GPS Data (BINEX) Streaming via Iridium Short Burst Data (SBD) Mode Preliminary Report

3/20/2007 Rev. 2

Performed under UNAVCO sub-contract by: Alberto Behar, PhD, Address: 518 11th Street, Unit C, Hermosa Beach, CA 90254. 818-687-8627

Purpose:

This is a report for a system that is able to stream GPS position data (BINEX open format) from a Trimble NetRS (easily adaptable to other units) to a microcontroller + Iridium modem box that can then send the data through the Iridium Network to the UNAVCO operations base where it is repackaged to look like the original stream (Figure 1).

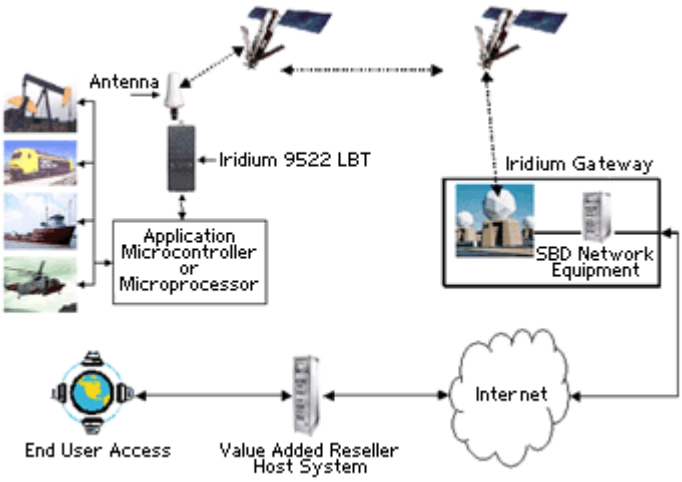

Figure 1 GPS Position Data Path

Remote Unit Configuration:

Data per day:

Records every position 30 sec, 35k/hour
7200 epochs/day, 100-220bytes/epoch) ~1mbyte/day

Download/receive frequency: Continuous as data comes in.
Receiver and Output Format: Trimble NetRS in BINEX Form, 9600bps
Connection Method: Iridium Modem, LBT9522 with DOD Sim card

Operations base details:

PC Computer located at UNAVCO, Boulder, Colorado running: a Linux application (shell script) that reassembles the data into 24hr UTC break files. Communication with Iridium Network is via TCP/IP Direct IP Sockets.

Current description of work performed:

Phase 1 – Proof of Concept:

Testing Plan:
Set up a bench test system with a Trimble NetRS GPS receiver (provided by UNAVCO) and an NAL A3LA series Iridium modem (provided by UNAVCO) that delivered data to a system computer using the Iridium SBD mode.

Developed and built a hardware data buffer unit that is able to send GPS position through an Iridium modem and deliver the data to an application that reliably reconstitutes the data into 24-hr, UTC delineated files.

Deliverables:

1. A system performance report (this document), including ability of system to handle 1 Mb/day throughput, reliability, latency, scalability, and ability for "catch-up" from system outages.
2. The reconstituted BINEX data files.

Since this is a completion of Phase 1, UNAVCO will now determine whether to continue with Phase 2. If the decision is made to not continue, all UNAVCO equipment will be promptly returned.

Phase 2 – Delivery of system prototype:

An operational data collection system was installed at UNAVCO with a remote field unit (installed by UNAVCO) at the Marshall test site.

Deliverables:

1. Two hardware data buffer units.
2. System software installed on a UNAVCO system computer (LINUX OS).
3. Basic software documentation.
4. Source code and the right to modify as required and apply to UNAVCO GPS data retrieval and IRIS seismic data retrieval.
5. A site visit and debugging assistance to implement the operational prototype system.

Facilities, materials, and/or services provided by UNAVCO during this work:

1) Iridium A3LA series modem with DOD Sim Card
2) Trimble Net RS GPS receiver system
3) Assistance with GPS receiver configuration and operation
4) Assistance in setting up receiving software at the UNAVCO operations base

1.1.General Introduction

UNAVCO provides support for scientific applications of the Global Positioning System to the National Science Foundation Office of Polar Programs (NSF/OPP) Artic Sciences Section. This support includes pre-deployment planning, field support and post-deployment follow up for a number of artic projects. (UNAVCO, 2004) During the year 2004 UNAVCO supported eleven different projects in the artic. One of such projects is located in Greenland and monitors the movement of the ice sheet to validate and calibrate the remote sensing estimates of basal melting.

