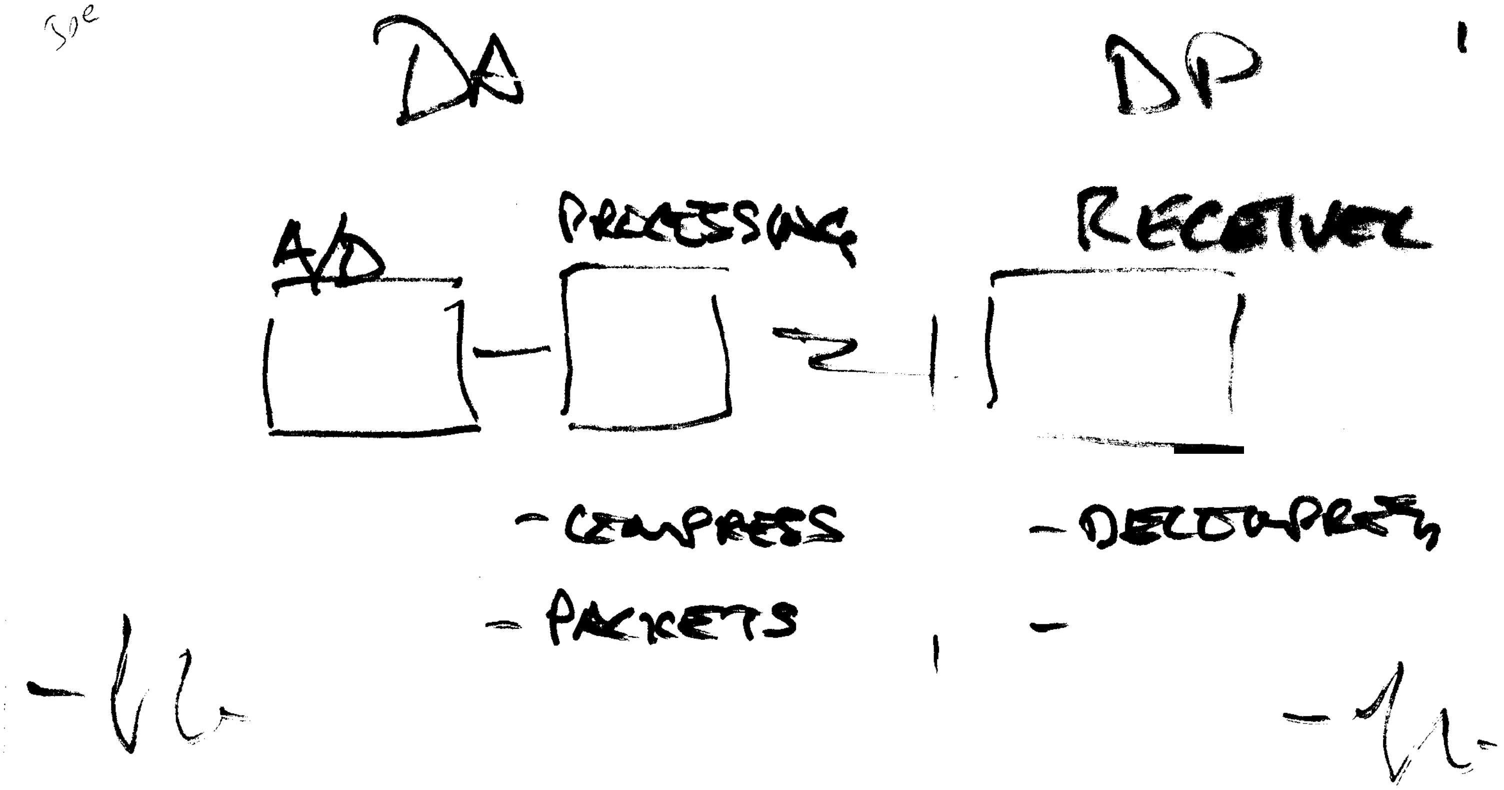
Joe Steim

Nov. 13,1997

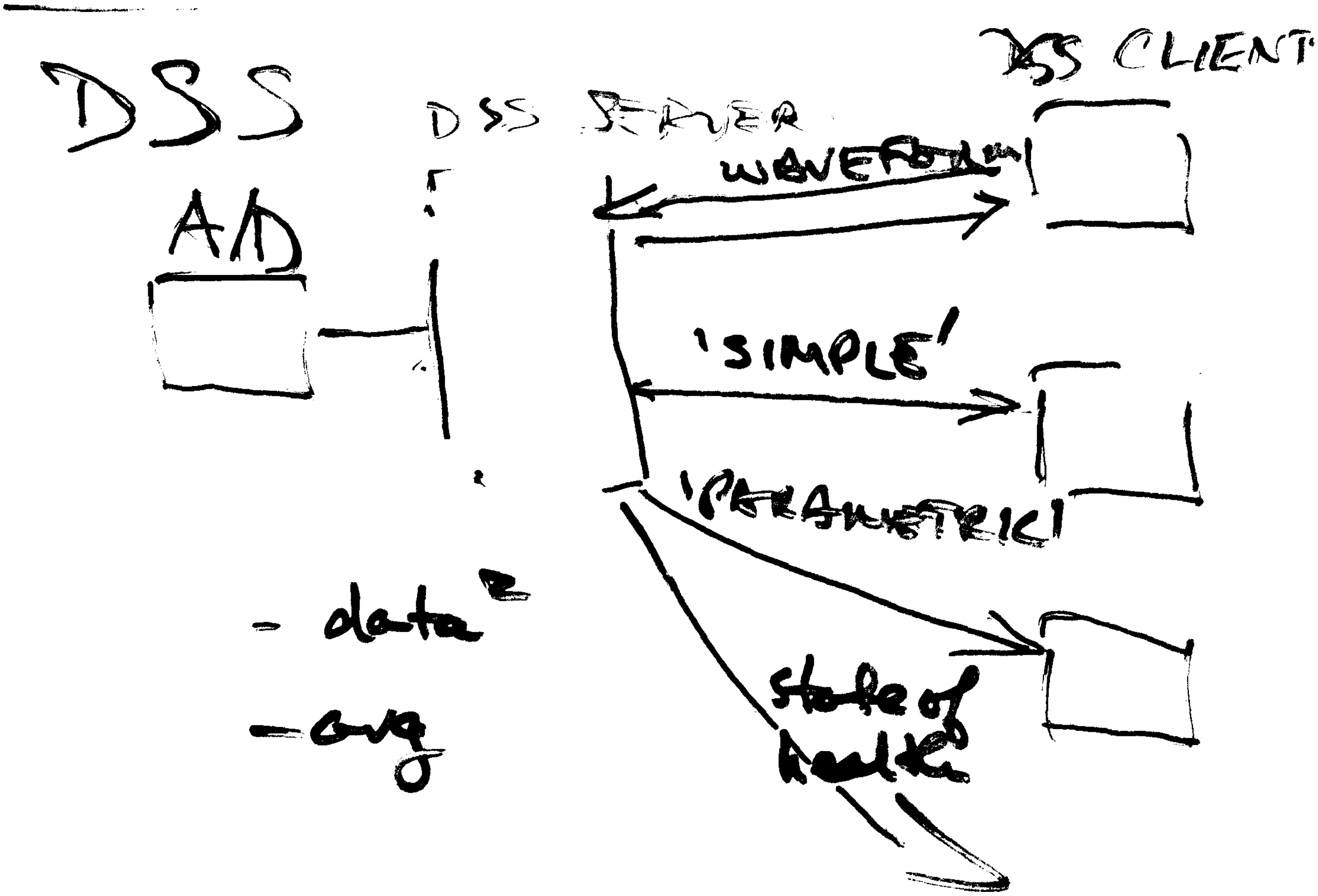
9:00 AM

30e

1



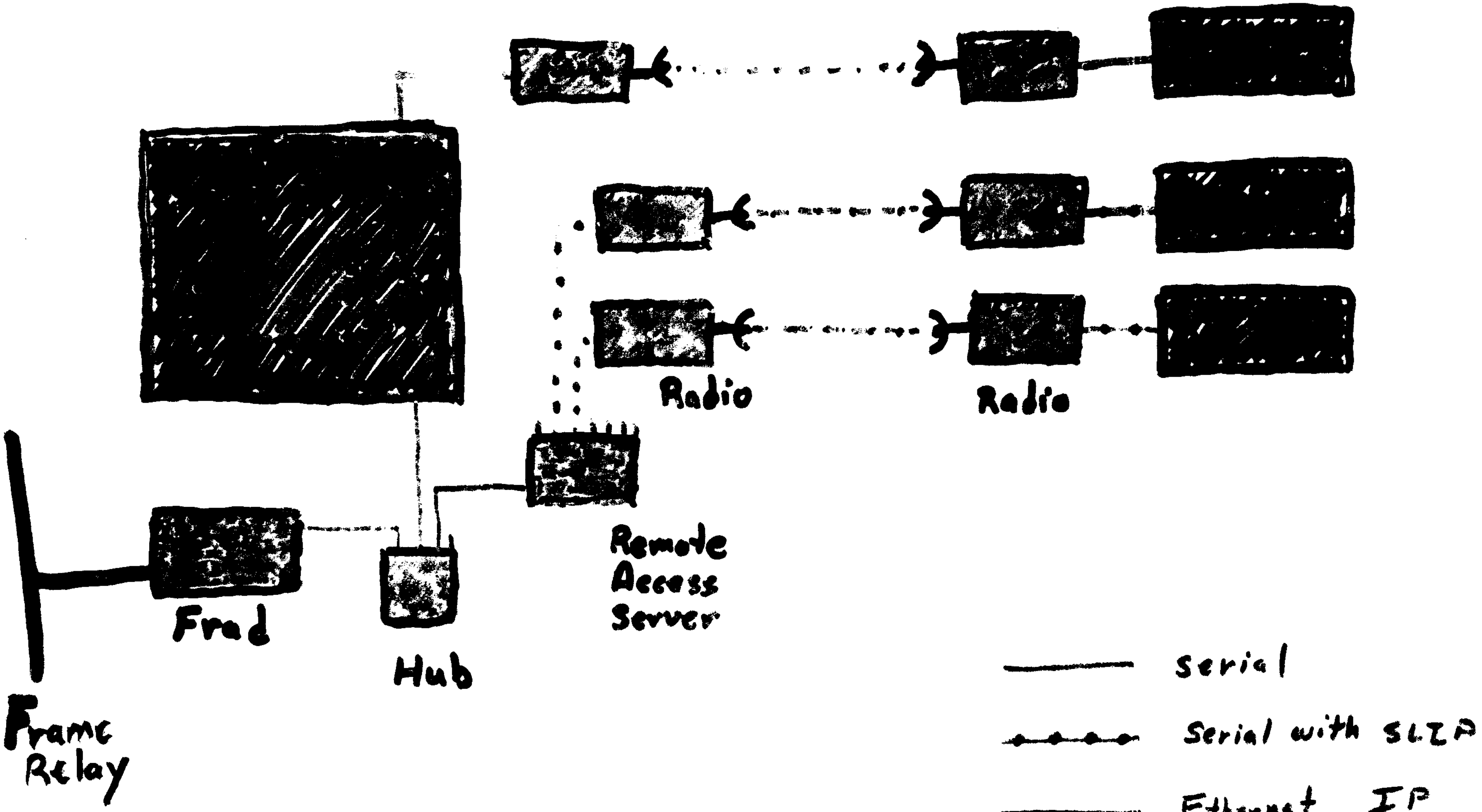
DA
DP
A/D
PROCESSING
RECEIVER
- COMPRESS
- PACKETS
- DECOMPRESS
-



DSS
DSS SERVER
DSS CLIENT
A/D
WAVEFORM
'SIMPLE'
'PARAMETRIC'
state of health
- data²
- avg

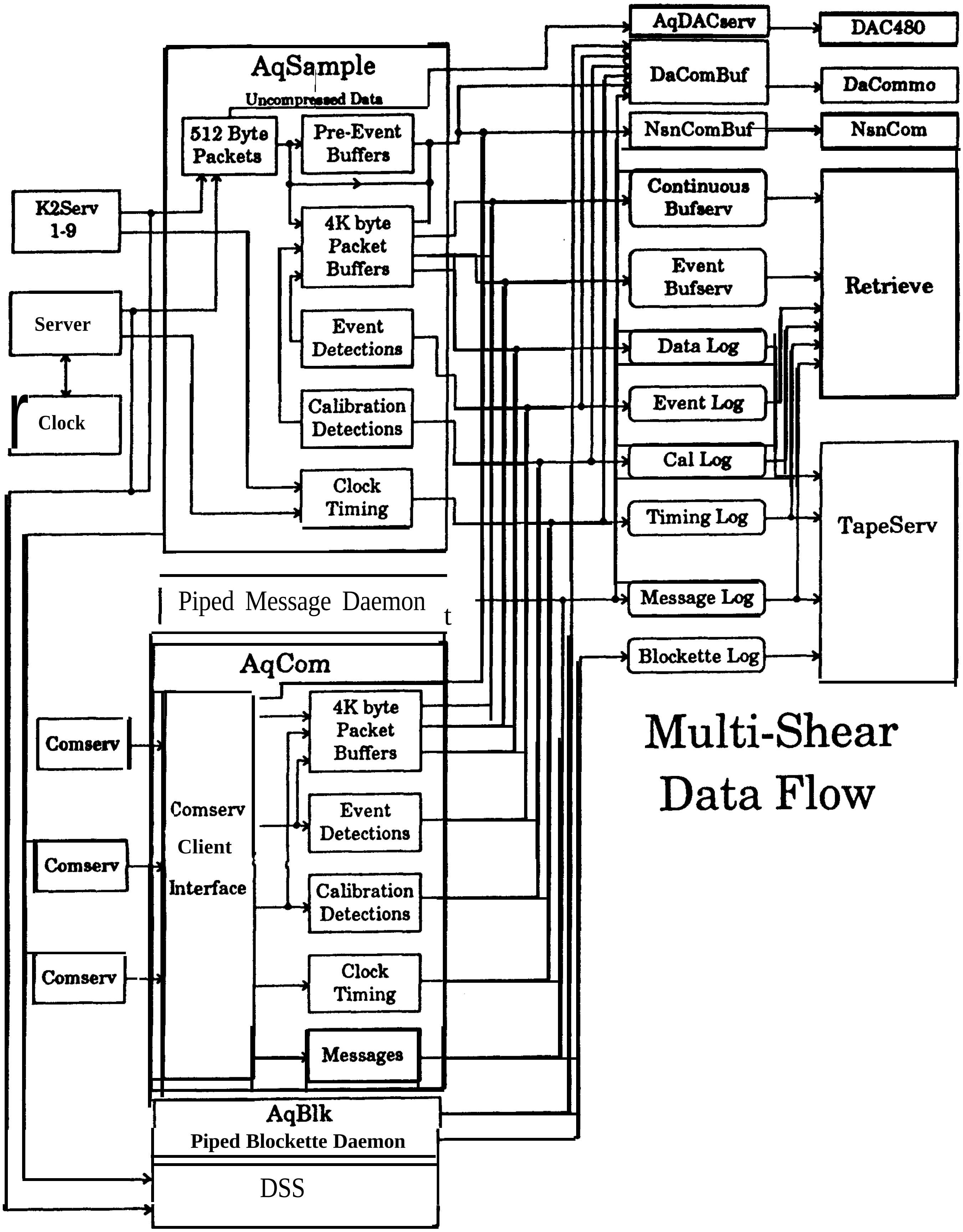
m

Hypothetical Network



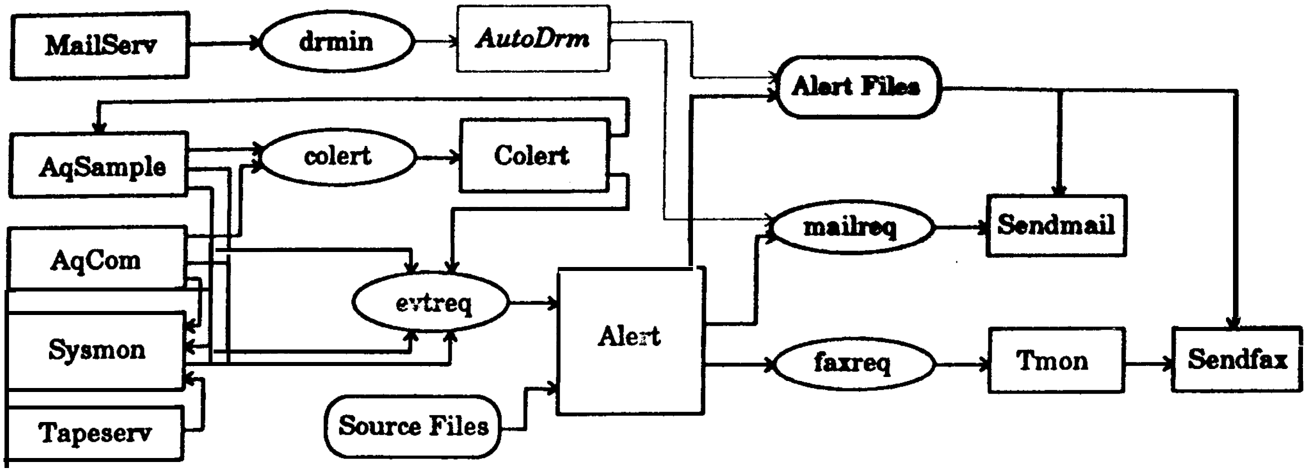
3

MULTI-SHEAR SOFTWARE CONFIGURATION
30 OCTOBER 1997, PRE-RELEASE VERSION



Multi-Shear Data Flow

This document describes Multi-Shear configuration. Additional background information can be obtained in th Quanterra SHEAR SOFTWARE CONFIGURATION GUIDE Revision B. Multi-Shear is a software package that is based o Ultra Shear and adds the capability to handle data for multiple stations. This product is still under development an capabilities and configuration are subject to change.

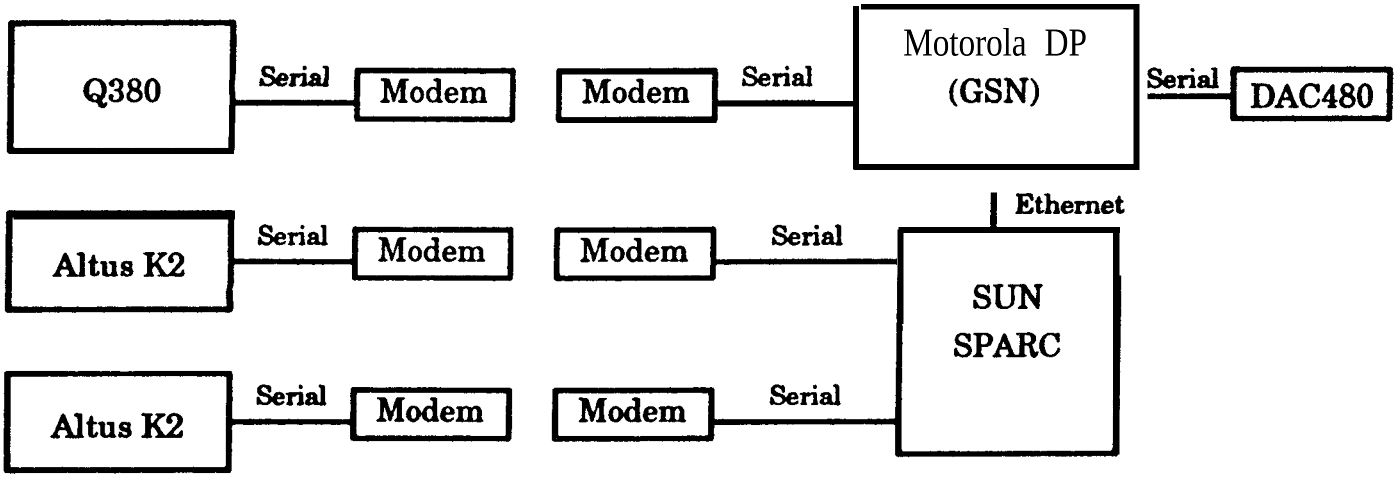


Multi-Shear Alarm & Event Flow

Configuration File

The configuration file (/r0/aqcfg) determines what data is to be recorded, how it is recorded, where it is sent, and where it comes from.

As an example of a config file, the following system is used :



The SUN is used as the final destination for the data from all three stations, as well as providing the serial port to Ethernet handling for the K2 digitizers. The DP provides tape storage and dial-up retrieval for the first station (provided by the Q380), in effect, the equivalant of a GSN DP. It also does the processing and compression of the K2 data and sends that data, and the data received from the Q380 to the SUN. The DAC480 is driven from the Z axis channels from the K2 digitizers. The ability to decompress the data received from the Q380 is not currently available.

The configuration file that runs on the DP is :

```
[aqdirs]
filtdir=/HO/MSHEAR/FILTERS temodir=/RO datadir-/HO/OAT aadir=/HO/MSHEAR/LOGS alog=AQLog mlog=MsgLog
```

The *filtdir* must contain the file FIRFilters for use by AqSample. The combination of*tempdir/mlog* is the file name of the message log used to refresh the status screen on AqShell. This is normally on Ramdisk for fast access. The combination of *aqdir/alog* is the file name of the message log that will be available for later access by retrieve.*Aqdir* is also the directory where the timing, detection, calibration, and data logs are kept. Datadir will contain the data files created by bufserv that can be accessed by retrieve.

```
[aqstationl
Station-HRVO  lat-43.5392  lon-72.3758  elev-386  naz-0  eaz=90  source-0
Station=HRV1  lat-43.5392  lon-72.3754  eleve-399  naz-0  eaz-90  source=1
Station-HRV  lat.-43.5388  lon-72.3765  elev=403 naz=0 eaz=90  source=hrv  ret-y  to=360
```

This section is used to map a data source to a station name. There is one entry per line. More than one data source can be mapped to a station, however, this can cause problems with comlinks using the current comlink protocol due to lack of routing information. *Station* is the name that the data will carry, regardless of the source.*lat*, *lon*, *elev*, *naz*, and *eaz* are only used for SAC outputfilesfrom retrieve.

Source is either a single digit from 0 to 9 in which case it indicates a local digitizer, or a station name indicating it is from a remote station received by comserv. The source station can be different from the*station* name, in which the aqcom program will change all incoming packets to conform to the*station* name. If *ret* is enabled for a station, then it indicates to retrieve that there might be data available for that station.*to* is used to change the data timeout for a remote station. If no new data is received during this period (in seconds) then the synthetic clock will stop running, which results in the station time being highlighted in Aqshell, indicating a lack of data. It defaults to 120 seconds.

```
[aqsignon]
Quanterra  Multi-Shear  Seismic  Processor  -  Example
```

This line is used for the Aqshell Banner.

```
[retsignonl
Quanterra  VBB  Data  Retrieval  System
```

This line is used for the Retrieve sign on banner.

```
[aqglobal]
pass=4.668772527287011ld+025 netid=EX  clkqflt-1200
det=500 tim=100 cal=100  data-2000  bl  k=1000  verbosity=2
chartname-/pipe/chart    chartsize-    level-3a
l1cense=12345678901234567890
```

This section sets values *that are* used by one or more programs in the acquisition package :

- *level=n* where *n* is 1, 2, 3F, 3S, or 3A. This is the default compression level if not specified in a LCQ. If not specified it is 1. 3F is Steim level 3, first differences. 3S is level 3, second differences. 3A is level 3, adaptive, which means that the first frame of each com record (normally 7 data frames) is compressed using both first and second differences, and which is best is used for the rest of the frame.
- *pass=n* where *n* is encoded password (moved here from the old ctlch section).
- *det=n* where *n* is the number of records for the detection log file. If not specified, then no detection log is used.
- *tim=n* where *n* is the number of records for the timing log file. If not specified, then no timing log is used.
- *cal=n* where *n* is the number of records for the calibration log file. If not specified, then no calibration log is used.
- *data=n* where *n* is the number of records for the data log file (data to be written to tape, using tapeserv).
- *blk=n* where *n* is the number of records for the blockette log file. If not specified, then no blockette log is used.
- *verbosity=n* sets the default verbosity level for Aqsample, can be overridden on command line.
- *blockettes=y* inserts calibration and detection blockettes into 4K records written to disk and tape by aqsample.
- *chartname=pipename* sets the default chart pipe name. In an LCQ, if you specify "chart" then this name is substituted for the name.
- *chartsize=size* sets the size of the default chart pipe. Aqsample is designed to not write to the pipe if doing so would overflow the pipe, thereby hanging aqsample.
- *timeout=n* where *n* is the number of seconds to use as a timeout in Aqshell when determining whether Aqsample has stopped. The default is 5 and the maximum is 300. This value should be increased for digitizers that communicate through modems, such as possible with K2 digitizers.

