Transmitting GPS position by an Iridium phone

F. SIGERNES AND JEFF HOLMES

University Centre in Svalbard (UNIS), Box 156, N-9170 Longyearbyen, Norway

Abstract
This document describes how to send GPS information using an Iridium portable phone. The core component in the system is a Basic Stamp II microcontroller from the company Parallax, Inc. Position is extracted from the NMEA sentences of the GPS by the controller and translated to a PDU formatted SMS. The message is submitted to the iridium phone by using standard AT – commands. The SMS message contains an e-mail address which is gated by the Iridium satellite network to the addressee via *internet*.

1. Motivation

The rather harsh environment of the arctic requires us to keep track of our students out in the field. If an emergency situation occurs, the exact location of our students is reported automatically to the logistical department at UNIS. Fig. 1 shows a typical experiment where one of our students is out in the field to sample snow temperatures. It illustrates that it is important to act quickly and respond with accurate information to any deployed rescue team.
This is the main motivation for this project.

Figure 1. Snow temperature recording.

2. System layout

The system contains an Iridium 9505A portable phone with an AT- Command compatible modem connected to a Stamp II (BS2-IC) microcontroller from Parallax, Inc. The Stamp II chip itself is mounted onto a Basic Stamp 2 program board which provides easy access to the programming port and the connected components. A RS-232 Serial LCD screen is connected to the Stamp to provide status information. The whole system is designed to run on a +9 Volt battery. The GPS (Garmin-eTrex) is setup to transmit the NMEA protocol at a RS-232 serial speed of 4800 baud, 8 data bits, 1 stop bit and no parity. Correspondingly, the Stamp communicates with the Iridium modem but at a higher speed (19200 baud). A push button is connected to send the SMS and a switch is included to optionally provide continuous transmissions at a pre selected interval (~15 minutes). Fig. 2 shows how the system is connected together.

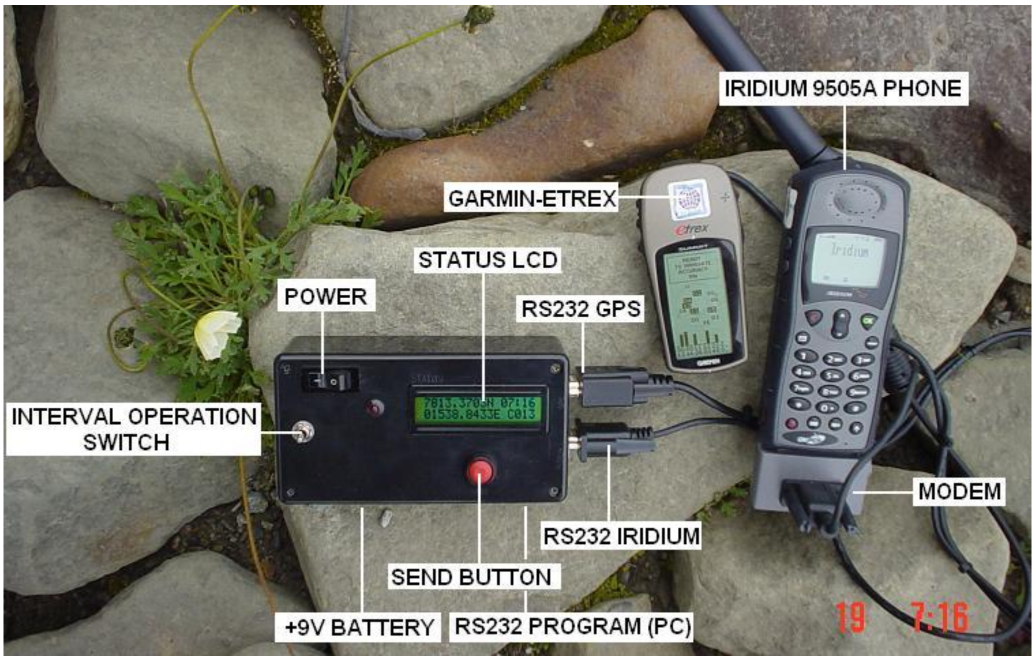


Figure 2. The modules of the system connected together. The units are ready for operation.