The isolation and climatic conditions that are characteristic of the artic regions pose big challenges to these projects. The GPS receivers are left alone for a complete year, they must work reliably and their power source must last the whole year. Some of the GPS receivers currently installed in Greenland are logging data the whole year, and once a year a person must go to the site and upload the recorded data. There is no real time data transmission in these units.

The main objective of this experiment is to develop system that is able to stream GPS position data in Binex format to a microcontroller. The microcontroller would then send this data through the Iridium network to the UNAVCO operations base where it will be repackaged to look like the original stream. This system must be reliable, low cost and have very low power consumption.

1.2. General System Description

This experiment consists of a NetRS GPS receiver, developed by Trimble, a microcontroller that receives information from the GPS and an Iridium modem that will send information in the form of Short Burst Data (SBD mode). A brief description of each of these elements is discussed in the following paragraphs.

Iridium modem:

The unit being used is an Iridium satellite modem; model LBT9522 with a SIM card capable of sending data through the SBD DirectIP protocol. Figure 2 shows the LBT9522 modem.

The data service utilized will be the Short Burst Data (SBD). SBD is a simple a new data service from Iridium that enables value added applications to send and receive short data transactions efficiently over the Iridium network. The size of the messages can be between zero and 1960 bytes.

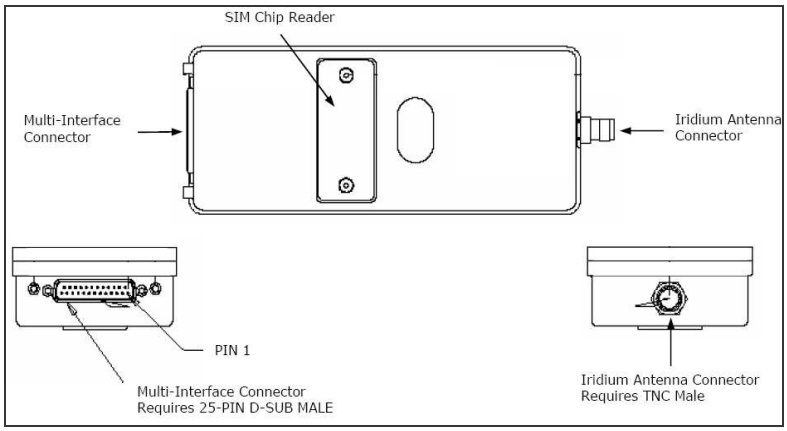

Figure 2. Iridium Modem

The sequence of events for sending an SBD message is the following (see Figure 3):

- The Mobile Application (MA) loads the Mobile Originated (MO) SBD data message into the L-band transceiver.

- The MA instructs the L-band transceiver to send the SBD message to the Iridium Gateway SBD Subsystem (GSS).

- The GSS stores and forwards the message from the MA to the vendor application (VA). This is commonly done through terrestrial communications.