- ***netid=xx*** where *xx* is the SEED network name. The network name is now required.
- ***clkqflt=n*** where *n* is the number of seconds to use as a clock quality filter. This value is used by Aqsample and Aqcom to avoid recording frequent clock messages when the quality is moving between 40-100%. It has no effect if the clock quality should drop below 40%. It is also used by the clock program to filter out clock status dumps from GPS engines when it drops in and out of lock. The default is no filtering.
- ***license=n*** where *n* is a 20 digit number used to terminate the function of Aqsample and Aqcom after a certain date.
- ***showall=y*** enables aqshell to show data values in the status screen for channels that an not written to comlink, tape, or disk by aqsample. This is a debugging feature to allow showing data for DSS source channels.

```
[aqdetect]
ehon col
```

This line is used to redefine the names of the 32 available comm events. The default names are COMM:1 through COMM:32. You can replace these with more descriptive names by just listing the names. The first name listed renames COMM:1, the second COMM:2, etc. An individual comm event is actually the logical or of 3 flags, local, remote, and coincidence. The local flag can be set using the "Local Events" menu option in Aqshell. The remote flag can be set over a comlink connection from the DP. The coincidence is set by the "colert" program.

```
[comlinkl
levels-32 mprio=8 cprio-4 dprio-30 tprio-6 port=23400ipaddr=128.103.105.230
resend=20 pkts-500 delay=10 rce-y ws-16 resendpkts-2 notify-y udp-y
station=hrv

[commo0]
levels-32 mprio=8 cprio-4 dprio-30 tprio=6 port-23401 ipaddr=128.103.105.230
resend=20 pkts=500delay=10 rce-y ws-16 resendpkts-2 notify-y udp=y
station=hrv0

[commol]
levels=32 mprio-8 cprio=4 dprio-30 tprio=6 port-23402 ipaddr-128.103.105.230
resend=20 pkts-500 delay=10 rce-y ws=16 resendpkts-2 notify-y udp-y
station=hrvl
```

These three sections describe the comlinks for the remote station (hrv), and the two local stations (hrv0 and hrv1). The are all identical except the station and IP port. These IP ports must match up with comservs running the receiving DP (whether it be a Multi-Shear system, or Comserv running on Unix).

Note that calibrations, messages, and timeing are given a fairly low priority, while detections are given a high priority. Detections are given high priority because downstream systems may be trying to correlate detections from various sources to generate alarms.

Since these connections are used over ethernet there are a few defaults that are overridden. Window size and the resend interval are increased to allow return packet latency caused by routers and busy computers. Udp is set indicating to use UDP instead of TCP packets. This is more efficient in terms of network traffic and avoids connections being lost. Only use TCP if UDP results in excessive sequence errors. The parameters available for comlinks are:

- ***STATION=name*** *specifies* which station this dacommo is attached to. Data cannot be mixed in comlinks except if the comlink uses the data module name "anycom" and the station is "any".
- ***LEVELS=n*** where *n* is the maximum number of priority levels to be used. Must be between 1 and 100 and there is no default.
- ***MPRIO=n*** where *n* is the default priority level of message packets, this entry must be specified and be in the range of 0 to the number of levels. If 0 is specified, messages will not be transmitted.
- ***DETPRIO=n*** where *n* is the priority level of detection packets, if not specified, defaults to message priority.
- ***TIMEPRIO=n*** where *n* is the priority level of time correction packets, if not specified, defaults to message priority.
- ***CALPRIO=n*** where *n* is the priority level of calibration packets, if not specified, defaults to message priority.
- ***BLKPRIO=n*** where n is the default priority level of blockette packets, if not specified, defaults to message priority.
- ***DELAY=n*** where *n* is the polling delay in ticks for the comlink program "dacommo". This value defaults to 20 which is probably fine for 9600 baud links, 10 is recommended for 19200 or faster links as well as TCP/IP links.
- ***RESEND=n*** sets a timeout in seconds used to resend packets in the transmission window if no valid acknowledgements are received in this amount of time since the last packet was sent. The &fault is 5 seconds. If this is set to zero, then resends are disabled, which is not recommended, use a large value instead to avoid link lockup.

- *WARNAT=size* - specifies when tape changing warning messages will start appearing. The size is in megabytes. If not specified then 130MB will be used for SCSI-1 drives and 1100MB for SCSI-2 drives.
- *FULLAT=size* - specifies when a tape change will be forced. If not specified then 140MB will be used for SCSI-1 drives and 1200MB for SCSI-2 drives..
- *RETRY=count* - the number of times tapeserv will try to write to a tape if there is an error when starting a write. This is intended for Archive DAT drives which sometimes think they are offline after a week or two. The default is one.
- *THROTTLE=ticks* - this inserts sleeps (for the specified number 01 ticks) between writing SEED records to the tape. If this value is not set, it may cause the system to lose digitizer data during tape writes.
- *AUTOUNLOAD=y* - indicates that you want to do an unload command when a tape is full. This has no effect on a cartridge tape, but will eject a DAT tape.
- *RETENSION=y* - indicates that the drive has a retension command available. Cartridge tapes do, DAT tapes don't
- *OVERWRITE=y* - indicates that the program may overwrite data on this tape if there are not other available tapes and this tape has the oldest volume header.
- *TIMEOUT=count* - specifies how many times to retry a tape SCSI command if the drive returns a "busy" status. There is a one second delay between each attempt.
- *VERIFY=y* - indicates that the data written to a tape should be verified.
- *CACHE=y* - indicates that data should be first written to the cachefile on disk (or ramdisk) and then copied to the tape directly from the disk. Only Archive 2150S drives support this feature.
- APPEND-y - indicates that if a valid SEED volume header is found, that the drive should seek to end of data and check for filemarks. If no filemarks are found, then tapeserv will append data to this tape. If filemarks are found, or the APPEND flag is off, then the tape will be considered full.
- *STREAM=y* - indicates that the drive does not reliably support the normal mode of operation and that "dumb mode" should be used. Required for Archive 2525S drives.
- *SEMILOAD=y* - indicates that the drive should be put into a "semi-loaded" state between writing to logs. This is only valid on Hewlett Packard DAT drives.
- *PREVENT=y* - indicates that removal of the tape should be prevented on an active drive. Has no effect on cartridge drives.
- *COMPRESS=y* - enables compression on Hewlett Packard DAT drives. To allow this option to take effect, SW 1 must be OFF and SW2 must be ON.
- *REPORT=y* - enables writing extended status information after an error condition.
- *FATALTICKS=ticks* - where *ticks* is the maximum number of systems ticks (normally 10ms) before a SCSI command is considered hopeless. The default is 360000 ticks (or about 1 hour). If this timeout occurs it is due to a drive firmware error and the drive will be hidden from tapeserv. To use that drive again the system must be reset.

```
[bufserv]
hrv_20hz 5000 2000 hrv.bh?
hrv_1hz 5000 0  hrv.1h?
hrv_80hz 0 5000  hrv.e??
hrv_0.1hz 1000 0  hrv.v??
```

This section tells the bufservs how many 4K records to allocate for each data file and what channels to put in each file. The first parameter in each line is the root file name. For instance, using the first line as an example, the continuous bufserv will construct files with the names HRV_20HZ_C and HRV_20HZ_C_DIR as the data and directory files. The event bufserv will construct HRV_20HZ_E and HRV_20HZ_E_DIR. The 2nd parameter in the line is the number of data records for the continuous bufserv, while the 3rd parameter is the number for the event bufserv. The 4th through nth parameters are the seedname "masks". The masks are normally used to group data with the same sampling rate (and from the same station) into the same file.

When aqsample or aqcom connect to a bufserv they perform a series of requests to map seednames for the channels that will be written to a 1 byte component number and 1 byte file number. Bufserv will search to find the first file that will accept the seedname and still has available components. If no match is found aqsample/aqcom will generate a warning message and that data cannot be written to bufserv. There are a maximum of 32 components (individual seed channels) allowable in each file. If you are making frequent changes in your configuration file during installation you may want to stop and restart the bufserv's to get rid of any components in files that were requested by aqsample/aqcom but where no data was ever written. Jf there is actually data written then bufserv will find these components when it starts up, tk only way to get rid of tkm would be to delete the data files so bufserv starts with a clean slate.

This section sets values for use by the sysmon program :

- *tapes=nodrive,tapeok,tapeerr,tapenod,tapechg* - *nodrive* specifies the number of minutes that can elapse with no tape in the drive before the TAPENOD alert is generated. This is required since there will be no drive selected when the system first starts up, and when changing drives. The number for*tapeok* is the minimum number of minutes between successive generations of the TAPEOK alert signifying that the tape is running OK. Likewise*tapeerr* is the filter value for the TAPEERR alert (error on drive), *tapenod* is the filter value for the TAPENOD alert (no drive selected), and*tapechg* is the filter value for the TAPECHG alert (drive changed).
- *station=aqstart,timeout,ok,to,poor,deg* - *aqstart* specifies the number of minutes that can elapse when sysmon first starts up before it will generate an alert indicating that acquisition is not operating (TO). This is required since sysmon is started before either the Aqsample or Aqcom programs. *Timeout* is used to determine how often to make sure new data has been registered by Aqsample or Aqcom, in minutes. The*ok, to, poor,* and *deg* parameters are the filter values in minutes to avoid generating alerts that flop back and forth between clock values. Tbe alerts generated will all be in the format of <station>.alert where alert is OK (acquisition OK, clock quality >= 60%), TO (acquisition timeout, no new data within the timeout period), POOR (acquisition running, but with quality < 40%), and DEG (acquisition running, but with quality = 40%). An example would be "HRV.OK".
- *clocks=station{,station}* - determines which station's data reception times can be used to reset the OS9 system clock.
- *verbosity=n* - sets the verbosity level, default is zero. Setting to one will enable writing a message for each station in the "clocks" list indicating whether the clock was good during the sample period, and what the deviation was.
- *tolerance=n* - sets the clock tolerance level between the calculated data clock value and the OS9 clock. If the difference is above this value (in seconds), the OS9 clock will be reset. The default is 10 seconds.
- *interval=n* - sets the interval in minutes between checking clocks, the default is 60 (1 hour).
- *threshold=n* - sets the threshold above which a station is not considered stable enough to use for clock calculations, the default is 20.
- *aqstatus=station{,station}* - uses the listed stations to determine what status to display on the 4120 acquisition light. The WORST status of the stations listed is used.

```
[dacserv]
device=dac480
port=/tl
soeedcor-5
```

This section is required if running the "aqdacserv" process to produce analog output from aqsample. Generating analog output from remote station is not currently supported. This will likely be a process that obtains the data directly from one or more comserv processes, based on work done at Berkeley.

Available options are :

- *device=DAC480 | MZ8610 | XVME-505* this option is required and specifies the Digital to Analog convertor board or unit to use.
- *port=path* is required only for the DAC480 and indicates which serial port it is connected to.
- *blind=yes* does not expect any response from a DAC480, assuming it is connected via a oneway connection.
- *speedcor=n* is used with the DAC480 to adjust how fast data is transferred to the DAC480.

A word of explanation is in order about *thespeedcor* parameter. The system "tick" counter is used to generate an alarm at 1Hz rate. Should this clock be fast then the DAC480's buffer would fill up and the internal buffer would empty. Likewise, if too slow, the DAC480 will run out of data and the internal buffer will overflow. This can only be set by watching the Status display on the DAC480. There are 2560 samples for each channel on a DAC480, so each change of the display represents 256 samples. For instance, if, after 7 days the display change from averaging a "5" to a "3" that indicates 512 samples change. If the output rate is 80Hz this indicates that the incoming data rate was (512/80) 6.4 seconds slow over the 7 days, or (6.4/7) 0.914 seconds per day. The *speedcor* parameter should then be set to 0.914, or to avoid having to slop the aqdacserv program, you can use the setcorr utility to change it in real time :

```
$ setcorr 0.914 ↵
```

```
[dss]
high=2.5220277614434208D+035
middle=3.0412526690162778D+035
lowest=2.9398707340601414D+035
verbosity-1
maxcpu-10
maxbps-300
maxmem-20
```

```
[aqlcqs]
•
• Altus K2 channels
•
seed-HHZ source=hrv0.100.2
mkmhh-spww fhi-2 flo-40 iw-160 nht-4 xl-33 x2-23 x3-8 xx-13 tc-500 wa-507 av-8
detector-k2z
comlink-commo0 epri-26 wecom-y
•
seed-HHN source=hrv0.100.1
detector-k2z
comlink-commo0 epri-26 wecom-y
*
seed-HHE source=hrv0.100.3
detector-k2z
comlink-commo0 epri-26 wecom-y
*
seed-ACE source-hrv0.clk
comlink-commo0 cpri-DEF wccom-y
*
seed=HHZ source-hrvl.100.2
nkmhh-spww fhi-2 flo-40 iw=160 nht-4 xl-33 x2=23 x3=8 xx-13 tc-500 wa-507 av-8
jetector-k2z2
:omlink=commol epri=26 wecom-y
*
;eed-HHN source-hrvl.100.1
letector-k2z2
:omlink=commol epri-26 wecom-y
*
;eed=HHE source-hrvl.100.3
letector-k2z2
:omlink=commol epri-26 wecom-y

;eed=ACE source-hrvl.clk
:omlink=commol cpri-DEF wccom-y
```

Yes/No entries can be shortened to Y and N. Some parameters also allow the Available option which can be shortened to A. The first line of a LCQ is the identification line and can have the following parameters :

- ***SEED=[location-]seedname SOURCE=data-source***. *SEED specifies* the seedname and optional location for this data. ***data-source*** has different formats based on the kind of data :
  - ***station.frequency.channel*** - data from a digitizer. station can be the station name, or the digitizer number.
  - ***station.aux.channel*** - digitizer auxiliary channels, these are considered 1Hz data.
  - ***station.status.channel*** - status channels, considered 1Hz data, these are :

    1 through ***10*** will be the number of free packets in the first ten dacommo links attached to this station.
    11 is the clock drift in microseconds for this station.
    12 is the clock frequency control value for this station, if relevant.
    13 is the percentage (times 100) fullness of the timing log, -1 if tapeserv not running.
    14 is the percentage (times 100) fullness of the message log, -1 if tapeserv not running.
    15 is the percentage (times 100) fullness of the data log, -1 if tapeserv not running.
    16 is the percentage (times 100) fullness of the first tape drive, -1 if tapeserv not running or "fullat" parameter not specified for this drive.
    17 through ***19*** are the fullness of the second through fourth tape drives, if configured.
    20 is the clock quality percentage (0 - 100).
  - ***station.clk*** - digitizer clock timing events.
  - ***station.seedname/filter*** - is used to take data recorded at a higher rate and filter it down to a lower frequency. The station and seedname must already be defined in the lcqs before this one. Filter is the name of the FIR filter to use. The resulting recording frequency will be the frequency of the source lcq divided by the decimation of the filter.

Subsequent lines may be one of three types :

- Detectors, either Murdock-Hutt, or Threshold. The entire line is used for parameters of the detector.
- Comlinks, the entire line is used for parameters.

- ***MAMP=n*** sets the amplitude in DAC counts for timemarks and defaults to 40. This must be specified and set to 0 for use with DAC480.
- ***DACON=yes*** enables analog output on this channel, this option should be last in the series of analog output parameters to guarantee they are all processed.
- ***FIRFIX=n*** sets a multiplier that is used when FIR filtering down data from a higher frequency to a lower frequency. It defaults to 1.0.

```
[aqcontrol]
k2z-hrv0.hhz:spww
k2z2-hrv1.hhz:spww
```

The section defines the control detectors that are referenced in the [aqlcqs] section. The control detector allow logical combinations of previously defined Murdock-Hutt, Threshold, and Comm detectors. The format is either the actual Comm detector name or in the form : station.[location-].seedname:detector_name. Detector-name may also be "CALON" indicating the current calibration status on that digitizer.

Detectors may be combined using the following logical operators :

- ***&*** Logical AND - Precedence Level 2
- **+** Logical OR - Precedence Level 3 (lowest)
- ***/*** Logical NOT - Precedence Level 1 (highest)
- • Logical Exclusive OR - Precedence Level 3 (lowest)
- *(. . . .)* Parenthesis can be used to modify the precedence ordering.

For instance, "xyz=*(hrv.bhz:spww+hrv.hhz:vspww)&hrv.bhn:spww*" would only trigger event recording if either the first two detectors are on, and the third detector is on. "xyz" is the control detector name. All of the different types combined so that "*xyz=hrv.bhz:remote+hrv.bhz:calon+comm:5+hrv.bhz:spww* " is valid. The control detector specification can also be following by an optional comma and a Y or N. If Y is specified, such as "xyz=hrv.bhz:spww,y" then a log message will be generated when the control detector goes on or off.

AQCOM Configuration

```
[aqcom]
source=hrv.bhz
wcbuf-y wcseq-Y
comlink wccom-y wecom-y cpri=25 epri-27
*
source-hrv.bhn
wcbuf-y wcseq-y
comlink wccom-y wecom-y cpri=25 epri=27
•
source=hrv.bhe
wcbuf-y wcseq-y
comlink wccom-y wecom-y cpri-25 epri-27
*
source=hrv.ehz
webuf-y weseq-y
comlink wccom-y wecom-y cpri-14 epri-15
•
source=hrv.ehn
webuf-y weseq-y
comlink wccom-y wecom=y cpri-14 epri=15
*
source=hrv.ehe
webuf=y weseq-y
comlink wccom-y wecom=y cpri-14 epri=15
•
source=hrv.log
wcseq=y
comlink wccom-y cpri-DEF
•
source-hrv.ace
wcseq=y
comlink wccom-y cpri=DEF
*
```

The [aqcom] section is very similar to [aqlcqs] :

- *SEED* is. not normally required, if not specified the location and seedname is obtained from the source parameter.
- ***SOURCE=station.[location-seedname]*** **determines which comserv to get the data from (possibly remapped in the** [aqstation] section. If remapping occurs (STATION <> SOURCE in [aqstation]) then **aqcom is the one that** changes stationnames.
- ***COMLINK, CPRI,*** *EPRZ, WCCOM,* ***WECOM,*** *DETP, CALP, WCSEQ,* ***WESEQ,*** *WCBUF,* ***WEBUF, WDET,*** *WCAL, GAP,* ***4KFR,*** and *FLUSH* are also supported.

Note that aqcom is handling messages instead of pmsgd, so we need a LCQ for the message packets (LOG).

AQBLK Configuration

```
[aqblk]
source-hrv.gps
wcseq-y
comlink wccom-y cpri-8
```

The [aqblk] section is a very small subset of the [aqlcqs] section :

- ***source=station.[location-]seedname*** allows received piped packets with the designated **station/location/seedname** to be processed.
- ***wcseq=y*** enables combining received blockettes into 4K records and to be written to **the blockette** bg. **The blockette** log can be written to tape and downloaded using **the** retrieve program. You can also we **(A)vailable** to **setup, but** not enable recording.
- ***Comlink[=modname]*** indicates to allow received blockettes into 512 byte records for transmission by **dacommo**. wccom and cpri have their usual meanings.

```
echo . . . starting continuous data buffer server with profiler
shell 'bufserv -d=/h0/DAT -q=C -s=2000 -1-3 -m=100 ^2000 >>-/pipe/log.BufC <>/nil&"
sleep -S 3
bufwait bufsvcq_C 60
•
echo . . . starting event data buffer serverwithprofiler
shell 'bufserv -d=/h0/DAT -q=E -s-20000 -1=3 -m=50 ^2000 >>-/pipe/log.BufE <>/nil&"
sleep -S 3
bufwait bufsvcq_E 60
```

This set of commands starts up the two buffer servers, the first for continuous data, the second for event data. Bufserv has as its command line options :

- ***-c=file*** specifies the name of the configuration file, defaults to /r0/aqcfg.
- ***-d=dir*** specifies the directory where the data and index files are to be created.
- ***-q=x*** where x is C for continuous and E is for event.
- ***-r=n*** specifies the number of queued service requests and defaults to 20.
- ***-s=n*** specifies the number of segment descriptors to allocate., defaults to 1000.
- ***-l=n*** specifies the debug level, defaults to 0.
- ***-m=n*** specifies the number of seed channels allowed in it's map, defaults to 200.

Bufwait is run after starting a buffer server to wait for it to initialize and has as it's first parameter the name of the server module and the second parameter is the maximum number of seconds to wait.

```
echo . . .starting K2 servers'
servk2 <>/nil >>-/pipe/log.hrv0.servk2 ^1500 &
sleep -S 2
servk2 1 <>/nil >>-/pipe/log.hrvl.servk2 ^1500 8
sleeo -S 2
```

The server number is zero if not specified as the first command line parameter. Following the optional server number can be the name of the configuration file, which defaults to /r0/aqcfg. Only the K2 server has this server number parameter, all other servers (which use the name "server") have only the optional configuration file override option.

```
echo . . . . starting remote command manager'
rcman -v-2 >>-/pipe/log.rcman <>/nil &
sleeo -S 2
```

Rcman has the following command line options :

- ***-v=n*** sets the verbosity level, default is zero.
- ***-l*** will lockout calibrations during event detections.

```
echo "...starting comlink process for remote 0680"
dacommo-v=l >>-/pipe/log.hrv.dacommo <>/nil 8
sleep -S 5
echo "...starting comlink processes for K2s"
dacommo -v=l -m=commo0 >>-/pipe/log.hrv0.dacommo <>/nil &
sleep -S 5
dacommo -v=l -m=commol >>-/pipe/log.hrvl.dacommo <>/nil &
sleep -S 5
echo "...starting comlink process for Log'
dacommo -v=l -m-logcommo >>-/pipe/log.logcommo <>/nil &
sleep -S 5
echo "...starting comlink process for Any .
dacommo -v-l -m=anycom >>-/pipe/log.any.commo <>/nil &
sleep -s 5
```

Dacommo has the following command line options :

- ***-c=file*** specifies the configuration file, default is /r0/aqcfg.
- ***-v=n*** specifies the verbosity level.
- ***-m=modname*** specifies the data module name, default is "dacombuf".

o *<!--EMAIL=address-->* indicates that this is an Email template and specifies the recipient's Email address.

- *<!--RETRY=retrystring-->* is used to override the "faxretry" parameter in the config file for this recipient.
- *<!--PRI=priority-->* where priority is used to sort the recipients in the alert program. This doesn't have much effect on Email messages because they are all sent in a batch once the SMTP server is contacted. For faxes this will determine the order that the initial attempt to send a fax is done when a detection occurs. Since it can take a minute or more to send a fax, the person with the lowest priority may see a significant delay before they receive their fax. Higher priorities are sent first, the default priority is zero.
- *<!--DET=detectorlist-->* This determines under what conditions a recipient will be sent a fax or Email. The detectorlist is a series of detector names, separated by spaces. To specify an event detector you use the station name, a period, the SEED channel name, a colon, and the detector name, such as HRV.BHZ:SPWW. Coincidence detector names can also be used, and the SYSMON program can also generate alerts. Note that the detector list is limited to 80 characters, however, multiple Detector tags can be used. A particular type of detection can be accepted from any station by using an asterisk "*" instead of the station name.

FAX Template Structure

A typical fax template :

```
<!--FAX-Txxxxxxx-->  <!--RETRY=30to200--> <!--PRI=2-->
<!--FINE--> <!--DET=CO1 HRV.BHZ:SPWW ●  .TO-->
<BODY >
<P>
<HR>
<IMG SRC="detalert.qub">
<HR>
FROM: Station RAND<BR>TO: Bob Reimiller<BR>
<HR>
($INCLUDE)
<HR>
<IMG SRC="seisconn.qub">
</BODY>
```

The "xxxxxxx" of course was replaced with the actual telephone number, and the T indicates tone dialing. The retry in the file overrides the default retry time-out. The priority is 2. The fax is to be sent in fine resolution. The following detector are defined : CO1, HRV.BHZ:SPWW, and the acquisition timeout alert on any station generated by Sysmon. The <IMG SRC="detalert.qub"> puts in the "DETECTION ALERT" file, and the <IMG SRC="seisconn.qub"> puts in the "QUANTERRA YOUR SEISMIC CONNECTION" file. The ($INCLUDE) line is replaced by the "alert" program with the details of the detection.

If you wanted to identify the sender of the fax as something other than it's phone number, and your fax machine supports it, you could have something like <!--ID=STATION HRV--> anywhere before <BODY>.