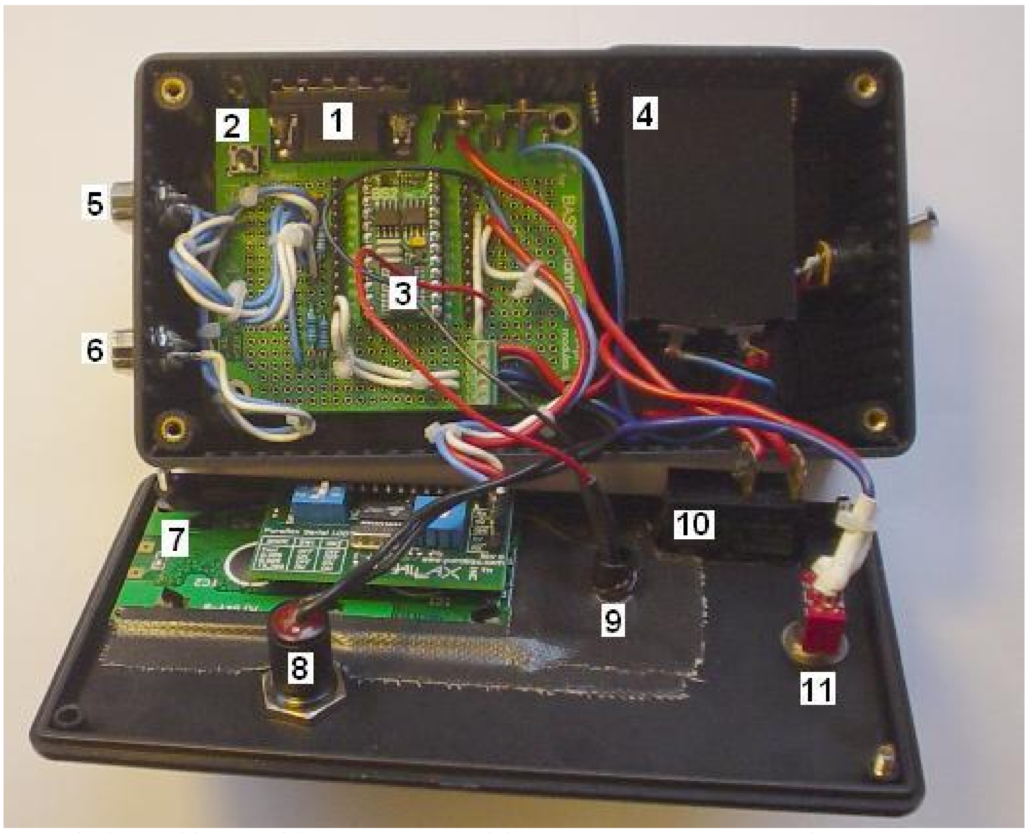


Figure 3. The internal layout of the system. (1) Serial program port to PC, (2) Basic Stamp II prototype board, (3) Basic Stamp II, (4) battery compartment, (5) serial port to Iridium, (6) serial port to GPS, (7) status LCD, (8) send button, (9) activity LED, (10) power switch, and (11) interval switch.

The internal layout and wiring of the black box that contains the Stamp II module is shown in Fig. 3. Each of the key parts is labelled with numbers. The corresponding schematic is shown in Fig. 4.

