

# GLOBAL DIAGNOSTICS SYSTEM FINAL DESIGN REVIEW

Rolf Bork, Peter Fritschel, David Shoemaker,  
Daniel Sigg and John Zweizig

## Agenda

- 20' Introduction/Overview (David S.)
- 30' Data Viewer and JDclient (Rolf B.)
- 60' Diagnostics Tests (Daniel S.)
- 20' Analysis Algorithms (Peter F.)
- 60' Data Monitor Tool (John Z.)
- 10' Schedule/Costs (Daniel S.)
- 10' Answer to PDR action items (Daniel S.)

May 7, 1999



# Global Diagnostics Goals

---

- Assist the operators in the control room and in the experimental areas to successfully run the machine.
- Provide immediate answers:
  - > What are the quality of the GW data written to disk?
  - > Are all of the subsystems working properly?
- Establish & automate diagnostics procedures.
- Give assistance to:
  - > learn about the behavior of the instrument,
  - > classify abnormal environmental events,
  - > identify the exact machine state,
  - > correlate the signals of different sensors and,
  - > ultimately, reduce the large amount of measured data to a set of relevant and comprehensible statistical quantities.

# Design principles

---

## **Scalable**

- can grow with users
- can grow with experience
- incrementally useful

## **Maximally independent modules**

- to simplify interfaces within GDS (Global Diagnostics System)
- again, ease staged implementation

## **Powerful infrastructure**

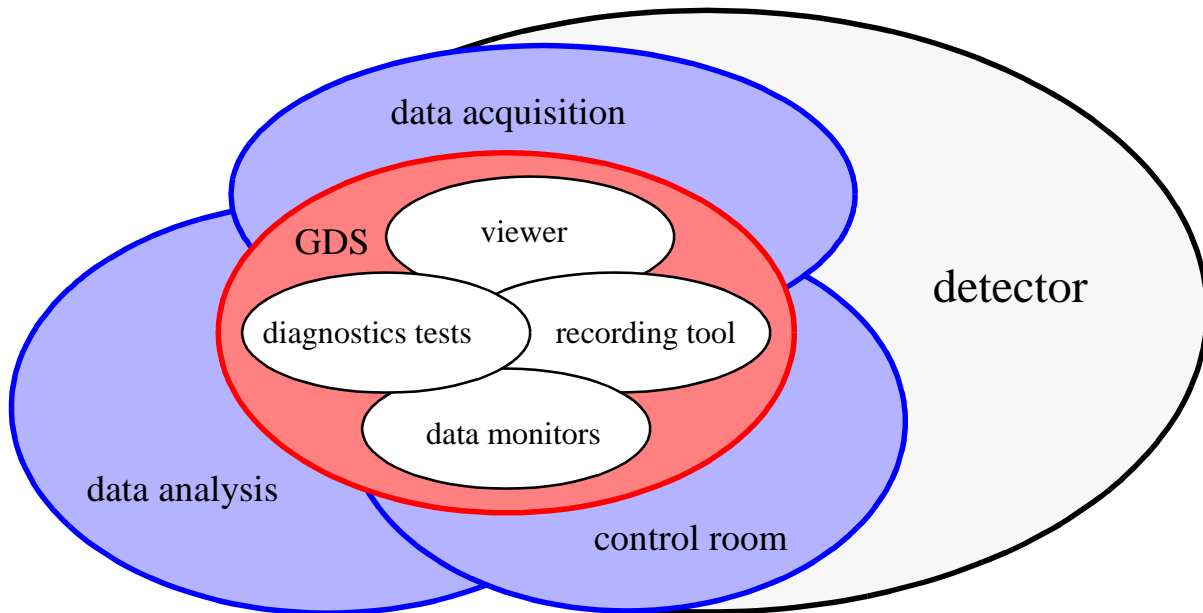
- access to all data in 'real time'
- processing power to ingest, evaluate, reduce all data in 'real time'
- analysis tools to give 'HP-box' versatility and ease-of-use

## **Expandable**

- aid in creating reduced datasets
- closed-loop stimulus-response
- system identification/adaptive control
- element in GW data analysis

# Scope

---



- Viewer based on Xmgr (shared with DAQS)
- Recording tool or 'JDClient' (for e.g., trend frames)
- Diagnostics tests (sequences of measurements on interferometer)
- Data monitors (the data reduction needed for a control panel)
- Computing hardware for analysis, signal generators

## Does Not Include

- data acquisition system (CDS)
- cameras, oscilloscopes, PEM-related equipment (detector subsystems)
- archived data analysis (LDAS)

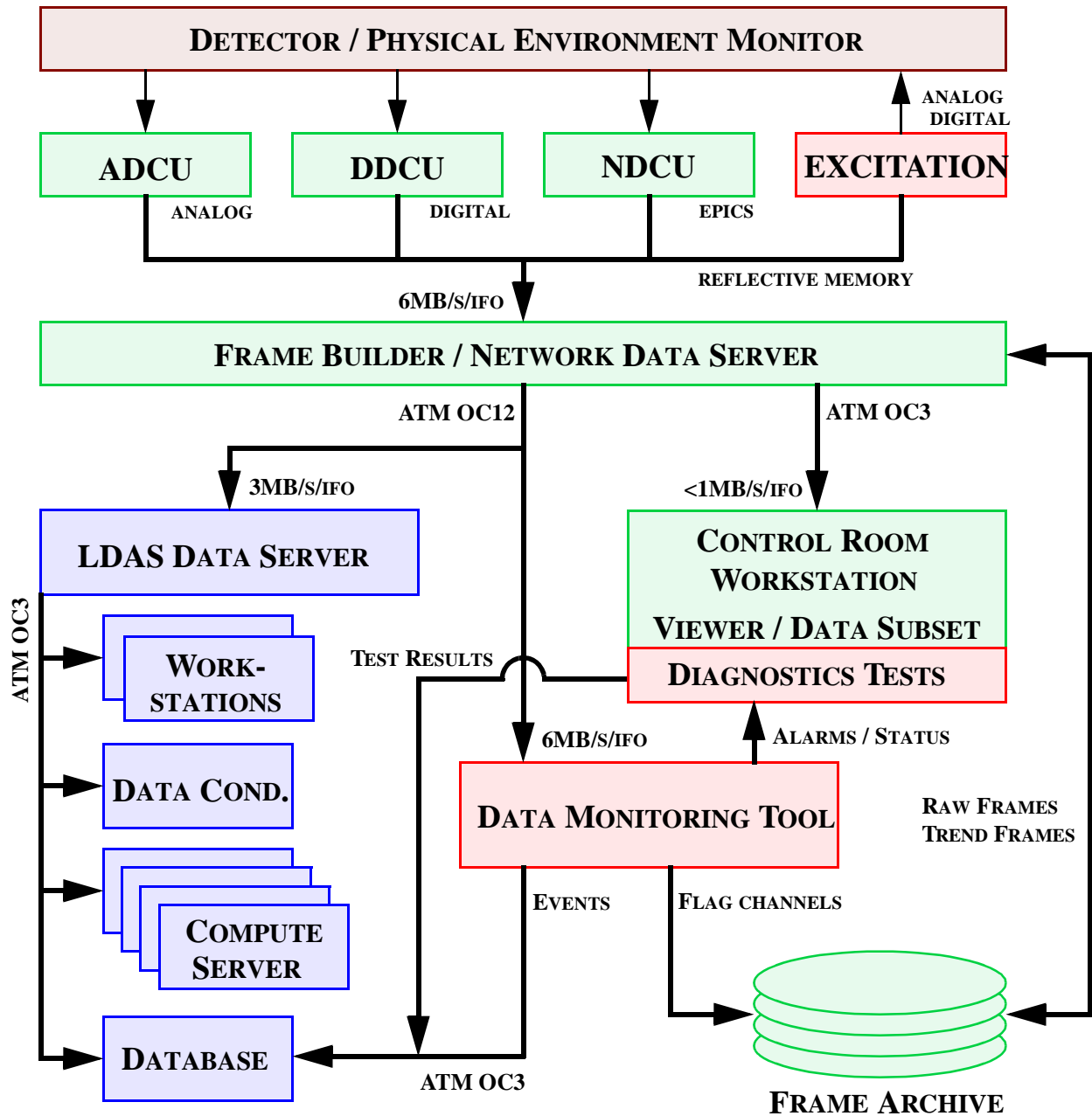
# Changes from PDR

---

## **Many interfaces to LDAS and DAQ simplified; overlaps eliminated**

- The data monitoring tool (formerly known as search tool) has been moved from a real-time CPU to a (SUN) workstation which receives the data from a network data server.
- The compute engine at the front-end has been eliminated. Instead, the data monitor tool makes use of a symmetric multi-processor machine to do the number crunching.
- The real-time data distribution unit has been eliminated in favor of transferring the data to the diagnostics test tool through the network data server.
- The conceptual design of the software has been advanced significantly; most of the software for the excitation engine and the diagnostics kernel have been written.
- **Lots done.**

# GDS in the Data Handling System



# Diagnostic Tests

---

## What kinds of tests?

- Sensing noise; e.g., coupling of laser frequency noise to the strain output
  - > apply swept sine excitation to laser frequency modulation input
  - > record LSC digital test point corresponding to frequency input to determine net modulation
  - > record LSC digital test point corresponding to strain output
  - > form transfer function, visualize, compare with models
  - > record quiescent frequency noise; predict influence on strain output
  - > calculate correlation (coherence) with strain output
- Optimization of optical phase sensitivity; e.g., optimization of the RF Modulation index
- Noise due to random forces; e.g., coupling of seismic noise to suspended component motion
- Minimization of random force noise; e.g., search for 'sweet spot' on suspended component
  - > apply excitation from LSC test point to suspension controllers
  - > correlate with synchronous strain output
  - > also, GW sensitivity level (with help from the Data Monitor Tool)
  - > ...can also step a parameter (beam position), remeasure, optimize
- Tests of the facility-detector interface; e.g., correlation of facility monitors with interferometer signals
  - > ...need access to FMCS signals!

# Data Monitors

---

- designed to give the look and feel of a sports car, not a freight train
- good feedback to operator
- allowed to take significant computation
- data input/output such that interfaces and re-use of output easy

## What kinds of Monitors?

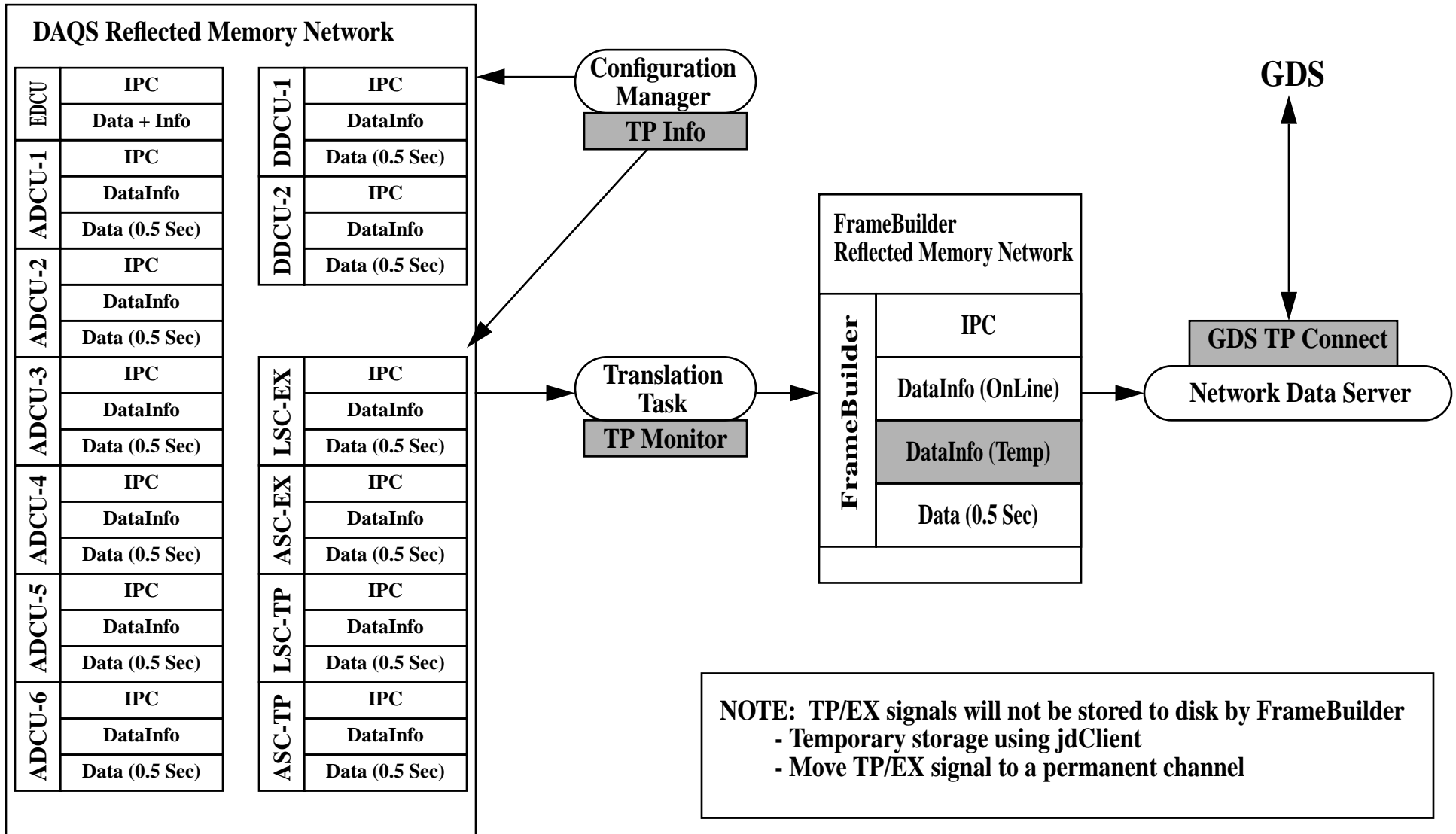
- Detect and tag known terrestrial signals or disturbances to the interferometer, *e.g.*,
  - > seismic activity: form a combination of seismometer, accelerometer, and tiltmeters which best reflects interferometer sensitivity (first guess, then refined notion with experience)
- Search for pathological conditions, *e.g.*,
  - > servo gain peaking.
- Gauge current state or performance of the instrument, *e.g.*,
  - > measure in-spiral observation volume
  - > (can be used in a diagnostic test as part of a parameter search)
- Check data integrity, *e.g.*,
  - > repeated data
- Look for broken channels
  - > rms = 0, or spectral feature indicating oscillation, etc.