Email Template Structure

A typical Email template :

```
<!--email=steim@quanterra.harvard.edu--> <!--PRI=2-->
<!--DET=CO1 HRV.BHZ:SPWW *.TO-->
<BODY>
TO: steim@quanterra.harvard.edu<BR>
SUBJECT: Detection Alert<BR>
<HR>
($INCLUDE)
<HR>
</BODY>
```

This file has the same detector list and the same priority as the fax template. Retry is not specified for Email templates because the retry is for contacting the SMTP server, not for individual recipients. Note that the first line of the body needs to be in the form "TO: name", or else your Email will say "Apparently To", just a quirk of SMTP. The second line needs to be the "SUBJECT: description" line, or else your Email won't have a subject. If you wanted a return address that is someone

Opaque Data Blockette Piped Record Format

The Aqblk program was created to allow GPS or other data to be written as a series of blockettes. Aqblk does not care about the contents of the blockettes, other than it be SEED opaque blockette, so that it is not specfic to handling GPS data, but for the purpose of this docuement, that is the intended use. The data sent to the pipe from the user's GPS receiver communications program consist of a 20 byte header, the actual blockette, and a 2 byte checksum. The pipe name will be in the form of "/pipe/blk.<station>.gps", such as "/pipe/blk.hrv.gps". The header is :

Byte Offset	Contents
0-1	SEED Location, normally spaces
2-4	SEED Channel, such as "GPS"
5	Flush Flag : Bit 0 TRUE indicates that comlink output should be flushed after this blockette. Bit 1 TRUE indicates that blockette log output should be flushed after this blockette.
6-7	16 Bit integer blockette length (not counting this header or checksum)
8-17	Time stamp for blockette, standard "BTIME" SEED format.
18-19	Reserved, must be zero.

This header is followed by the blockette, which is a maximum of 456 bytes in length, and then a two byte checksum which is calculated by doing a 16 bit addition over the length of the header and blockette. If the blockette length is odd, then the checksum is calculated based on a zero pad byte.

Our software will build data records using the standard 48 byte data record header followed by a blockette 1000, followed by the blockettes, up to the appropriate 512/4K byte limit, or until the "flush" flag is encountered.

So that we can assemble data records without knowing the format of individual records we insist that all blockettes have the following format as the first 14 bytes (Variable length opaque data blockette) :

Byte Offset	Contents
0-1	Blockette Type = 2000
2-3	Next blockette's byte number. Always zero when you write the blockette to the pipe.
4-5	Length of blockette in bytes.
6-7	Offset from start of this blockette to start of data.
8-11	Record Number
12	Word Order. 1 = Motorola Format
13	Data Flags. Bit 1 set indicates that a new record is required.

The command line for the Aqblk program is :

```
aqblk [c=configfile] [-v=n] >>logdevice_or_file
```

The config file defaults to /r0/aqcfg. n is the verbosity level. Error/status messages are written to the standard error path.

DSS

All DSS (Data Subscription Service) packets use the QSP (Quanterra Serial Protocol) header structure (QDP). The basic transport mechanism is UDP/IP with the QDP header providing a CRC and sequencing control information.

QDP Format (12 bytes)

CRC		
Command	Version	Length
Sequence #	ACK #	Spare (Zero)

- CRC is the standard Quanterra CRC value, and includes this header (not including itself) and any data may follow.
- Command is a Quanterra specific command, indicating the content of any data that follows.
- Version is used to accommodate updates to the protocol.
- Length is the actual length of the data, and may be between 0 and 512 and must be even.
- Sequence number is a modulus 256 record number to providing sequencing of datagrams. This is normally set by the client so that it can track responses from the server. Timeout resets can leave this field zero, since there is no response. The server leaves this field zero for data packets.
- ACK number is used to acknowledge previously received datagrams. The server sets this to the sequence number of the client request so the client can associate responses with requests. The server leaves this field zero for Data packets.

DATA Format

The format of the DATA area is dependent on the QDP Command byte :

DSS_REQ ($F0)

Used to request a data item be added from the list. Each request has the following header :

Data Identification Word	Priority
Reporting Interval (seconds)	

The data identification word is a unique word generated by the client that will be attached to the data so the client can identify it. This word is also used when deleting requests. Priority is local to this client, with higher priority requests being sent first. The reporting interval indicates how often the DSS server sends this data to the client. For simple data and GPS requests, a value is picked out of the data stream at this interval, no filtering is done. For other requests, multiple samples at the source data rate will be processed for that many seconds before reporting. For instance, if the source data is 20Hz, and the reporting interval is 5 seconds, there will be 100 data values processed. Selecting an interval of zero results in one report being generated and the request removed.

Following the header is the data request. The format of each request is still under development, but the following have been defined so far :

$01=Simple Data	Report Flag	Station (first 2 characters)
Station (second 2 characters)		Location
Seedname		Format

Data is sampled from the data stream at the reporting interval. Bits 0-2 of format are used to determine data format :
1=16 Bit Integer
3=32 Bit Integer
4=IEEE floating point (32 bits)
5=IEEE double precision floating point (64 bits)

The Report Flag determines what action the server will take when the requested data is not available, such as when acquisition is not operating. A value of zero indicates that the server merely skips this report. Values between 1 and 249

JWL
15

DSS_REF ($FE)

Server refuses request

Refusal Code

Refusal Codes :

1	REF_TO	Timeout
2	REF_IDS	Invalid Data Source
3	REF_SEI	Seconds exceeds Interval
4	REF_INP	Invalid Password
5	REF_URC	Unregistered Client
6	REF_TMR	Too many requests
7	REF_URQ	Unknown Request
8	REF_DUP	Duplicate Data_ID
9	REF_OOM	Out of Memory
10	REF_IDF	Invalid Data Format

DSS_DAT ($FD)

A data packet consists of one or more records. The offset to the next entry is relative to the start of actual data record (past the QDP header). An offset of zero indicates the last record. The offset is the lowest 9 bits (OFF_MASK=$1FF) of the "Status and offset to next entry" field. The upper bits are status, such as OFF_DNA ($8000).

Simple Data entries have the following format :

Data Identification Word	Status and offset to next entry
Time field (IEEE double precision) - seconds since 1984	
Value (16-64 bits)	

GPS Location Data entries have the following format :

Data Identification Word	Status and offset to next entry
Latitude (-xx.xxxxxx degrees) 10 bytes	
Longitude (-xxx.xxxxxx degrees) 11 bytes	
Elevation (-xxxx.x meters) 7 bytes	

Dead Channel requests have the following format :

Data Identification Word	Status and offset to next entry
Number of Dead Channels	

Detection Count requests have the following format :

Data Identification Word	Status and offset to next entry
Number of Detections	

Record Count requests have the following format :

Data Identification Word	Status and offset to next entry
Number of Records	

Time since Boot requests have the following format :

Data Identification Word	Status and offset to next entry
Seconds since reboot (IEEE double precision)	

Min,Max, & Avg requests have the following format :

Data Identification Word	Status and offset to next entry
Time field (IEEE double precision) - Seconds since 1984	
Minimum Value (16-64 bits)	
Maximum Value (16-64 bits)	
Average Value (16-64 bits)	

$\int Val^2$ & Avg requests have the following format :

Data Identification Word	Status and offset to next entry
Time Field (IEEE double precision) - Seconds since 1984	
Sum of Squares (16-64 bits)	
Average Value (16-64 bits)	

DSS_ACK ($FC)

This acknowledges packets from the client when there is no error or return value, these include DSS_REQ and DSS_DEL.

DSS_PRG ($FB)

This packet tells a user that a report has been purged.

Data Identification Word	Purge Code

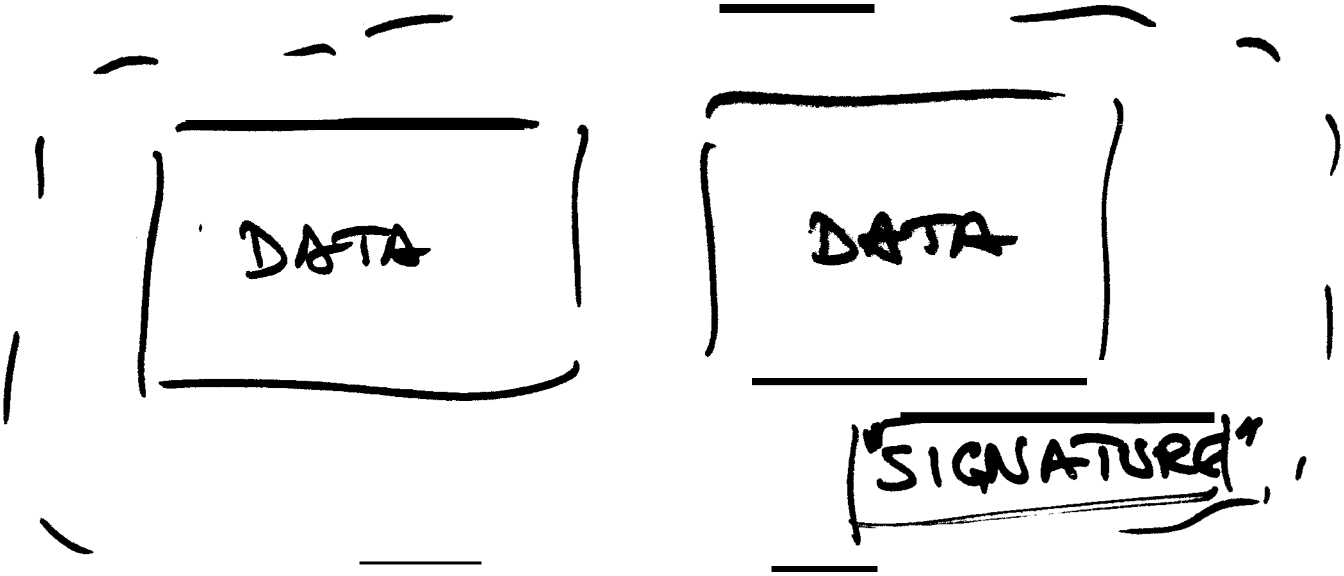
Purge Codes :

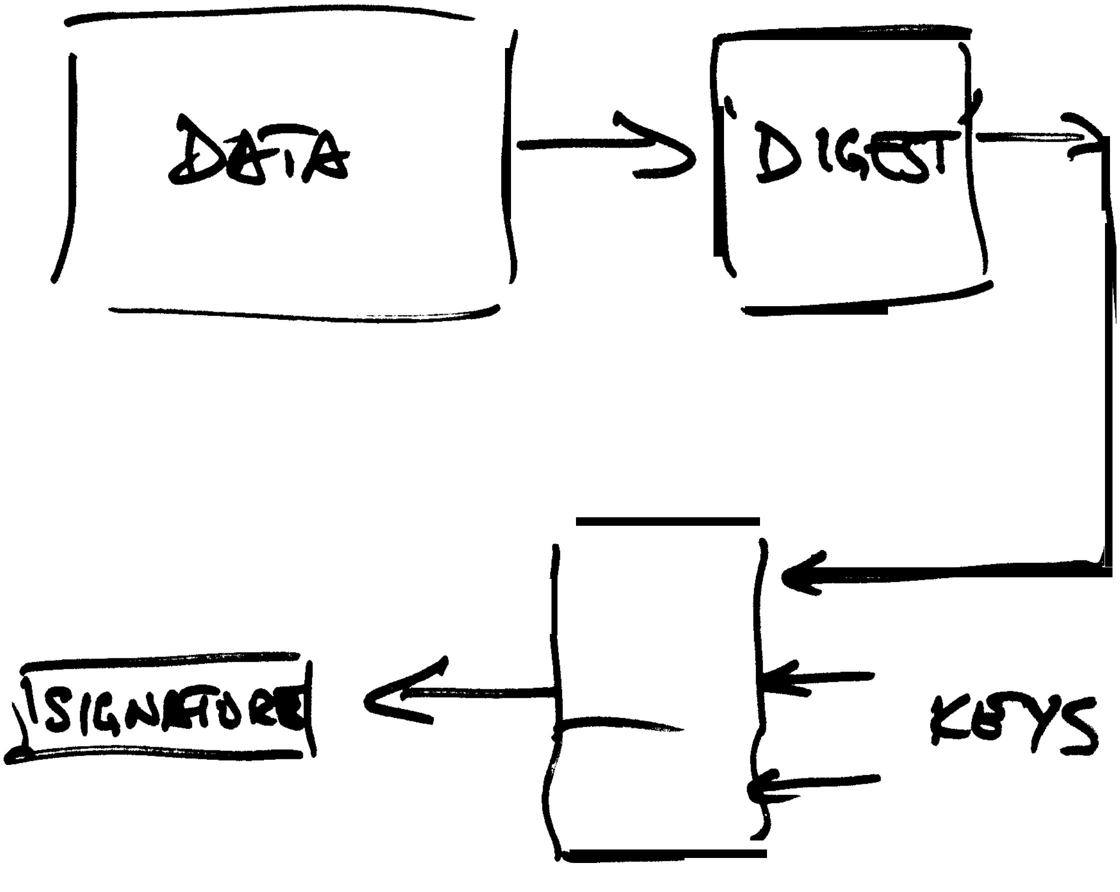
1	PRG_ESL	Excessive Server Loading (DSS is using too much CPU time)
2	PRG_BPS	Bytes Per Second (Too much data being transmitted by DSS)

Adding New Reports

Request to add new reports to the DSS server need to be submitted to Quanterra. In order for us to validate our implementation we must have some example input data along with the expected output. Any parameters that DSS needs must be passed as part of the DSS_REQ packet. The format and purpose of these parameters must be explained in full so we can document them.

We are currently looking into adding Hiroo Kanamori's Peak Acceleration algorithim, pending meeting the above requirements


DATA
DATA
"SIGNATURE"


DATA
DIGEST
SIGNATURE
KEYS

Doug Neuhauser

Nov. 13,1997

AM Session

doug neuhauser

Proposed Blockette for polynomial representation of a non-linear sensor.

Bob Uhrhammer, UC Berkeley Seismological Laboratory

Version: 1.0
Date: 1997/07/24

Name : Response (Polynomial) Blockette
Blockette Type: ??? (to be determined)

Use this blockette to characterize the response of a non-linear sensor.

Note	Field Name	Type	Length	Mask or Flags
1	Blockette Type	D	3	"###"
2	Length of Blockette	D	4	"####"
3	Transfer Function Type	P	1	[U]
4	Stage Sequence Number	D	2	"##"
5	Stage Signal Input Units	D	3	"###"
6	Stage Signal Output Units	D	3	"###"
	Polynomial Approximation Type	A	1	[U]
8	Lower Bound of Approximation	D	4	"-#.#####E-##"
9	Upper Bound of Approximation	D	4	"-#. #####E-##"
10	Maximum Absolute Error	D	4	"-#.#####E-##"
11	Number of Polynomial Coefficients	D	3	"###"
	REPEAT fields 12-13 for each polynomial coefficient.			
12	Polynomial Coefficient	F	12	"-#.#####E-##"
13	Polynomial Coefficient Error	F	12	"-#. #####E-##"

Notes for Fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A single character "P" describing the type of stage.

4 The identifying number of this stage.

5 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter.

6 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stages output signal.

7 A single character describing the type of polynomial approximation (this field is mandatory):

C - Chebychev

pn(x)=a0*T0(x)+a1*T1(x)+a2*T2(x)+...+an*Tn(x)

L - Legendre

pn(x)=a0+a1*L1(x)+a2*L2(x)+...+an*Ln(x)

M - MacLaurin

pn(x)=a0+a1*x+a2*x^2+...+an*x^n

8 This field is mandatory and it is the lower bound (a) for which the polynomial approximation is valid.

9 This field is mandatory and it is the upper bound (b) for which the

polynomial approximation is valid. This upper bound (b), along with the lower bound (a) is used in the linear transformation:

```
x = (2*s-a-b)/(b-a)
```

to map the arbitrary finite interval a <= s <= b into -1 <= x <= 1.

10 The maximum absolute error of the polynomial approximation. Put 0.0 if the value is unknown or actually zero.

11 The number of coefficients that follow in the polynomial approximation. The polynomial coefficients are given lowest order first and the number of coefficients is one more than the degree of the polynomial.

12 The value of the polynomial coefficient.

13 The error for field 12. Put 0.0 here if the value is unknown or actually zero. This error should be listed as a positive value, but represent a +/- error (ie 2 standard deviations).

Example:

Polynomial representation of the temperature response of a thermistor.

A thermistor temperature sensor may have a response which can be represented by a sixth order McLaurin series approximation with a maximum error of 0.21 degrees Celsius over a temperature range of -12.4 to +82.4 degrees Celsius.

The calibration data are:

Counts	Temperature
100	82.4
200	59.2
300	45.6
400	35.0
500	26.4
600	17.8
700	9.6
800	-0.8
900	-12.4

The McLaurin polynomial representation is:

Coefficient	Value	Error
a0	2.61846E+01	6.59725E-02
a1	-3.37632E+01	2.16967E-01
a2	6.65491E+00	8.16682E-01
a3	-7.73147E+00	7.77808E-01
a4	-7.85504E+00	2.14852E+00
a5	-5.90769E+00	5.95239E-01
a6	1.00124E+01	1.41479E+00

with bounds of:

Bound	Value
Lower	1.00000E+02
Upper	9.00000E+02

The Polynomial Blockette representation is:

Field Name	Type	Length	Value
Blockette Type	D	3	"???"
Length of Blockette	D	4	" 200"
Transfer Function Type	P	1	"P"
Stage Sequence Number	D	2	“??”
Stage Signal Input Units	D	3	“???”
Stage Signal Output Units	D	3	“???”
Polynomial Approximation Type	A	1	"M"
Lower Bound of Approximation	D	4	" 1.00000E+02"
Upper Bound of Approximation	D	4	" 9.00000E+02"
Maximum Absolute Error	D	4	" 2.15385E-01"
Number of Polynomial Coefficients	D	3	" 7 "
a0 Polynomial Coefficient	F	12	" 2.61846E+01"
a0 Polynomial Coefficient Error	F	12	" 6.59725E-02"
a1 Polynomial Coefficient	F	12	"-3.37632E+01"
a1 Polynomial Coefficient Error	F	12	" 2.16967E-01"
a2 Polynomial Coefficient	F	12	" 6.65491E+00"
a2 Polynomial Coefficient Error	F	12	" 8.16682E-01"
a3 Polynomial Coefficient	F	12	"-7.73147E+00"
a3 Polynomial Coefficient Error	F	12	" 7.77808E-01"
a4 Polynomial Coefficient	F	12	"-7.85504E+00"
a4 Polynomial Coefficient Error	F	12	" 2.14852E+00"
a5 Polynomial Coefficient	F	12	"-5.90769E+00"
a5 Polynomial Coefficient Error	F	12	" 5.95239E-01"
a6 Polynomial Coefficient	F	12	" 1.00124E+01"
a6 Polynomial Coefficient Error	F	12	" 1.41479E+00"

The new variable length blockette contains an opaque byte stream or variable length opaque data record.

Blockette 2000: Variable length opaque data blockette.

Note	Field Name	Type	Length	Offset
1	**Blockette type - 2000**	B	2	0
2	**Next blockette's byte offset**	B	2	2
3	**Total blockette length in bytes**	B	2	4
4	Offset to Opaque Data	B	2	6
5	Record number	B	4	8
6	Data Word order	B	1	12
7	Opaque Data flags	B	1	13
8	Number of Opaque Header fields	B	1	14
9	Opaque Data Header fields	V	V	15
a	Record type			
b	Vendor type			
d	Model type			
	Software			
e	Firmware			
10	Opaque Data . . .	Opaque		

Notes for fields   * = indicates mandatory information

1 * UWORD : Blockette type (2000). Opaque Data blockette.

2 * UWORD : Byte number of next blockette (Calculate this as the byte offset from the biginning of the logical record - including the fixed section of the data header; use 0 if no more blockettes will follow.)

3 * UWORD : Blockette length. The total number of bytes in this blockette, including the 6 bytes of header. The only restriction is that the blockette must fit within a single SEED data record for the channel. Otherwise, the blockette must be partitioned into multiple blockettes.

4 * UWORD : Offset to Opaque Data.
Byte offset from beginning of blockette to Opaque Data.

5 * ULONG: Record number.
The record number may be used for sequence identification of stream, record, or file oriented data. If a record is partitioned into multiple opaque blockettes, each blockette containing a portion of the record should contain the identical record number. It is strongly recommended that record number be used to aid in the detection of missing data aid to process of merging data from different telemetry streams. Use 0 if data is not record oriented, or if record number is not required.

6 * UBYTE: Word order of binary opaque data. See field 4 of blockette 1000, and fields 11 and 12 of blockette 50.
0 = little endian (VAX or 80x86 byte order).
1 = big endian (68000 or SPARC byte order).

7 * UBYTE: Opaque Data flags.
[bit 0] Opaque blockette orientation.
0 = record oriented.
1 = stream oriented.
[bit 1] Packaging bit.
0 = Blockette 2000s from multiple SEED data records with different timetags may be packaged into a single SEED data record. The exact original timetag in each SEED Fixed Data Header is not require for each blockette 2000.
1 = Blockette 2000s from multiple SEED data records

```
                            with differing timetags may NOT be repackaged
                            into a single SEED datarecord.  Set this bit
                            if the timetag in the SEED Fixed Data Header
                            is required to properly interpret the opaque data.
                [bits 2-3] Opaque blockette fragmentation flags.
                        00 = opaque record identifed by record number is
                             completely contained in this opaque blockette.
                        01 = first opaque blockette for record spanning
                             multiple blockettes.
                        11 = continuation blockette 2...N-1 of record
                             spanning N blockettes.
                        10 = final blockette for record spanning N blockettes.
                [bits 4-5] File blockette info.
                        00 = not file oriented
                        01 = First blockette of file
                        10 = continuation of file
                        11 = last blockette of file
8   *   UBYTE:  Number of Opaque Header fields.
                Each opaque header field is a variable length ascii string,
                terminated by the character "~".
9       VAR:    Opaque Data Header string, which contains the ascii variable
                length fields.  Each field is terminated by a "~".  The
                definition of the fields may be defined the by the originator
                and receiver, but the following are recommended.  Any of the
                fields may be empty.
    a           Record Type - name of the type of record (eg "GPS", "GPS MBEN").
    b           Vendor Type - name of equipment vendor (eg "ASHTECH").
    c           Model type - model type of equipment (eg "Z12").
    d           Software Version - software version number (eg ""),
    e           Firmware Version - firmware version number (eg "1G0C");
10      OPAQUE: Opaque Data - bytes of opaque data.
                Total lenght of opaque data in bytes is
                blockette_length - 15 - length(opaque_data_header_string)
```

NOTES:

1. More than one blockette 2000s may be stored in a SEED data record.
if the SEED data record timetag is not require for precise timing of
the data in the opaque blockette.

2. Under normal usage, there would be no data in the data portion of the
SEED data record. However, it is possible that the blockette 2000 could
be used to provide additional information for a normal timeseries data
channel.

Phil Maechling

Nov. 13, 1997

1145 PM

Increasing Need for Job Specialization

- Sensor & Station Installtion
- Communications/Networking
  - Microwave
  - Spread Spectrum
  - Serial
  - TCP/IP
  - Frame Relay
  - SLIP
- Realtime Earthquake Processing System (Software)
- Data Archival
- Data Quality Verification
- Administration of People/Funding

Phil Maechling
13 Nov 97 1:45pm

Consistentcy of Information & Tracking Changes

- Varieties of Sensors & Dataloggers
- Versions of Software on Dataloggers
- Station Configuration Variations
- Station information LAT/LONG/ELEV
- Data processing Software Versions & Updates
- Upgrading Datalogger Software

Phil Maechling
13 Nov 1997 1:45pm

Problem Detection & Reporting

- Need Problem Reporting system
  Tracking system
  Resolution Reporting System

- Need Dectetion of Problem methods, procedures
  sensors GPS
  Dataloggers comm link

- We use Netmon & monitor.pl

- adopting operational procedures formally

Phil Maechling
13 Nov 1997 1:45pm

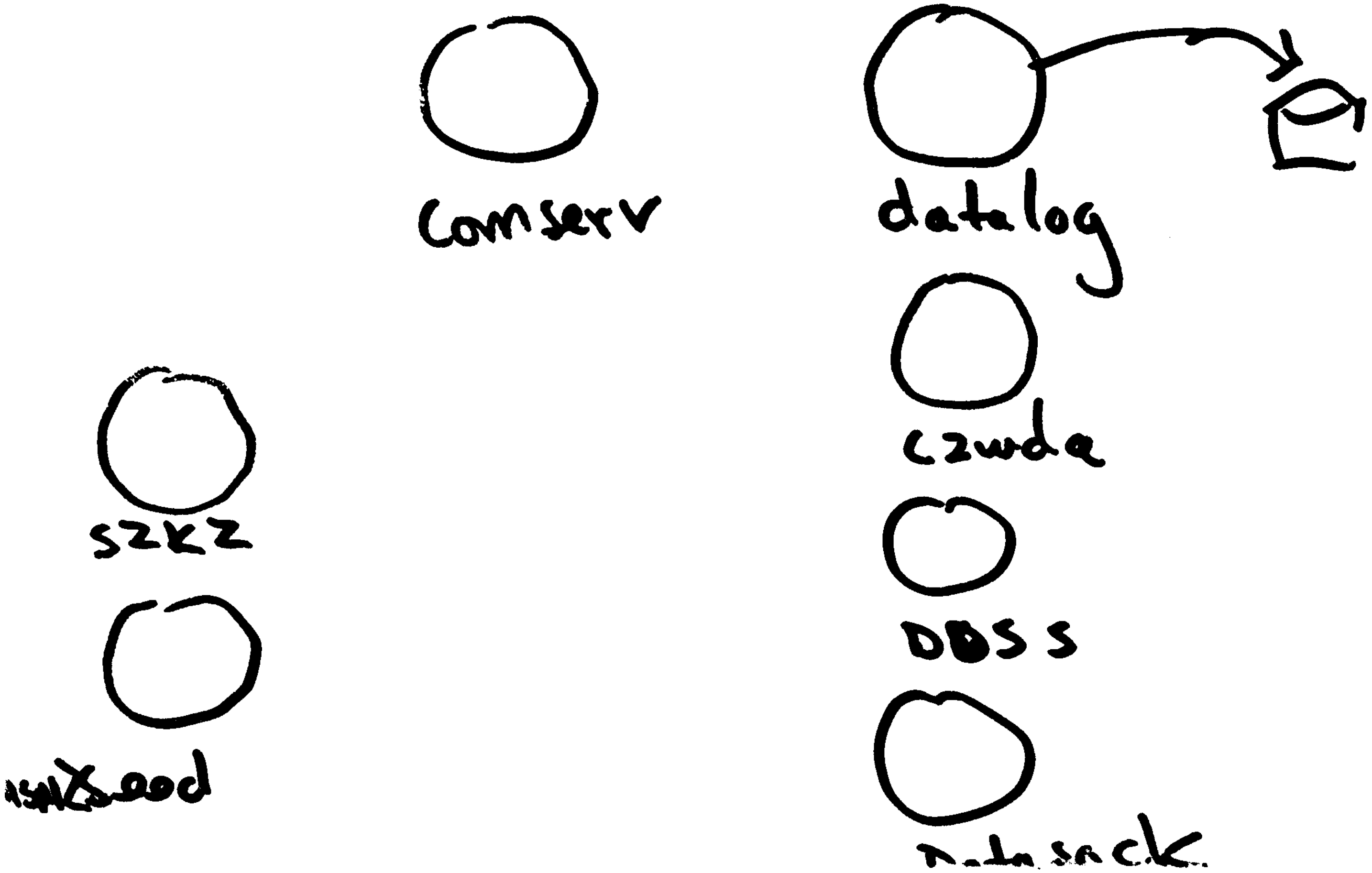
3

Proliferation of Processes on Central site

X * Number_of_stations = Number_of_processes on central site

How can we limit X ?