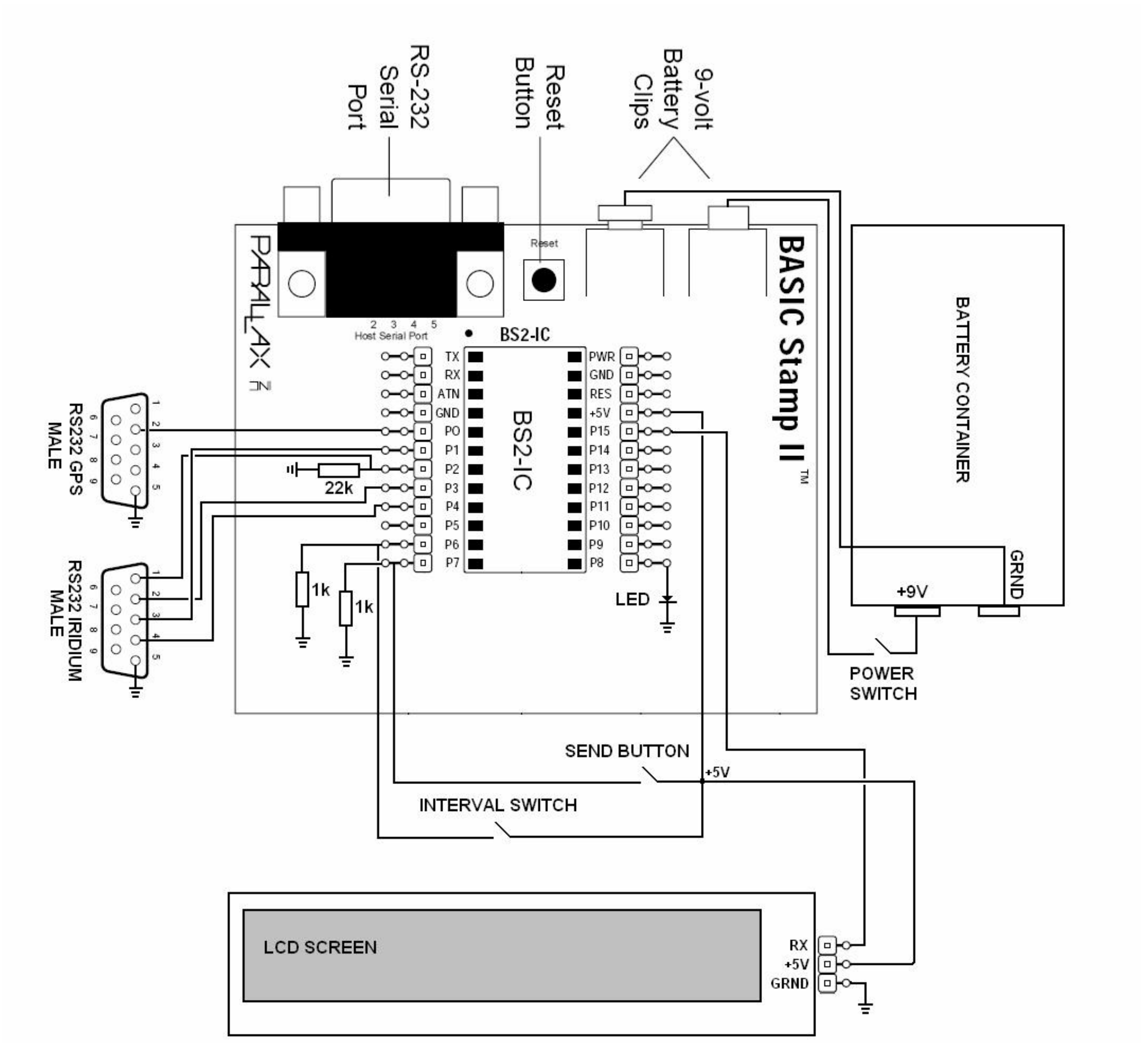


Figure 4. Schematics of the black box connecting the GPS and the Iridium phone to the Basic Stamp II microcontroller. A battery compartment is added to the unit and a serial LCD screen is used to get status information.

3. Programming the Basic Stamp II

The program requires the Basic Stamp II to communicate with three different devices using standard serial RS232 ports.

3.1 Interfacing the GPS and the LCD with the BS2-IC

The procedure is to first extract position data from the NMEA sentences that the GPS transmits. This is done by the SERIN command using the WAIT command to pick out the right NMEA sentences. We use the sentences starting with $GPGSA, $GPRMC and

$PGRMZ (GARMIN) for validation, position and altitude, respectively. Below is an example on how to extract altitude.

```
$PGRMZ, 707, f, 3*1B                                          NMEA sentence
SERIN 0, 16572, [ WAIT ( "$PGRMZ" ),SKIP 1, DEC3 ALTITUDE]    BS2-IC command
```

The Stamp waits for the $PGRMZ condition to occur, then it skips the comma and reads the altitude as a 3 digit decimal text (707 feet).

Next, the status of the process is displayed on the serial LCD connected to pin 15 of the Stamp. The communication with the LCD is straight forward. For example,

```
SEROUT 15, 32, 25, [12, "STATUS..." ] .
```

3.2 Communication with the Iridium modem

The last step in the procedure involves interfacing with the Iridium modem. The Stamp needs to emulate a *hyper terminal* in order to communicate with the modem. This is done by setting the DTR (Data Terminal Ready) high (Iridium RS232 pin 4). In addition, the DCD (Data Carrier Detect) line can be used to signal to the Stamp that the modem is on line and ready to talk. DCD is pin 1 on the Iridium RS232 port and is connected to pin 2 on the Stamp. The Stamp is now ready to send and receive AT - commands through ports 1 and 3, respectively (see Fig. 4).