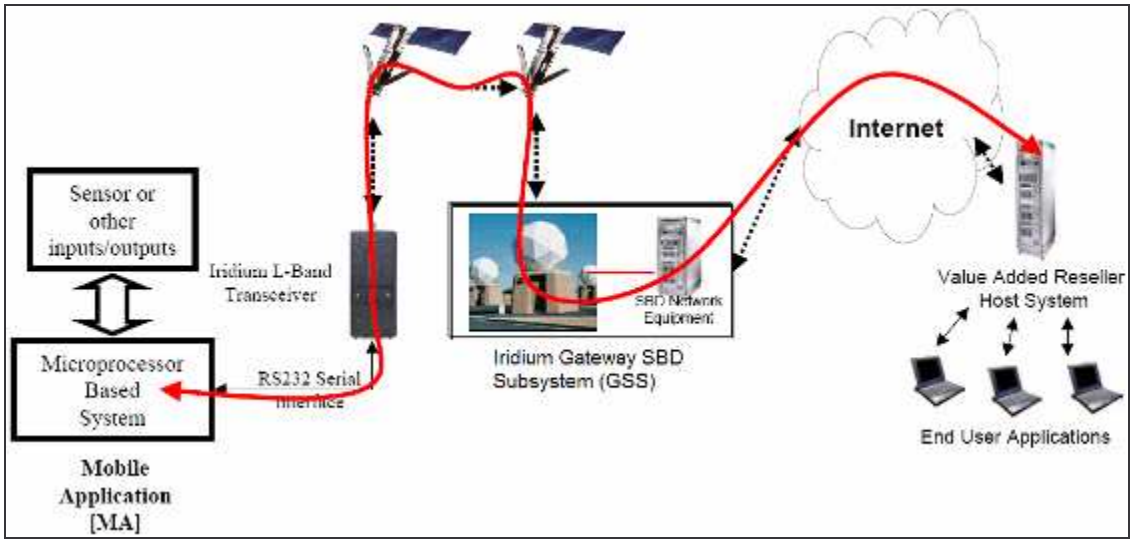

Figure 3. GPS position data path (Iridium, 2003)

To transfer a message from the mobile application to the GSS (first two steps listed above), the following AT commands must be sent to the transceiver:

AT+SBDWB=<message length> <CR> Write binary data to the SSD:

This command will send an SBD message to the SBD subscriber device; this message will be stored in the device's buffer. Once the command is entered, the SBD Subscriber Device (SSD) will indicate to the field application that it is prepared to receive the message by sending the ASCII encoded string "READY<CR><LF>".

The <message length> parameter represents the length, in bytes of the SBD message not including the mandatory two byte checksum. The maximum message length is 1960 bytes.

Once the field application receives the READY indication from the SSD, the SBD message must be sent followed by the 2 byte checksum. The SSD will wait for one minute to receive this information. The two byte checksum is the least significant 2 bytes of the summation of the entire SBD message. The high order byte must be sent first.

Once the message and the checksum are sent, the SSD will respond with any of the following:

0: SBD message successfully written to the SSD.
1: SBD message timeout, an insufficient number of bytes were transferred to the SSD during the transfer period of 60 seconds.
2: SBD message checksum sent from DTE does not match the checksum calculated at the SSD.
3: SBD message size is not correct, the maximum message length is 1960 bytes and the minimum is 1 byte.

AT+SBDI<CR> Initiate an SBD session:

This command initiates an SBD session between the SSD and the Iridium Gateway Subsystem (GSS). If there is a message in the send buffer it will be transferred to the GSS. Similarly, if there is one or more messages in the GSS, they will be transferred and placed into the SSD receive buffer, in the same order as they were received at the GSS.

When the field application sends this command to the transceiver, it will respond one of the following:

+SBDI: <send status>, <MOMSN>, <receive status>, <MTMSN>, <receive length>, <receive queued>

<send status> provides an indication of the send transaction status. The possible values of this field are:

0: No SBD message to send from the SSD.
1: SBD message successfully sent from the SSD to the GSS.
2: An error occurred while attempting to send SBD message from SSD to GSS.

<MOMSN>, the Mobile Originated Message Sequence Number, is a value assigned by the SSD when sending a message to the GSS. This value is incremented each time and SBD session is successfully completed.

The subsequent fields (<receive status>, <MTMSN>, <receive length>, <receive queued>) are used when messages are sent to the field application. Since they are not used in the application described in this report, they will not be explained.

The following table summarizes the commands that the field application must be sent to the Iridium modem, in order to send an SBD message. The table also shows the response that the modem must send back and a small description on each of them.

COMMAND	RESPONSE	DESCRIPTION
AT		Attention code. Checks the status of the Iridium Subscriber Unit (ISU).
	OK	An “OK” response must be sent back by the ISU, indicating that the modem is OK.

AT+SBDWB=length		The field application (FA) instructs the SBD subscriber device that it will write a message into the SSD. The length must be specified.
	READY	The SSD informs to the FA that it is ready to receive the message, it will wait for 60 seconds.
(binary transfer)		The FA sends message, followed by the two byte checksum to the SSD.
	0	The SSD responds with a number, a "0" will indicate that the message was loaded without error.
AT+SBDI		The FA instructs the SSD to initiate an SBD transfer.
		The SSD informs the FA that the message was sent successfully.
AT+SBD0		The FA instructs the SSD to clear the message from the send buffer.
	0	The SSD informs the FA that the message buffer was cleared successfully.