Typically
X = 3



Phil Maechling
13 Nov 1997 1:45pm

Increasing Need for Job Specialization

- Sensor & Station Installtion
- Communications/Networking

  microwave   Serial   Frame Relay
  spread Spectrum   TCP/IP   SLIP

- Realtime Earthquake Processing System (Software)
- Data Archival
- Data Quality Verification
- Administration of People/Funding

Phil Maechling
13 Nov 97   1:45pm

Consistentcy of Information & Tracking Changes

- Varieties of Sensors & Dataloggers
- Versions of Software on Dataloggers
- Station Configuration Variations
- Station information LAT/LONG/ELEV
- Data processing Software versions & Updates
- Upgrading Datalogger Software

Phil Maechling
13 Nov 1997 1:45pm

Problem Detection & Reporting

- Need Problem Reporting system
  Tracking system
  Resolution Reporting System

- Need Dectetion of Problem methods, procedures
  Sensors GPS
  Dataloggers " m link

- We use Netmon & monitor.pl

- adopting operational procedures formally

3

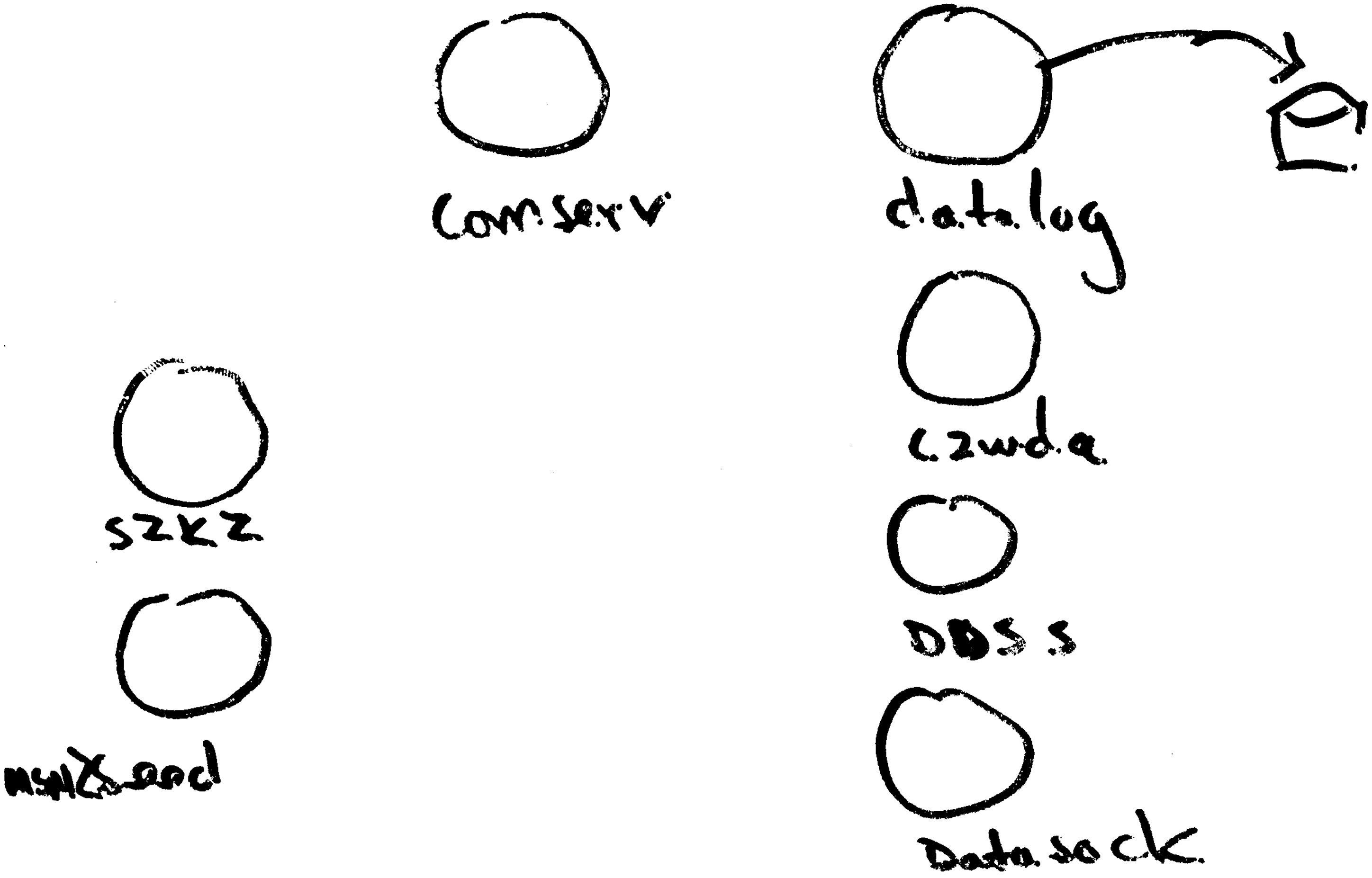
Phil Maechling
13 Nov 1997 1:45pm

Proliferation of Processes on Central site

X * Number_of_stations = Number_of_processes on central site

How can we limit X ?

Typically X = 3



Phil Maechling
13 Nov 1997 1:45pm

Robert Busby

Nov. 13, 1997

12:30 PM

Busby
11/13/97 12:30
Page 1

Time Delay in FIR Filters?

Robert W. Busby
Channel Z Seismometry

An apparent delay between data from different streams is observed on some Quanterra dataloggers. The cause of this delay and the appropriate method for correction of the time labels is the subject of this report.

On the Quanterra system, the delay associated with the hardware FIR filter for each sample rate is kept in a table. A description of the calculation for each delay is included below. The delay can be broken into three parts. The first and most significant part of the delay is related to the number of coefficients used in the filter. The second part of the delay is a small correction to time align the output samples of the filter. The third part is an experimentally determined time shift so that the data filtered by WWSSN filters give a first break time consistent with earlier estimates.

Taken from a memo June 3 1992 from Joe Steim to Bob Hutt:
Internal filter delay for each sample rate is given by the following formula;

$$D_m = 1/2 \sum_{n=1}^{m} (L_n \ / f_n) - T_m + I_m - 1.0$$

where:
D_m is the delay in seconds of filter stage *m*.
L_n is the length in samples of filter stage *n*.
f_n is the input sample rate in samples per second of filter stage *n*.
T_m is the half width at half amplitude of the impulse response of filter stage ***m*** (in seconds).
I_m is a constant related to the data buffering of filter stage *m*. It is a function of the initial preloading of all filter accumulators so that samples come out of the filters at all rates at the same time. It is essentially $1.0 - 1.0/L_m$.

The table below differs from the original in that I have added a column, O_n, for the output sample rate and changed the value of I_1 from 0 to 1 to make D_1 consistent with formula above.

TABLE of time delay values.

m	T_m	I_m	f_m	L_m	D_m	O_m
1	0	1	5120	64	0.006	320
2	0.021	0.986	320	72	0.083	80
3	0.040	0.984	80	64	0.462	40
4	0.080	0.984	40	64	1.222	20
5	0.160	0.984	20	64	2.742	10
6	1.800	0.996	10	260	14.114	1

Busby
11/13/97 12:30
page 2

There is a slight error in the calculation of the first part of the FIR filter delay. The time delay associated with N coefficients is midway between (N-1) sample intervals, or a delay term of 1/2((L-1)/f). More important however I believe the third part of the delay should not be applied at all and that this is the major cause of discrepancy between Quanterra data streams.

I calculate a new table below which has only the (corrected) term for the coefficients and the small fixed delay for time alignment. The difference, S_n between the delay actually applied and this new delay calculation should be observed as a time shift between streams.

$$F_n = 1/2 \sum_{m=1} \left((L_m - 1) / f_m \right) + I_n - 1.0 \quad \text{and} \quad S_n = D_n - F_n$$

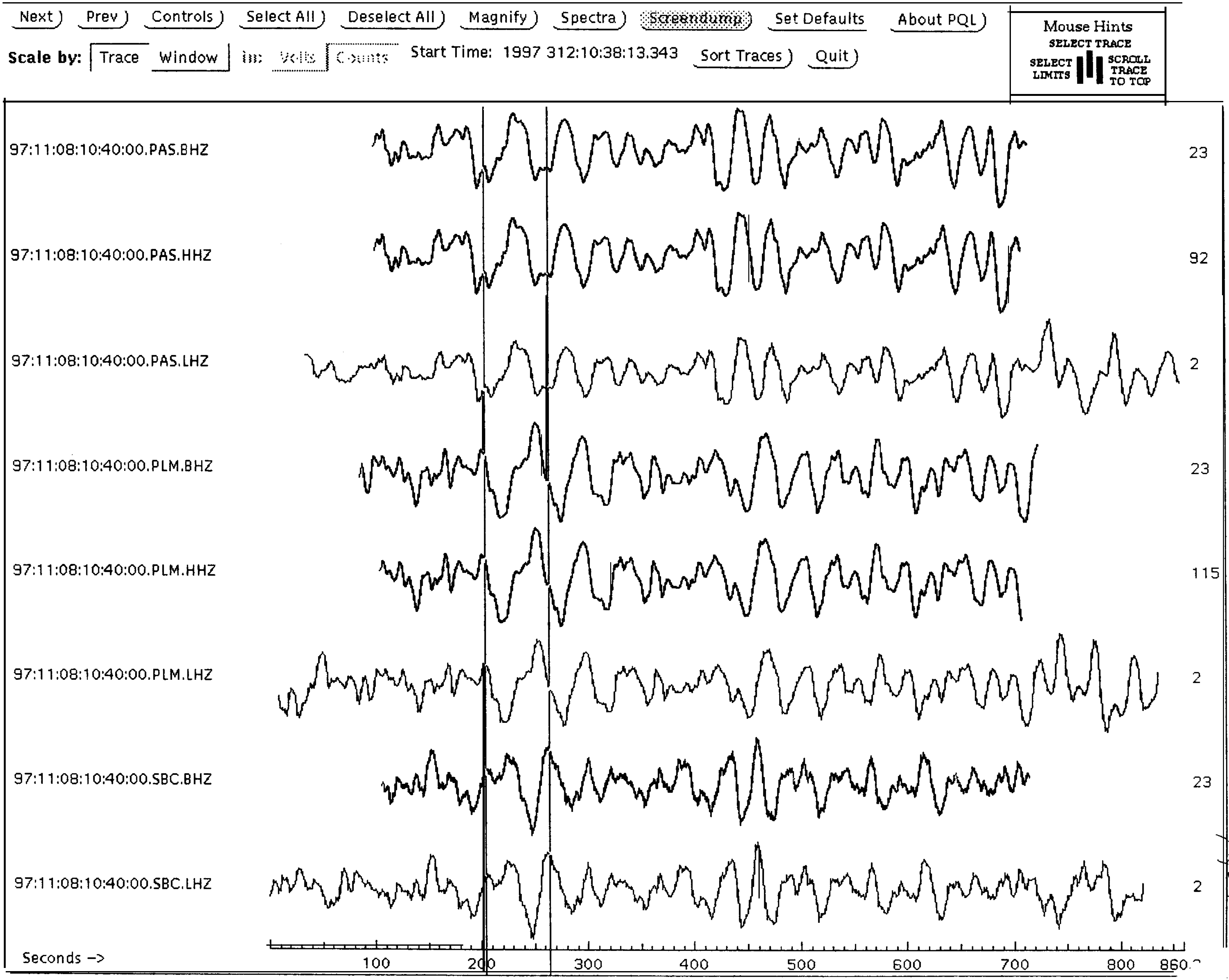
TABLE of shifts to time align streams. S_n should be added to quanterra time to yield UTC.

n	I_n	f_n	L_n	D_n	F_n	S_n	O_n	
1	1	5120	64	0.006	0.006	0.000	320	
2	0.986	320	72	0.083	0.103	-0.020	80	HHZ
3	0.984	80	64	0.462	0.495	-0.033	40	
4	0.984	40	64	1.222	1.282	-0.060	20	BHZ
5	0.984	20	64	2.742	2.857	-0.115	10	
6	0.996	10	260	14.114	15.8193	-1.705	1	LHZ

The observed time delay between streams is shown by overlapping traces of different streams for the same time period. Three types of dataloggers are displayed. The Q52K with software FIR filters, the Q680 with hardware filters and the Q4128 with hardware filters. Since plotting programs may not always plot multiple sample rate data correctly on an absolute time axis I show examples from both PQL and SAC. The conversion to SAC used Doug Neuhauser's ms2sac program.

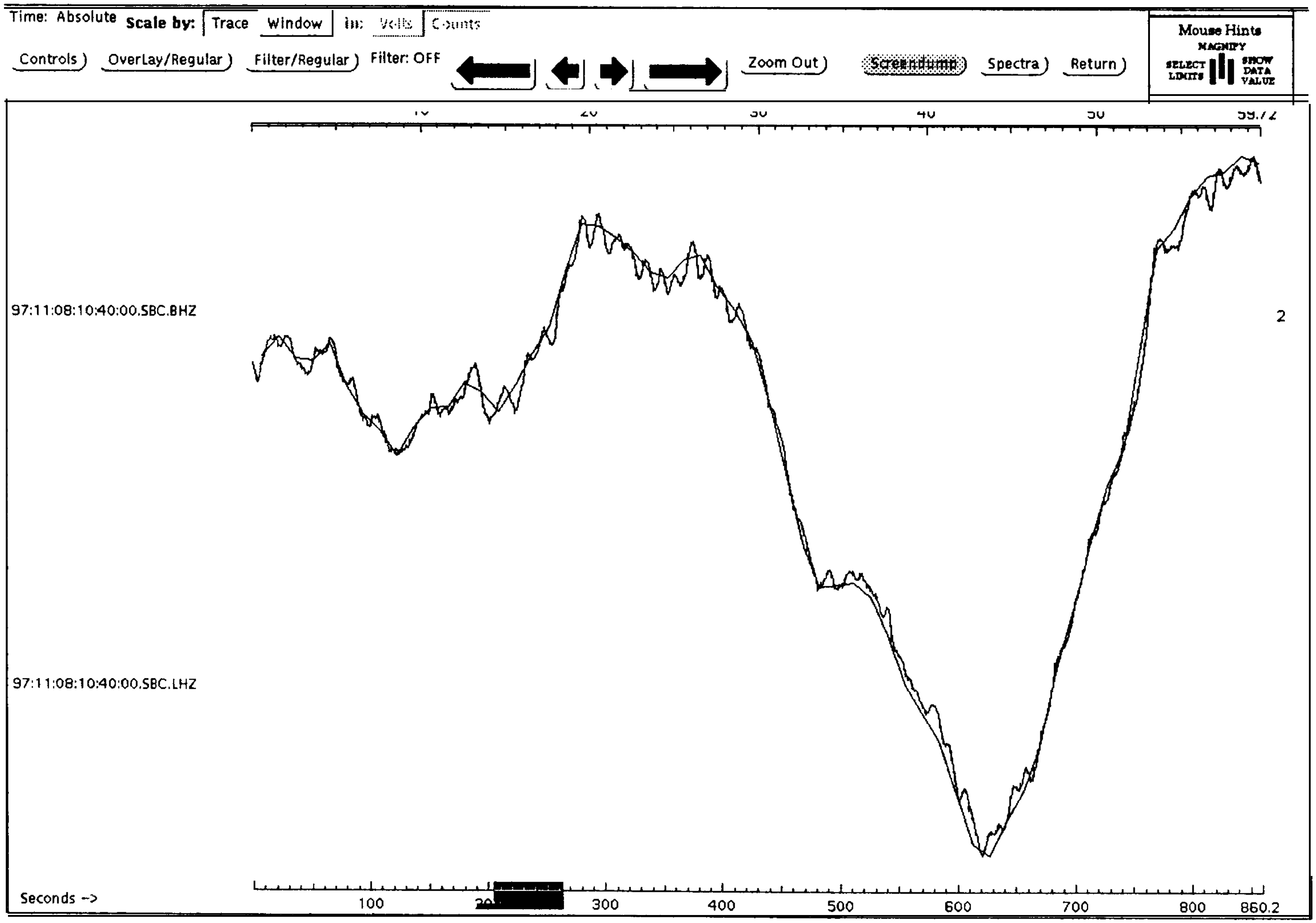
To estimate the exact delay between streams I recorded a sine wave of 0.1Hz on a Q680 on HHZ, BHZ and LHZ streams. I interpolate the BHZ and LHZ back to 80 sps and perform a cross-correlation with the HHZ data. The maximum correlation between LHZ and BHZ is at 906 +/- 10 msec. I used a 1 Hz sine wave to determine a maximum correlation between HHZ and BHZ at 13 +/- 2 msec.

Observed Delay	Expected Delay
HHZ-BHZ = 0.013 sec	0.040
BHZ-LHZ = 0.906 sec	1.645



Start Time: 1997 312:10:38:13.343

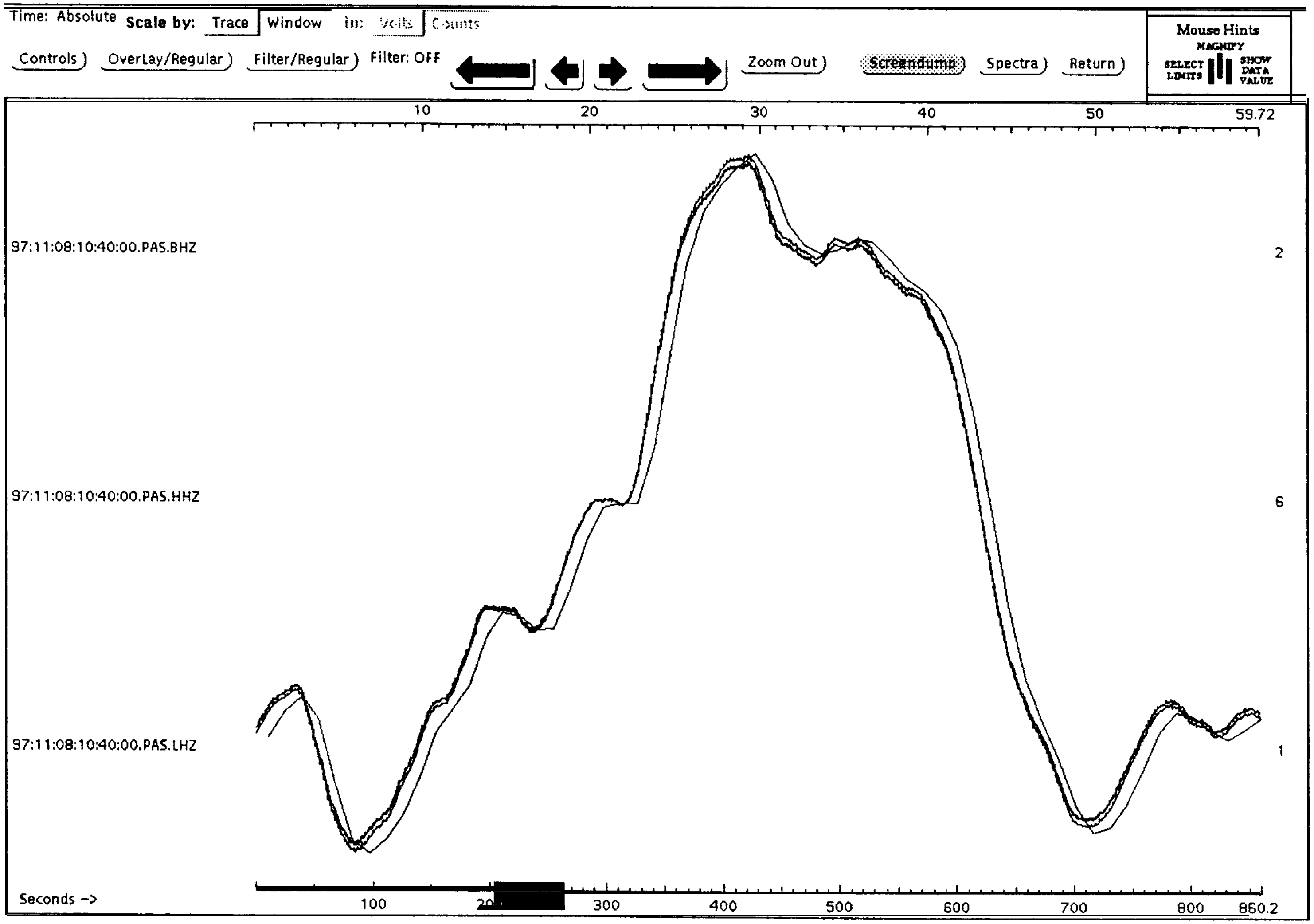
[illegible]



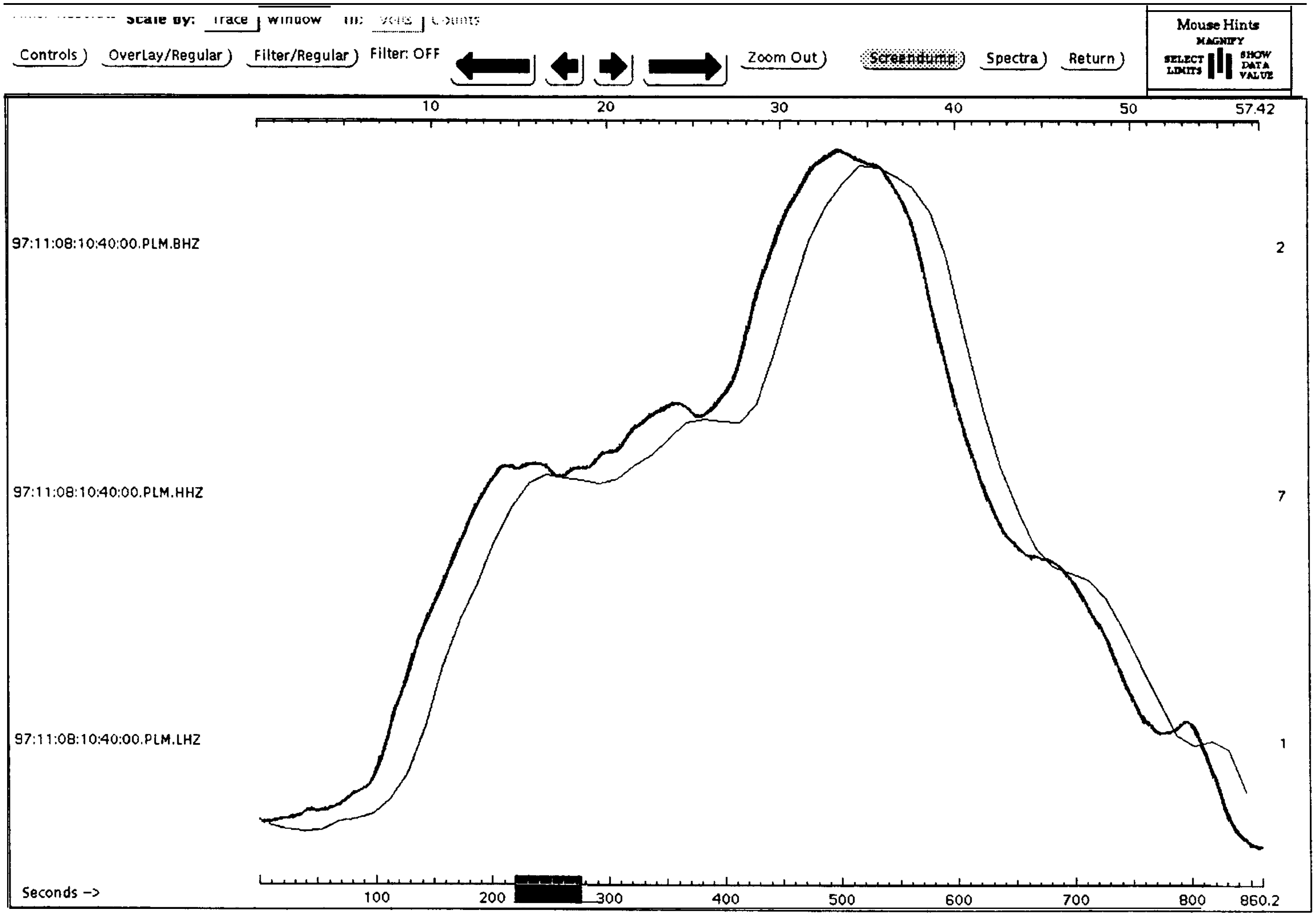
Time: Absolute Scale by: Trace Window in: Volts Counts