# **DAQS Support for GDS**

- Software Changes**
- Hardware Changes**
- Data Viewing and Extraction Tools**

# DAQS Software Changes to Support GDS

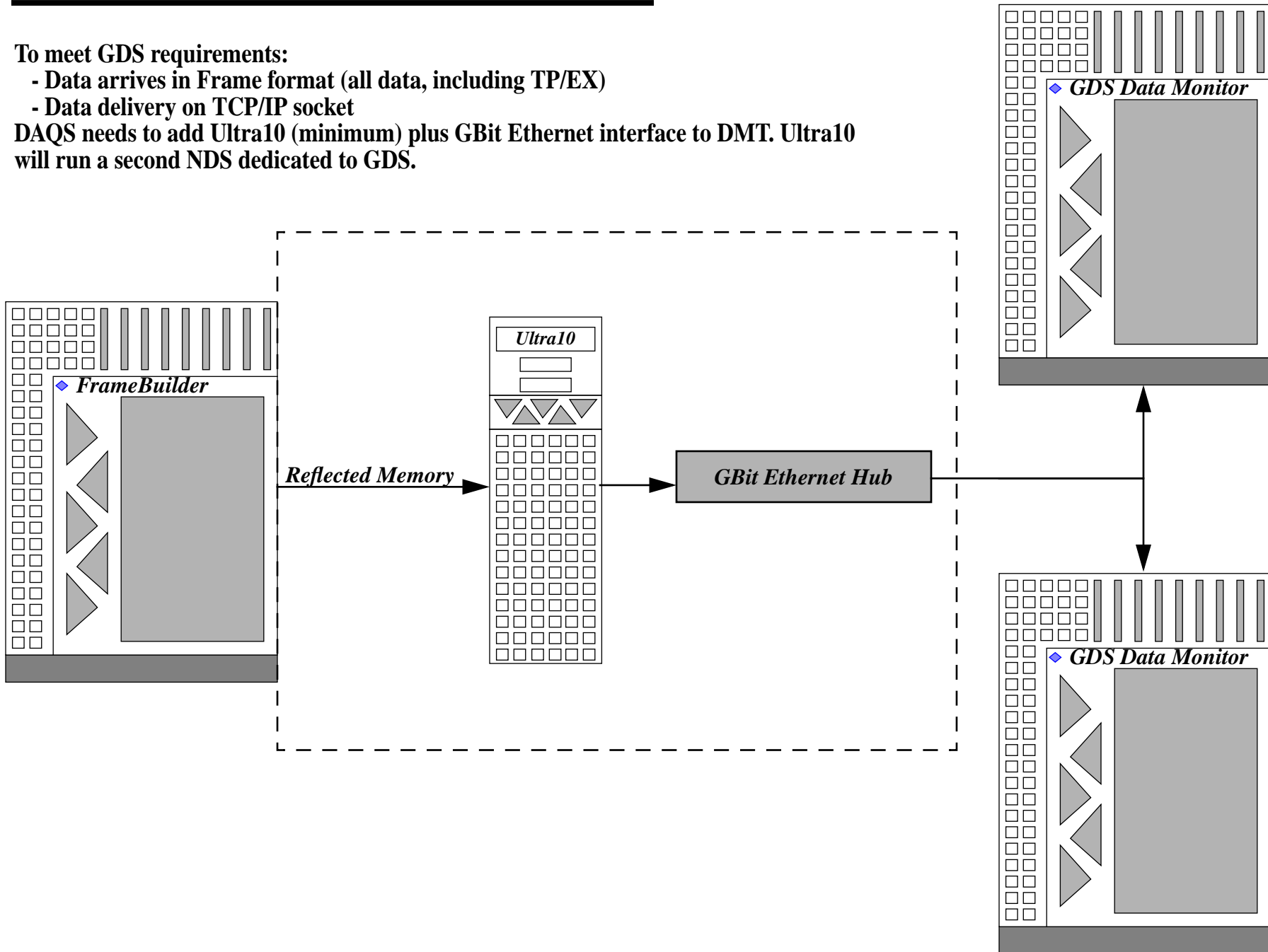


# DAQS Hardware Additions to Support GDS

To meet GDS requirements:

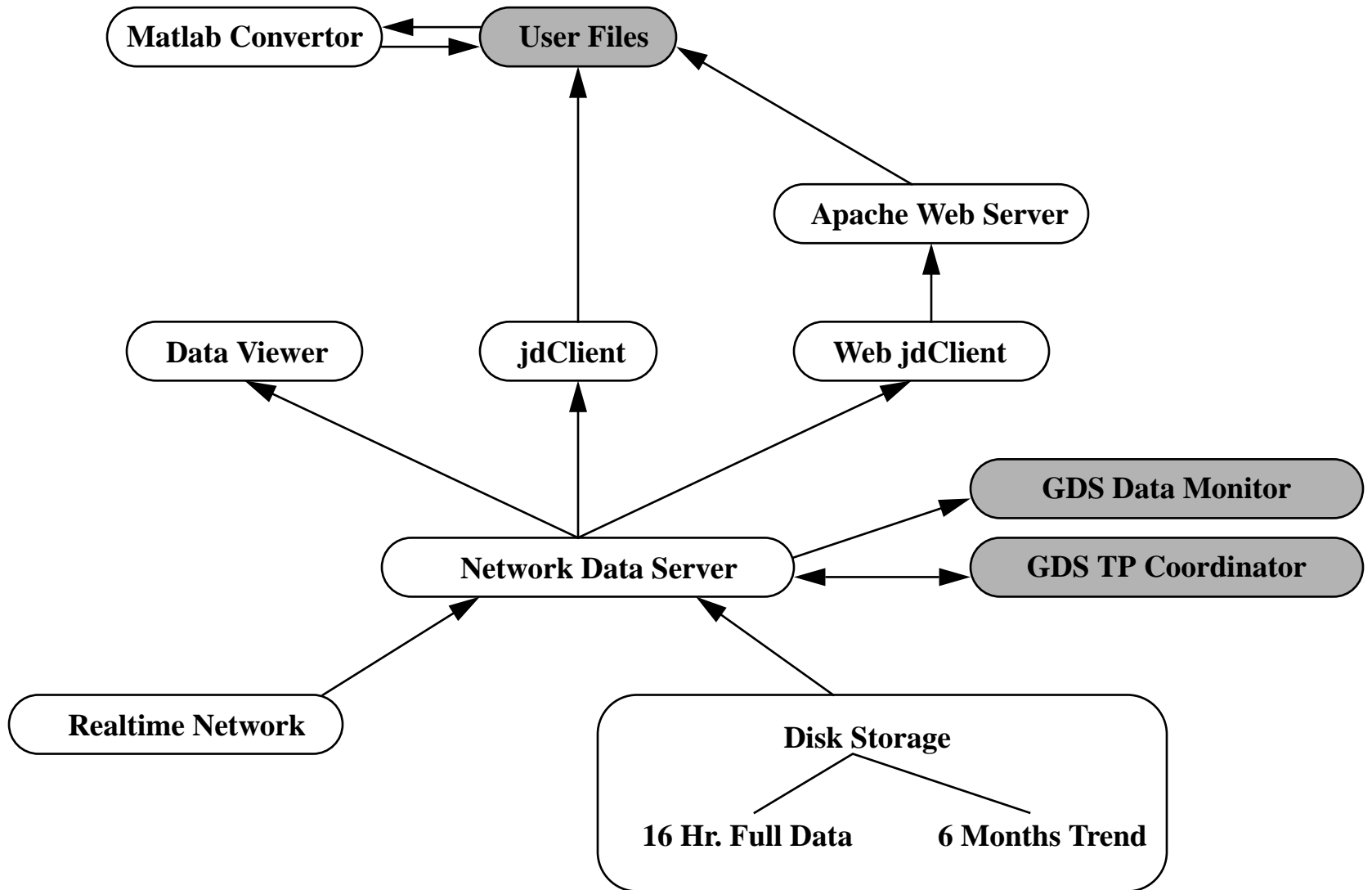
- Data arrives in Frame format (all data, including TP/EX)
- Data delivery on TCP/IP socket

DAQS needs to add Ultra10 (minimum) plus GBit Ethernet interface to DMT. Ultra10 will run a second NDS dedicated to GDS.

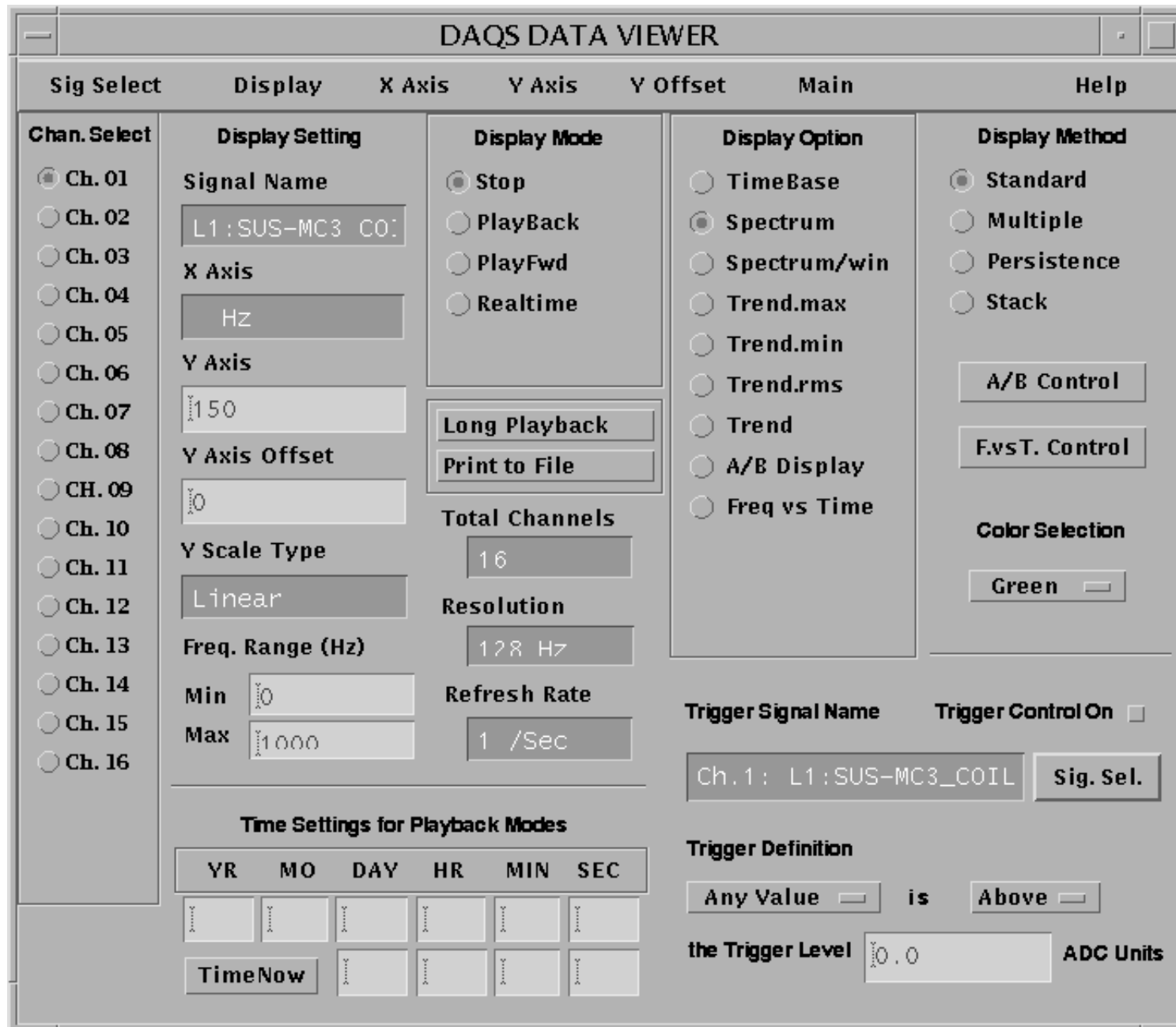


# DAQS Data Viewing and Extraction Tools

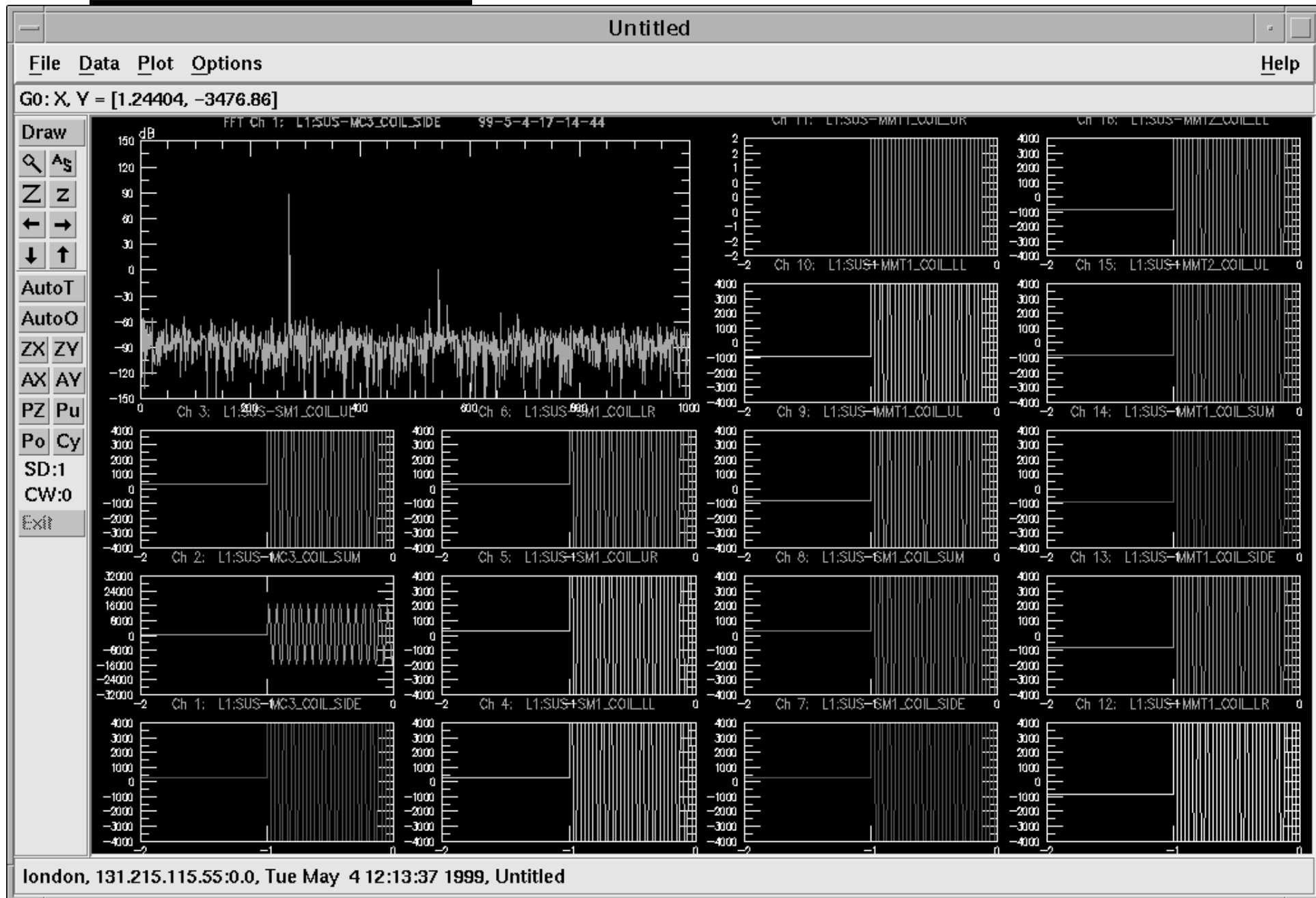
---



# Dataviewer Main Panel



# Time Series Display Mode



# Signal Selection Display

**Signal Selection**

Current Selected Channels	Group Names	Signal Names
Ch. 1: L1:SUS-MC3_COIL_SIDE (2048)	0: 4k_DAO	L1:SUS-MC1_COIL_UL (2048)
Ch. 2: L1:SUS-MC3_COIL_SUM (16384)	1: 4K_PSL	L1:SUS-MC1_COIL_LL (2048)
Ch. 3: L1:SUS-SM1_COIL_UL (2048)	2: 4K_IOO	L1:SUS-MC1_COIL_UR (2048)
Ch. 4: L1:SUS-SM1_COIL_LL (2048)	3: 4K_ASC	L1:SUS-MC1_COIL_LR (2048)
Ch. 5: L1:SUS-SM1_COIL_UR (2048)	4: 4K_SUS	L1:SUS-MC1_COIL_SIDE (2048)
Ch. 6: L1:SUS-SM1_COIL_LR (2048)	5: 4K_LSC	<b>L1:SUS-MC1_COIL_SUM (16384)</b>
Ch. 7: L1:SUS-SM1_COIL_SIDE (2048)	6: 4K_GDS	L1:SUS-MC2_COIL_UL (2048)
Ch. 8: L1:SUS-SM1_COIL_SUM (16384)	7: Future1	L1:SUS-MC2_COIL_LL (2048)
Ch. 9: L1:SUS-MMT1_COIL_UL (2048)	8: Future2	L1:SUS-MC2_COIL_UR (2048)
Ch. 10: L1:SUS-MMT1_COIL_LL (2048)	9: Future3	L1:SUS-MC2_COIL_LR (2048)
Ch. 11: L1:SUS-MMT1_COIL_UR (2048)	10: Future4	L1:SUS-MC2_COIL_SIDE (2048)
Ch. 12: L1:SUS-MMT1_COIL_LR (2048)	11: Future5	L1:SUS-MC2_COIL_SUM (16384)
Ch. 13: L1:SUS-MMT1_COIL_SIDE (2048)	12: Future6	L1:SUS-MC3_COIL_UL (2048)
Ch. 14: L1:SUS-MMT1_COIL_SUM (16384)	13: Future7	L1:SUS-MC3_COIL_LL (2048)
Ch. 15: L1:SUS-MMT2_COIL_UL (2048)	14: PEM	L1:SUS-MC3_COIL_UR (2048)
Ch. 16: L1:SUS-MMT2_COIL_LL (2048)	15: VACUUM	L1:SUS-MC3_COIL_LR (2048)
		L1:SUS-MC3_COIL_SIDE (2048)
		L1:SUS-MC3_COIL_SUM (16384)

New signal selected for Ch.