Once an SBD message has been sent from the field application to the Iridium Gateway SBD Subsystem (GSS), the message will be sent to the vendor application (VA) in one of the following ways: email, Direct IP or FTP. Email delivery is simple but delivery latency is indeterminate due to traversing the internet. DirectIP requires a custom server application to receive transfer but delivery latency is near zero. FTP has very low delivery latency, as well. The method used in this application is DirectIP.

DirectIP is a socket based delivery mechanism for Mobile Originated (MO) and Mobile Terminated (MT) SBD messages. The application discussed in this report is a mobile originated SBD message. Upon the completion of an SBD session between an SSD an the GSS, the GSS opens a socket, connects to the vendor application and delivers the MO message including SBD session descriptors. These messages are delivered in a first-in-first-out manner, and only one message is delivered per socket connection. Once a socket connection is established, a single MO message is delivered and then the socket is closed. This sequence is repeated for every MO message queued for delivery to the vendor server.

GPS receiver:

The Trimble NetRS GPS is designed to communicate using standard internet protocols, such as FTP and http for both receiver control and data retrieval. It is also possible to use the serial ports to send information in several formats. The NetRS GPS is basically a computer running on Linux operating system, and it is able to save up to 950 MB of raw data observables.

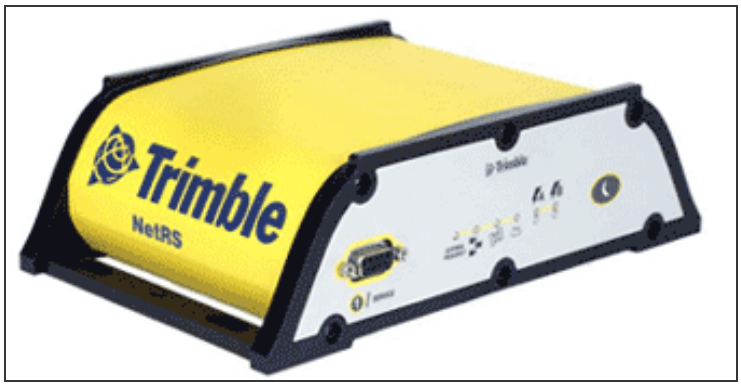

Figure 4. NetRS GPS unit

This GPS unit must send information and log this data in its memory every 30 seconds, which equals to about 1 MB per day of information. The information will be sent immediately, to the Iridium Vendor Application (VA). The format of the data being used is Binex, which is a standard developed by UNAVCO. Binex (Binary Exchange) is a binary format standard for GPS/GLONASS/SBAS and is used for research purposes.

The communications, data logging and security can be configured through the GPS service port. The service port is a serial connection located in the front part of the unit and allows the user to configure the GPS using a web browser. It is also possible to save and download configuration files to one or more GPS units.

Microcontroller:

The microcontroller's function is to receive the messages sent by the GPS, store these messages and send them to the Iridium modem. In order to properly communicate to the modem, the adequate commands must be sent. The microcontroller selected for this application is the Javelin. The Javelin series is the one of the basic stamps that performs well in low temperatures. The following figure and table describe the main characteristics and pinout of the Javelin microcontroller.

Figure 5. Basic Stamp Javelin

PIN	NAME	DESCRIPTION
1	SOUT	Serial out. Connects to PC serial port Rx pin for programming.
2	SIN	Serial in. Connects to PC serial port Tx pin for programming

3	ATN	Attention. Connects to PC serial port DTR pin for programming
4	VSS	System ground.
5 – 20	P0 – P15	General purpose I/O pins.
21	VDD	5 volt DC input / output. If an unregulated voltage is applied to the VIN pin, then this pin will output 5 volts. If no voltage is applied to the VIN pin, a regulated voltage of 5 volts must be applied to this input.
22	RES	Reset pin
23	VSS	System ground (same as pin 4).
24	VIN	Unregulated power input, accepts from 5.5 to 12 VDC, which is then internally regulated to 5 volts. Must be left unconnected if 5 volts are applied to the VDD pin.

1.3.System Design

The final system design (Figure 7) involved the use of an internal memory on the Javelin to store the information sent by the GPS. This information was retrieved and then sent then to the Iridium modem. The following steps summarize the final program.