Filter: OFF

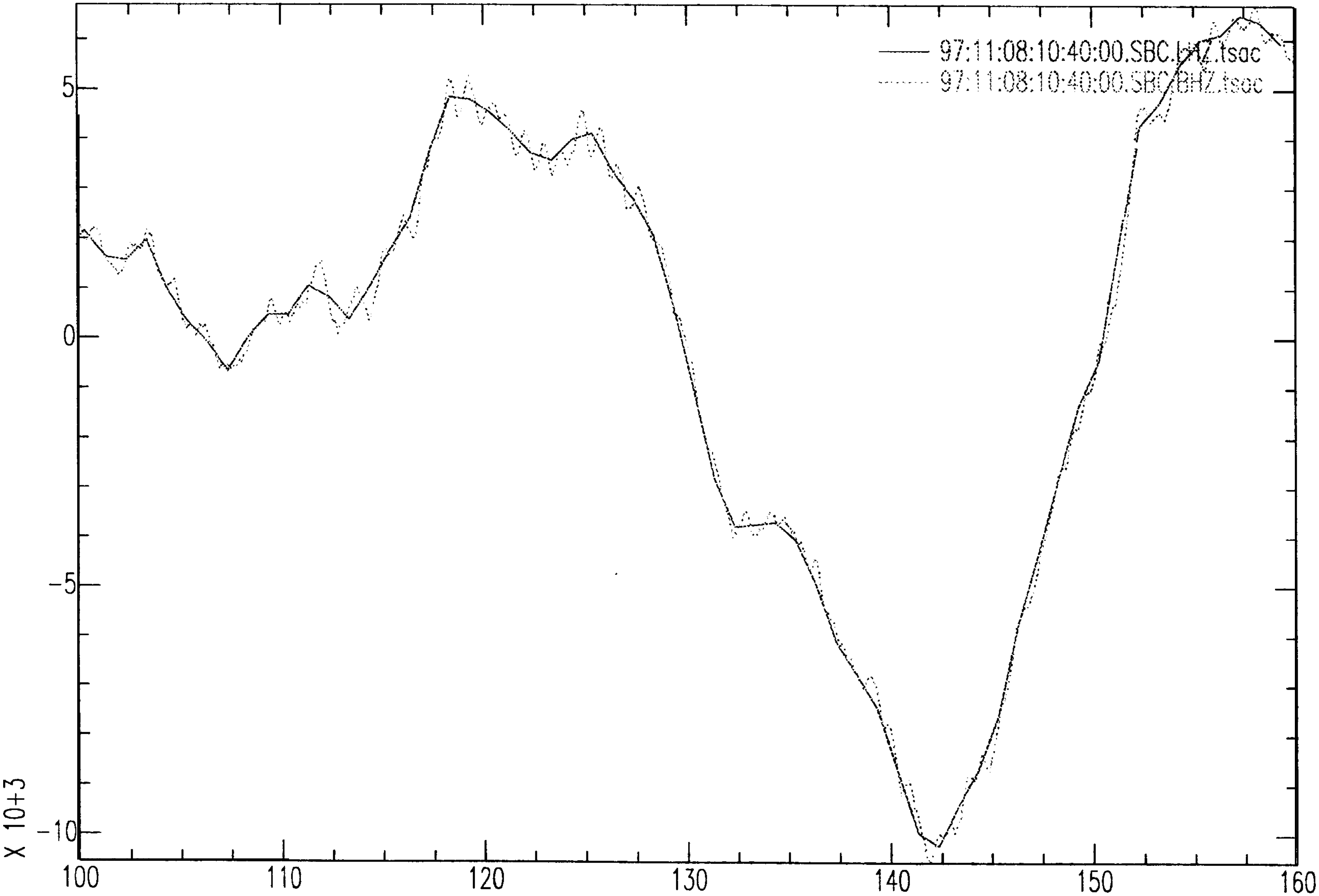
[illegible]


Time: Absolute
Scale by: Trace Window in: Volts Counts
Filter: OFF

Busby
11/1/97 10:30
[illegible]

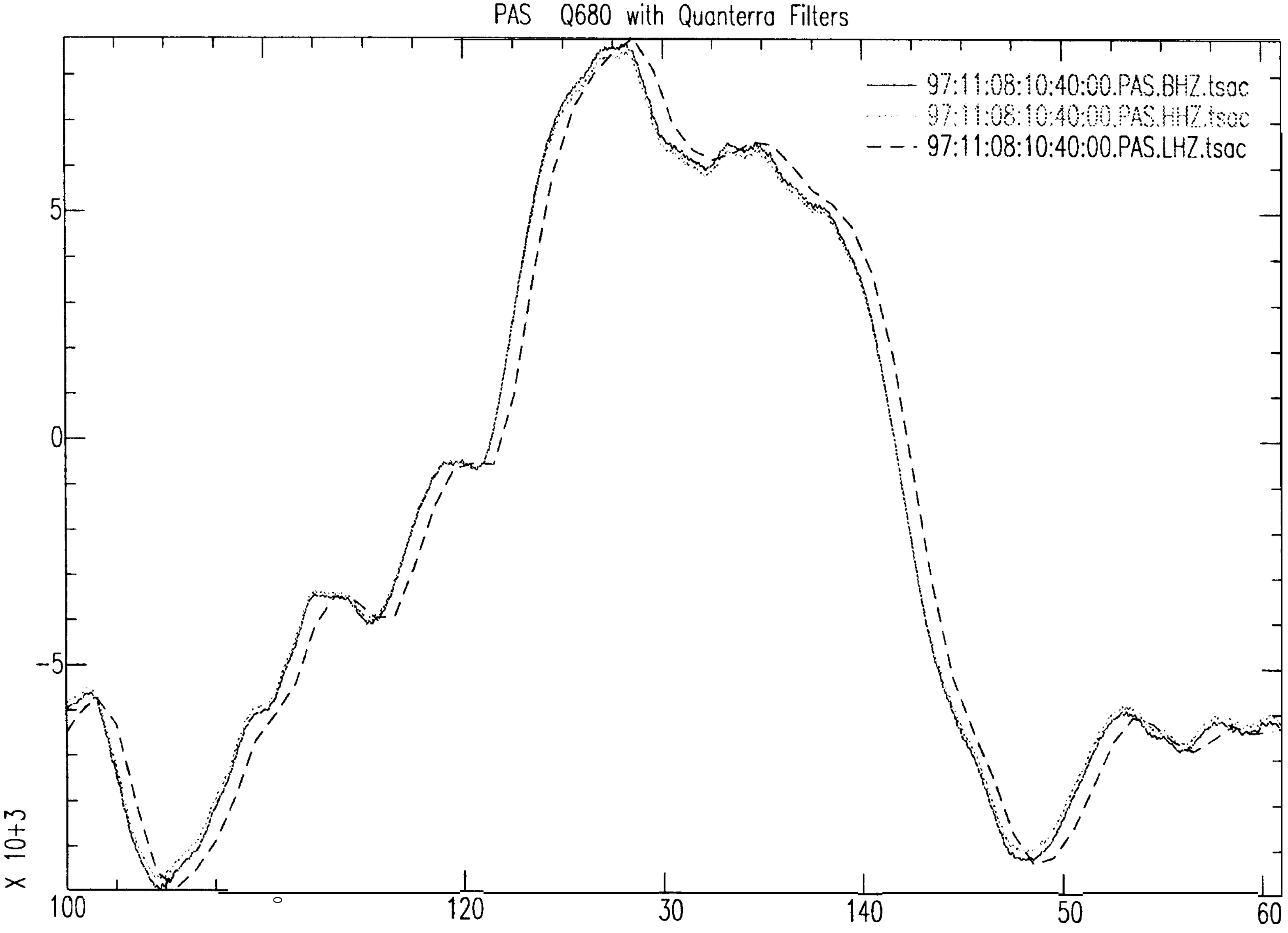

Scale by: Trace | Window
In: Volts | Counts
Controls
OverLay/Regular
Filter/Regular
Filter: OFF
Zoom Out
Screendump
Spectra
Return
Mouse Hints
MAGNIFY
SELECT LIMITS
SHOW DATA VALUE
10
20
30
40
50
57.42
97:11:08:10:40:00.PLM.BHZ
2
97:11:08:10:40:00.PLM.HHZ
7
97:11:08:10:40:00.PLM.LHZ
1
Seconds ->
100
200
300
400
500
600
700
800
860.2

[illegible]