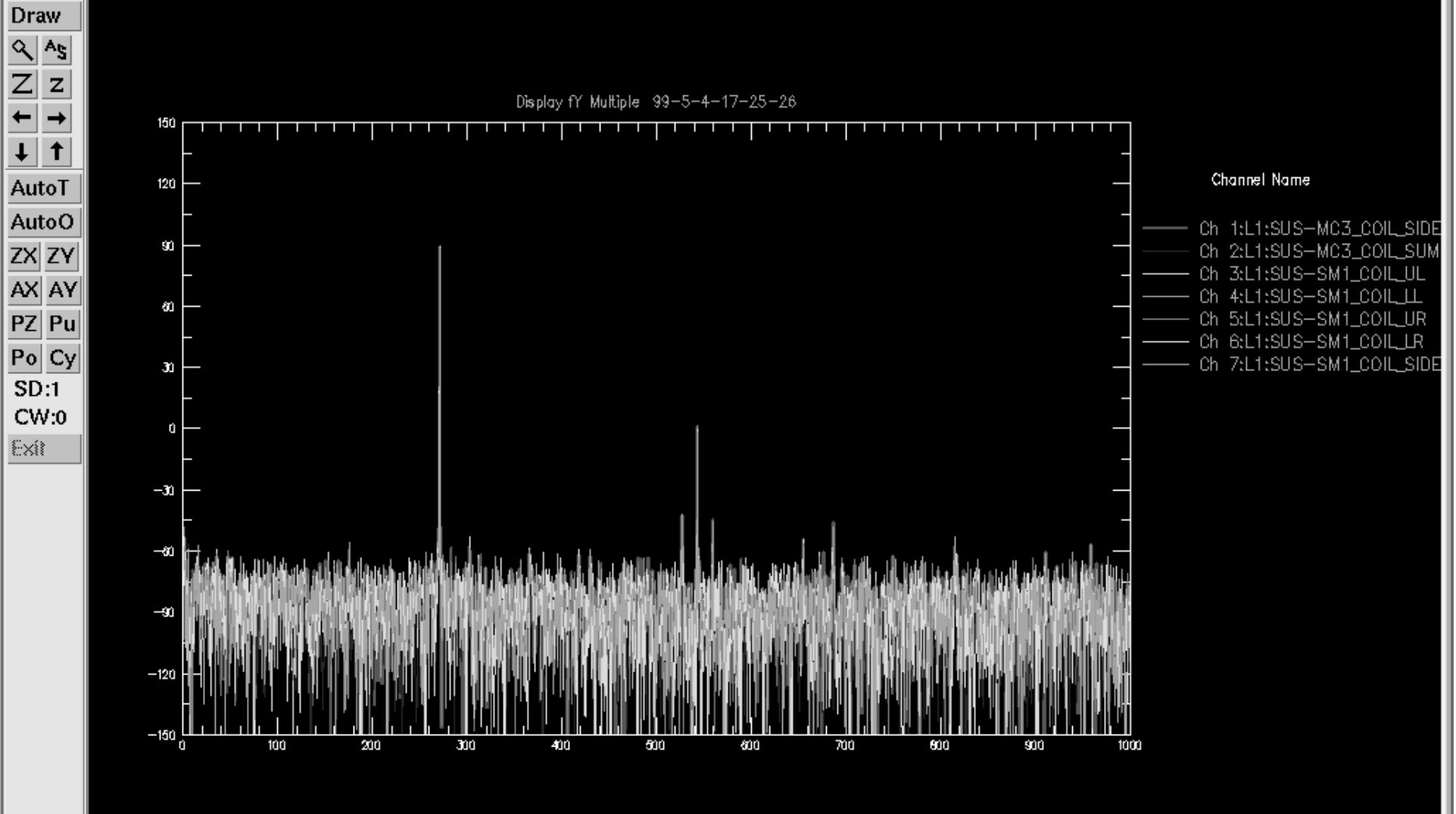
# Overlay Display Mode

Untitled

File Data Plot Options

Help

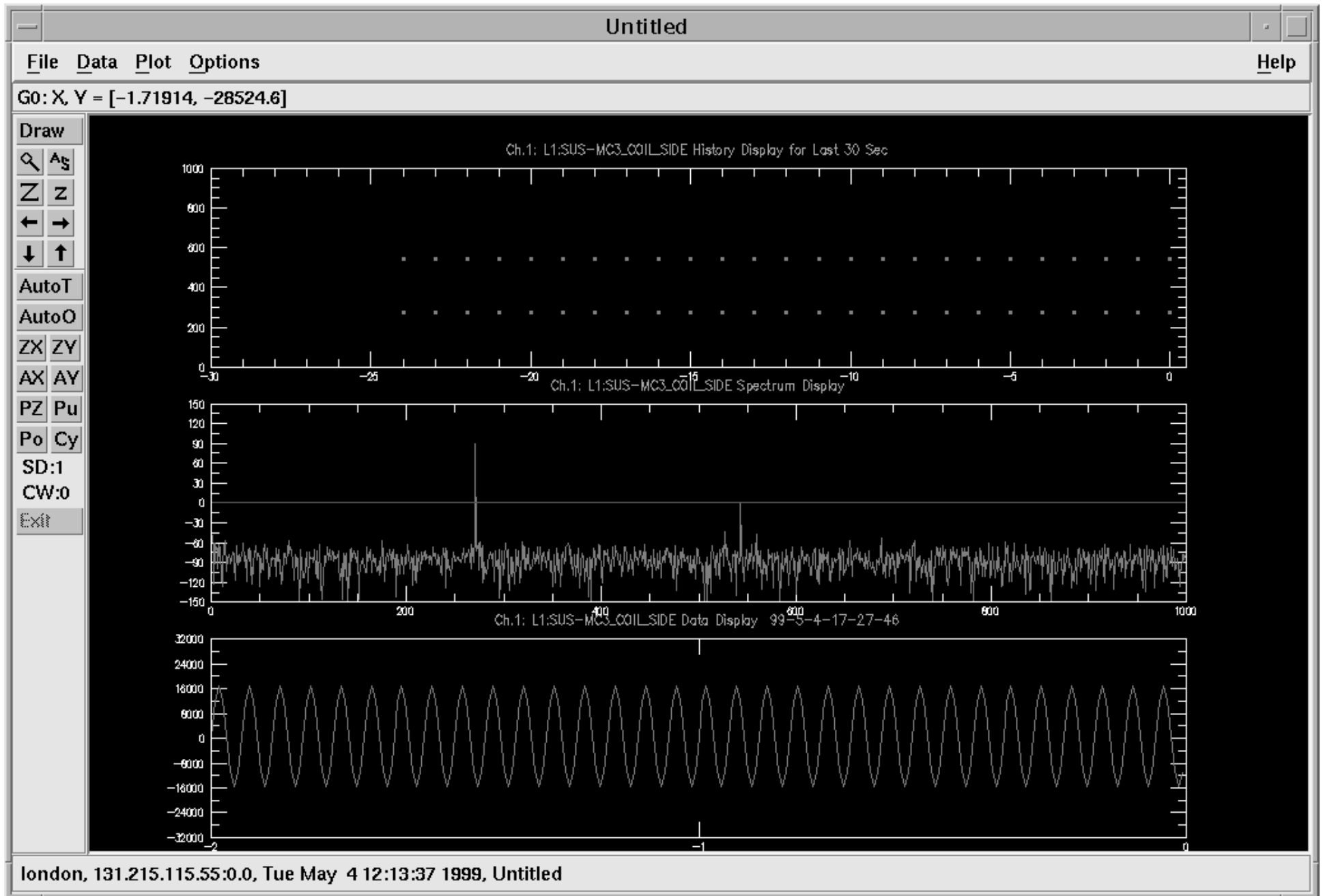
G0: X, Y = [736.82, -186.534]



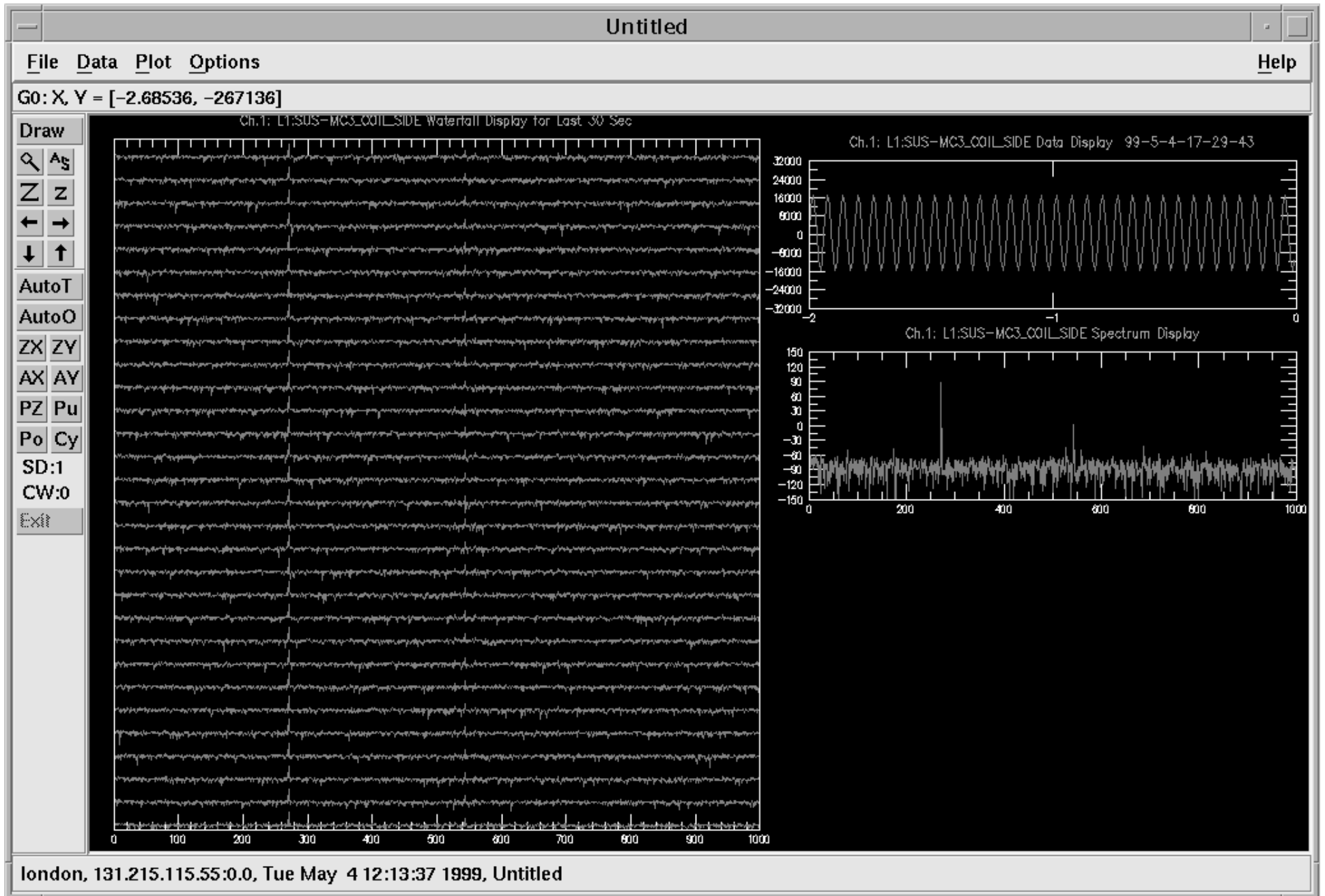
london, 131.215.115.55:0.0, Tue May 4 12:13:37 1999, Untitled