The Iridium modem needs to be in initialized by sending the following AT commands

```
SEROUT 1, 16416, 25, [ "AT" , CR]          Check communication with modem
SERIN 3, 16416, [ WAIT ( "OK" )]           Wait for response
SEROUT 1, 16416, 25, [ "AT &F &C1" , CR]   Bring up factory defaults and enable DCD
SERIN 3, 16416, [ WAIT ( "OK" )]           Wait for handshake
```

The text format of the SMS message we want to send contains an e-mail address and the extracted GPS information. Below is a typical example.

TRACKGPS@unis.no	*E-mail address*
STRK 001	*Track unit identifier*
2316UT	*Universal time*
N 79 50 8459	*Latitude*
E 014 04 2661	*Longitude*
013 355 00062	*Speed, course and altitude*

This message needs to be transformed to the PDU (Protocol Description Unit) format, since the Iridium phone does not support text mode. The PDU string contains not only the message, but also a header which includes key parameters about where and how the message will be sent to the SMSC (Short Message Service Center). Each character in the string is all of hexa-decimal octets or decimal-semi octets. The user data or the actual message consists of hexa-decimal 8-bit octets, but these octets represent 7 bit data.

As seen above, our messages all have an e-mail address and a unit identifier string in the first line "TRACKGPS@unis.no STRK 001". If we for simplicity send only the username of the e-mail, the message will in PDU mode look like

00110002912a0000AA08546970B83C42A7.

There are 32 characters in this message, or 16 octets. The first octet 00 does not count. The blue part of the string represents the header and the red part is the user message, respectively. The below table below shows the content of the PDU string.

Octet(s)	Description	Example
00	Length of SMSC information	00 - Information is set by phone
11	First SMS-SUBMIT octet	PDU Format indicator (layout)
00	Message Reference	00 - Reference number is set by phone
02	Length of phone number	2
91	Format of phone number	International
2a	Phone number	*2 – Uses number stored in phone
00	Protocol identifier	Type 0
00	Data coding scheme	7-bit
AA	Validity period	AA - Means 4 days
08	Length of message	8
54 69 70 B8 3C 42 A7	Users data	TRACKGPS

Table 1. Simple PDU - SUBMIT message for the Iridium phone.

The message contains 8 characters. These are called septets in the 7-bit alphabet. The transformation by the BS2-IC to octets is done by shifting bits of 8 septets sequentially. A simple BS2-IC program below demonstrates the process.

```
'{$STAMP BS2}
'{$PORT COM1}
'Stamp II program to Convert ASCII to PDU format
'F. Sigernes University Centre on Svalbard (UNIS)
'Variables
PDU VAR BYTE(7)
CH VAR BYTE(8)
'Assign ASCII string to convert: TRACKGPS
CH(0)="T":CH(1)="R"
CH(2)="A":CH(3)="C"
CH(4)="K":CH(5)="G"
CH(6)="P":CH(7)="S"
'PDU shift routine
PDU(0)=(CH(1) << 7)+(CH(0) >> 0)
PDU(1)=(CH(2) << 6)+(CH(1) >> 1)
PDU(2)=(CH(3) << 5)+(CH(2) >> 2)
PDU(3)=(CH(4) << 4)+(CH(3) >> 3)
PDU(4)=(CH(5) << 3)+(CH(4) >> 4)
PDU(5)=(CH(6) << 2)+(CH(5) >> 5)
PDU(6)=(CH(7) << 1)+(CH(6) >> 6)
'PDU output: 54 69 70 B8 3C 42 A7
DEBUG HEX2 PDU(0),HEX2 PDU(1),HEX2 PDU(2),HEX2 PDU(3),HEX2 PDU(4),HEX2
PDU(5),HEX2 PDU(6)
'END
```

In this way, the trick is to make sure that we write messages that have blocks of 8 septets. The next block in the procedure would then be the string “@unis.no”, and so on.