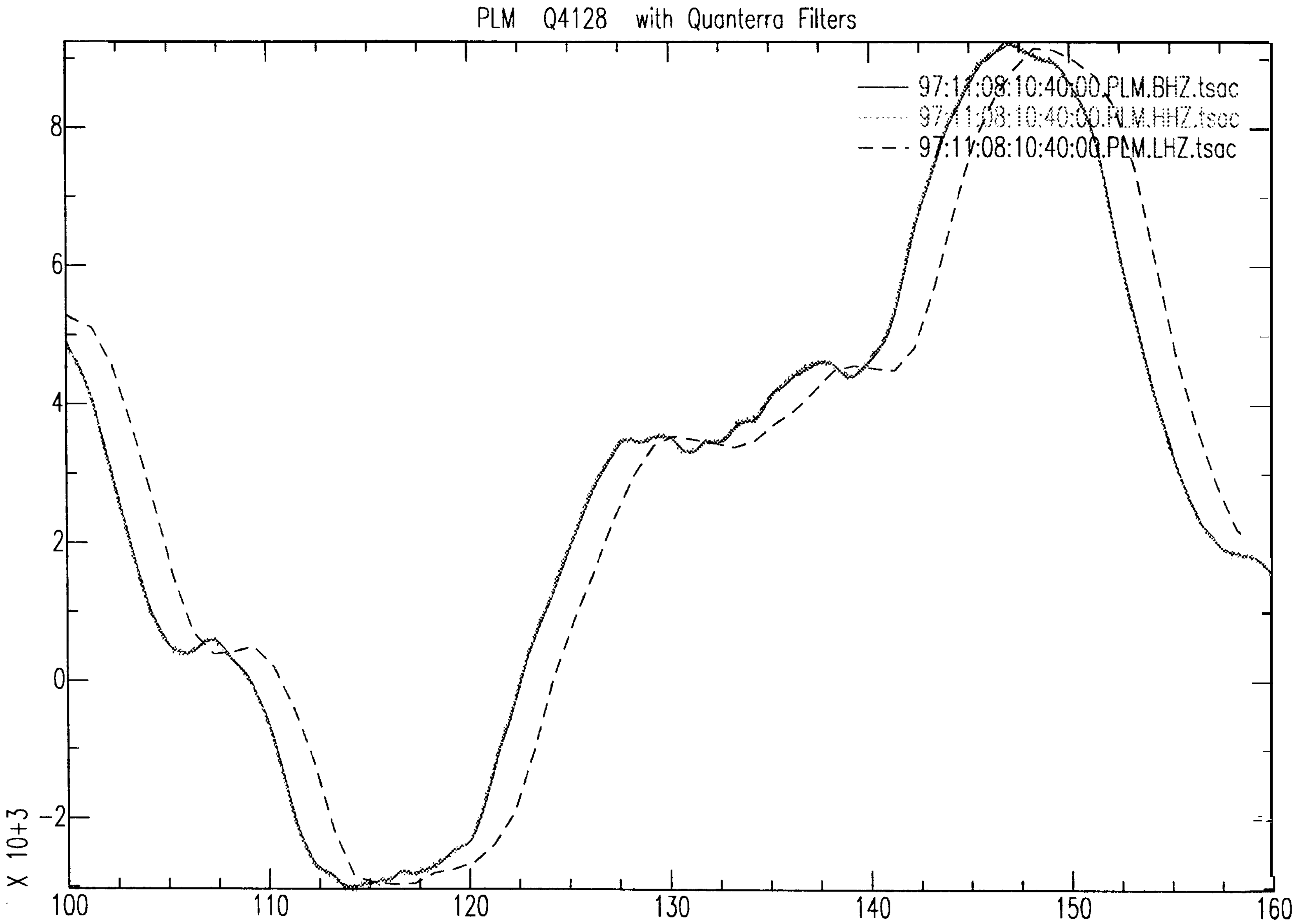


SBC Q52K with software filters

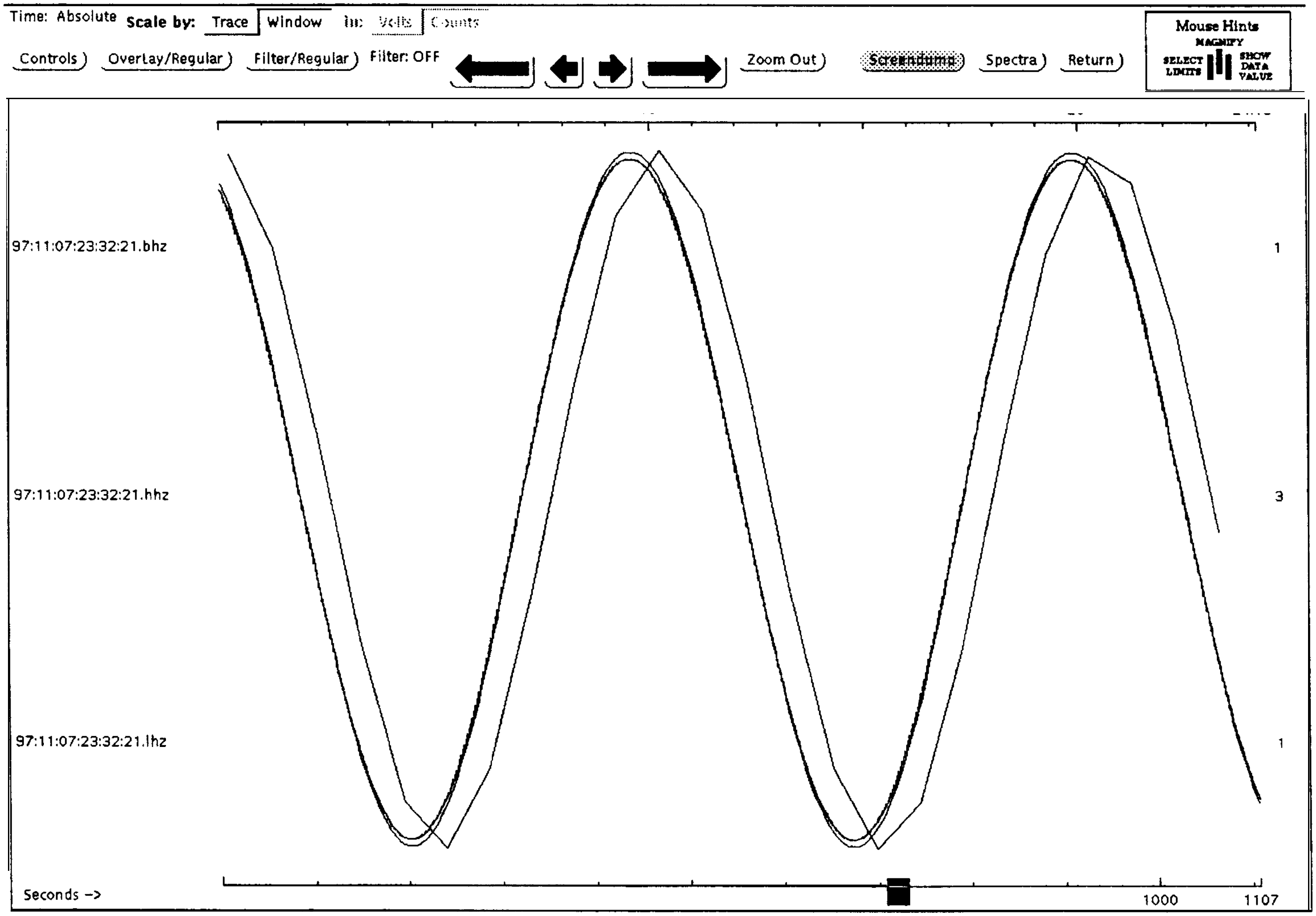
Busby
11/13/97 [illegible]
page 6


PAS Q680 with Quanterra Filters

Busby
11/13/97 12:30
Page 7



Busby
11/11/97 12:20
Page 8

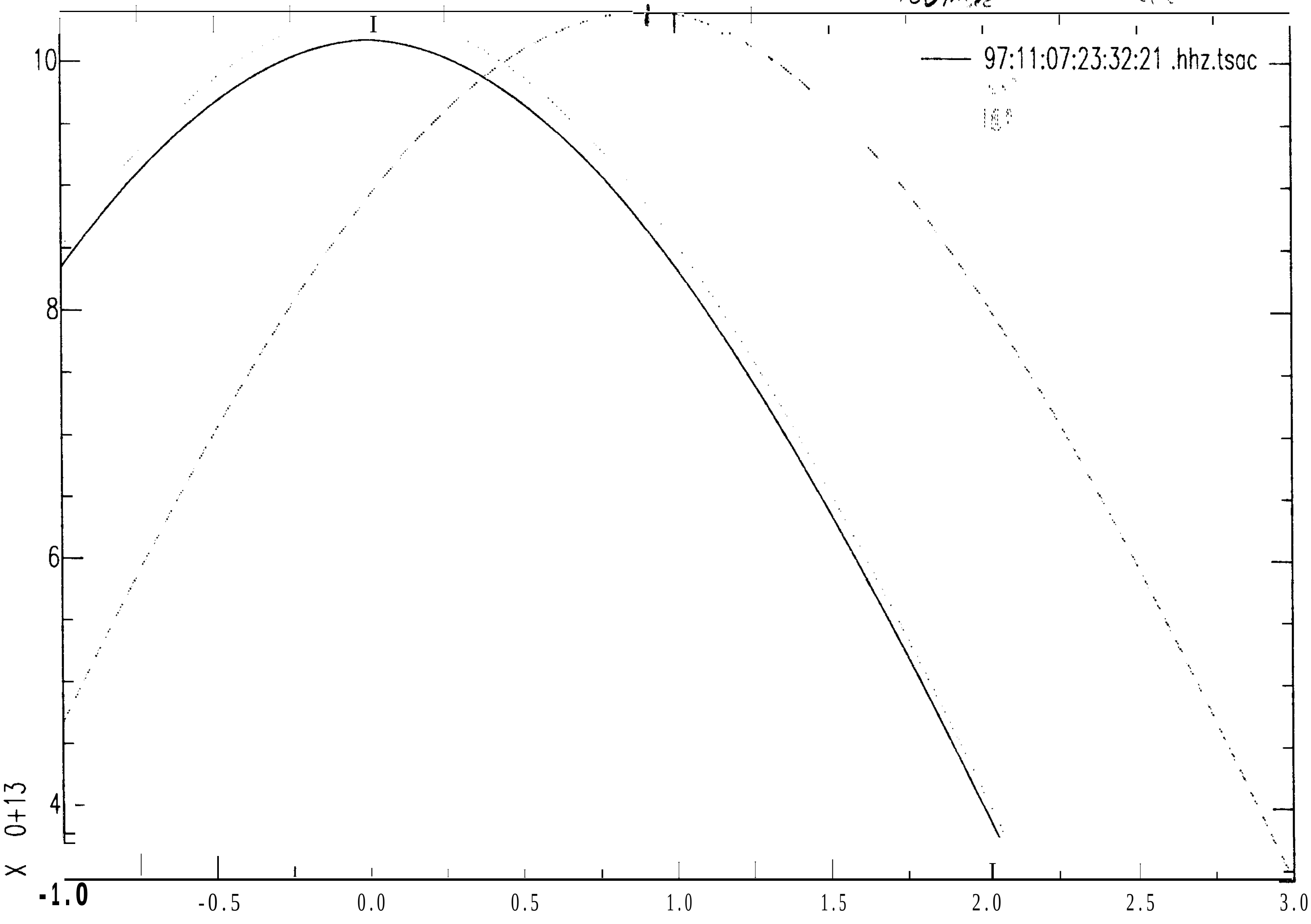


Time: Absolute **Scale by:** Trace Window in: Volts Counts

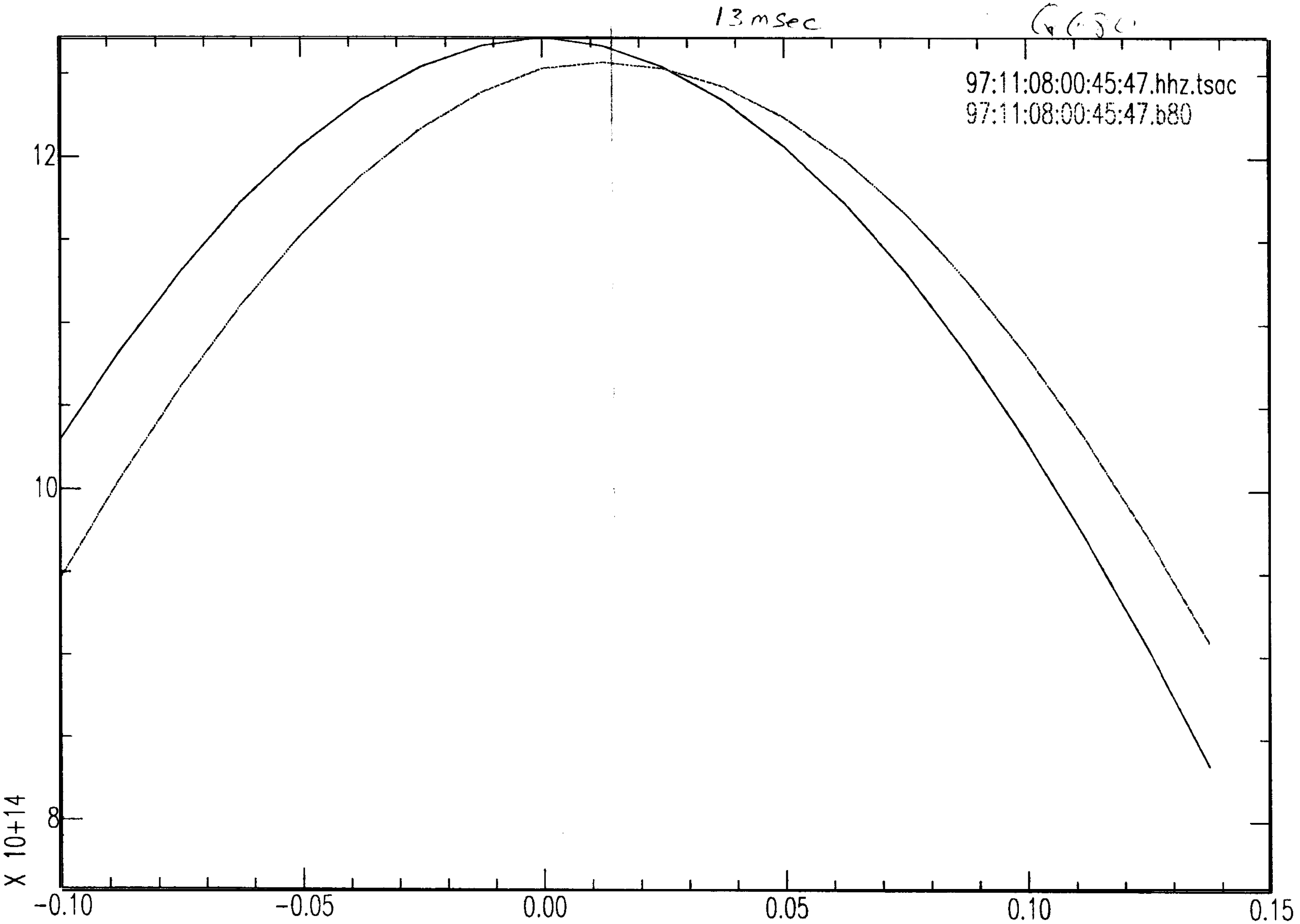
Filter: OFF

Busby 11/13/97 12:50
1 of 3

Correlation of 10 Second Sine Wave 906 nsec Q68C



Busby
11/13/97 12:30
Page 10



Busby
11/13/97 12:30
page 11

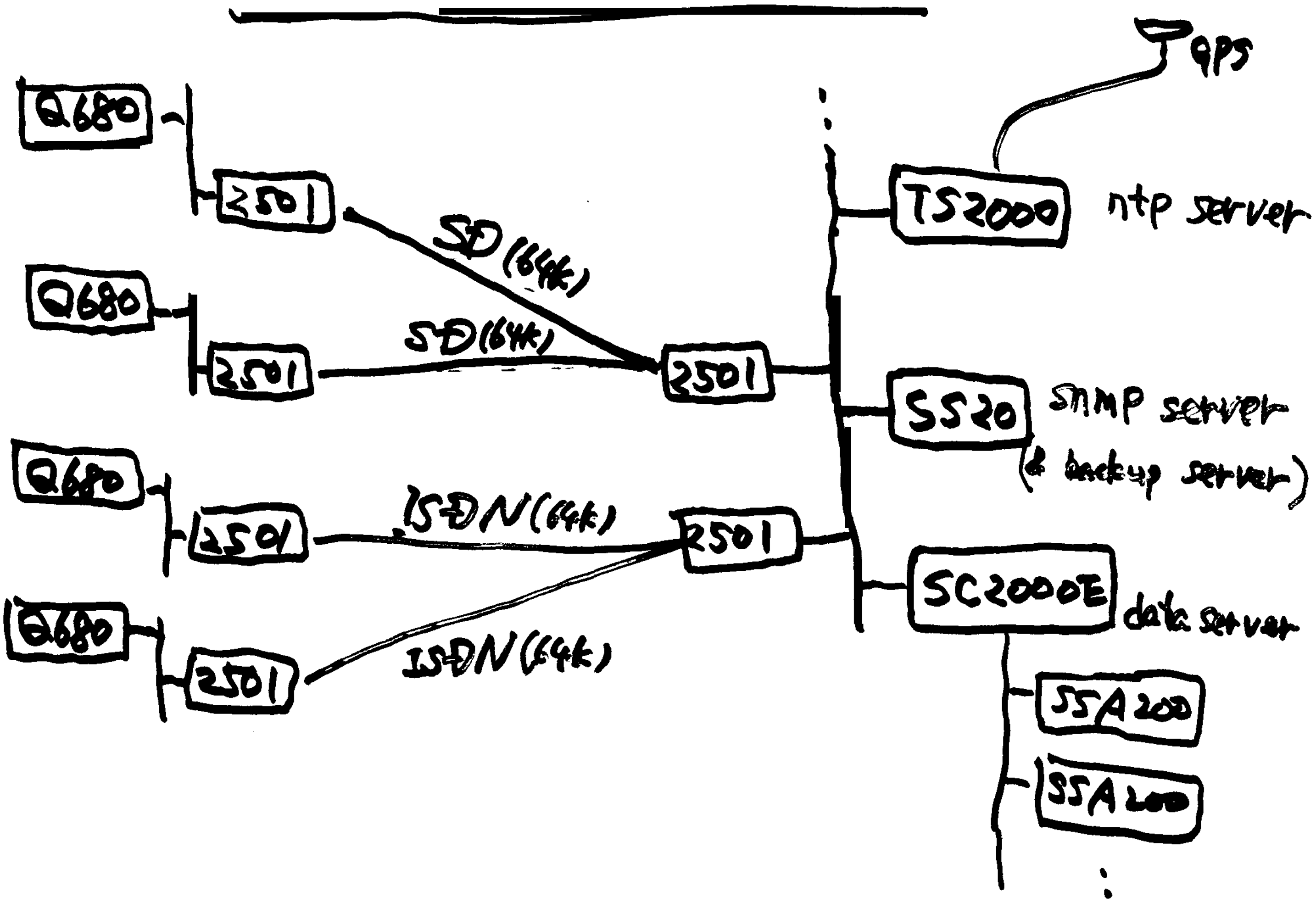
Eiichi Fukuyama

Nov. 13,1997

2:30 PM

E. Fukuyama
97/11/14 P.1
14:30

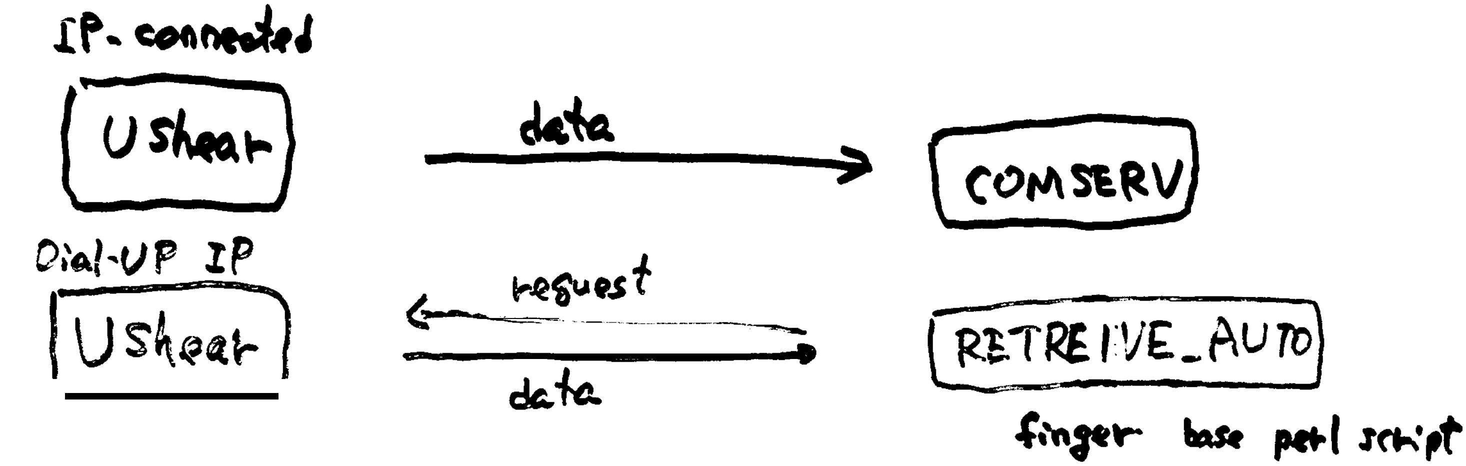
FREESIA NET



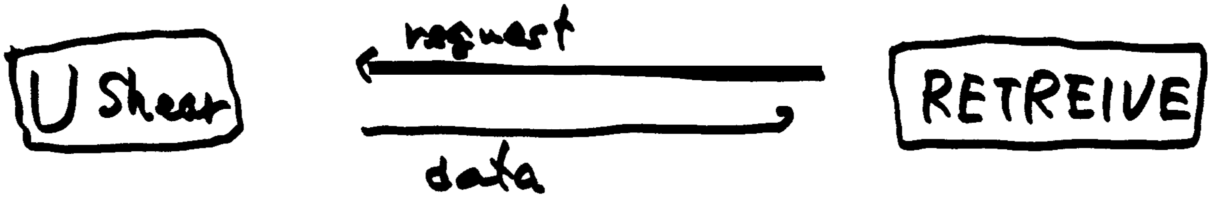
http://argent.geo.bosai.go.jp

E. Fukuyama
97/11/14 P. 2
14:30

Routine Process



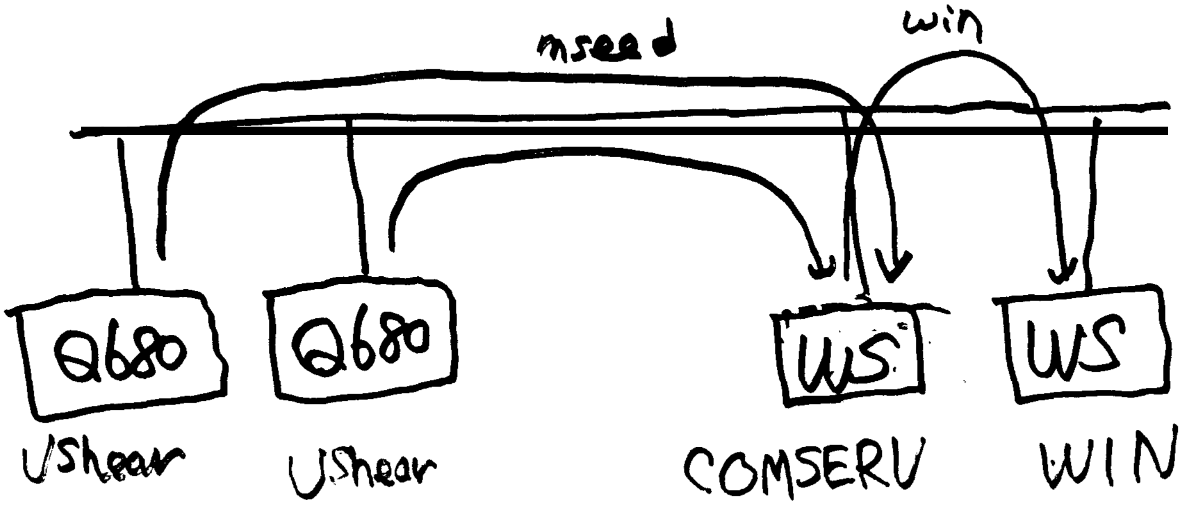
Transmission Fail.



gconv : gmerge rapper perl script
unify comserv & retreive.

E. Fukuyama
97/11/14 P.3
14:30

Testing Environment

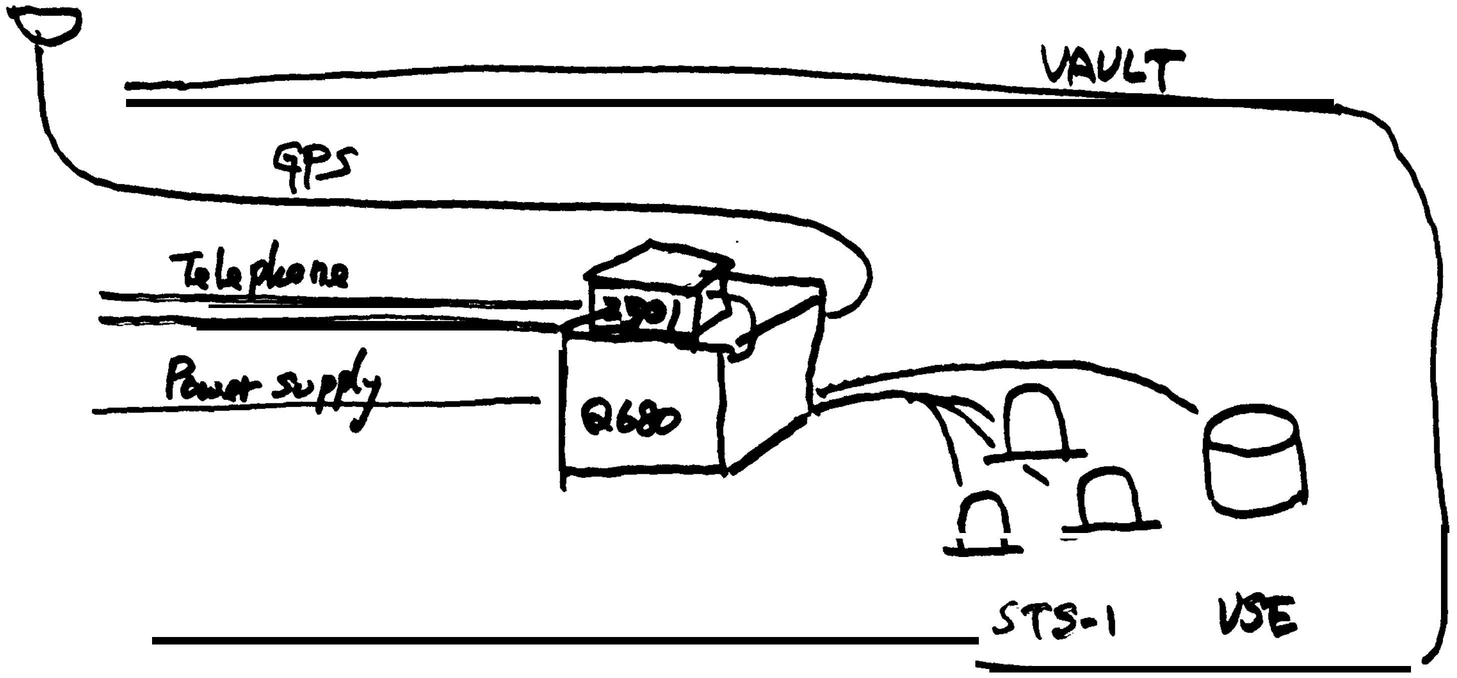


MSEED : fixed Packet length
Variable time window

WIN : variable Packet length
fixed time window

E. Fukuyama
97/11/14 P.4
14:30

Typical Configuration at Station



Joe Steim

Nov. 13,1997

3:00 PM

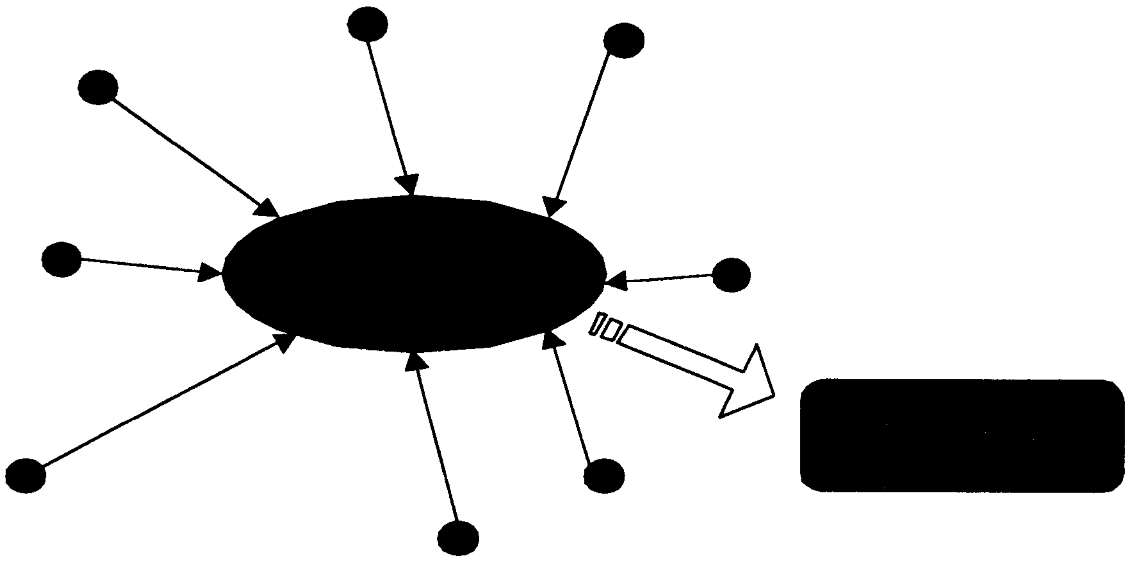
Part A

Robust Telemetry for Modern Seismic Networks

J.M. Steim
Quanterra, Inc.

Abstract

Most telemetered networks are now organized in a conventional hub-and-spoke configuration, with dedicated communications links from each field station to a single central collection point. The hub-and-spoke arrangement is inherently subject to failure at a single point: the network center. An arrangement of field stations and data collection points as a “well-connected” network eliminates the single point of failure, and provides for multiple simultaneous parametric or waveform data “consumers” that do not depend on the collection and distribution of data from a central site. Such an organization inherently combines the function of traditional “repeaters”, but in effect permits repeating information across the entire dimensions of a network. A simple, robust routing protocol is required to run at each node in the network to permit automatic self-identification of new nodes, and to handle “self-healing” of the network when individual “hops” between nodes become unavailable. Low-bandwidth links, such as radio, may preclude complete waveform transmission across a large (say, 100 station) “well-connected” network, but would likely permit highly robust parametric data transmission. A sophisticated software operating environment is needed at each field node. The “well-connected” routing protocol may be implemented using industry-standard stateless UDP/IP protocols. To improve network efficiency, waveform data may be compressed very quickly to near entropy-coding levels using Level 3 compression, without introducing latency.



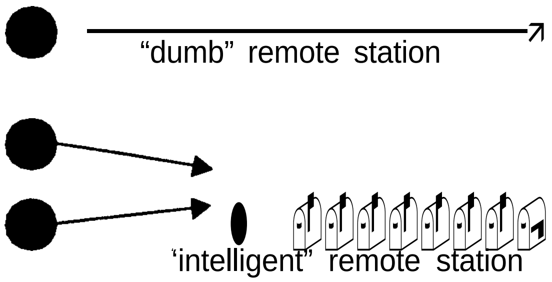
Traditional, “singly-connected” network with no redundancy.

Background

There is an increasing trend toward multiple uses of data telemetered in near-real-time for critical public information, in addition to traditional research. **Most** research applications can tolerate an occasional network shutdown because of central computer or communications "hub" failure. Critical applications must, however, be able to provide timely, accurate information even when a central network system or communications system is down. Only a technique that eliminates **every** single point of failure, by distributing telemetry and data collection functions can achieve the required reliability.

The "dumb" remote station

In traditional hub-and-spoke networks, each "dumb" remote site reports directly to a central system. There is no provision for acquisition of data from field sites if either the central site is down or communications links into the central site are down. The latter may be caused by a comparatively simple fault. Some real-time telemetered networks have achieved some degree of fault tolerance by equipping field sites with telephone dial-up and local recording. The real-time component, however, remains subject to single-point failure.

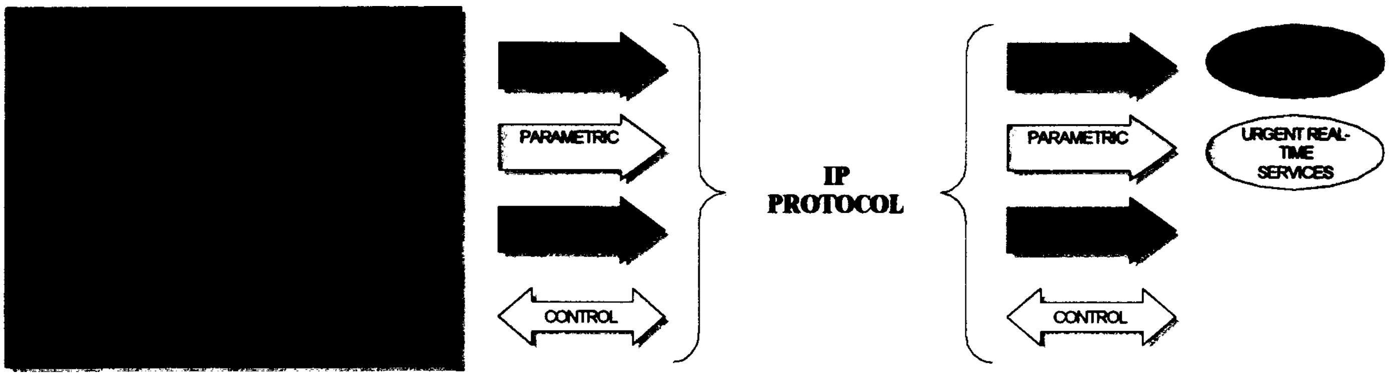


An intelligent remote station

An "intelligent" remote station, however, may implement in software the ability to forward data from other remote stations. This may be handled by specific application programs, or standard IP network protocols. By controlling the "forwarding" function in the remote ***station's software***, the function of a "repeater" or "router" becomes inherent in the remote station itself. Any physical communications medium, such as spread-spectrum radio, commercial services, or **hardwire** lines can be used.

Incorporation of industry-standard IP protocols directly at each remote station enables the station to be used simultaneously as a source of real-time waveform and parametric data and buffered data on request, and as a router for other stations' traffic. Each type of data may be "beamed" directly to the recipient, which may reside on separate computers in separate locations. IP protocol enables multiple usage of a single physical link, but need not imply excessive complexity or power consumption.

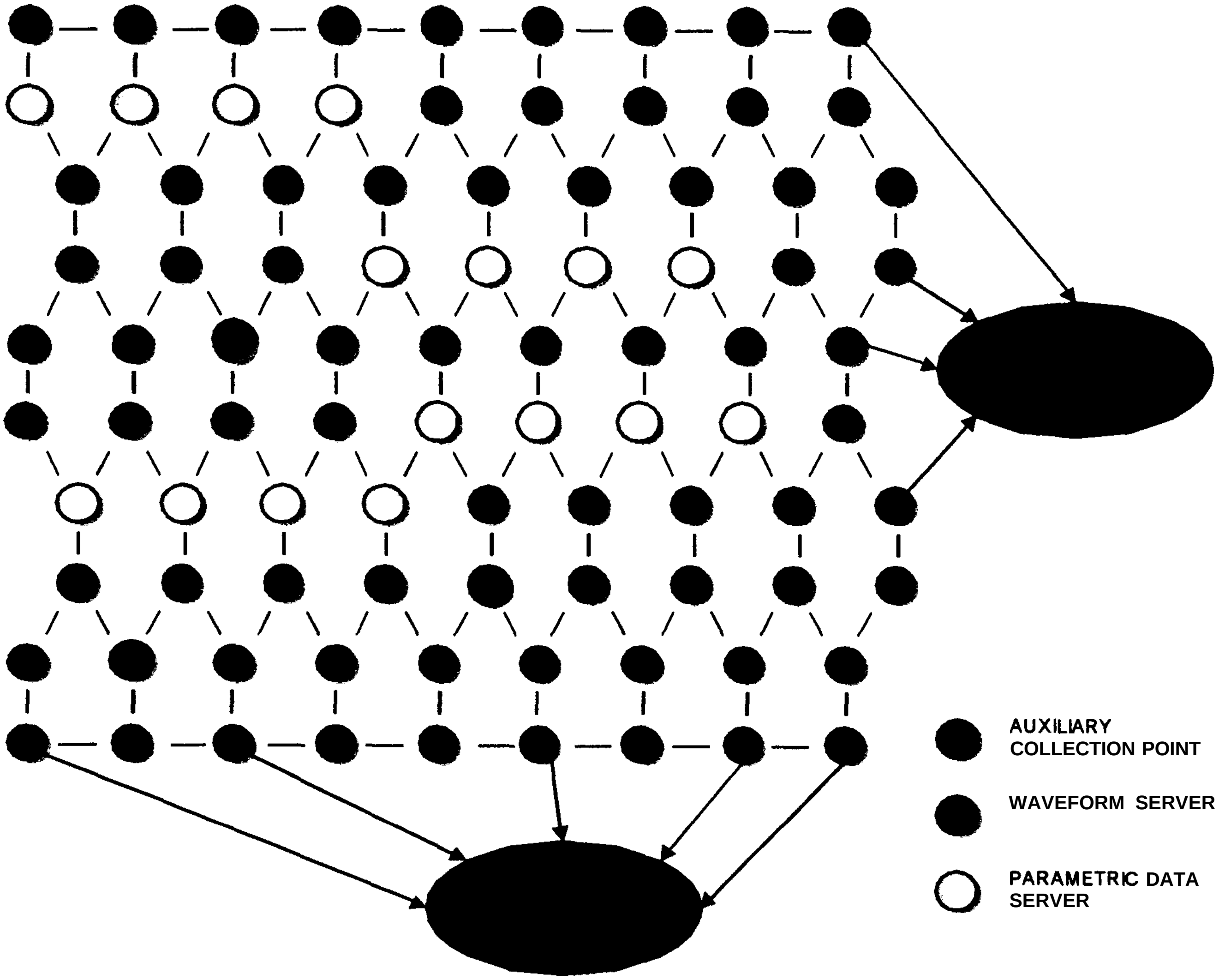
IP-protocol remote station



The "well-connected" network.

The figure below. Illustrates the "well-connected" concept. Each remote station is actually also a router, and is connected to up to 3 nearest neighbors by point-to-point physical or logical links that may be radio or some form of commercial service. With radio links, care must be taken in the placement of radios near one another, and the selection of frequencies or "hop" patterns to reduce the likelihood of interference, and the adoption of some security technique to prevent "break-ins". Not all links between nodes are necessary to achieve redundancy.