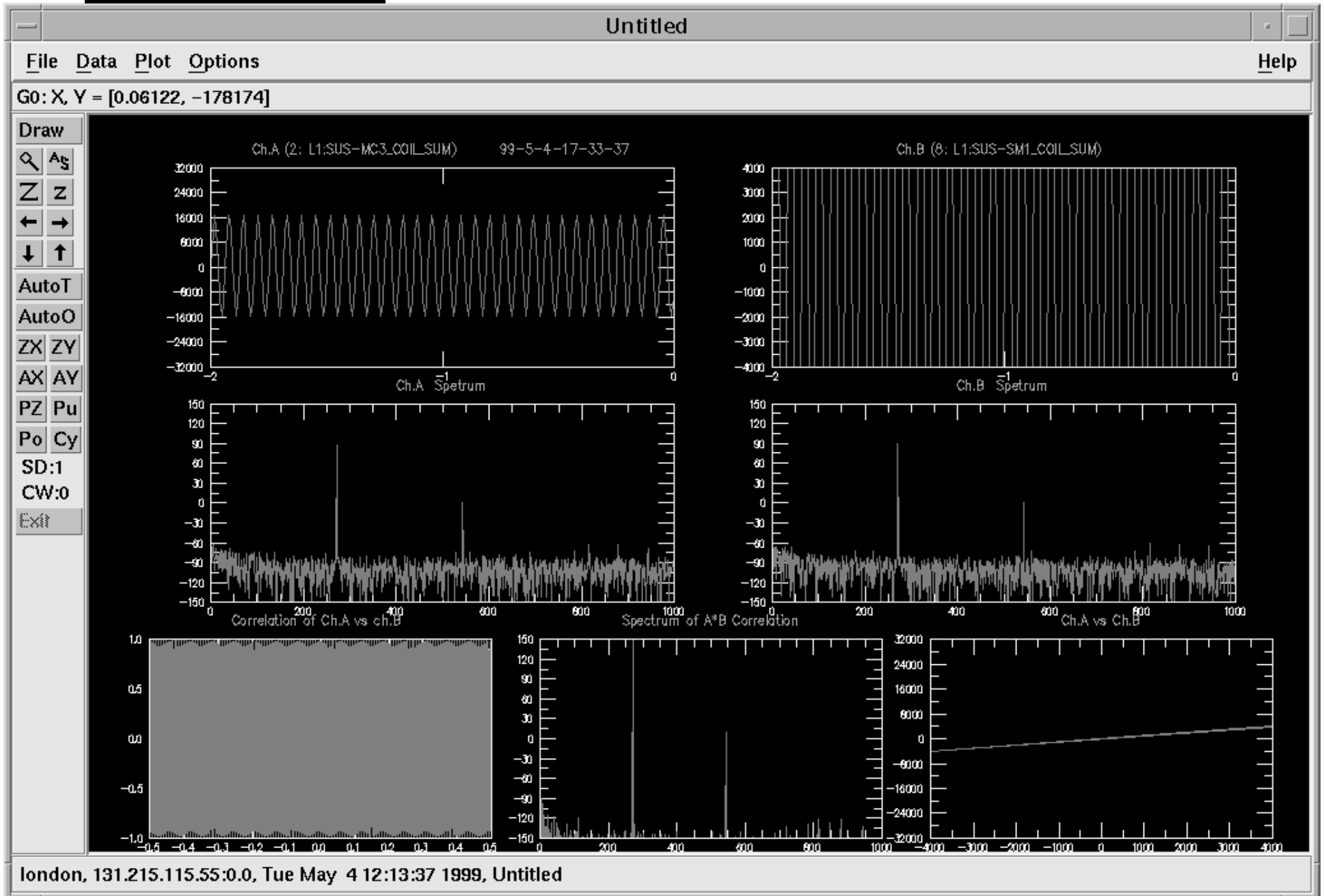
# Frequency vs. Time Display Mode



# Waterfall Display Mode

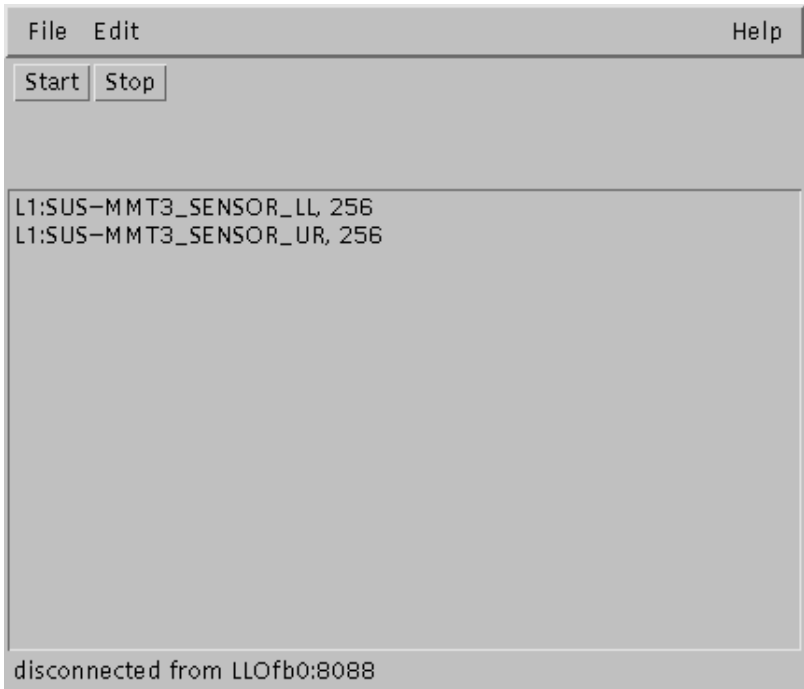
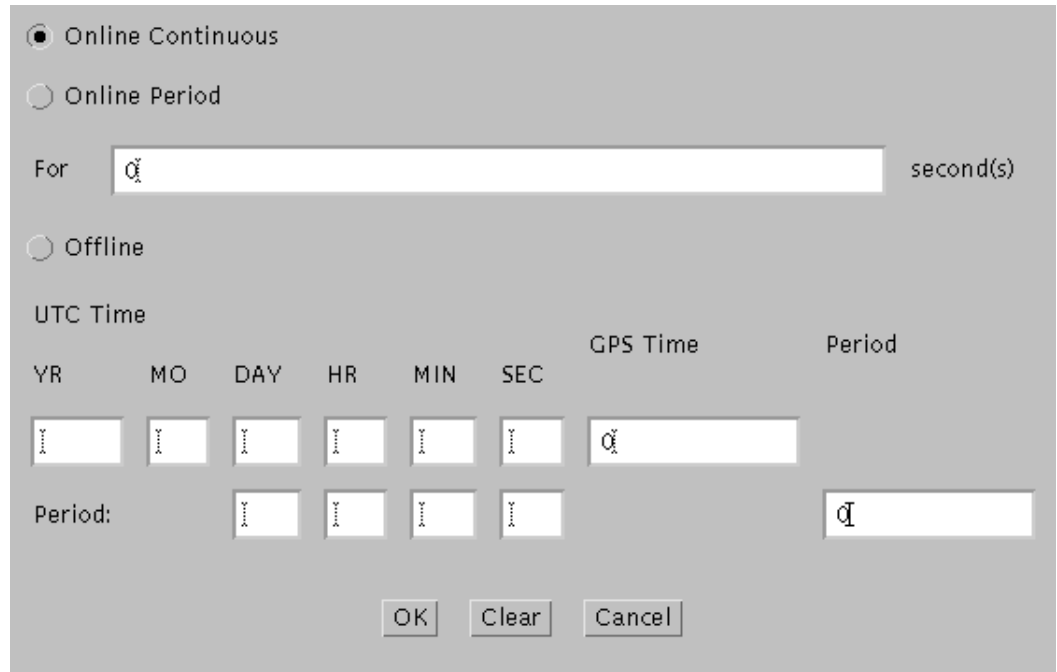
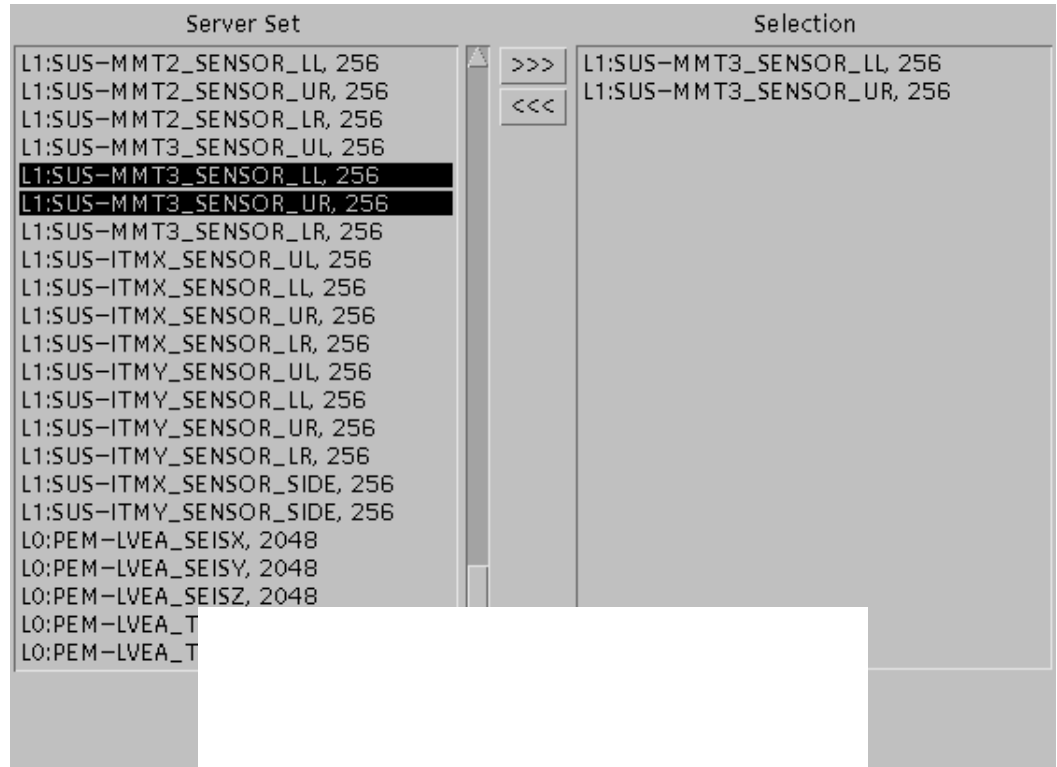


# A vs. B Display Mode

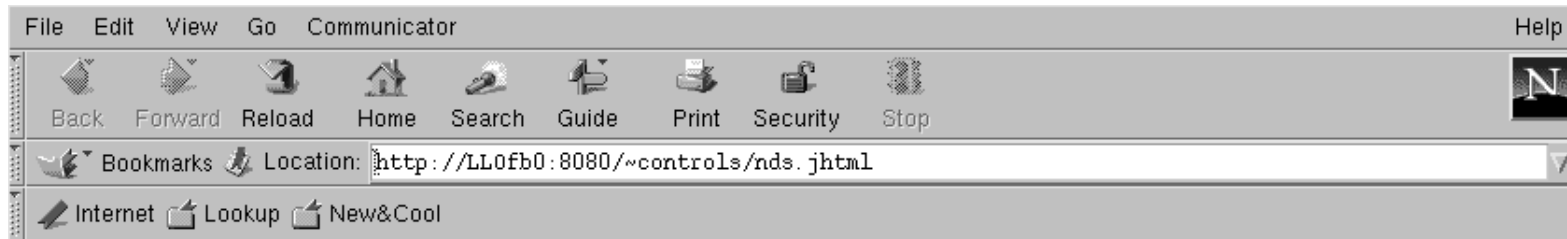


# jdClient Software

- Extracts data via NDS and writes to user specified files.
- May be used for short term storage of signals not archived by the FrameBuilder (ie GDS temporary signals).



# Web jdClient Software



## Network Data Server

Fill out the form below to get the data from the Network Data Server.

When you have completed and submitted the form the data for one or more data channels will be returned to you in the format you requested. You will be able to save the data in a file on the computer your browser is running on.

LVE-LX:X1\_670ATORR  
LVE-LX:X1\_670BTORR  
LVE-LX:X2\_634ATORR  
LVE-LX:X2\_634BTORR  
LVE-LX:X3\_644ATORR  
LVE-LX:X3\_644BTORR  
LVE-LY:Y4\_624ATORR

Data format:

Acquisition rate:

The minimum of the original channel data acquisition rate and this selection applies to all selected channels.

Play Back

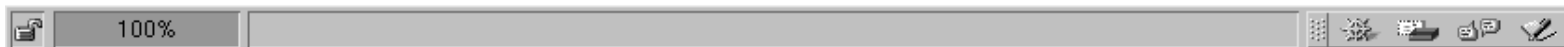
Data is acquired for the past time period determined by the "Period" field in the form below. If "GPS Time" field in that form is empty the data for the last collected "Period" seconds is sent.

Play Forward

"GPS Time" entry field is ignored in this mode. The data is acquired online as it becomes available on the server. This means the saving process will continue for the "Period" number of seconds.

UTC time						GPS Time (seconds)	Period (seconds)
YR	MO	DAY	HR	MIN	SEC		
<input type="text" value="99"/>	<input type="text" value="5"/>	<input type="text" value="4"/>	<input type="text" value="16"/>	<input type="text" value="48"/>	<input type="text" value="25"/>	<input type="text" value="609875318"/>	
Period:			<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="Time Now"/>	<input type="text"/>

Alex Ivanov <aiivanov@ligo.caltech.edu>  
Last modified: Fri Apr 21 14:45AM PDT 1999



# DIAGNOSTICS SYSTEM KEY FEATURES (1)

---

## □ Main design goals

- Distributed system (signals/excitations in all buildings)
- Probes inside of digital servo systems

## □ Excitation system

- Provides excitation signals for all subsystems
- Signal waveforms: sine, square, ramp, triangle, white noise, colored noise and arbitrary
- Sweeps over frequency or amplitude
- Interfaces:
  - Analog: digital-to-analog converter in each building, 8 kHz BW
  - Digital: through reflective memory, 8 kHz BW
  - Stand-alone: remote control of SR DS340, 15 MHz BW
- GPS synchronization



# DIAGNOSTICS SYSTEM KEY FEATURES (2)

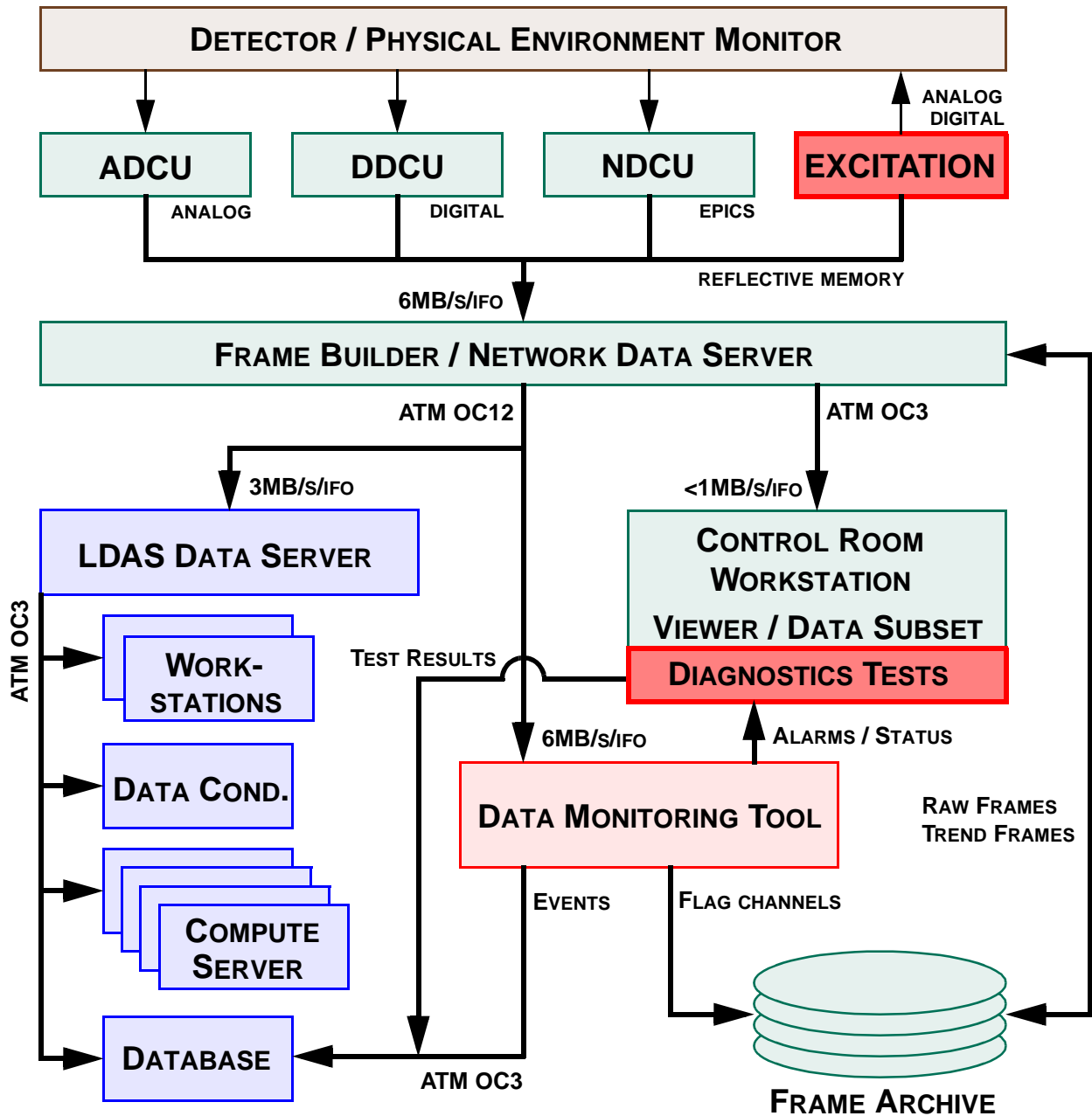
---

## □ Stimulus-response tests

- Sine response tests:
  - Multiple input, multiple excitations
  - Harmonic distortion
  - Two-tone intermodulation
- Swept sine
- Fourier tools:
  - Power spectral density
  - Cross-correlation
  - Pseudo-random stimulus
- Triggered time response
  - Step response
  - Impulse response (decay transients)
  - Ramp scans
- Test iterations
  - Simple repeat
  - Parameter scan
  - Parameter optimization
- Command line interface and GUI
- Data stored in LIGO lightweight data format (XML)