The BS2-IC sends the SMS message in PDU mode by the following AT-commands

```
SEROUT 1, 16416,25, [ "AT+CMGF=0" , CR]            Set phone to PDU mode
SERIN 3, 16416,[ WAIT ( "OK" )]                    Wait for OK
SEROUT 1,16416,25, [ "AT+CMGS=16" , CR]            Send SMS message of 16 octets
SEROUT 1,16416,25,[ “00110002912a0000AA08”]        Send the header
SEROUT 1,16416,25,[“546970B83C42A7”]               Transfer the  user data
PAUSE 2500                                         Wait a second
SEROUT 1,16416,25, [26]                            Terminate session with <ctrl-Z>
```

A complete listing of the main program is given in the section 6.

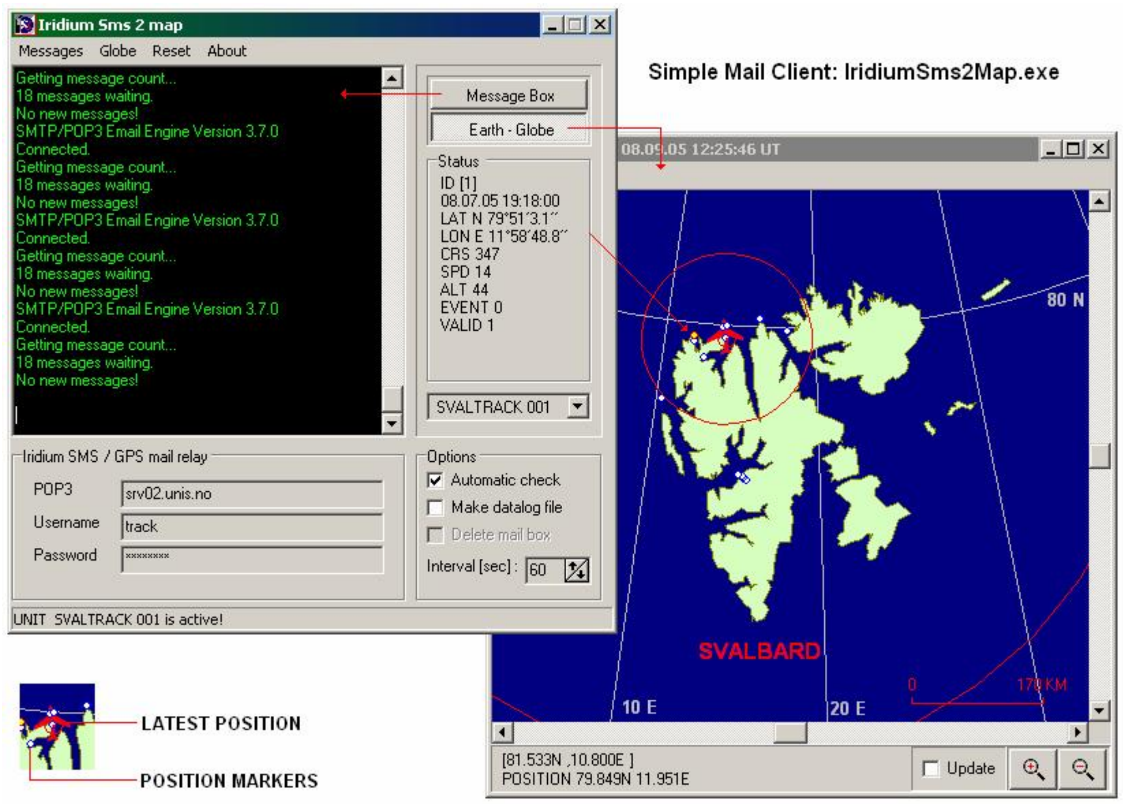
4. Test of the assembled system

A field trip around the archipelago of Svalbard was conducted by ship in July 2005. Fig. 5 shows the system in operation. The location is 79^o 50’ 50.8’’ North and 14^o 4’ 16’’ East. The main limitation is the start up time of the GPS, which in can take up to 2 minutes. As soon as both the Iridium phone and the GPS are ready, the system takes about 5-10 seconds to send the SMS. A total of 18 messages were sent. All were received by the Iridium Short Message Service Center (SMSC), and ported successfully to the UNIS mail server.

