OpenVMS Technical Journal V7

Bringing Seismic Data to the Web with OpenVMS

Dipl. Math. Bernd Ulmann

Overview

This article describes the techniques employed in gathering online seismic data from various geophysical instruments, filtering these data, and creating plots of seismic events that are made publicly available by a web server. All of these tasks are performed by a VAX-7820 running OpenVMS, which has run flawlessly for more than five years now. Excerpts from programming examples show the various interesting parts of the programs involved.

Introduction

When I first began to build seismometers many years ago, I used a conventional strip chart recorder as the main output device. This is a nearly perfect tool for the development of seismometers and geophones, because it allows direct access to the data delivered by the instruments and is readily available in most lab setups. When the instruments matured to a degree where it was possible to detect teleseismic events all over the world from my location in Germany, as well as making it possible to run the instruments unattended for extended periods of time, the strip chart recorder became a burden. At that point, it became necessary to build something which would allow access to the data being gathered by the instruments and to present plots to people located far away from the instruments.

Due to noise considerations, it was necessary to place the instruments as far away from the next house as possible. This required building a sturdy hut on a heavy concrete base plate. This hut is about 20 meters away from the house, so the first problem which needed to be solved was the data transmission over this distance. At first glance, two solutions came to mind.

The first idea involved using a commercially available analog/digital-converter system in conjunction with a PC (normally running Windows). These types of systems are available from a number of companies, such as National Instruments. The alternate idea was to develop a new analog/digital-converter that could send the converted data over a serial line and then develop a method of processing these data.

The first variant could be easily ruled out for at least two reasons. First of all, most commercially available analog/digital-converters either have a resolution too small for seismic applications (12 bits are definitely not enough) or they are far too expensive. More importantly, there is no PC with Windows or anything similar being used, since everything is done on my VAX-7820 running OpenVMS. I would never consider using anything other than my VAX system, so the second variant became the obvious choice.

The first step was to build a multi-channel analog/digital-converter with a resolution of at least 16 bits. This resulted in a design based on the ADS7807 chip from TI, which is a monolithic, high speed, 16-bit converter. This converter was preceded by some signal forming stages, incorporating the necessary low pass filters and a one-out-of-eight analog multiplexer. General control of this setup is done with a 68HC11 microcontroller, which was programmed in assembler language. This control program contains a main loop that samples each of the eight input ports, starts a conversion for each port, and builds a datagram containing two synchronization characters (`0xFF`) followed by eight 16-bit values, as shown in the following diagram:

```
-------------------------------------------------------------------------
! 0xFF ! 0xFF ! LOW_0 ! HIGH_0 ! LOW_1 ! HIGH_1 ! ... ! ... ! LOW_7 ! HIGH_7 !
-------------------------------------------------------------------------

 Synchroniza-   Channel 0    Channel 1          Channel 7
 tion bytes
```

This datagram is then transferred to a host computer system using an asynchronous serial line running at 9600 baud. This converter can run in standalone mode (that is, it has a software time base to trigger conversions in a regular fashion), or it can be used with an external clock, which allows multiple converters to be used together to increase the number of analog input channels. The current installation uses a custom-built time base based on a highly stable crystal oven.

Getting the data to and into the VAX

Because the instrument hut is about 20 meters away from the house, it was decided to buffer the serial line with two leased-line modems which were readily available from another project. (Apart from this converter there are some additional instruments, such as a precision magnetometer, etc., which also communicate by serial lines.) Using these modems, the output from the analog/digital-converter is brought into the house at a speed of 9600 baud. The serial lines are connected to a DECserver 900TM sitting in a DEChub 900, which is in the same network as the previously mentioned VAX-7820. The datagrams are purely binary in their nature, so the configuration of the LAT device to which the converter is connected is a bit tricky. On the OpenVMS side, for example, the terminal setup is as follows:

```
$ SET TERM TA57:/PERM/NOHOSTSYNC/NOWRAP/NOBROAD/NOMODEM/NOECHO-
$_/NOSCOPE/NOTTSYNC/NOLINE/EIGHT/NOHANGUP/PASTHRU/NOINTER-
$_/TYPE_AHEAD/ALTYPEAHD
```

It is necessary to use the alternate type-ahead buffer because datagrams are frequently dropped when the VAX is heavily loaded. This in turn requires setting the system parameter TTY_ALTYPAHD to a value of 4096, which turned out to be enough even for a heavily loaded system. The LAT device used to connect to the serial output line of the converter is defined as follows:

```
$ MC LATCP CREATE PORT LTA57:/APPL
$ MC LATCP SET PORT LTA57:/PORT=PORT_7/NODE=DSRV02
```