# OVERVIEW





# STATUS

---

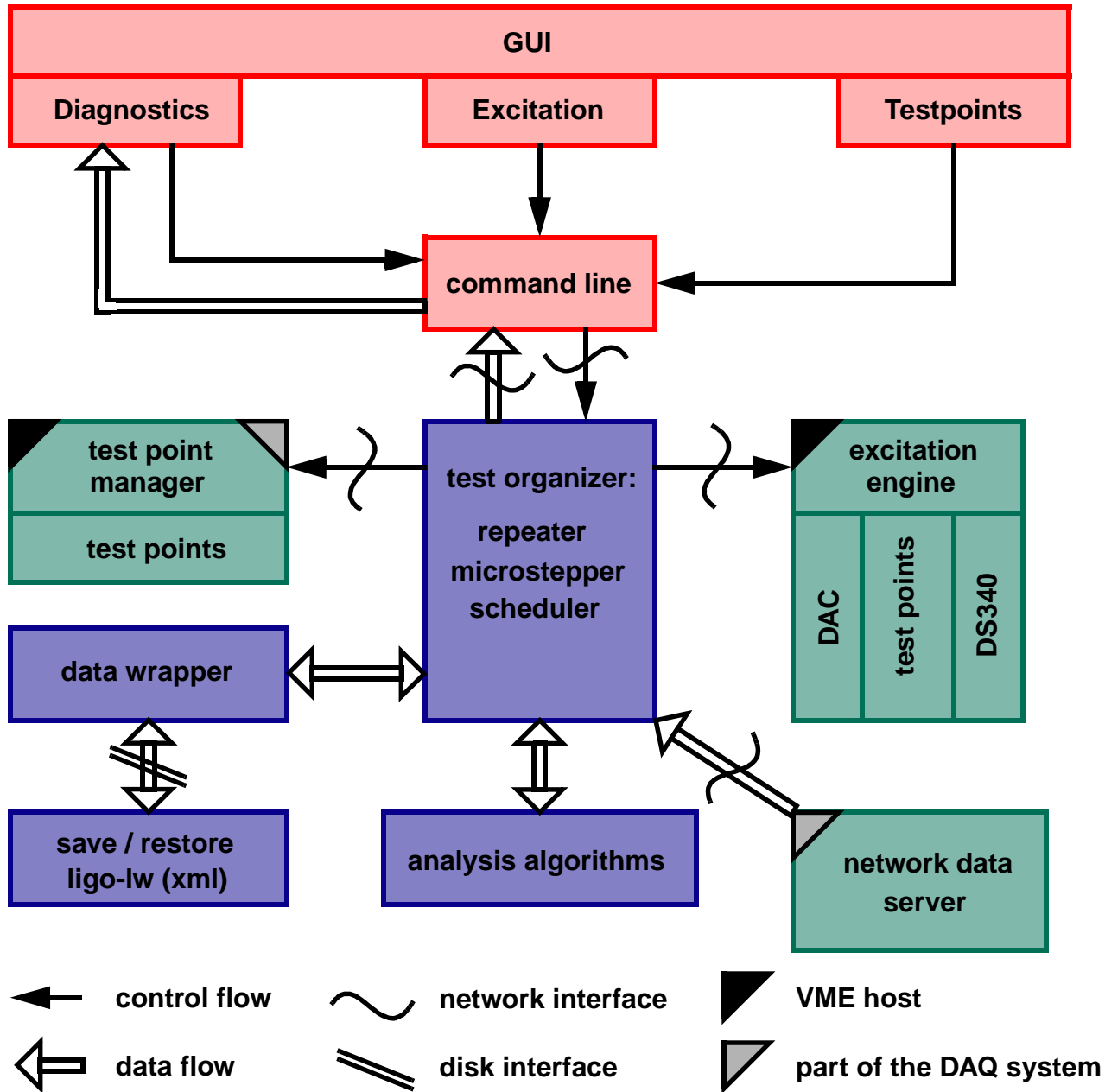
<b>Category</b>	<b>complete</b>
Utilities (timing, rpc, task, error handling)	95%
Algorithms	50%
Arbitrary waveform generator	90%
Command line interface	90%
Graphical user interface	5%
Data acquisition interface (RTDD)	25%
Diagnostics supervisory	50%
Diagnostics test iterators	15%
Diagnostics tests	15%
Hardware drivers (gps, rm, dac, ds340)	80%
Test points	80%
Storage	85%

**currently:**

**~90 software modules, ~50k lines of code and comments**



# KEY COMPONENTS



# USER INTERFACES

---

## ❑ Command line interface

- unix command prompt like
- options for setting parameters, start and stop a test, and save and restore results
- control the excitation engine
- set/release digital test points

## ❑ GUI

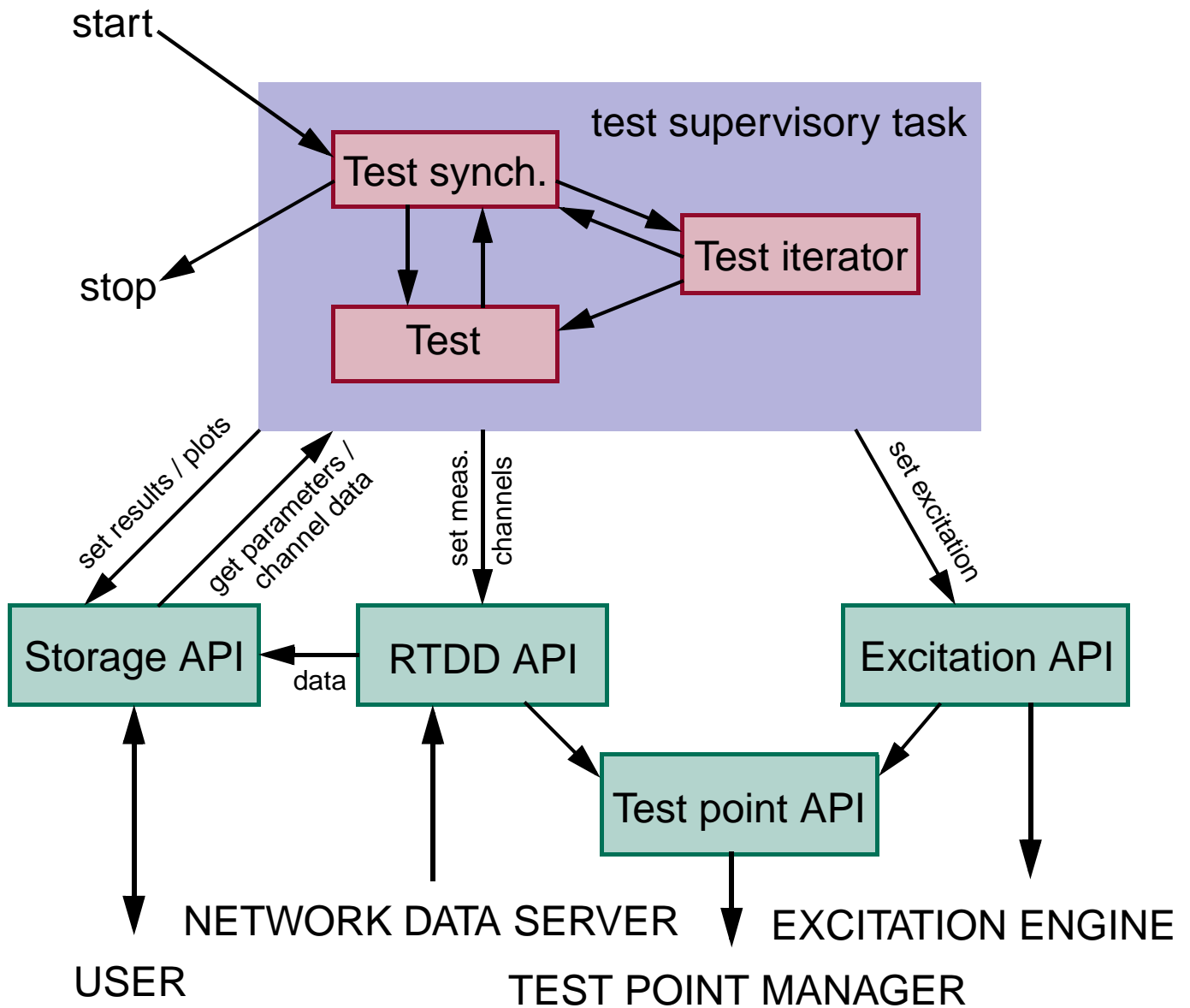
- identical capability
- dialog windows for sine response, swept sine, fourier tools and time series measurements
- control screen for arbitrary waveform generator
- control screen setting test points

## ❑ File format

- LIGO lightweight XML
- commercially available browsers



# TEST ORGANIZATION

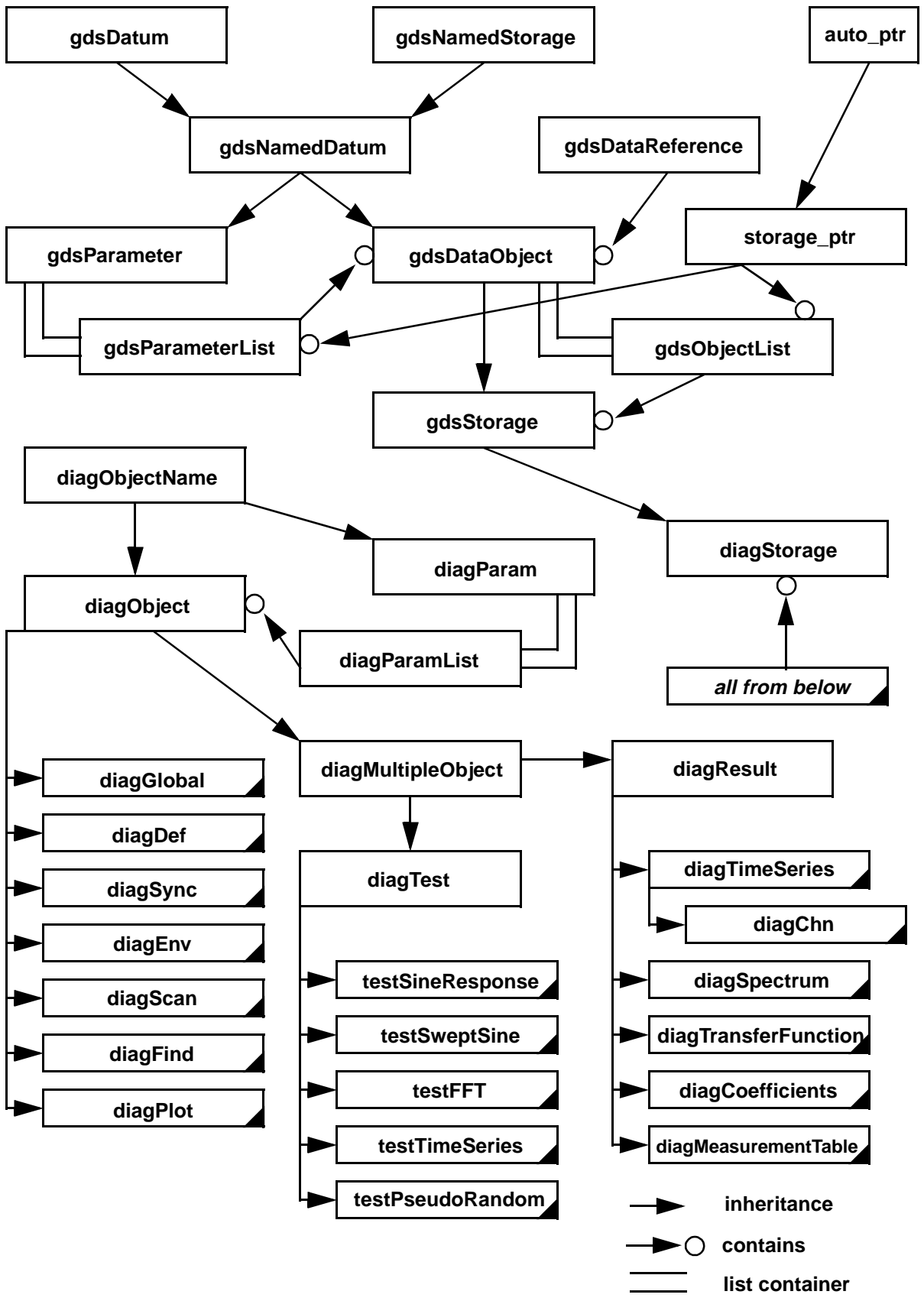


# STORAGE API

---

- ❑ All test parameters, raw data, results, plot settings, etc. are stored in the storage object
- ❑ Organization:
  - Storage object = list of data objects + list of (global) parameter objects
  - Data object = name + data array + list of parameter objects
  - Parameter object = name + value(s)
  - Names can contain array indices
  - Example: Env[1].Active
    - “Env”: data object name
    - “[1]”: index
    - “Active”: parameter name
- ❑ Supported data types:
  - string, channel name
  - integer formats: 8, 16, 32, 64 bit, boolean
  - number formats: single/double precision, complex
- ❑ Includes full type/name checking & save/restore





# REAL-TIME DATA DISTRIBUTION (RTDD) API

---

- ❑ Obtains data from the NDS (network data server)
- ❑ Makes sure test points are selected
- ❑ Optional preprocessing:  
filter/decimate → down-conversion → filter/decimate
- ❑ Data partitioning
- ❑ Stores data in storage object



# EXCITATION API

---

- ❑ All excitation generators have the same interface
- ❑ Controls excitation engine(s)  
automatically selects the output device
- ❑ Handles multiple waveforms and outputs
- ❑ Makes sure test points are selected





# TEST POINT API

---

- ❑ Selects digital test points by communicating with the test point manager (one test point manager / site)
- ❑ Makes sure test points are active on time
- ❑ Keep alive feature



# TEST SUPERVISORY

---

## Separates:

- Tests, e.g., sine response
- Test iterators, e.g., parameter optimization
- Test synchronization, e.g., wait for data
- Tests and test iterator can be combined freely

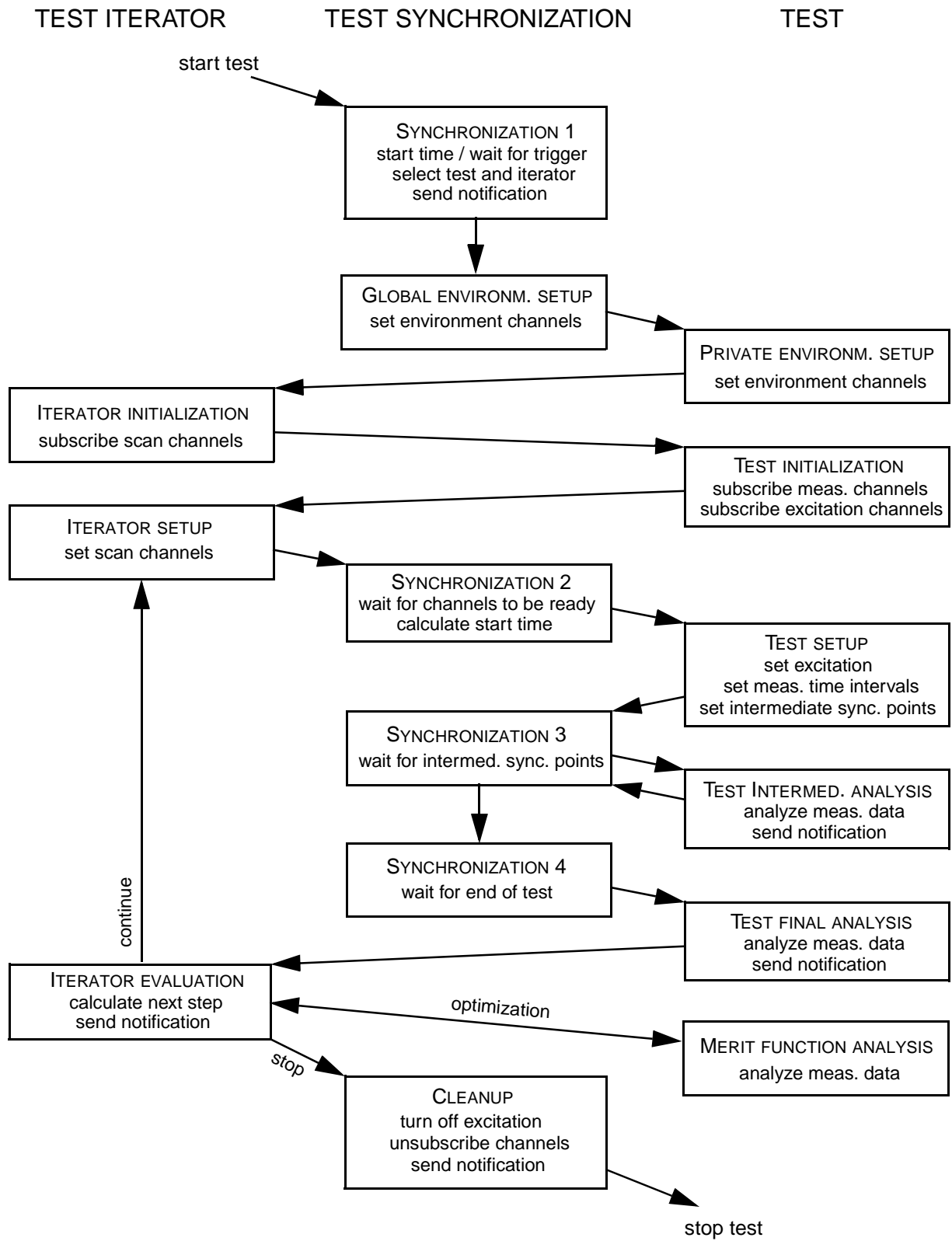
## Default settings

- Allow to abort test?
- Stimulus allowed?
- Do analysis?
- Default/mandatory site identifier for channel names
- Default/mandatory ifo identifier for channel names