Figure 5. H. F. Flå and F. Sigernes out in the field testing the GPS-SMS-Iridium system, 08.07.2005

The receiver or the home base uses a simple mail client to read each mail and plot the position on a map. The program checks the mail account on the POP3 server for any new messages automatically. The identifier string in the message sorts out which unit that is tracked by the program. That way, several units can be deployed out to different groups in the field.

Fig.6 shows a snapshot of the mail client. Note that the position on the map marked with yellow colour corresponds to the Status box in the main program module. This is the same location as seen in Fig. 5. The mail client has also been tested using a laptop and the iridium data kit to connect to the internet. In other words, the home base does not necessarily have to be located at a fixed position; it may be mobile as well.



```
Getting message count...
18 messages waiting.
No new messages!
SMTP/POP3 Email Engine Version 3.7.0
Connected.
Getting message count...
18 messages waiting.
No new messages!
SMTP/POP3 Email Engine Version 3.7.0
Connected.
Getting message count...
18 messages waiting.
No new messages!
SMTP/POP3 Email Engine Version 3.7.0
Connected.
Getting message count...
18 messages waiting.
No new messages!
```

Status
ID [1]
08.07.05 19:18:00
LAT N 79°51´3.1´´
LON E 11°58´48.8´´
CRS 347
SPD 14
ALT 44
EVENT 0
VALID 1

SVALTRACK 001

Iridium SMS / GPS mail relay
POP3: srv02.unis.no
Username: track
Password: xxxxxxxx

Options
[✓] Automatic check
[ ] Make datalog file
[ ] Delete mail box
Interval [sec]: 60

UNIT SVALTRACK 001 is active!

[81.533N ,10.800E]
POSITION 79.849N 11.951E

Figure 6. A snapshot of the mail client: IridiumSms2Map.exe. The program is written in Borland Delphi 5 Pascal and is win9x, NT and XP compatible.

5. Final remark

The Basic Stamp II (BS2-IC) is indeed capable of retrieving position from NMEA sentences transmitted by a GPS. Furthermore, it easily transforms the information to a SMS - PDU formatted message. The Stamp also communicates reliably with the Iridium modem in order to send the SMS message. The BS2-IC is the key element for our tracking system.