Thus, all of the subsequent communication takes place using the device `LTA57`. On the side of the DECserver, the corresponding port is configured as follows:

```
Port  7: (Remote)                    Server: DSRV02
```

```
Character Size:            8          Input Speed:              9600
Flow Control:           None          Output Speed:             9600
Parity:                 None          Signal Control:        Disabled
Stop Bits:                 1          Signal Select:  CTS-DSR-RTS-DTR

Access:                Remote          Local Switch:             None
Backwards Switch:        None          Name:                   PORT_7
Break:                Disabled         Session Limit:               4
Forwards Switch:         None          Type:                    Ansi
Default Protocol:         LAT          Default Menu:             None
Autolink Timer One:10 Two:10           Dialer Script:            None

Preferred Service: None
Authorized Groups:   0
(Current)  Groups:   0

Enabled Characteristics:
```

Reading the data from LTA57 is accomplished with a short FORTRAN program called GATHER.FOR, which gathers data sent to LTA57 by the analog/digital-converter on an hourly basis, and then writes the data into a file containing a time stamp in its name. A new file containing raw data is created every hour; meanwhile, the previous file is closed and ready for post processing. The file names look like this: CHNL_20051105_132304.RAW. The prefix CHNL is a relic from a time when not all of the eight channels were used and the numbers of the channels contained in a file were represented in the file name. The time stamp shows that this file was opened on 05-NOV-2005 at 13:23:04. The file is organized into eight columns: one column for each channel sampled. The converter samples the analog signals delivered by the seismometers with 25 samples per second. This is sufficient for most purposes because teleseismic events are very low frequency signals. The resulting file contains 25 lines of data per second.

The GATHER.FOR program converts the raw data, which is sent in binary (two's complement) from the converter, into single precision floating point numbers to facilitate post processing. These floating point values are written as clear text. While this admittedly wastes some disk space, it makes data transfer to and from other systems (sometimes even PCs) as easy as a copy command. On startup, GATHER.FOR first tests to see if the selected input device is indeed a terminal device. For example:

```
      STRUCTURE /ITMLST/
       INTEGER*2 BUFLEN, CODE
       INTEGER*4 BUFADR, RETLENADR
      END STRUCTURE
      RECORD /ITMLST/ DVI_LIST
C
      DVI_LIST.BUFLEN = 4
      DVI_LIST.CODE = DVI$_DEVCLASS
      DVI_LIST.BUFADR = %LOC (CLASS)
      DVI_LIST.RETLENADR = %LOC (CLASS_LEN)
C
      STATUS = SYS$GETDVIW ( , , INPUT_DEVICE, DVI_LIST, , , , , )
      IF ((.NOT. STATUS) .AND. (STATUS .NE. SS$_IVDEVNAM))
      1 CALL LIB$SIGNAL (%VAL (STATUS))
      IF ((STATUS .NE. SS$_IVDEVNAM) .AND. (CLASS .EQ. DC$_TERM)) THEN
...continue processing...
      ELSE
       ...abort program...
      ENDIF
```

Following these two initial steps, the GATHER.FOR program opens a new output file, reads 3,600 times 25 datagrams (an hour's worth of data), converts each datagram into eight floating point values, and writes these to the file.

Reading data from the LAT device is not completely straightforward. Because the data sent by the analog/digital-converter is strictly binary, there is no guarantee that the data part of a datagram will not contain two bytes containing 0xFF in direct succession, which would lead to confusion with the synchronization bytes. The GATHER.FOR program reads single bytes from LTA57 and waits for the first occurrence of two 0xFF-bytes. If two bytes like these are received, the program reads the following 16 bytes of raw data, converts it, and writes it into the output file. The program then checks the next two bytes, checking for the value 0xFF. If this test succeeds, the loop described is triggered again. If this test fails, the GATHER.FOR program writes an error message and skips data until it finds two bytes with the value 0xFF. As a result, the GATHER.FOR program leaves a file containing 90,000 lines of data (and an additional first line containing a time stamp), each of which has eight columns containing the data received from the seismometers.

Processing the raw data

The next step in the process is a DCL batch job called DISPLAY_HOURLY_PLOT.COM, which checks the directory to which the GATHER.FOR program writes the data files for the existence of at least two raw data files. The newest file is always the one the GATHER.FOR program currently writes to. Older files have already been closed and are ready for post processing. If there is a file to process, the DISPLAY_HOURLY_PLOT.COM batch job starts another program (about 1,600 lines of C code) called STK.