## Environment

- Set EPICS channels, offsets and simple signals while test is running





# TEST SYNCHRONIZATION

---

- ❑ Checks start time
- ❑ Makes sure test points, excitations and data channels are turned on on time
- ❑ Waits for (intermediate) results
- ❑ Steps through iterators
- ❑ Sends notifications to the user interface
- ❑ Optional synchronization through EPICS triggers:
  - Wait for start of test/step
  - Signal end of test/step



# TEST ITERATOR

---

## Repeat

- Repeat N times
- wait between tests

## Parameter scan

- Repeat test while changing a parameter (e.g., servo offset)
- Support multi-dimensional scans
- Logarithmic, linear or user defined scans
- Scan frequency, amplitude or offset

## Parameter Optimization

- Extends parameters scan by choosing optimal parameters
- Computes min/max, zero/value crossing of merit function



# TESTS

---

## Supported tests:

- Sine response
- Swept sine
- Fourier tests
- Triggered time response
- Pseudo-random stimulus/cross-spectrum readout

## Performs analysis

- Support analysis at intermediate steps
- Get data from storage object
- Save results to storage object

## Sets up plots

- Stores plot settings to storage object

## Merit functions:

- mean, (band-limited) rms
- amplitude or phase of sine wave
- harmonic distortion, intermodulation product



# SINE RESPONSE

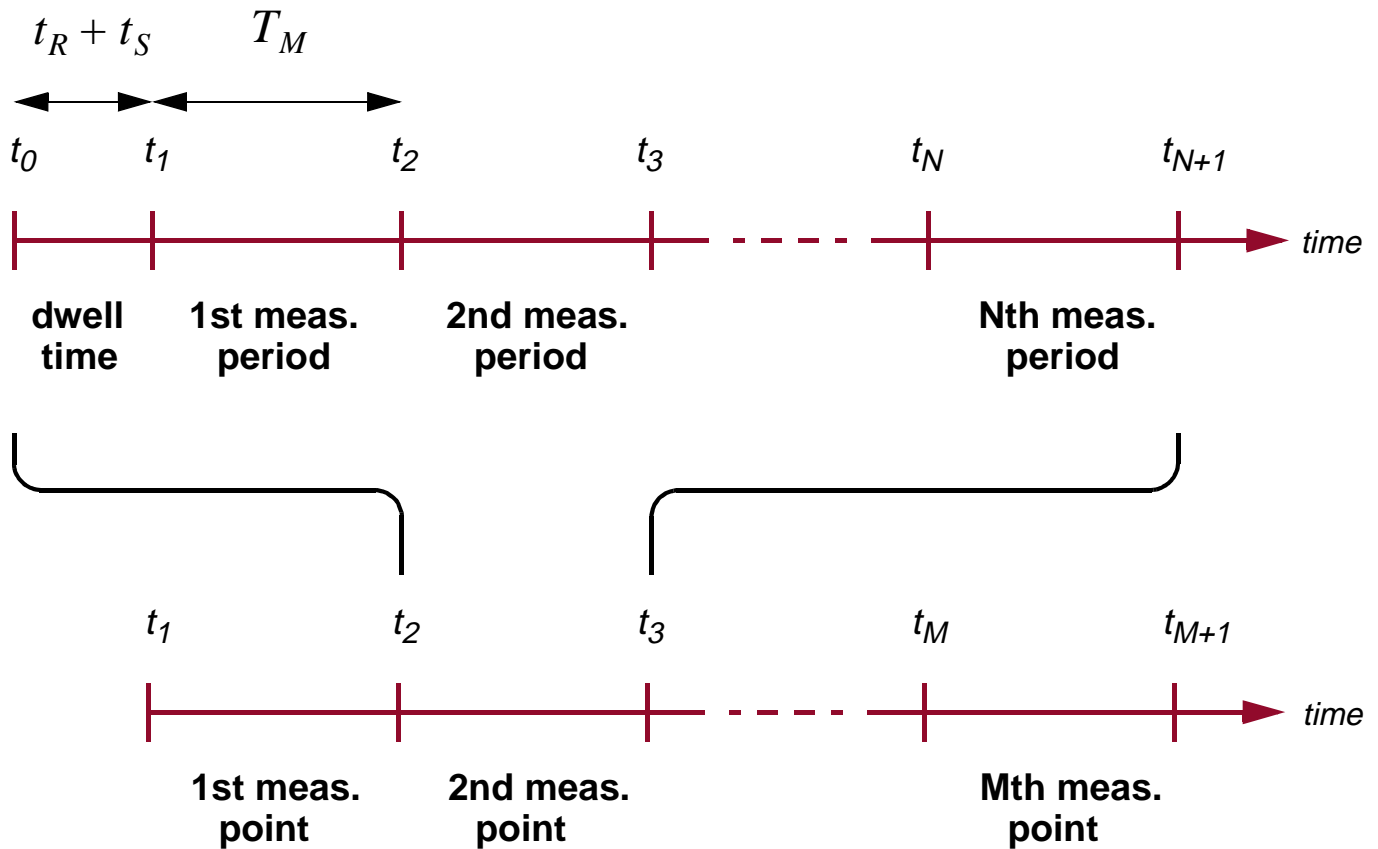
---

Name	Description
MeasurementTime	measurement time at each frequency; first value in sec, second value in cycles. The smaller one is taken and rounded up to the next cycle. Negative values are ignored. default: 100, 10.
Averages	number of averages; default is 1.
SettlingTime	settling time at each frequency step; first value in sec, second value in cycles. The smaller one is taken. Negative values are ignored. default: 10, 3.
StimulusChannel[M]	name of stimulus channel.
StimulusFrequency[M]	frequency of stimulus channel in Hz. default: 100.
StimulusAmplitude[M]	amplitude of stimulus channel. default: 1.
StimulusOffset[M]	offset of stimulus channel. default: 0.
StimulusPhase[M]	phase of stimulus signal in rad; the phase is relative to the last 0:00UTC. default: 0.
StimulusWait[M]	settling time of stimulus in sec. default is 0.25sec.
MeasurementChannel[N]	measurement channel.
FFTResult	if true (default), calculates averaged 1024 point FFTs of each measurement channel as part of the result.



# TIMELINE (1)

## Sine Response



## Swept Sine Response





# SWEPT SINE

Name	Description
SweepType	0 – linear, 1 – log, 2 – user supplied. default: 1.
SweepDirection	0 – upwards, 1 – downwards (default).
StartFrequency	start frequency of swept sine in Hz. default: 1.
StopFrequency	stop frequency of swept sine in Hz. default: 1000.
NumberOfPoints	number of frequency steps; default is 61.
FrequencySteps	user supplied frequency steps.
Averages	number of averages; default is 1.
MeasurementTime	measurement time at each frequency; first value in sec, second value in cycles. The smaller one is taken and rounded up to the next cycle. Negative values are ignored. default: 100, 10.
SettlingTime	settling time at each frequency step; first value in sec, second value in cycles. The smaller one is taken. Negative values are ignored. default: 10, 3.
StimulusChannel	name of stimulus channel.
StimulusAmplitude	amplitude of stimulus channel. default: 0.
MeasurementChannel[N]	measurement channels; must be at least two. By default the first channel is the A channel.
FFTResult	if true, calculates averaged 1024 point FFTs of each measurement channel for each frequency step as part of the result; default is false.



# FOURIER TESTS

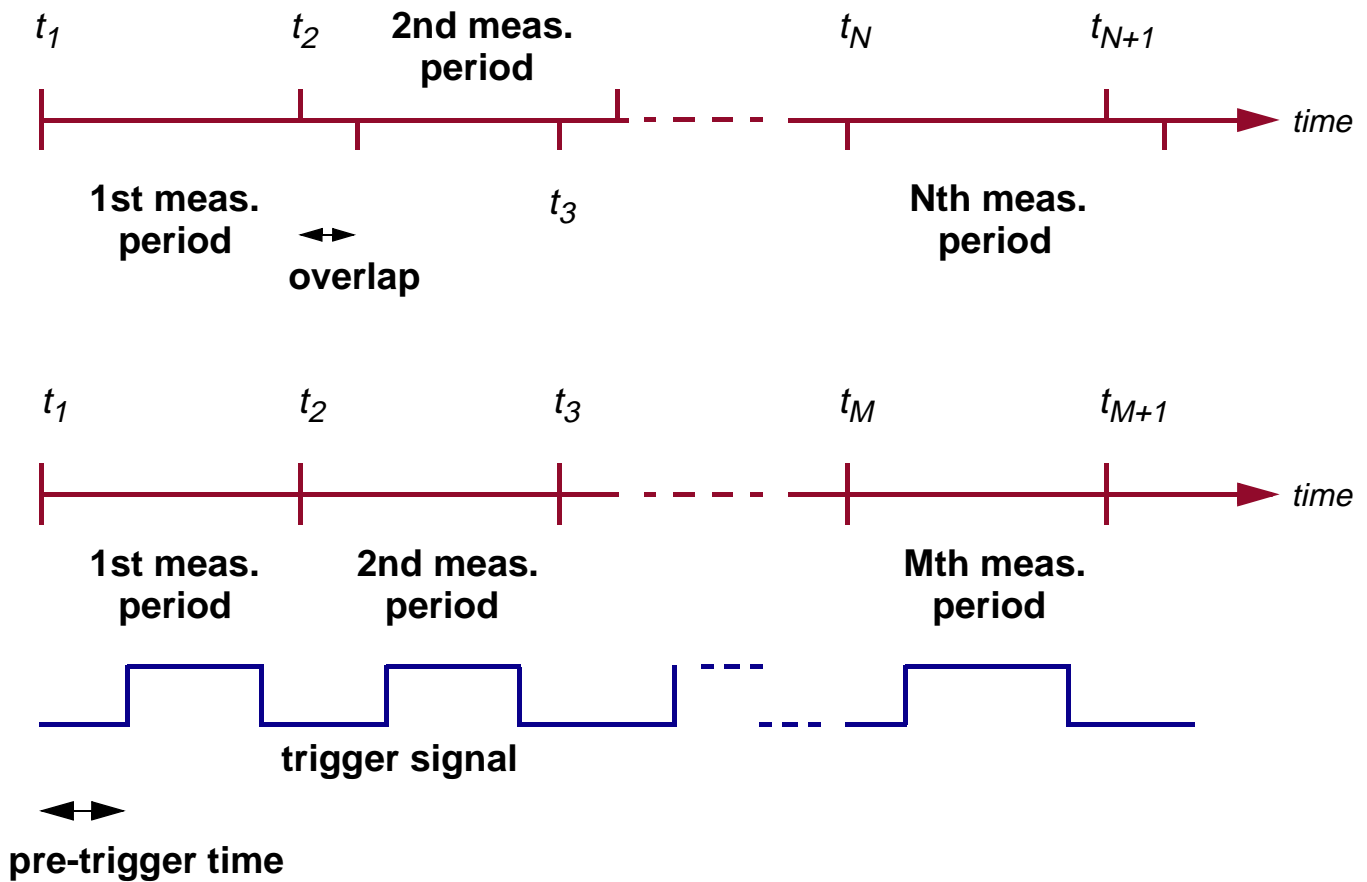
---

Name	Description
StartFrequency	start frequency of FFT in Hz. default: 0.
StopFrequency	stop frequency of FFT in Hz; internally always rounded up so that the frequency span is a power of 2. default: 1000.
BW	bandwidth in Hz; always rounded to the closest power of 2. default: 1.
Overlap	overlap of FFT windows (only useful when averaging); $t(i) = t_0 + i * (1 - \text{Overlap}) * dt$ . default: 0.5.
Window	0 – uniform (no window), 1 – Hanning window, 2 – Flat-top.
AverageType	0 – fixed number, 1 – running (exponential weight). default: 0.
Averages	number of averages; default is 10.
MeasurementChannel[N]	measurement channel.



# TIMELINE (2)

## Fourier test



## Triggered time series measurement



# TRIGGERED TIME SERIES MEASUREMENT

---

Name	Description
TriggerRate	trigger rate in sec; if no trigger is used, this is the measurement duration.
PreTriggerTime	measurement time before trigger is applied; default is 20% of the total measurement time.
TriggerType	0 – no trigger signal, 1 – square wave, 2 – impulse, 3 – ramp, 4 – triangle. default: 0.
TriggerNum	number of triggers/averages. 1 – single trigger response (no average), >1 – periodic trigger response (with average), default: 10.
AverageType	0 – fixed number, 1 – running (exponential weight).
TriggerChannel	name of trigger channel
TriggerAmplitude	amplitude of trigger signal
MeasurementChannel[N]	measurement channel.



# EXCITATION ENGINE

---

## □ Periodic waveforms

$$o(t) = s(2\pi f(t - t_0) - \phi_0 + \varphi(t - t_0 - t_2, t_{PO})) \times b(t - t_0, \Delta t, t_{PI}, t_{PO})$$

with  $\phi_0$  phase shift,  $\varphi$  frequency sweep and  $b$  amplitude phase-in/out.

sine wave

$$s(t) = A \sin(2\pi ft)$$

square wave

$$s(t) = \begin{cases} +A & 0 \leq 2\pi ft \bmod 2\pi < \pi \\ -A & \pi \leq 2\pi ft \bmod 2\pi < 2\pi \end{cases}$$

ramp

$$s(t) = A \frac{\phi}{2\pi} \text{ with } \phi = 2\pi ft \bmod 2\pi$$

triangle

$$s(t) = \begin{cases} A \left( \frac{2\phi}{\pi} - 1 \right) & 0 \leq 2\pi ft \bmod 2\pi < \pi \\ A \left( 3 - \frac{2\phi}{\pi} \right) & \pi \leq 2\pi ft \bmod 2\pi < 2\pi \end{cases}$$



# EXCITATION ENGINE (2)

---

## □ Phase-in/out:

$$b(t - t_0, \Delta t, t_{PI}, t_{PO}) = \begin{cases} 0 & t < 0 \text{ or } t > \Delta t \\ 1 & t_{PI} \leq t < \Delta t - t_{PI} \\ g_1(t, t_{PI}) & 0 \leq t < t_{PI} \\ g_2(t - t_2, t_{PO}) & \Delta t - t_{PO} \leq t \leq \Delta t \end{cases}$$

step

$$\begin{aligned} g_1(t, t_{PI}) &= 0 \\ g_2(t, t_{PO}) &= 1 \end{aligned}$$

linear

$$g_1(t, t_{PI}) = \frac{t}{t_{PI}}$$

$$g_2(t, t_{PO}) = 1 - (1 - c)g_1(t, t_{PO})$$

quadratic

$$g_1(t, t_{PI}) = -\left(\frac{t}{t_{PI}}\right)^4 + 2\left(\frac{t}{t_{PI}}\right)^2$$

$$g_2(t, t_{PO}) = 1 - (1 - c)g_1(t, t_{PO})$$



# EXCITATION ENGINE (3)

---

## □ Frequency sweeps

linear

$$\varphi(t, t_{PO}) = g_1(t, t_{PO})(2\pi\Delta f t - \overline{\Delta\phi})$$

$$o(t)|_{t_2 \leq t - t_0 \leq \Delta t} = s(2\pi f t_2 - \phi_0 + \varphi_{\log}(t - t_0 - t_2, t_{PO}))$$