Routing is performed not from host-to-host, but using a highly simplified "Shortest Path First" algorithm, similar in concept to RFC 1583 (OSPF), but tailored to the specific case of dealing with low-latency data. In the illustrated network below, some remote stations provide waveforms, and some only parametric data, such as peak acceleration. The actual number of waveform sources in such a multiply-connected network will be limited by the number and placement of low-bandwidth links; relatively many more parametric data sources may be accommodated within the same network. Essential features of the network are the ability to collect data from the entire network or any subset at any node, and inherent fault tolerance.



Joe Steim

Nov. 14, 1997

9:00 AM

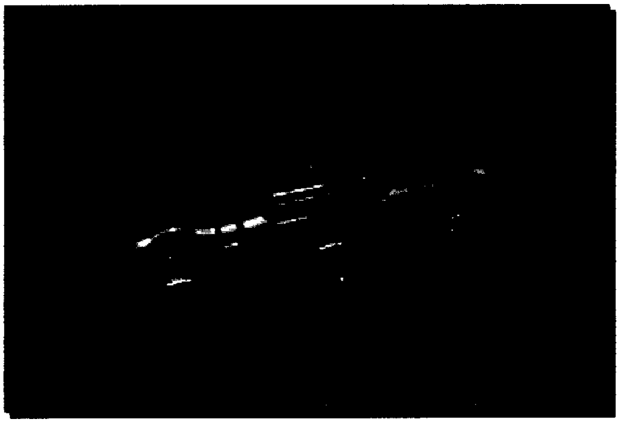
QUANTERRA *Q 7 3 0*

ADVANCED BROAD BAND REMOTE DATA ACQUISITION SYSTEM

General Description

The 4730 data acquisition system is an advanced low-cost remote broad-band data acquisition system incorporating Quanterra's leading, proven broad-band technology. The system combines a **3-channel 24-bit** digitizer having an independent digital signal processors (DSP), and a powerful CMOS computer system with RAM memory that supports Quanterra's real-time packetized communications protocol.

Q730 ***system with the cover open. The sealed polyethylene enclosure, standard on all Quanterra products, is water tight.***



Rear view of water-tight Q730 ***system enclosure showing connector panel.***

***140+dB** dynamic range A/D and DSP*

Quanterra set the world standard for data acquisition at 24 bits and beyond. Our systems are the acknowledged high performers in broad-band seismological instrumentation. The 4730 analog front-end incorporates Quanterra's own patented (US Patent 4866442, others pending) delta-sigma modulator and operates at a fixed sample rate of **20kHz,** with other rates derived by digital filtration and decimation in the DSP module. This is the same technique employed in Quanterra's 4680, and 44120 family 24-bit digitizers, in use world wide in leading programs such as the IRIS GSN, **TERRAScope,** and US National Seismic Network. Quanterra **A/D** technology consistently outclasses all others in side-by-side evaluation.

Proven Software - Ultra-SHEAR

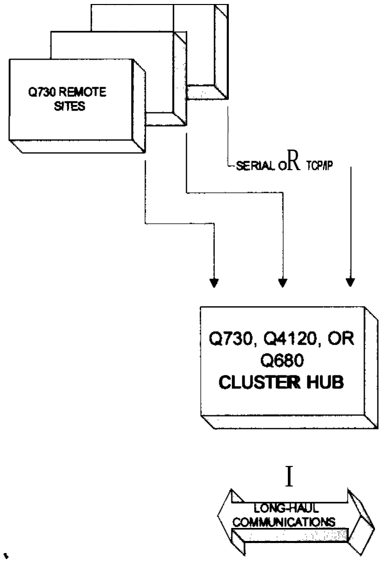
The **CPU/DSP** module is a CMOS 32-bit 68030, and 32-bit floating-point digital signal processor. The **CPU/DSP** supports Quanterra's proven *Ultra-SHEAR* comprehensive data acquisition **software** suite. *Ultra-SHEAR* is compatible across the entire line of Quanterra's products, and has been continuously refined over more than 10 years in highly scrutinized installations world-wide.

Minimum or Linear Phase Filters

The FIR digital filters may be either linear-phase (constant delay) or minimum-phase, which are causal. Causal filters may be essential where unambiguous onset times are a principal requirement. The user may also specify recursive filters used to filter broadband data before event detection.

Clustered Operation

Quanterra's new *Multi-SHEAR* data acquisition software suite allows any Quanterra system running *Ultra-SHEAR* or *Multi-SHEAR* to operate as a remote node transmitting data to a "cluster hub". In turn, within the limits of the number of I/O ports and processing power, cluster hubs may themselves report to another hub. Network topologies can be constructed to meet many needs without special hardware and software. Economical use can also be made of "long-haul" communications.



Real-time Packetized Transmission

All Quanterra processors support a real-time telemetry protocol, developed for and proven in leading networks such as the IRIS/GSN, Caltech and UC Berkeley. The protocol allows selective user-definable priority transmission of specific data types, such as broad-band event or long-period continuous, over a single link. The link may be temporarily completely cut, and the receiver powered down without loss of data. Extensive CRC error correction and sliding-window retransmission virtually eliminate the possibility of incorrectly-received data. The protocol includes advisory messages and event detections from the remote stations, and allows central-site triggering and reconfiguration without affecting data acquisition.

Serial or TCP/IP or UDP/IP links

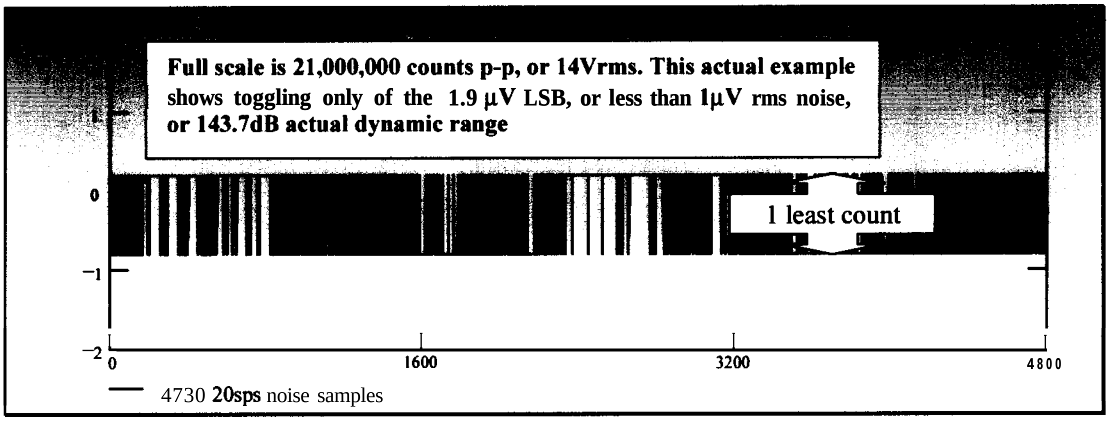
Telemetry may be through an ordinary serial asynchronous links, or a TCP/IP SLIP serial connection. TCP/IP communications with the 4730 allows remote maintenance and configuration using industry-standard networking protocols. Either ordinary serial or SLIP connections may use hardwire, modem or spread-spectrum duplex radio links. TCP/IP connections support advanced features such as origination of e-mail notification directly from the 0730, for example, when the power supply voltage drops too low. *Multi-SHEAR* supports advanced virtually-zero overhead stateless UDP/IP telemetry.

Data Compression

Data are stored using Federation of Digital Seismic Network standard Level 1 or 2 compression algorithms used on IRIS stations, which was also developed by Quanterra. Advanced Level 3 compression achieves near theoretical "entropy coding" levels.

High-Performance Analog – a Quanterra exclusive.. .

The figure below shows the actual acquired time series from a terminated-input test:

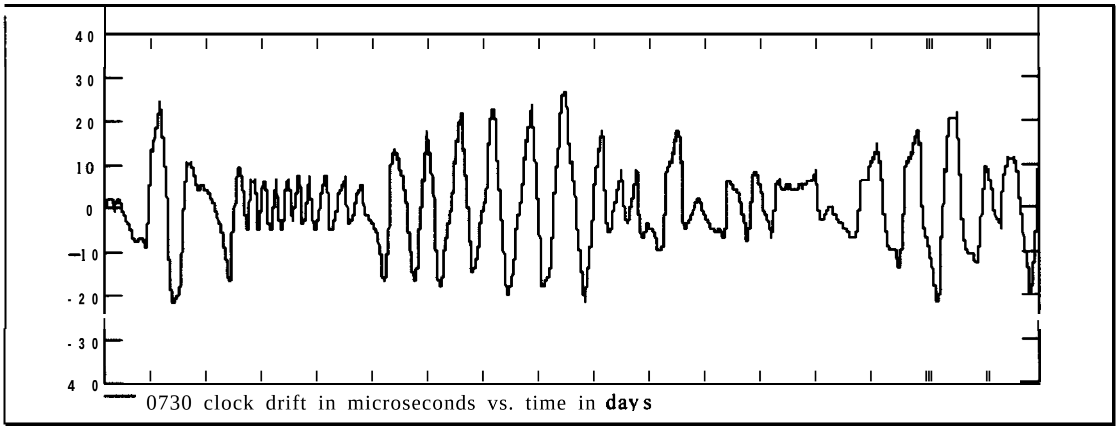


Full scale is 21,000,000 counts p-p, or 14Vrms. This actual example shows toggling only of the 1.9 µV LSB, or less than 1µV rms noise, or 143.7dB actual dynamic range

4730 **20sps** noise samples

. . .accurately timed

The figure below shows the uncorrected **drift** of the internal TCXO-derived **timebase** versus GPS time. The time stamp of recorded data are corrected using these measurements to maintain accuracy less than 1 **µs** relative to UTC. The ticks on the horizontal axis are days (total interval 20 days), and the vertical axis is microseconds. The maximum deviation in this period is 30 **µs,** while the RMS deviation is less than 10, and the long-term trend is zero. This data was taken from an actual deployed field station.



0730 clock drift in microseconds vs. time in **days**

Very affordable

Typical configurations of the Q730 are <u>very economical</u>. Contact Quanterra for a quote or to discuss special configuration requirements.

Q730 System Specifications

Specification	*Description*
Channels	3 standard, 3 additional optional. 3-channel groups separately galvanically and optically isolated for connection to separate sensors.
Sample Rate	20000 Hz, simultaneous. user rates: 250,125,200,100,50,40, 25,20,10,1Hz.
Resolution and Dynamic Range	LSB (Least Significant Bit) = 1.9µV. 148 dB max. (±10.5 $\times 10^6$ digital counts) Matched optimally to electronic broad-band force feedback sensors, e.g. STS-2
Bandwidth	O-O.8 Nyquist (-6dB point). Response controlled by digital FIR filter.
Noise and Distortion	Terminated input noise level typical -142 dBrmsrel to Full Scale, 0.25-50Hz. May exceed 146 dBrmsrel at sub-Hz frequencies and constant temperature.
Full Scale	±20V (40V p-p) differential input.
Operating Temp Range	-10 to 70 °C ambient temperature external to sealed enclosure.
Signal Processing	Digital. One fixed-point ADSP2 105 used per channel, one floating-point TMS320C3 1 master. Linear or minimum-phase FIR or IIR filters.
Sensor Calibration	Optional calibration/state-of-health module available for 3-channel sensor mass position acquisition, generation of mass-centering pulse, and sine/step/random noise sensor calibration signal. Plug-in module, field installable.
Timing	GPS C/A code. 1µsec accuracy to UTC. Position error 100 meter RMS. Sampling timebase phase-locked to GPS using software-controlled slew-limited low-distortion loop. Integral to 4730 processor module.
Main Processor	One 32-bit 10 MHz Motorola MC68EC030 microprocessor.
Recording Modes	Continuous or event, selectable by channel. Murdock-Hutt or STA/LTA detector
Parameter Setting	ASCII text "keys" stored in flash EEPROM set operating modes.
Memory	8Mb RAM on plug-in module. Optional additional 8Mb available. RAM used for program operation and temporary data storage. Up to 8Mb EPROM for permanent program storage.
Communications	2 asynchronous serial ports standard, 1 for terminal, others available for data transmission.
Networking	Supports standard TCP/IP and UDP/IP protocols, with remote login via telnet, and data transmission via finger and ftp. Email "alerts" sent to SMTP mail server.
Software	(*UltraSHEAR* or *MultiSHEAR*), resident in EPROM, fully installed.
Environmental	Sealed, polyethylene environmental enclosure. 9 X 16 X 23 in.
Auxiliary Monitoring	Ambient Temperature, Input DC power voltage digitized standard.
Construction	Fabrication to ANSI/IPC-A-600D Class 3. Single-board digital, factory replacement and repair only. Each 3/4 analog channel group on a single board. Field-replaceable individual analog-channel pre-processors.
Power	12VDC, 10-l 1 W average. Fully isolating DC/DC converters.

Bob Woodward

Nov. 14,1997

lo:15 AM

THE PROBLEM

DA-DP combination suffers various shortcomings because it is tied to design choices necessitated by the DA.

- some problems:
  - 0 unusual OS
  - 0 unusual language
  - o hard to take advantage of newer/faster CPU's
  - o hard to take advantage of new peripherals
  - o poor networking
  - o huge software support burden on Quanterra

ONE IDEA FOR A SOLUTION

Separate the hardware used for the “unique” and “routine” tasks performed by the DA-DP combination.

- DA: performs unique tasks

    o digitizes

    0 time stamps

    o emits data records

    0 only configure – never program

- DP: performs housekeeping chores

    o logs data

    o buffers data

    o networking

    o dial-up

    o AutoDRM

    0 etc.

DP WISHLIST

- robust operation

  0 never fails

  o never crashes

  o never loses data

- ease of programming and configuration

- easily upgraded and modernized

  o don’t lose software investment

PRACTICAL DESIGN GOALS FOR A DP

- COTS hardware
  - 0 low power
  - o rugged
- COTS software
  - ○ OS
  - 0 languages
- adapt to new peripherals
- adapt to new CPU’s
- does networking/WWW/modems well
- full suite of software tools
  - o development tools
  - o debugging tools

Hutt
11/14/97
12:30 PM

Making Data Recovery More Robust

- Recording Medium
  - More reliable tapes?
  - Removable hard disks?

    Jazz
    Zip
    Other?

- DA - DP Telemetry
  - Make telem. more reliable?
  - Increase storage capacity @ DA?

- Operational Changes
  - Make 2 of every data tape?
  - Much larger disks at station?
    (so data may be re-requested)
  -

Hutt

DP Discussion

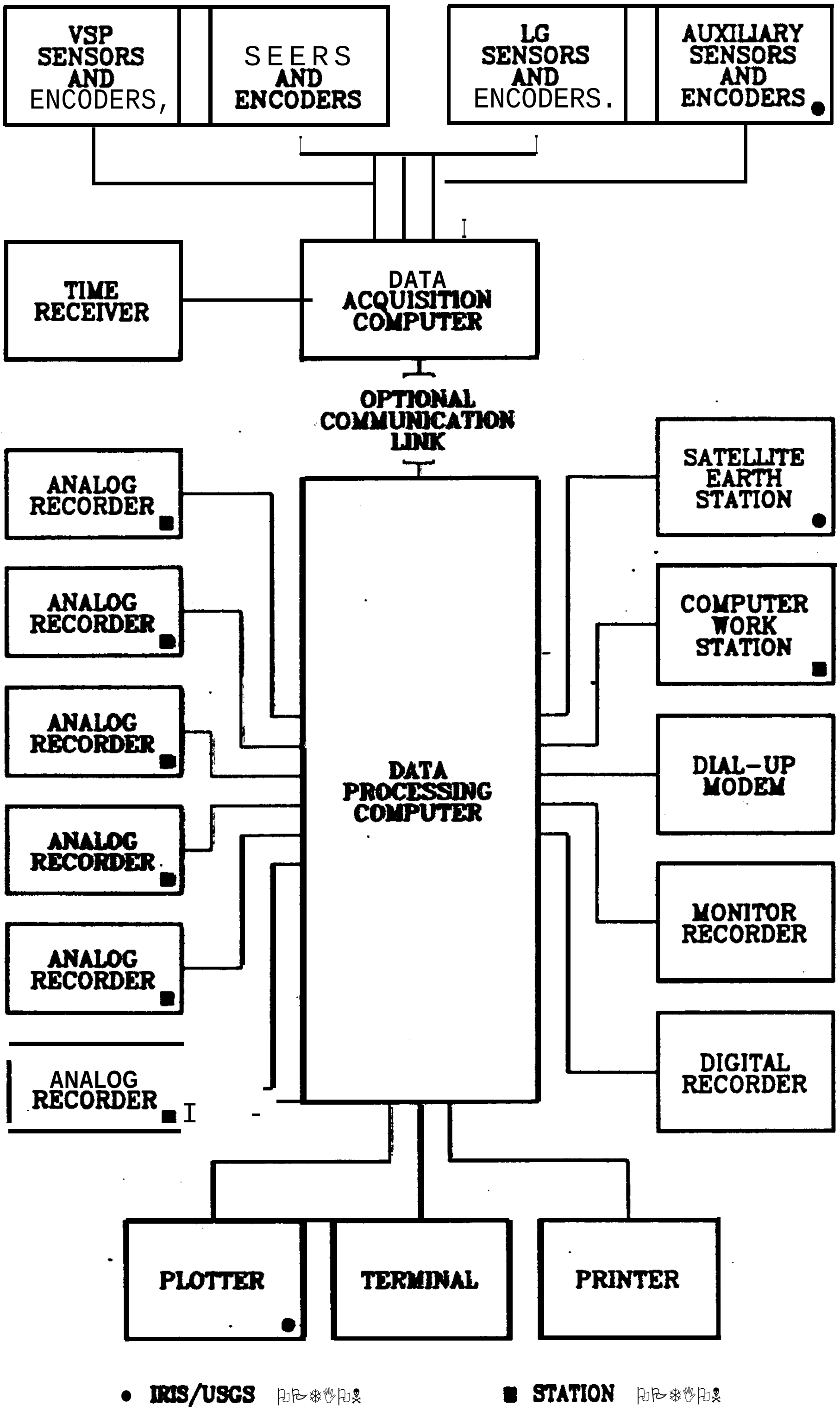


Figure S.-Block diagram showing major components of the IRIS-2 system. The data acquisition and data processing functions are combined in a single computer when a communication link is not used.

Bob Hutt

Nov. 14,1997

10:15 AM

To: gsnmaint@asl.cr.usgs.gov
From: Sue Mac <sue@asl.cr.usgs.gov>
Subject: Some thoughts on a UNIX DP
cc:
Bcc:
X-Attachments:

Much of what I have to say is based on my experiences in setting up SUN workstations (used only for data analysis) for the NCDSN stations and in working with Ed Lukens to trouble-shoot these systems for both NCDSN and GTSN stations.

The first consideration is the additional complexity of amintaining two entirely different systems in the field. Currently all of the Q/R and Q/L systems have the same editor, the same operating system amd the same backin/backout procedures. The system dependent files are the same or similar and half of the software runs on all the systems (the tape, online buffer and retrieve programs). This means that users logging into the IRIS/USGS systems don't have to know what kind of system they are logging into in order to retrieve data. The programs that ASL develops to access the online buffers (such as auto-helem and the network data requester) are the same for all systems.

A Q/R system with a non-OS9 DP will require learning and amintaining two different operating systems, two different editors, two different backin/backout procedures and sets of tapes, two different sets of system dependent files and two different sets of application software. It will require maintaining two different sets of documentation and taking twice the time to train the station operators.

The SUN systems require a careful shutdown procedure - if all of the files are not closed correctly, the system may not come up OK. This is not a robust design for places where it's a major accomplishment to get the operator to press the "red reset" button or where there are power outages.

The backin/backout procedures for the SUNs are more complex - because the disk(s) are partitioned, it requires one tape per partition. For the NCDSN and GTSN stations, a full station backup requires four tapes and about two hours to perform (as opposed to one tape and about fifteen minutes for an OS9 DP). Backing in the software takes even longer and you're in big trouble if you backin the wrong tape into the wrong partition!

The OS9 DP has been very robust due in part to the fact that all programs stay resident in memory and that the application software is designed to work with circular files instead of constantly creating and deleting files. This ensures that once the system has been initialized and is running correctly, the system will not run out of disk space or memory. The SUN systems generate system level files which need to be deleted on a regular basis and are designed to swap out memory to disk. The data center application software also creates and deletes files. There is the potential to run out of memory and/or disk space as the system is running. The UNIX system is much more complex - and there is much more than can go wrong. Ed Lukens has spent a lot more time trouble-shooting the GTSN Sun workstations than I have the DPs.

For the installation of the NCDSN systems (over a duration of about five years), I had to deal with three different SUN workstation platforms - IPC, SUN Classic and a SPARC5. It required the help of the resident station UNIX programmer to help me make the upgrade as the back-out tapes were not compatible between systems (even the backin/backout procedures had to be modified). In the mean time SUN had come out with several new versions of the UNIX operating system including a major change between SUN OS and SOLARIS. We didn't have the resources (time, source code and knowledge) to make the upgrade - however some of the hardware delivered (such as the SPARC5 video cards) could not be run under the old operating system. The problems encountered with the upgrade to SPARC5 in particular caused a delay of several months.

Sue

Sender: ed@asl.cr.usgs.gov
Date: Tue, 04 Nov 1997 13:28:31 -0700
From: Ed Lukens <ed@asl.cr.usgs.gov>
Organization: Albuquerque Seismological Laboratory
To: gsnmaint@asl.cr.usgs.gov
cc: sue@asl.cr.usgs.gov, ed@asl.cr.usgs.gov
Subject: [Fwd: Some thoughts on a UNIX DP]

I would like to add some more thoughts to the referenced message "Some thoughts on a UNIX DP":

1. Unix systems, in general, require a proactive approach to system administration. Someone needs to look for potential problems and know where to look. For example, disk clearing and maintenance jobs could be scheduled through a cron. However if the disk fills up because the cron daemon died then someone needs to know how to correct this.

2. When Unix systems begin to perform poorly, finding a solution usually requires a careful analysis of the processes that run in a typical day and their usage of resources. Someone needs to be proficient in the use of performance analysis tools such as sar, vmstat, netstat, etc. A example of this here in the GTSN is when an auto cal for a particular station is scheduled from our main server (through a cron). If the station is down at the time the auto cal is requested, the gcmd processes sit in the process queue trying to connect to the station periodically. The end result: the load averages on the server increase significantly and stay that way until someone physically kills the gcmd processes for that station.

3. Over time, I have tried to train the operators of the GTSN field Suns in Unix and Oracle troubleshooting. However, the most difficult concept to transfer is that a particular problem on a Unix system can have one to many causes and one to many solutions. For example, the e save program is a popular tool on the GTSN and China field Suns. This program is used to store an event in a separate partition so it will not get removed from the normal data spooling area. An "event" data object consists of a waveform file in CSS format and a header in the form of an Oracle database row. In the past, the field station operators will go into the e save directory and remove all of the waveform files without removing the database rows that reference these files. Eventually the database table fills to capacity even though there is plenty of Unix disk space for the raw waveform files. On the the other hand, the waveform disk space could fill to capacity but the Oracle database has plenty of free space. To the user, these two exceptional conditions appear to be the same

thus he attributes them to the same cause. The users will usually contact me and say that e save does not work even though there is plenty of e-save disk space. Since they do not have training in relational databases and SQL, they do not typically go into the database to check its free space or the amount of fragmentation (another possible cause) in the tablespaces.

Ed

Karl Jaeckel

Nov. 14, 1997

1:00 PM

Speckel
14.11.97
~~12:30~~ 1PM

1

Seismological Communication Processor - SeisComP

Motivation:

GEOFON needs station processors for the installation of various “distributed” stations and seeks for a flexible, future oriented, open solution for this task.

Concept:

* flexible configurable- and upgradable system,
* usage of cheap, worldwide available hardware,
* usage of standard protocols and software,
* integration of well known, open software packages,
* usage of most modern and/or adequate communication and storage technics,
* support of several parellel ADCs (network processor),
* usage for permanent and mobile stations,
* simple hard- and software upgrades,
* cooperation with other institutions.

Solution: Usage of normal PCs with Linux operating system and integration of open software

Jaeckel
14.11.97
12:30
2

Seismological Communication Processor - SeisComP

Hardware:	*Minimum*	*GEOFON-Systems*
	486 with FPP	Pentium (200MHz)
	8 MB memory	32 MB memory
	400 MB disk	2 GB disk
	2 add. serial ports	8 add. serial ports
	(or Ethernet)	SCSI
		Ethernet
		DAT tape drive
		UPS
	optional:	ISDN
		Modem(s)

For field usage (operation on batteries): Laptop with power save option needed (also under Linux)

Jaeckel
14.11.97
12:30
3

Seismological Communication Processor - SeisComP

Software:

Standard: Linux 2.0 incl.
- X-Windows
- TCP/IP
- PPP
- Kermit
- HTML, Java

Tools:	Comserv 1.0	(Quanterra, UC Berkeley)		*
	SeedStuff	(GFZ)		*
	Passcal-Software	(IRIS-PASSCAL)	> ported to Linux (GFZ)	*
	rdseed, weed	(IRIS-DMC)		(*)
	AutoDRM	(ETH)	(Stuttgart)	(*)

Object management: Client-server architectur
- Data Request Manager
- Station Operation Manager
- Adaption moduls
- Perl-Scripts
- FISSURES ??