6. Basic Stamp II program listing: IridiumSMSGPS.bs2

```
'{$STAMP BS2}
'{$PORT COM1}
'Stamp II program to read GPS and send SMS by Iridium
'F. Sigernes University Centre on Svalbard (UNIS)

'GPS variables
LATDEG VAR Word
LATDEC VAR Word
LONDEG VAR Word
LONDEC VAR Word
SPD VAR Word
CRS VAR Word
ALT VAR Word
TMP VAR Byte
DIRLAT VAR Byte
DIRLON VAR Byte
T VAR Byte(4)

'Iridium and main loop variables
C VAR Byte
D VAR Byte
PDU VAR Byte
SAT VAR Byte

'Initialization of loop variables and Iridium modem
'P2 Checks Data Carrier Detect (DCD) on Iridium RS232 pin 1
'Iridium RS232 pin 2 and 3 is connected to P3 and P1, respectively
INPUT 2
'P4 is set high to signal Data Terminal Ready (DTR) to Iridium RS232 pin 4
OUTPUT 4
OUT4 = 1
D=0
C=1
PAUSE 3000
'Show startup message on LCD (P15)
SEROUT 15,32,5,[22]
SEROUT 15,32,5,[12, "SMS GPS-IRIDIUM" ]
SEROUT 15,32,5, [148, "INITIALIZING...." ]

MAIN:
  OUTPUT 8 'P8 is activity LED indicator
  INPUT 7 'Sends SMS by Iridium by pushing button if P7=1
  INPUT 6 'Sends SMS periodically if P6 = 1

  'Get Position and validation from NMEA strings $GPRMC, $PGRMZ (GARMIN) and $GPGSA
  'GPS is connected to P0 and VSS - RS232 Pin 2 and 5, respectively
  SERIN 0,16572, [WAIT( "$GPGSA" ), SKIP 3,SAT]
  IF (SAT = 1) THEN NoFix:
  SERIN 0,16572,1000, NoFix,[ WAIT ( "$GPRMC" ),TMP,STR T\4, SKIP 3,
  DEC4 LATDEG,TMP, DEC4 LATDEC,TMP,DIRLAT,TMP, DEC5 LONDEG,TMP,
  DEC4 LONDEC,TMP,DIRLON,TMP, DEC3 SPD,TMP,TMP,TMP, DEC3 CRS]
  IF T(0) = "," THEN NoFix:
  SERIN 0,16572,1000, NoFix, [ WAIT ( "$PGRMZ" ),TMP, DEC5 ALT]
```

```
 ‘Show GPS information on LCD
 SEROUT 15,32,5,[12, DEC4 LATDEG, "." , DEC4 LATDEC,DIRLAT, " " ,T(0),T(1), ":" ,T(2),T(3)]
 SEROUT 15,32,5,[148, DEC5 LONDEG, "." , DEC4 LONDEC,DIRLON, " C" , DEC3 C]

‘Checks status of interval switch and push button
IF (IN7 = 0) AND (D=0) THEN co:

 ‘START OF SMS MESSAGE (P7 or P6 HIGH)
 SEROUT 15,32,25,[12, "SMS GPS DATA..." ]
  I_Modem: ‘Initialize Iridium modem
  OUT8  = 1 ‘Activity LED on
  PAUSE 250
  SEROUT 1,16416,25,[ "AT" , CR]
  SERIN 3,16416,2500,Error,[ WAIT ( "OK" )]
  SEROUT 1,16416,25,[ "AT &F &C1" , CR]
  SERIN 3,16416,2500,Error,[ WAIT ( "OK" )]
  'Set modem to PDU mode
  SEROUT 1, 16416,25, [ "AT+CMGF=0" , CR]
  SERIN 3, 16416,2500,Error,[ WAIT ( "OK" )]
  ‘Set total length of SMS to 79 octets
  SEROUT 1,16416,25, [ "AT+CMGS=79" , CR]
  'SMS header and phone numbers to call (8 octets)
  SEROUT 1,16416,25,[ "00110002912a0000AA" ]
  'Number of characters in message to send Hex 50 = 80 Dec (1 octet)
   SEROUT 1,16416,25,[ "50" ]
  'Email and Identifier: track@unis.no STRK 001$13$10' (21 octets)
   SEROUT 1,16416,25,[ "747978BC06D4DDE9B9CBFD064DA9D22508068B3514" ]

  ‘Convert rest of message to 7 PDU blocks of data (49-7bits octets)
  ‘Each block contains 8 septets shifted to 7 bit octets.

  '(1) - Time string to PDU format
  PDU=(T(1) << 7)+(T(0) >> 0)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(T(2) << 6)+(T(1) >> 1)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(T(3) << 5)+(T(2) >> 2)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=( "U"  << 4)+(T(3) >> 3)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=( "T"  << 3)+( "U"  >> 4)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(13 << 2)+( "T"  >> 5)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(10 << 1)+(13 >> 6)
  SEROUT 1,16416,25,[ HEX2 PDU]

  '(2&3) - Convert latitude to PDU
  PDU=(32 << 7)+(DIRLAT >> 0)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((LATDEG DIG 3) + 48) << 6)+(32 >> 1)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((LATDEG DIG 2) + 48) << 5)+(((LATDEG DIG 3) + 48) >> 2)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(32 << 4)+(((LATDEG DIG 2) + 48) >> 3)
```

```
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LATDEG DIG 1) + 48) << 3)+(32 >> 4)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LATDEG DIG 0) + 48) << 2)+(((LATDEG DIG 1) + 48) >> 5)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(32 << 1)+(((LATDEG DIG 0) + 48 ) >> 6)
SEROUT 1,16416,25,[ HEX2 PDU]