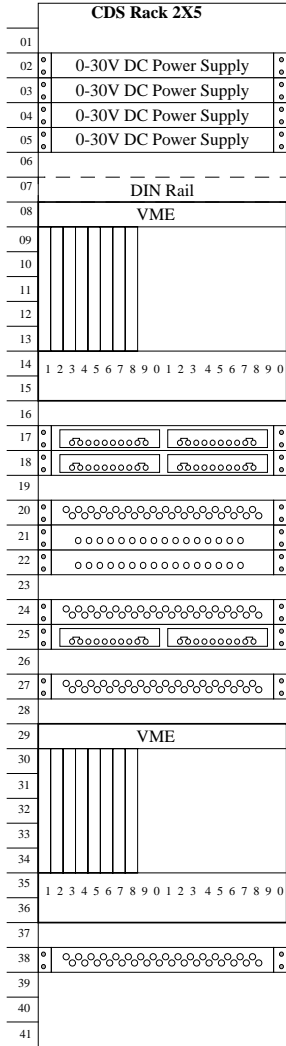
logarithmic

$$\varphi_{\log}(t, t_{PO}) = \left[ 2\pi f t_{PO} \left( \frac{f + \Delta f}{f} \right)^{\frac{t}{t_{PO}}} - \overline{\Delta\phi} \right] \frac{t}{t_{PO}}$$

## □ Outputs

- Reflective memory
  - single precision floats
  - 16K and 2K sampling rate
- Digital-to-analog converter
  - 16 bit
  - 16K sampling rate
- DS340
  - 15 MHz bandwidth
  - ethernet to RS232 converters





**Rack Mount Components w/Locations**

Loc	Description	Vendor	Model	Designator
02	+/- 0-30VDC Power Supply	Sorenson		PS-1
03	+/- 0-30VDC Power Supply	Sorenson		PS-2
04	+/- 0-30VDC Power Supply	Sorenson		PS-3
05	+/- 0-30VDC Power Supply	Sorenson		PS-4
07	DIN Rail w/terminal blocks (rear mount)			DIN-1
08	VME Crate (Data Acquisition)			VME-1
17	DAQS Interface Chassis (LEMO)	LIGO		DAQIC-1
18	DAQS Interface Chassis (LEMO)	LIGO		DAQIC-2
20	DAQS Interface Chassis (BNC/1KFilter)	LIGO		DAQIC-3
21	Accelerometer Signal Conditioner	Endevco		ACCSC-1
22	Accelerometer Signal Conditioner	Endevco		ACCSC-2
24	DAQS Interface Chassis (BNC/1KFilter)	LIGO		DAQIC-4
25	DAQS Interface Chassis (LEMO)	LIGO		DAQIC-5
27	DAQS Interface Chassis (BNC/1KFilter)	LIGO		DAQIC-6
29	VME Crate (Global Diagnostics)			VME-2
38	GDS BNC Patch Panel	LIGO		GDSIC-1

**NOTES:**

1) This drawing is PRELIMINARY and is only to be used for initial installation of rack mount components and connection of PEM signals. Cabling to other Detector components and GDS are pending final equipment designs for those subsystems.

**VME1 Modules / Slot Assignments**

Slot	Description	Vendor	Model	Designator
1	MIPS Processor	Heurikon	4700	CPU-1
2	MIPS Processor	Heurikon	4700	CPU-2
3	Reflected Memory	VMIC	5588DMA	RFM-1
4	GPS Slave	Brandywine		GPS-1
5	Timing Slave	LIGO		TS-1
6	32 channel ADC	ICS	110B1	ADC-1
7	32 channel ADC	ICS	110B1	ADC-2
8	32 channel ADC	ICS	110B1	ADC-3
9	32 channel ADC	ICS	110B1	ADC-4
10	32 channel ADC	ICS	110B1	ADC-4
11	Backplane Split -----	-----	-----	-----
12	68040 Processor	Motorola	162-333	CPU-3
13-21	Empty	-----	-----	-----

**VME2 Modules / Slot Assignments**

Slot	Description	Vendor	Model	Designator
1	MIPS Processor	Heurikon	4700	CPU-1
2	MIPS Processor	Heurikon	4700	CPU-2
3	Reflected Memory	VMIC	5588DMA	RFM-1
4	GPS Slave	Brandywine		GPS-1
5	Timing Slave	LIGO		TS-1
6	32 channel DAC	ICS	115	DAC-1

UNLESS OTHERWISE SPECIFIED		CURRENT REVISION APPROVAL	
DIMENSIONS ARE IN INCHES		DRAWN	GROUP
TOLERANCES:		CHECKED	SIGNATURE
FRACTIONAL ±			DATE 2/24/99
ANGULAR ± BEND ±			
TWO PLACE DECIMAL ±			
THREE PLACE DECIMAL ±			
FINISHED SURFACE THIS SIDE			
REMOVE ALL BURRS			
DO NOT SCALE THIS DRAWING		REV	DESCRIPTION
USED IN:		SHEETS	DATE
NEXT ASS'Y:		EFFECTED	
REFERENCE DRAWINGS		ISSUE DESCRIPTION	
DWG NO	DESCRIPTION	DWG NO	DESCRIPTION
6		5	
SCALE		SHEET	REV
D990016-00-C		1 of 10	01



# ANALYSIS ALGORITHMS

---

## □ Provide the functions typically found in a dynamic signal analyzer

### ›› FFT measurements (~50% done)

- power spectra
- frequency response
- coherence

### ›› Swept sine measurements (complete)

- swept sine frequency response – amplitude & phase, coherence
- (multiple) sine response

### ›› Pole-Zero curve fitting

- for closed loop gain measurements
- implementation TBD

### ›› Time measurements

- step/impulse response – e.g., diagnose loop bandwidth, phase margin
- implementation TBD

### ›› Correlation measurements

- auto- and cross-correlation
- implementation TBD

### ›› Octave band analysis

- measures power within logarithmic frequency bands
- digital filter bank followed by rms averaging
- implementation TBD

# FFT MEASUREMENTS

---

## ❑ Power spectrum estimation – Welch's method

›› Data is broken up into equal-length segments, possibly overlapping

›› Each segment is windowed:

- uniform (no window)
- Hanning (gen'l purpose)
- Flat-top (accurate amplitude measurement)
- BMH (large dynamic range for separating two close peaks)

›› An FFT is performed on each segment (using FFTW)

›› Periodogram of each segment is formed by proper normalization of the magnitude squared of each FFT bin

›› Periodograms are averaged to give the power spectrum estimate

## ❑ Cross-spectral density

›› similar to above, except instead of mag squared of a single FFT, use FFT coefficients from two channels:

$$P_{xy}(k) = \frac{1}{W} X(k) \cdot Y(k)^* \quad , \quad k = 0, 1, \dots, N-1$$

## ❑ Transfer function & coherence

›› formed from cross-spectral & power spectrum estimates

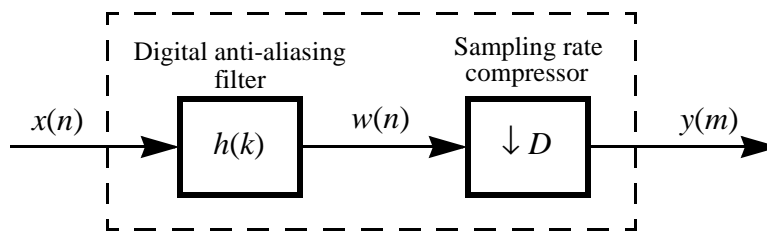
# DATA PRE-PROCESSING FOR FFT MEASUREMENTS

---

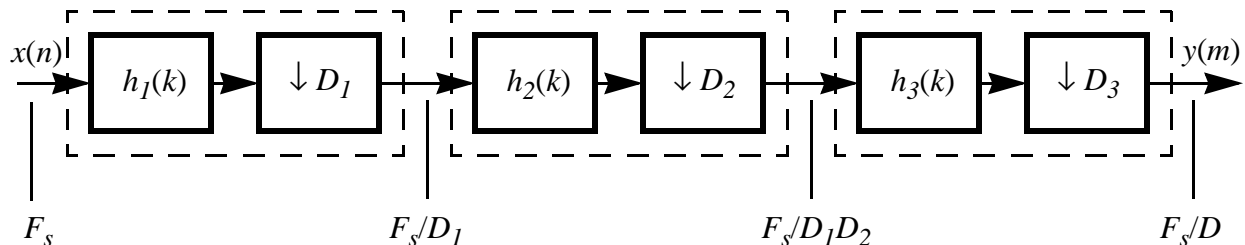
## □ Data decimation

- ›› For a FFT measurement over a bandwidth  $\Delta f$ , reduce the sampling rate to  $2/\Delta f$
- ›› Decimation is a process of low-pass filtering and down-sampling, to avoid aliasing

single stage:



multi-stage decimation:



# DECIMATION

## Implementation

›› Decimate-by-2 function that can be cascaded any number of times to reduce the sampling rate from  $f_s$  to  $f_s/2^N$

›› Filter is a linear-phase FIR low-pass filter

○ can be chosen from a predefined, but expandable set of filters

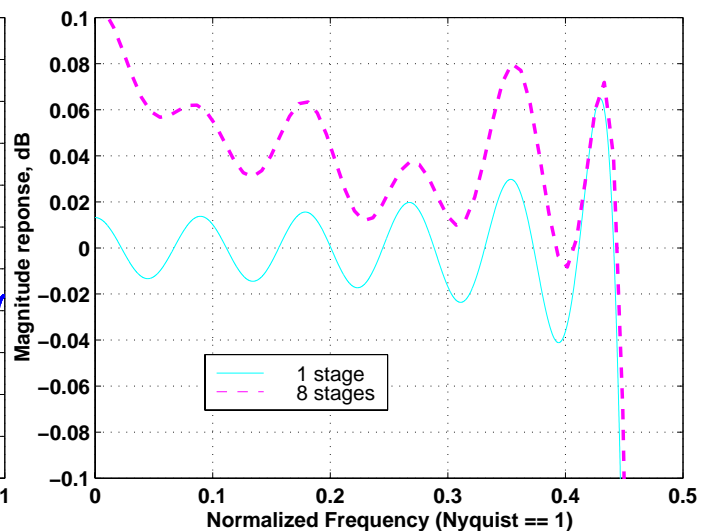
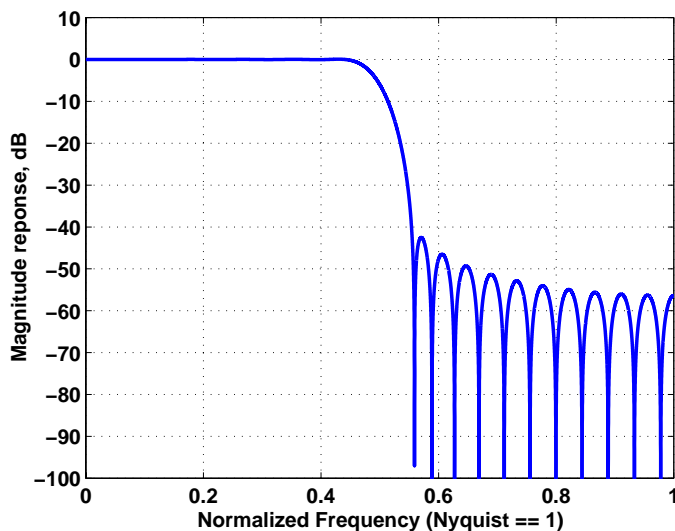
○ trade-off between passband/stopband error & filter length; width of transition zone & filter length; etc

›› Standard calculation techniques for efficient computation

○ instead of low-pass filtering all the data and then down-sampling, the down-sampling is embedded in the filter (filter is advanced by 2 points for each output point), reducing the number of multiplications by factor of 2

○ linear phase FIR filter is symmetric –  $h(n) = h(N - 1 - n)$ ; reduces number of multiplications by factor of 2

○ use a half-band filter: if passband ripple = stopband ripple &  $f_{pb} + f_{sb} = f_N$  then about 1/2 the filter coefficients are zero (not computed)

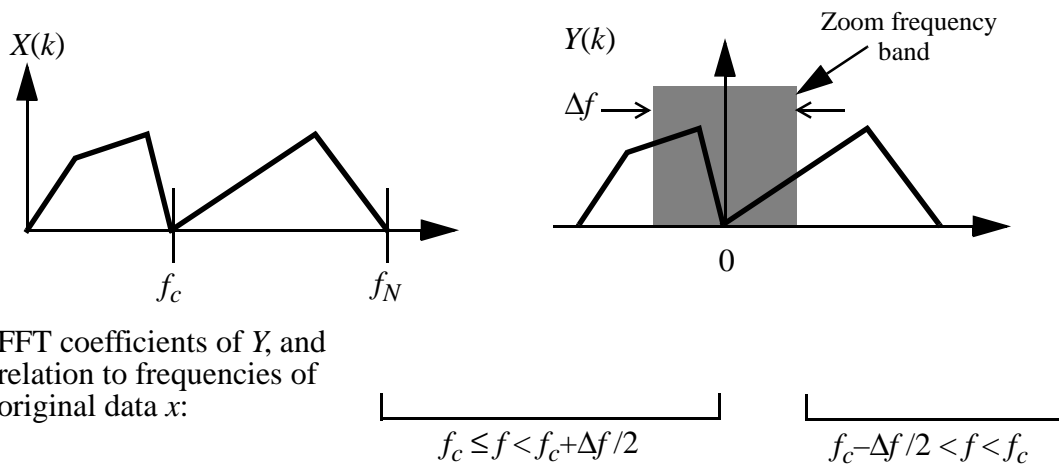
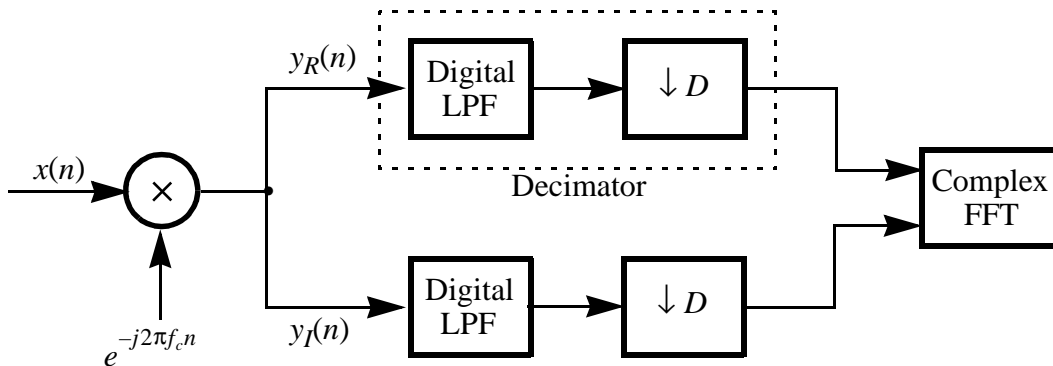


# ZOOM ANALYSIS

- For FFT spans starting above DC, first complex heterodyne the data:

$$y(n) = e^{-j2\pi f_c n} \cdot x(n)$$

then decimate to desired bandwidth, & perform a complex FFT on  $y(n)$



# SWEPT SINE ANALYSIS

---

- To measure the amplitude of a sine wave of frequency  $f_0$  in a signal  $s(t)$ , compute:

$$c_1 = \frac{1}{T} \int_0^T s(t) e^{-i\omega_0 t} dt$$

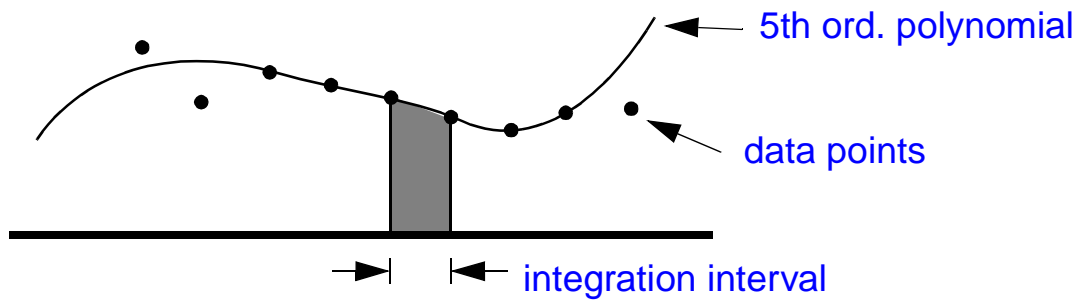
where  $T$  is an exact multiple of periods of  $f_0$

- Steps:

- ›› obtain the number of data points needed by the integration algorithm
- ›› optionally remove a DC offset from the data
- ›› multiply the data set by  $e^{i\omega_0 t}$
- ›› integrate the real and imaginary parts of the above over an integral number of cycles
- ›› repeat the above steps for the desired number of averages
- ›› form the average  $\langle c_1 \rangle$ , and use to compute the frequency response with other channels; the individual  $c_1$  are also kept for computing the coherence with other channels