Jaeckel
14.11.97
12:30
(3)

Seismological Communication Processor - SeisComP

Software:

Standard: Linux 2.0 incl.
X-Windows
TCP/IP
PPP
Kermit
HTML, Java

Tools:

Comserv 1 .O	(Quanterra, UC Berkeley)			*
SeedStuff	(GFZ)			*
Passcal-Software	(IRIS-PASSCAL)	> ported to Linux (GFZ)		*
rdseed, weed	(IRIS-DMC)			(*)
AutoDRM	(ETH)		(Stuttgart)	(*)

Object management: Client-server architectur
Data Request Manager
Station Operation Manager
Adaption moduls
Perl-Scripts
FISSURES ??

Jaeckel
14.11.97
12:30

4

Seismological Communication Processor - SeisComP

Basic system:

Server:	Communication with station(s) (comserv)	*
	Message server (msgserv)	
	Email server (mailserv)	
Clients:	Network monitor (netmon)	*
	Message monitor (msgmon)	*
	Command tool (cmds)	*
	Data reading (dataread)	*
	Data logging (datalog)	*
	Data storage (datadump)	*
	Data plotting (dataplot)	
	Data charts (chart)	
	Message printer (msgprint)	
	Modem control (mdmcntr)	

* already implemented

Jaerbel
14. 11. 97
12:30

5

Seismological Communication Processor - SeisComP

Data Request Manager (version 1):

* Listing of available data
* Selection of time windows from different data streams
* Format conversion (Miniseed, SEED, ASCII, GSE, SAC, CSS, AH...)
* Transfer of selected data (ftp, kermit, direct)
* Visualisation of selected data (pql)
* Listing of message, detektion, timing, calibration logs

Station Operation Manager (version 1):

* Dialog with ADC
* ADC and SeisComP configuration
* Downloading of software into ADC
* Control of SeisComP clients
* Visual data and SOH monitoring
* Tape and disk storage control
* X monitors for log control

Later (version 2):

* Network Email Service (time depended transfer of informations and data)
* Network Data Service (transfer of informations and data from other stations or network DCCs)

22:30

6

```
        Welcome to the SeisComP Station Operation Manager
         at GII Data Collection Center in Holon, Israel
                   SeisComP Version 0.5

Data for station(s) EIL + FDN + JER are available on this SeisComP
system.

Please enter your initials ---> wh

Select  station ---> EIL

  SOM Main Menue

    a  - Start/stop comserv data aquisition
    c  - Connect to Quanterra master console
    d  - Download USHEAR system to Quanterra
    l  - Start/stop X-monitor
    m  - Network mail service
    o  - Escape to OS9 shell
    q  - Quit
    r  - Retrieve data from other network stations/DCCs
    s  - SeisComP setup tool
    t  - Tape control
    u  - Start user DRM
    v  - View continuous data stream
    w  - Switch to other station

Command  --->
```

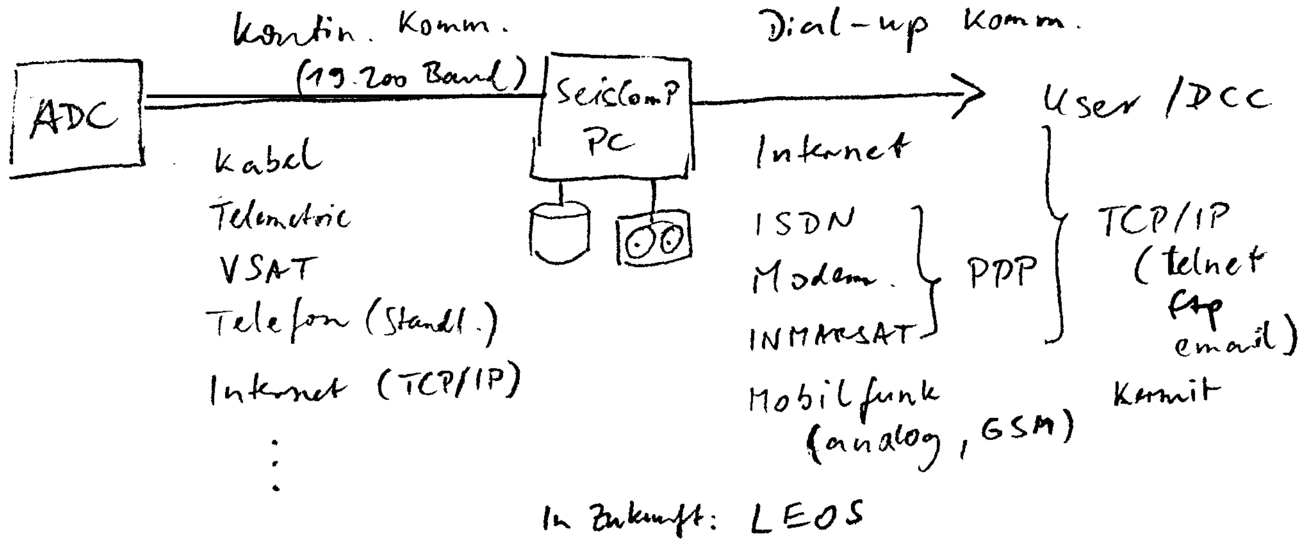
10:30

7

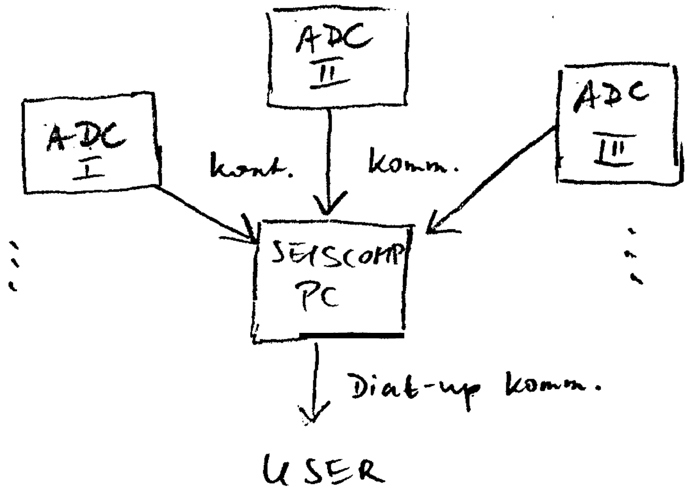
```
        Welcome to the SeisComP Data Request Manager
       at GII Data Collection Center in Holon, Israel
                SeisComP Version 0.5

 *** Please note that this software is still under development ***
      Please report problems to ruesing@gfz-potsdam.de

Data for station(s) EIL + FDN + JER are available on this DRM.

Please enter your name and institution:
---> Winfried Hanka, GFZ Potsdam

Select station ---> EIL

 DRM Main Menue for station EIL

   e       - Examine available data on disk
   d       - Download selected data
   l[cdt]  - List message [calib, detection, timing] log
   m       - Mail message to station & network operators
   q       - Quit
   s       - Select time window and extract data
   v       - View selected data

Command --->
```

12:30

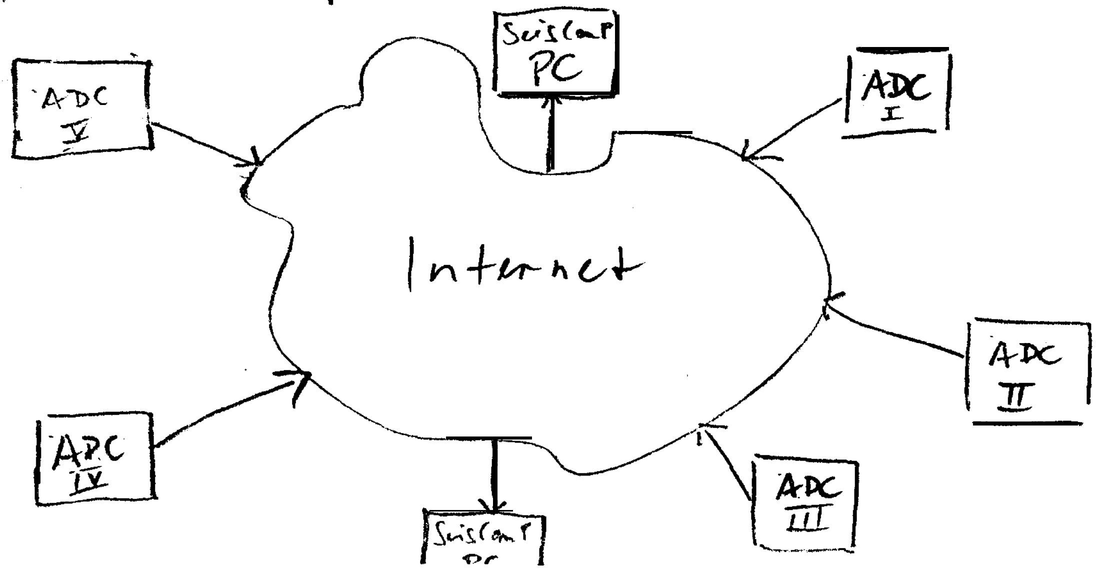
① Einzelstation (Open Station oder Feldstation)



② Stationsnetz (Open Network – Open DCC)



③ Internet-Netz (Open Station and Network/DCC)



Bob Woodward

Nov. 14,1997

2:00 PM

Near-Real-Time Stations of the IRIS/USGS Network

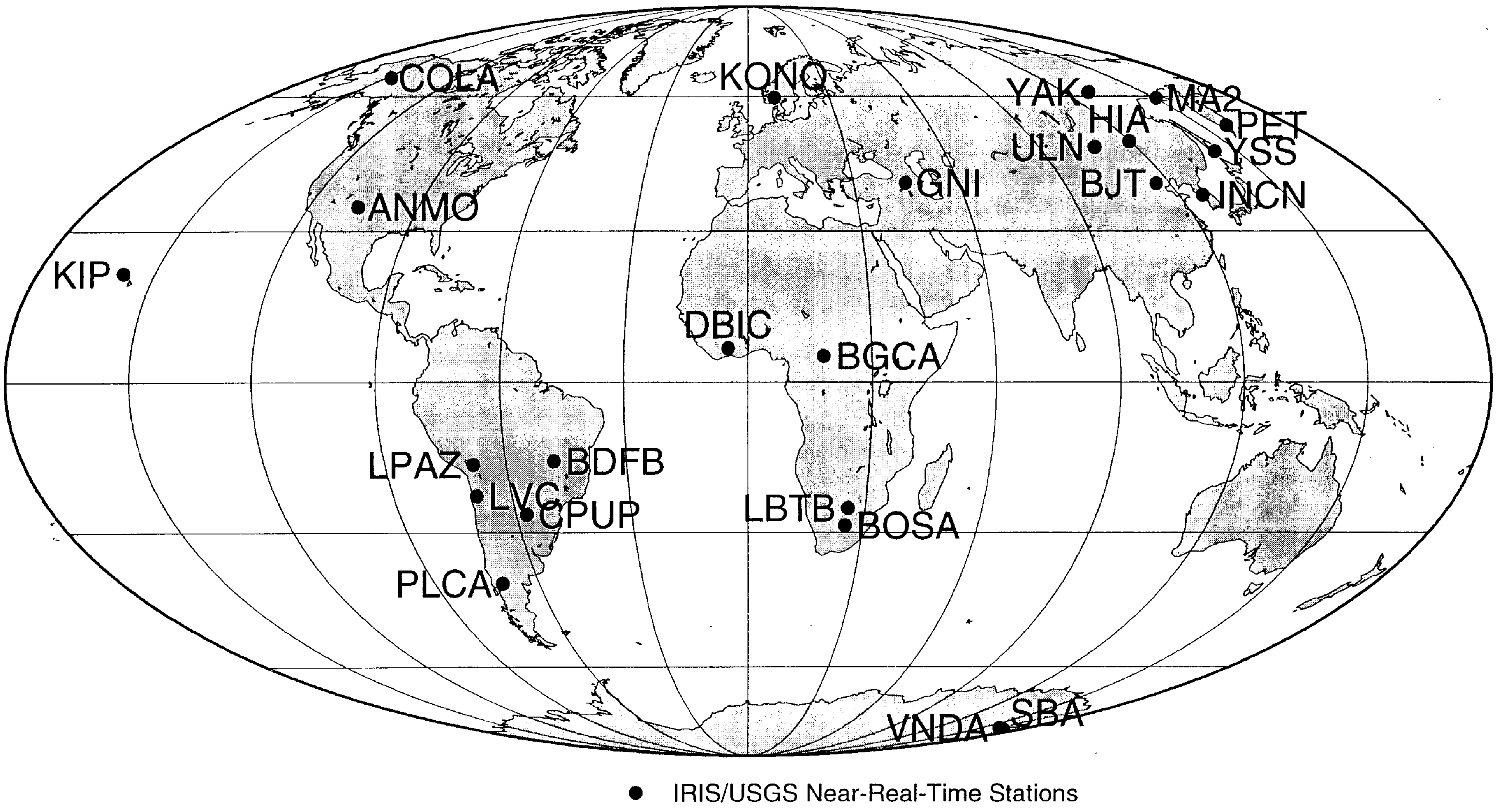


Figure 3

USGS Albuquerque Seismological Laboratory

September 1997 (RLW)

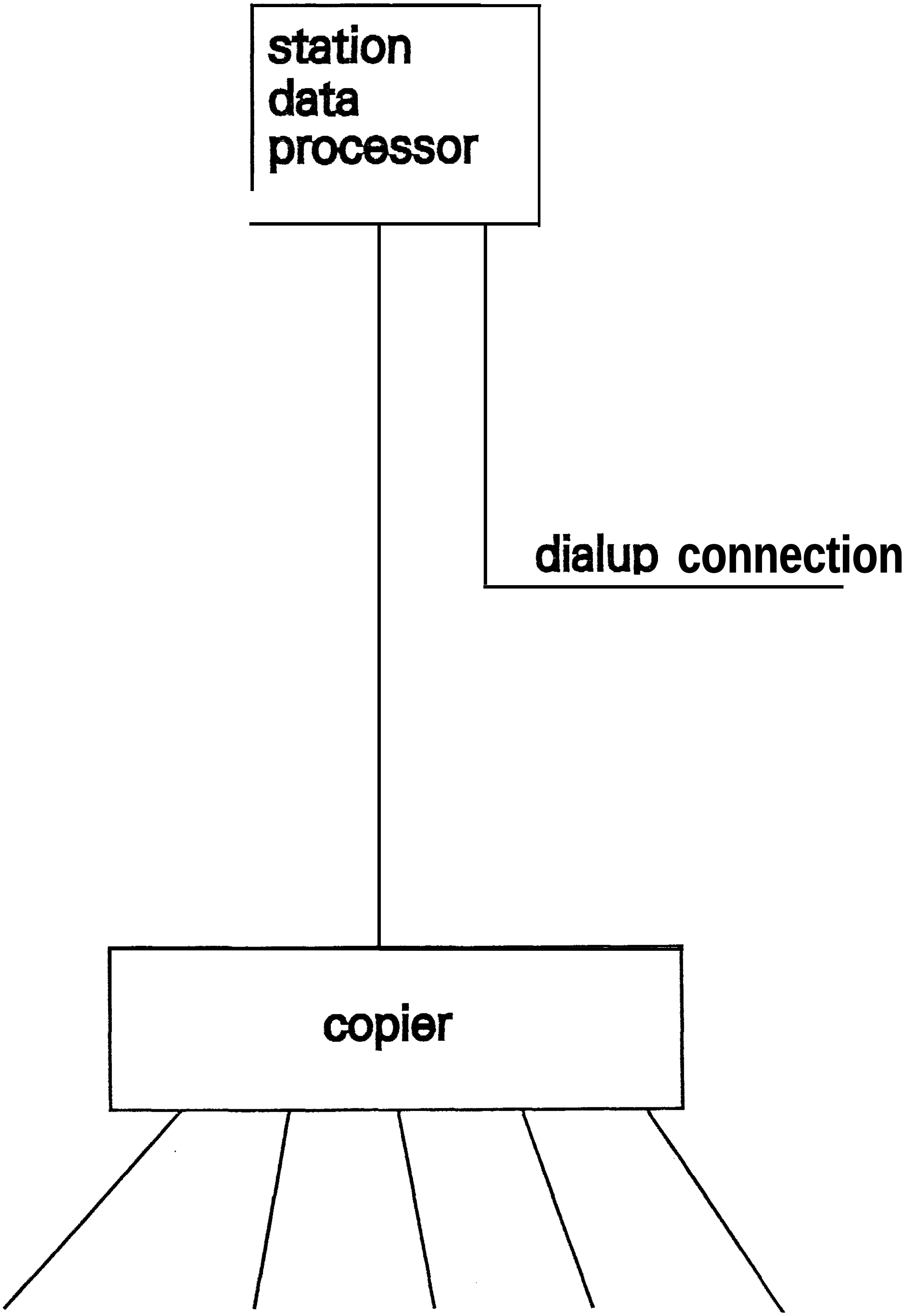


Diagram showing the basic design of the Albuquerque Seismological Laboratory's near-real-time data copier system. The station data processor (DP) sends data records to another computer which is running the copier software. The "copier" computer can be co-located with the DP, or could be located thousands of miles away and the two speak via the Internet. The copier accepts connections from an unlimited number of clients. As each data record is received from the DP, copies of the data record are immediately made for each client.

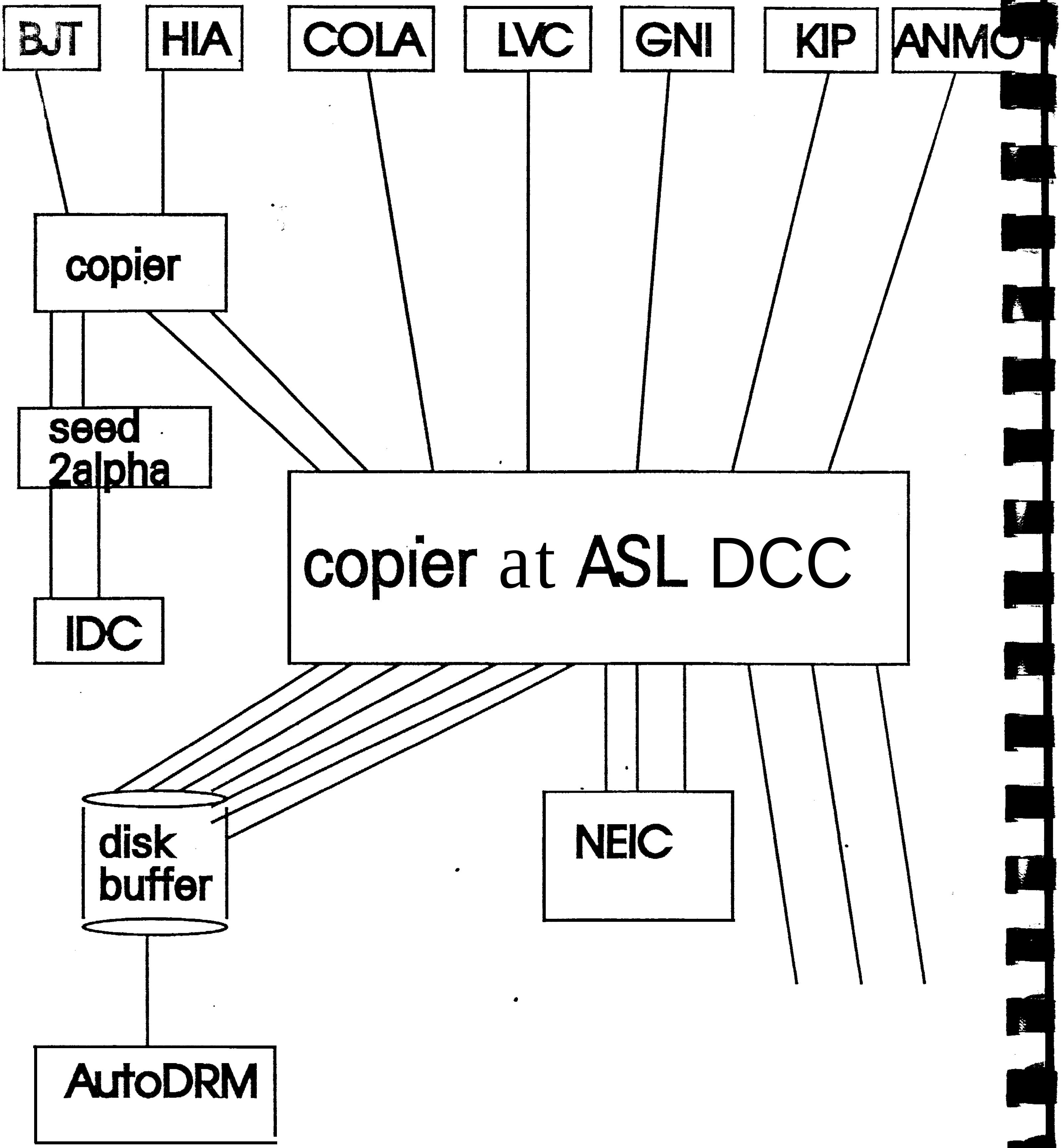
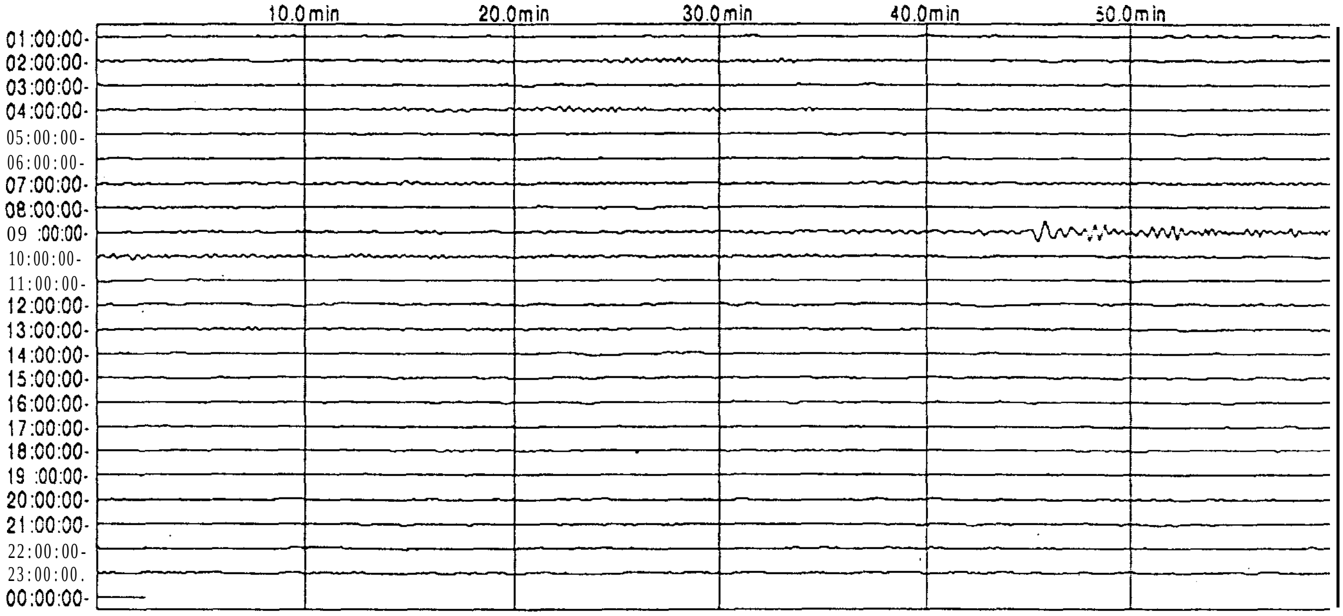


Illustration of the full copier system presently in place at the Albuquerque Seismological Laboratory.

Data from station GNI (Garni, Armenia)

last updated at

Fri 06/06/97 18:09 MDT (Sat 06/07/97 00:09 GMT)



GNI,LHZ Start Date:06.06.97 Filter: band-pass Displacement Magnification =3000.00 @ 0.020 Hz

Example of an application which is receiving near-real-time (SRT) data from the data copier running at the Albuquerque Seismological Laboratory. In this application, a process runs continually, receiving and storing the LHZ data (vertical component, 1 Hz sampling) from the near-real-time stations. Every ten minutes this process makes a helicorder style plot of the preceding 24 hours of data for each NRT station. In this example, the plot was made at 00:09 GMT, and contains data through 00:03 GMT (i.e. data less than ten minutes old). These plots can be found at:

http://aslwww.cr.usgs.gov/Seismic_Data/tele.htm