PDU=(((LATDEC DIG 2) + 48) << 7)+(((LATDEC DIG 3) + 48) >> 0)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LATDEC DIG 1) + 48) << 6)+(((LATDEC DIG 2) + 48) >> 1)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LATDEC DIG 0) + 48) << 5)+(((LATDEC DIG 1) + 48) >> 2)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(32 << 4)+(((LATDEC DIG 0) + 48) >> 3)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(32 << 3)+(32 >> 4)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(13 << 2)+(32 >> 5)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(10 << 1)+(13 >> 6)
SEROUT 1,16416,25,[ HEX2 PDU]

'(4&5) - Convert longitude to PDU
PDU=(32 << 7)+(DIRLON >> 0)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEG DIG 4) + 48) << 6)+(32 >> 1)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEG DIG 3) + 48) << 5)+(((LONDEG DIG 4) + 48) >> 2)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEG DIG 2) + 48) << 4)+(((LONDEG DIG 3) + 48) >> 3)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(32 << 3)+(((LONDEG DIG 2) + 48) >> 4)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEG DIG 1) + 48) << 2)+(32 >> 5)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEG DIG 0) + 48) << 1)+(((LONDEG DIG 1) + 48) >> 6)
SEROUT 1,16416,25,[ HEX2 PDU]

PDU=(((LONDEC DIG 3) + 48) << 7)+(32 >> 0)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEC DIG 2) + 48) << 6)+(((LONDEC DIG 3) + 48) >> 1)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEC DIG 1) + 48) << 5)+(((LONDEC DIG 2) + 48) >> 2)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(((LONDEC DIG 0) + 48) << 4)+(((LONDEC DIG 1) + 48) >> 3)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(32 << 3)+(((LONDEC DIG 0) + 48) >> 4)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(13 << 2)+(32 >> 5)
SEROUT 1,16416,25,[ HEX2 PDU]
PDU=(10 << 1)+(13 >> 6)
SEROUT 1,16416,25,[ HEX2 PDU]
```

```
  '(6) - Speed and Course to PDU
  PDU=(((SPD DIG 1) + 48) << 7)+(((SPD DIG 2) + 48) >> 0)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((SPD DIG 0) + 48) << 6)+(((SPD DIG 1) + 48) >> 1)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(32 << 5)+(((SPD DIG 0) + 48) >> 2)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((CRS DIG 2) + 48) << 4)+(32 >> 3)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((CRS DIG 1) + 48) << 3)+(((CRS DIG 2) + 48) >> 4)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((CRS DIG 0) + 48) << 2)+(((CRS DIG 1) + 48) >> 5)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(32 << 1)+(((CRS DIG 0) + 48) >> 6)
  SEROUT 1,16416,25,[ HEX2 PDU]

  '(7) - Convert altitude to PDU
  PDU=(((ALT DIG 3) + 48) << 7)+(((ALT DIG 4) + 48) >> 0)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((ALT DIG 2) + 48) << 6)+(((ALT DIG 3) + 48) >> 1)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((ALT DIG 1) + 48) << 5)+(((ALT DIG 2) + 48) >> 2)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(((ALT DIG 0) + 48) << 4)+(((ALT DIG 1) + 48) >> 3)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(32 << 3)+(((ALT DIG 0) + 48) >> 4)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(13 << 2)+(32 >> 5)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PDU=(10 << 1)+(13 >> 6)
  SEROUT 1,16416,25,[ HEX2 PDU]
  PAUSE 2500
  SEROUT 1,16416,25, [26] 'Equivalent to CTRL + Z in ASCII
  'At this point the SMS message has been sent out!
  SEROUT 15,32,5,[148, "DONE" ]
  OUT8 = 0 'Activity LED off
  D=0
  C=0
  GOTO co
  'END OF SMS MESSAGE

 Error:
   PAUSE 1000
   GOTO I_Modem
 NoFix:
  SEROUT 15,32,5,[12, "SMS GPS-IRIDIUM!" ]
  SEROUT 15,32,5,[148, "STATUS:NO GPS " , STR SAT\1]
 co:
 IF C<20 THEN go
  C=0
  IF IN6 = 0 THEN SK
   D=1
  SK:
 go:
 C=C+1
GOTO MAIN 'END
```

Terms and Abbreviations

AT Commands
A group of commands that can be sent by a terminal or host computer to control a modem in Command mode

GPS
Global Position System

NMEA
National Marine Electronics Association

PDU
Protocol Description Unit

SMS
Short Message Service

SMSC
Short Message Service Center

Parallax, Inc
599 Menlo Drive
Suite 100
Rocklin, California 95765
USA
Web: http://www.parallax.com

Useful links

1) Lars Pettersson, "SMS and the PDU format",
http://www.dreamfabric.com/sms/

2) Swen-Peter Ekkebus, "SMS-PDU Converter",
http://home.student.utwente.nl/s.p.ekkebus/portfolio/resource/sms_pdu.html

3) The National Marine Electronics Association,
http://www.nmea.org/

4) The Basic AT Command reference,
http://www.phys.unsw.edu.au/~mcba/iridium/Motorola_AT_Command_Set.pdf