Figure 7. Test bench for the GPS – Iridium communications

a) Wait for GPS data, Once the GPS starts sending information, wait 2 seconds so the rest of the message is stored in the memory.
b) Initialize the modem and send the message length.
c) Send the data received from the gps and calculate the checksum.

d) Send the last two bytes of the checksum to the Iridium modem.
e) Start again from step (a).

Because the NetRS is set to send information every 30 seconds, all these steps must run in less than that time so the next message is not lost. Once this program was operational, several tests were performed and the program was timed to see if it ran in less than 30 seconds. The program was optimized to perform as fast as possible. The steps that took longer time were the Iridium modem communications because the microcontroller has to wait for a response from the modem before continuing to the next step. It was found that when the signal from the Iridium satellites is low, the program is slower. On average the steps take about 12 seconds to run and then the microcontroller waits for about 18 seconds before the next batch of data from the NetRS.

The final circuit is shown in the following figure. The figure shows the microcontroller board, and an LCD display (used to monitor the communications status).

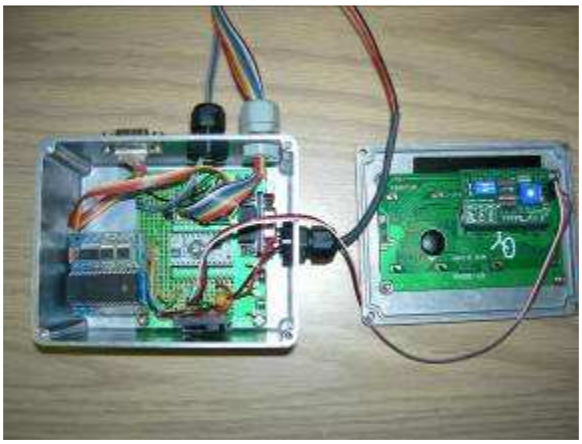

Figure 8. Final electronic circuit.

1.4. System Current Use

The system's current use was measured during testing and was found to be the following:

Transfer box: 90mA avg. from 5.5 to 16V
Iridium modem: 0.24-0.32 mA avg. at 4.6V. During SBD transmit ~1.1A for ~0.5sec

1.5. Operations base code

At the operations base a Unix/Linux shell script is run to receive the data from Iridium. Data latency from when the an SBD packet is sent to when it is received is < 1 sec. The following code is used to process the data coming in. It collects and creates two files (one binary and one hex) of the data entering for each day. The filenames include the date of the data in a Year, Month, Day fashion (Sck_Data_Hex_20060903,

Sck_Data_Bin_20060903). File size for one day's data is ~640K for Binary and 1.4MB for Hex.

```
#!/bin/csh -f
# Sampler Socket Listener

set CallNum = 0                                          # sets var used for number of calls
echo "NetRS Binex Over Iridium"
while (1)                                                # Loop until proces gets killed
   @ datevar = `date +%Y%m%d`                            # Get today's date

   echo "Waiting on Data for Call Count: $CallNum ..."       # Display Header for Status
   nc -l -p 10800 > "Sck_Call_Bin_Tmp"                  # Make listener socket at port 10800

   od -An -tx1 -j60 --width=40 Sck_Call_Bin_Tmp          # Echo to the screen
   echo ""                                               # Add linefeed to Screen

   cat Sck_Call_Bin_Tmp >> "Sck_Data_Bin_$datevar.bin"  # Make Summary Bin File
   echo "" >> "Sck_Data_Bin_$datevar.bin"                # Add linefeed to Summary Bin File

   od -An -tx1 -j60 --width=40 Sck_Call_Bin_Tmp >> "Sck_Data_Hex_$datevar.hex"    # Make
Hex File
   echo "" >> "Sck_Data_Hex_$datevar.hex"                # Add linefeed to Hex File

   @ CallNum = $CallNum + 1                              # Increment Call Number
end                                                      # End While Loop
```

1.6.System Tests Results

The system has been run for several weeks with the longest test being about a month. All indications are that it is working very well.

The current loss rate is ~1 epoch (one reading ~150 bytes at 30 sec intervals) on average per hour without a retransmit. If a resend of the current lost SBD message is done 10 seconds after the failure the rate goes down to 1 loss every 3 hours (8 per day).

The system is very stable and the losses are due to Iridium not accepting the send due to low signal strength. Right now the antenna is on a residential roof (Hermosa Beach, CA) in a small valley. Better results may be obtained in an open field (Boulder, CO) or in the Polar Regions.