The STK program reads the contents of a raw data file into memory and applies a digital filter which is configurable by channel. This is necessary because the raw data contains lots of artifacts resulting from noise (traffic, microseismic events caused by storms and waves in oceans, and so forth). Most of these artifacts can be removed easily by using a low-pass filter that is implemented as an FFT-filter (Fast-Fourier-Transform-Filter). First the raw data of a single channel is transformed from the "normal" time domain into the frequency domain (that is, the data is split into spectral information representing the data). This spectral data (the data in the frequency domain) is then modified; in the simplest case, the amplitudes of all frequencies above the desired low-pass frequency are set to zero. Then another FFT-transformation is employed, but in the reverse direction, to transform the data from the frequency domain back into the time domain, leaving data filtered by the low-pass filter. This process is quite time consuming. On the VAX-7820, it takes several minutes to apply the low-pass filter to all eight channels.

Note

The Fast-Fourier-Transformation is accomplished using FFTW, the "Fastest Fourier Transformation in the West." The project homepage for this collection of highly-optimized routines for performing Fourier transformation is located at http://www.fftw.org. The OpenVMS port of this package is available at http://www.vaxman.de/openvms/fftw/fftw-2_1_3_vms.zip.

The filtered data forms the basis for an hourly plot containing one line of data for each channel sampled. There are many tools available to plot scientific data (gnuplot, to name one of the best known). It was tempting to use an out-of-the-box tool like this, but normally these tools are quite powerful, containing lots of techniques which do not apply to the simple problem of plotting some lines of data. Thus, it was decided to create the plots directly from within the STK program. The program generates Postscript code that represents an hourly (or, selectively, daily) plots of seismic data. After the STK program creates an hourly plot as a Postscript file, ghostscript is used to convert the file to .JPG format. For example:

```
$ GS "-sDEVICE=jpeg -sPAPERSIZE=ledger -dNOPAUSE
-sOutputFile=" destinationfile postscriptfile
```

Once the raw data file is processed, the file is copied to another disk (DISK$SEISMIC_0:[DATA.RAW]) for archiving and some additional processing, then the cycle is repeated. The DISPLAY_HOURLY_PLOT.COM batch job looks for two files in the directory used by

the GATHER.FOR program to which to write its raw data files, and the process continues as described.

Bringing the data to the web

Finally, the DISPLAY_HOURLY_PLOT.COM batch job copies the resulting .JPG file to the following directory: DISK$USER_0:[ULMANN.PUBLIC_HTML.SEISMIC_ONLINE]. This directory is visible to the Internet through a web server running on the VAX-7820 and contains 24 .JPG files per day, which represent the seismic data gathered from the instruments in the instrumentation hut.

The web server used is WASD. The reasons that I use WASD instead of the featured CSWS are as follows:

- There is no CSWS port for OpenVMS/VAX. Since I have no intention of switching to an Alpha (because the VAX-7820 is rock stable, solid, and a truly wonderful system), this alone would rule out the CSWS.
- I decline to use software that is written for UNIX and then ported to OpenVMS, when there is equivalent software already developed with OpenVMS in mind.

The decision to use WASD turned out to be a good one. It is simple to configure and maintain, and when it comes to CGI scripts (DCL as well as Perl-based scripts), the WASD running on the VAX-7820 is substantially faster than the CSWS on a DS10.

It is cumbersome to scroll through a directory listing containing hundreds of thousands of entries with a web browser. Normally, only the last few dozens of plots are of interest, so I decided to write a small DCL-based CGI script to facilitate the selection of plots to be displayed in a remote web browser. This script is called SEISMIC_ONLINE.COM and resides in DISK$USER_0:[ULMANN.PUBLIC_HTML.CGI-BIN]. When called initially, it displays a simple web page. This page contains a form with four Submit buttons that are used to select the time frame for which seismic plots should be displayed. The selections are: "One day," "10 days," "30 days," and "All," each allowing the display of a more or less limited list of relevant files.

