

 DEBUG MoveTo, FieldLen, 9, DEC2 tmHrs, ":", DEC2 tmMins, ":", DEC2 tmSecs

 DEBUG MoveTo, FieldLen, 10, DEC2 day, " "

 eeAddr = (month - 1) \* 4 + MonNames ' get address of month name

 GOSUB Print\_Z\_String ' print it

 DEBUG " 20", DEC2 year

 RETURN

' ----------------------------------------------------

Get\_Lat:

 SEROUT Sio, Baud, ["!GPS", GetLat]

 SERIN Sio, Baud, 3000, No\_Response, [degrees, minutes, minutesD.HIGHBYTE, minutesD.LOWBYTE, dir]

 ' convert decimal minutes to tenths of seconds

 workVal = minutesD \*\* $0F5C ' minutesD \* 0.06

 DEBUG MoveTo, FieldLen, 12, DEC3 degrees, DegSym, " ", DEC2 minutes, MinSym, " "

 DEBUG DEC2 (workVal / 10), ".", DEC1 (workVal // 10), SecSym, " "

 DEBUG "N" + (dir \* 5)

 ' convert to decimal format, too

 workVal = (minutes \* 1000 / 6) + (minutesD / 60)

 DEBUG " (", " " + (dir \* 13), DEC degrees, ".", DEC4 workVal, " ) "

 RETURN

' ----------------------------------------------------

Get\_Long:

 SEROUT Sio, Baud, ["!GPS", GetLong]

 SERIN Sio, Baud, 3000, No\_Response, [degrees, minutes, minutesD.HIGHBYTE, minutesD.LOWBYTE, dir]

 ' convert decimal minutes to tenths of seconds

 workVal = minutesD \*\* $0F5C ' minutesD \* 0.06

 DEBUG MoveTo, FieldLen, 13, DEC3 degrees, DegSym, " ", DEC2 minutes, MinSym, " "

 DEBUG DEC2 (workVal / 10), ".", DEC1 (workVal // 10), SecSym, " "

 DEBUG "E" + (dir \* 18)

 ' convert to decimal format, too

 workVal = (minutes \* 1000 / 6) + (minutesD / 60)

 DEBUG " (", " " + (dir \* 13), DEC degrees, ".", DEC4 workVal, " ) "

 RETURN

' ----------------------------------------------------

Get\_Alt:

 SEROUT Sio, Baud, ["!GPS", GetAlt]

 SERIN Sio, Baud, 3000, No\_Response, [alt.HIGHBYTE, alt.LOWBYTE]

 DEBUG MoveTo, FieldLen, 14, DEC (alt / 10), ".", DEC1 (alt // 10), " meters "

 workVal = alt / 10 ' remove tenths from altitude

 ' convert altitude from meters to feet

 workVal = (workVal \* 3) + (workVal \*\* $47E5) ' 1 meter = 3.2808399 feet

 DEBUG " ( ", DEC workVal, " feet ) "

 RETURN

' ----------------------------------------------------

Get\_Speed:

 SEROUT Sio, Baud, ["!GPS", GetSpeed]

 SERIN Sio, Baud, 3000, No\_Response, [speed.HIGHBYTE, speed.LOWBYTE]

 DEBUG MoveTo, FieldLen, 15, DEC (speed / 10), ".", DEC1 (speed // 10), " Knots "

 ' convert speed from knots to MPH

 workVal = speed + (speed \*\* $2699) ' 1 knot = 1.1507771555 MPH

 DEBUG " ( ", DEC (workVal / 10), ".", DEC1 (workVal // 10), " MPH ) "

 RETURN

' ----------------------------------------------------

Get\_Head:

 SEROUT Sio, Baud, ["!GPS", GetHead]

 SERIN Sio, Baud, 3000, No\_Response, [heading.HIGHBYTE, heading.LOWBYTE]

 IF speed = 0 THEN

 DEBUG MoveTo, FieldLen, 16, "N/A "

 ELSE

 DEBUG MoveTo, FieldLen, 16, DEC (heading / 10), ".", DEC1 (heading // 10), DegSym, " "

 ENDIF

 RETURN

' ----------------------------------------------------

No\_Response:

 DEBUG MoveTo, 0, 18, "Error: No response from GPS Receiver Module"

 PAUSE 5000

 GOTO Initialize

 '

' ----------------------------------------------------

Signal\_Not\_Valid:

 DEBUG MoveTo, FieldLen, 7, "?", ClrRt ' clear all fields

 DEBUG MoveTo, FieldLen, 9, "?", ClrRt

 DEBUG MoveTo, FieldLen, 10, "?", ClrRt

 DEBUG MoveTo, FieldLen, 12, "?", ClrRt

 DEBUG MoveTo, FieldLen, 13, "?", ClrRt

 DEBUG MoveTo, FieldLen, 14, "?", ClrRt

 DEBUG MoveTo, FieldLen, 15, "?", ClrRt

 DEBUG MoveTo, FieldLen, 16, "?", ClrRt

 GOTO Main

' ----------------------------------------------------

' adjust date for local position

Correct\_Local\_Time\_Date:

 workVal = tmHrs + UTCfix ' add UTC offset

 IF (workVal < 24) THEN Adjust\_Time ' midnight crossed?

 workVal = UTCfix ' yes, so adjust date

 BRANCH workVal.BIT15, [Location\_Leads, Location\_Lags]

Location\_Leads: ' east of Greenwich

 day = day + 1 ' no, move to next day

 eeAddr = DaysInMon \* (month - 1) ' get days in month

 READ eeAddr, char

 IF (day <= char) THEN Adjust\_Time ' in same month?

 month = month + 1 ' no, move to next month

 day = 1 ' first day

 IF (month < 13) THEN Adjust\_Time ' in same year?

 month = 1 ' no, set to January

 year = year + 1 // 100 ' add one to year

 GOTO Adjust\_Time

Location\_Lags: ' west of Greenwich

 day = day - 1 ' adjust day

 IF (day > 0) THEN Adjust\_Time ' same month?

 month = month - 1

 IF (month > 0) THEN Adjust\_Time ' same year?

 month = 1 ' no, set to January

 eeAddr = DaysInMon \* (month - 1)

 READ eeAddr, day ' get new day

 year = year + 99 // 100 ' set to previous year

Adjust\_Time:

 tmHrs = tmHrs + (24 + UTCfix) // 24 ' localize hours

 RETURN

' ----------------------------------------------------

' Print Zero-terminated string stored in EEPROM

' -- eeAddr - starting character of string

Print\_Z\_String:

 READ eeAddr, char ' get char from EE

 IF (char = 0) THEN Print\_Z\_String\_Done ' if zero, we're done

 DEBUG char ' print the char

 eeAddr = eeAddr + 1 ' point to the next one

 GOTO Print\_Z\_String

Print\_Z\_String\_Done:

 RETURN

' ----------------------------------------------------