Selecting one of these buttons causes the SEISMIC_ONLINE.COM batch job to be called as a CGI script again, but this time with a parameter. The WASD web server supports a simple mechanism to access parameters like these from within DCL by maintaining symbols, like WWW_QUERY_STRING for example. Parameters are sent in the form NAME=VALUE; therefore, the first thing to do is split the contents of WWW_QUERY_STRING on the first equal sign found:

```
$ ACTION = F$EDIT (F$ELEMENT (1, "=", WWW_QUERY_STRING), "UPCASE,
COLLAPSE")
```

In the next step, the SEISMIC_ONLINE.COM batch job determines the first date for which existing seismic plots shall be displayed:

```
$ IF ACTION .EQS. "ONE+DAY" THEN DATE = F$CVTIME ("TODAY-1-00:00:00")
$ IF ACTION .EQS. "10+DAYS" THEN DATE = F$CVTIME ("TODAY-10-00:00:00")
$ IF ACTION .EQS. "30+DAYS" THEN DATE = F$CVTIME ("TODAY-30-00:00:00")
$ IF ACTION .EQS. "ALL"     THEN DATE = "0000-00-00 00:00:00.00"
$ DATE = F$EXTRACT (0, 4, DATE) + F$EXTRACT (5, 2, DATE) + -
         F$EXTRACT (8, 2, DATE)
```

Then the directory containing the plots (DISK$USER_0[ULMANN.PUBLIC_HTML.SEISMIC_ONLINE]) is scanned, and every file satisfying the selected time restriction is displayed as a link in a table with six columns:

```
$ COLUMN_COUNTER = 0
$ FILE_LOOP:
$   FILE = F$SEARCH ("''BASE_DIRECTORY'*.JPG")
$   IF FILE .EQS. "" THEN GOTO END_LOOP
$   FILE = F$ELEMENT (0, ";", F$ELEMENT (1, "]", FILE))
$   FILE_DATE = F$ELEMENT (1, "_", FILE)
```

```
$   IF FILE_DATE .LT. DATE THEN GOTO FILE_LOOP
$   DESCRIPTION = FILE_DATE + "/" + F$ELEMENT (0, ".", F$ELEMENT (2,
"_", FILE))
$   WRITE SYS$OUTPUT "      <TD><A
HREF=""''BASE_URL''FILE''""><PRE>''DESCRIPTION'</PRE></A></TD>"
$   COLUMN_COUNTER = COLUMN_COUNTER + 1
$   IF COLUMN_COUNTER .EQ. COLUMNS
$    THEN
$     TYPE SYS$INPUT
     </TR>
     <TR>
$     COLUMN_COUNTER = 0
$   ENDIF
$ GOTO FILE_LOOP
$ END_LOOP:
```

Additional tasks and features

Another batch job called `DISPLAY_DAILY_PLOT.COM` does many of the same things as the `DISPLAY_HOURLY_PLOT.COM batch job`, except that it runs only once a day. The processing of 24 files containing 90,000 lines of data with eight columns each takes a considerable amount of CPU time on the VAX-7820 (a normal run takes about an hour). This processing chain is almost identical to the one described above.

The CGI script allowing access to daily plots is called `SEISMIC_DAILY.COM` and after processing is complete, all files of a day's processing are compressed using ZIP and stored on a large disk for later retrieval. In addition to the batch mode of the `STK` program, this tool also allows an interactive display of seismic data. Using its filter techniques, it is possible to analyze a set of data varying the low-pass filter cutoff frequencies for individual channels, scale the data displayed, and so forth. This interactive mode makes heavy use of the `MESA` graphics package available for OpenVMS on the freeware disk.

A similar set of scripts and programs exists for processing data gathered from a high resolution magnetometer, with the exception that this data is displayed in another way. The JPG picture created every hour contains the last 24 hours of data. There is no direct way to access older data via the web.

Conclusion

The system processing data gathered from various geophysical instruments, including seismometers, a magnetometer, and more, has proven to be incredibly reliable. (In all honesty, I expected nothing less, since the basis of all of this work is OpenVMS!) The VAX experiences normal uptimes of up to about 200 days before a power outage takes it offline. Unfortunately, I have no three-phase UPS, so there is no way to survive power failures, which occur about twice a year where I live. Using QIO system calls and an alternate type-ahead buffer for terminal devices, it is possible to eliminate the potential loss of data caused by a heavy load on the VAX.

For more information

Current hourly and daily plots of seismic data may be viewed on the following locations:
http://fafner.dyndns.org/~ulmann/cgi-bin/seismic_online.com
http://fafner.dyndns.org/~ulmann/cgi-bin/seismic_daily.com

Data from the magnetometer is available at
http://fafner.dyndns.org/~ulmann/magnetometer_online/

The author may be reached at ulmann@vaxman.de, and the VAX-7820 is available to the public at http://fafner.dyndns.org.