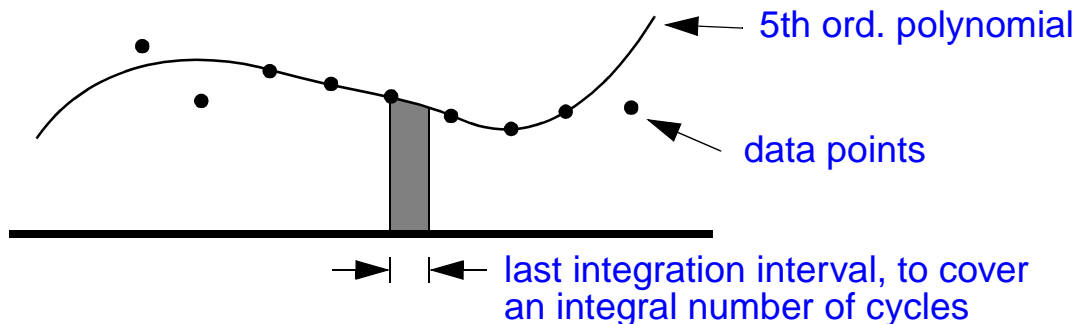
# NUMERICAL INTEGRATION (1)

- Modified Newton-Cotes method: find the 5th ord. poly. passing



through 6 adjacent pts; integrate over the middle two points

- Move forward one point and repeat until end of data set:



- Computationally simple over the whole data set:

- a single integration step can be reduced to a constant set of six coefficients for the six data points; the sum of which is 1
- almost all data points are multiplied once by each coefficient, thus most data points are simply added together
- the coefficients for the last, fractional integration step must be computed for each new frequency (for a given sampling rate)

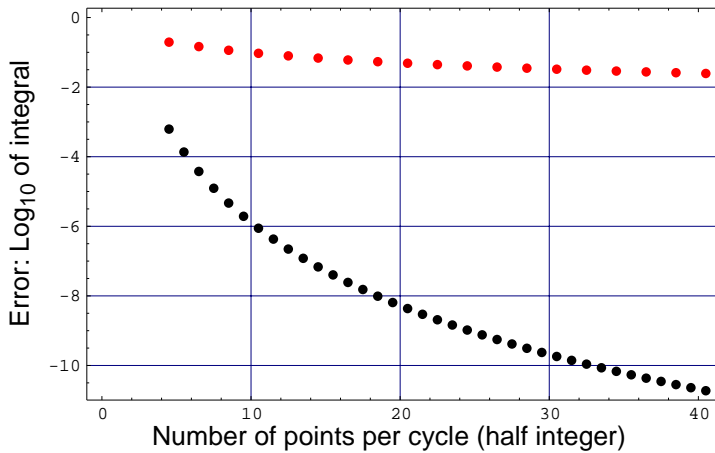
# NUMERICAL INTEGRATION (2)

## □ Add FIR filter

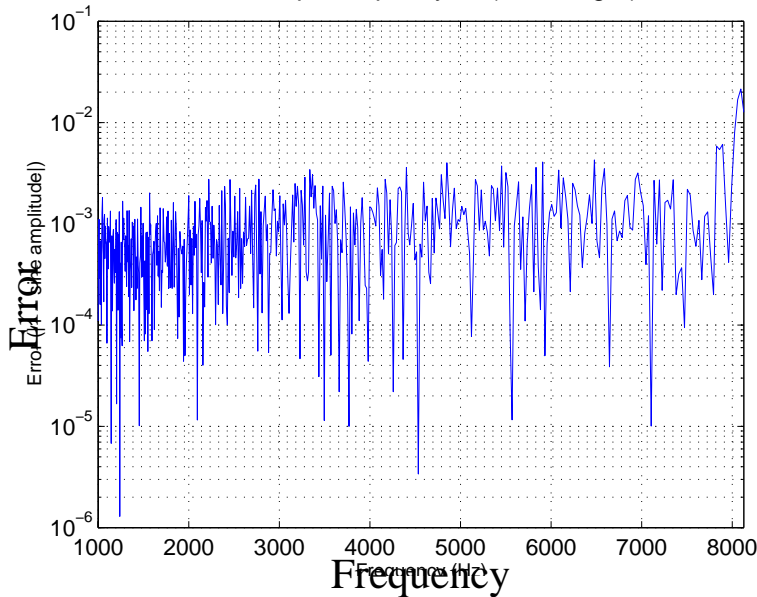
>> to remove power far from the sine detection frequency, the demodulated data is also low-pass filtered at  $0.1 \times f_{\text{Nyquist}}$  with a 20-tap FIR filter

>> FIR filter convolution combined with the integration to produce a new set of coefficients, most of which are unity

○ increases number of leading coefficients to 24, trailing coefficients to 25



computation of  $\frac{1}{T} \int_0^T e^{-in\omega t} dt$



Detection of

$$s(t) = 100 + \sin(2\pi f t_s) + 0.1 \text{ noise}$$

$$t_s = 1/16384$$

100 cycles per freq., 2 avgs.



# The LIGO Data Monitoring Tool

John G. Zweizig  
*LIGO, Caltech*

GDS Final Design Review  
Pasadena, May 7, 1999

---

# Contents

## A) Introduction

1. Purpose
2. S/W Requirements
3. H/W Requirements
4. Architecture

## B) S/W Components

1. Data Distribution
2. Trigger Management
3. Process Environment and Control

## C) Background Applications

1. /cpp/ Environment
2. C Environment
3. Example

## D) Foreground Applications

1. Foreground Overview.
2. ROOT Classes
3. ROOT Macros
4. ROOT session example

---

## Introduction: General

### Features:

- Access to all current data.
- Sufficient computational power for complex calculations.
- Interactive data analysis and presentation.
- Limited output (triggers, status displays, data summaries).
- On-line development and computation environment.
- Geared to provide immediate feedback.

### Uses:

- Detect and tag known disturbances or signals.
- Search for pathological conditions.
- Gauge interferometer state and performance.
- Check Data integrity.
- Interactive testing and diagnosis.

---

## Introduction: Usage Model

**Innovation:** Identify faults, noise sources.

- Immediate access to all current data.
- Analysis with a high-level data manipulation language.
- Graphical data presentation.
- Rapid development cycle (interpretive scripts).

**Display Manager:** Display current running state.

- Graphical user interface.
- Complex calculations on current data.
- Some process management (launch processes)

**Development:** Test and debug monitor algorithms.

- Interpretive environment.
- Extensive scientific function library.

**Production Monitoring:** Continuous data scanning.

- Process supervision.
- Run-time function library.
- Online data access.
- Result presentation/recording.

---

## Introduction: Sample Monitors

### Stationary Behavior of IFO Channels

- 60 Hz (& harmonics) contamination
- Violin mode amplitudes
- Stack vibration level
- Internal mirror resonance amplitudes
- Non-Gaussian noise level (outlier frequency, contribution to RMS)
- Distortion from *normal* ambient power spectrum
- Correlation among longitudinal & orientational DOF's.
- General operational state (*e.g.* good lock, marginal lock, unlocked)
- Strain sensitivity at various frequencies.
- Maximum viewing distance for in-spiral standard candle(s)

### Stationary Behavior of Environmental Channel

- Distortion from *normal* ambient power spectrum
- Correlation with key IFO channels.

---

## Introduction: Sample Monitors

### Transient Behavior of IFO Channels

- Known shape impulses *e.g.* wire relaxations, dropped objects, railed actuators, etc.
- Flickering optical modes
- Violin mode ring-up, ring-down.
- Servo instability
- Generic band-limited RMS growth
- Out-of-band resonance excitation
- Large-amplitude excursion
- Onset of electronic analog or digital saturation
- Readout malfunction (lost/duplicated data, sticky bits)
- Dust particle falling through beam (tough!)
- Non-stationary time-frequency behavior (waterfall/carpet plots, wavelet analysis)

### Transient Behavior of Environmental Channels

- Seismic tremors, gunshots, trucks, lightening, wind.

---

## Introduction: Requirements

### The DMT must provide

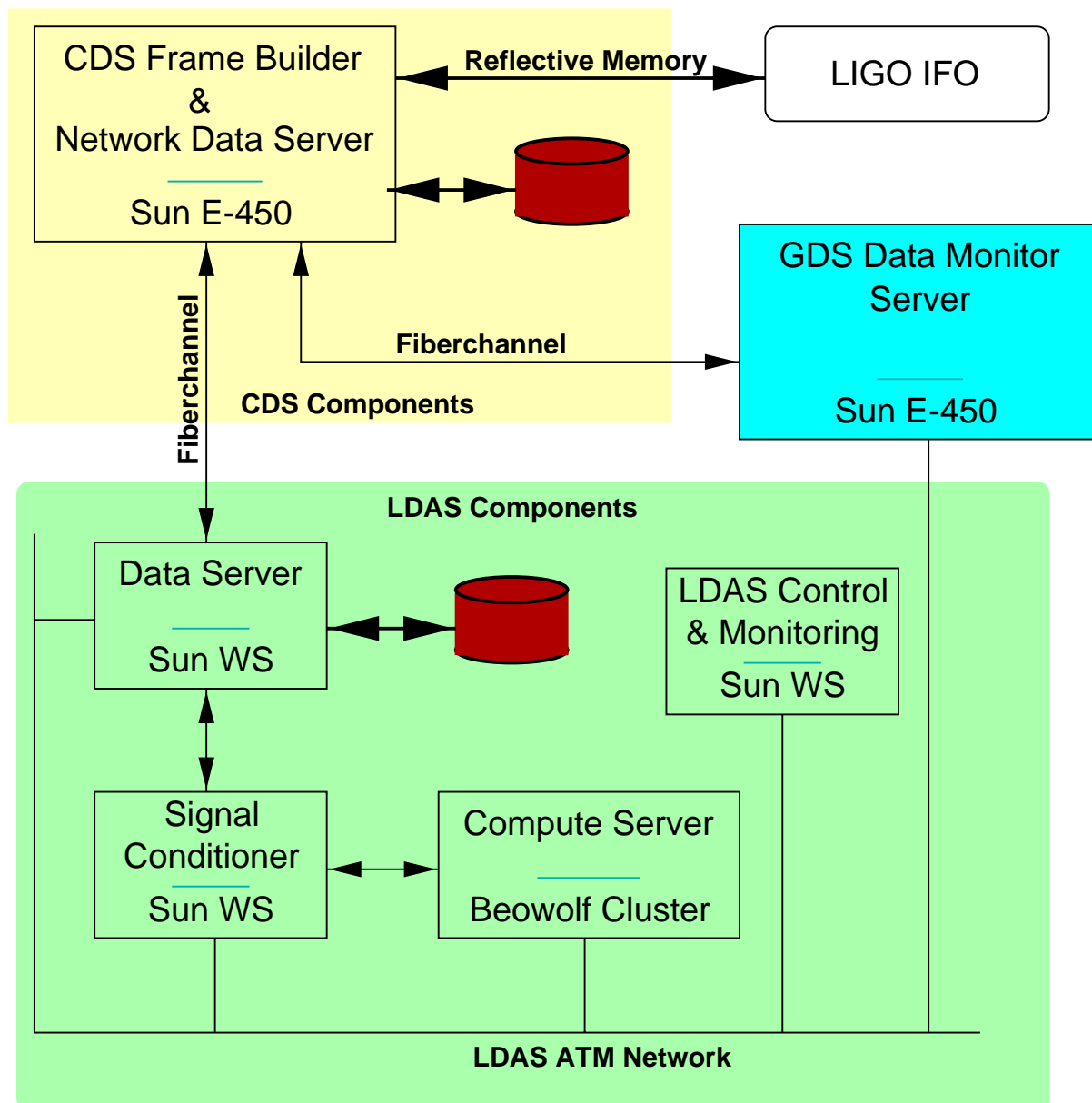
- Online access to current data
- A means to specify & record faults or unusual events (triggers) discovered in the data.
- A means to present data in a way conducive to immediate understanding of the IFO state.
- Sufficient computational power to do all this.

### Example techniques that will be used

- Bit manipulations.
  - Matched filters.
  - Convolutions.
  - Fourier transforms (Power spectra).
  - Exotica (time/frequency plots, wavelets).
-

---

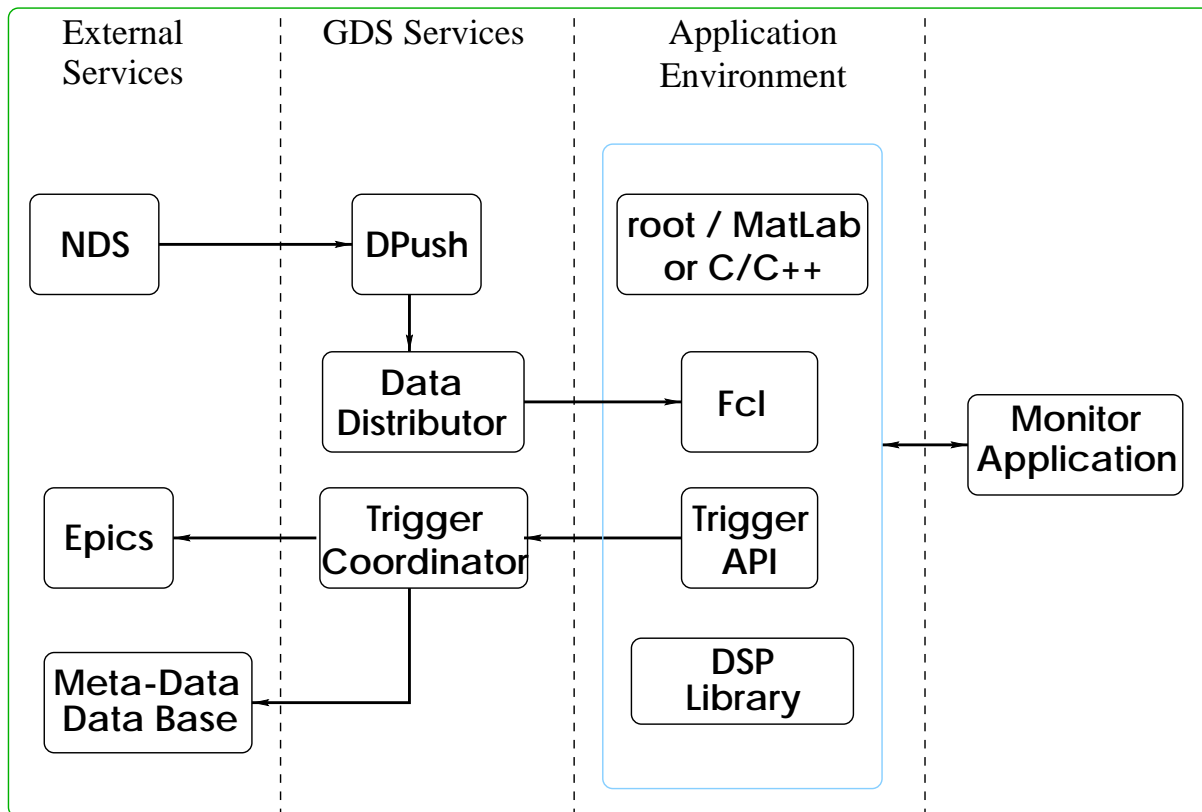
## Hardware Architecture





---

## Software Architecture



---

## Software: Data Distribution

### Frame Data

- Allows off-line testing
- Exchange of Monitors and off-line analysis functions.
- Software available.

### Shared Memory Buffer Manager.

- Allow efficient parallel access to data.
- Provide synchronization primitives.
- Tested: Robust, no bottlenecks.

### Data Pusher

- Copies data from NDS to shared memory.
- Currently running: uses  $\sim 4.5\%$  of Sun Ultra-1.
- Constant use,  $\sim 1\text{MB/s}$  from 40m IFO.

### Shared Memory Utility programs

- **smcreate**: Create a shared memory partition.
- **smdump**: Print partition status.
- **smraw**: Hex+ASCII dump of raw data.
- **smrepair**: Clean up after dead consumers.

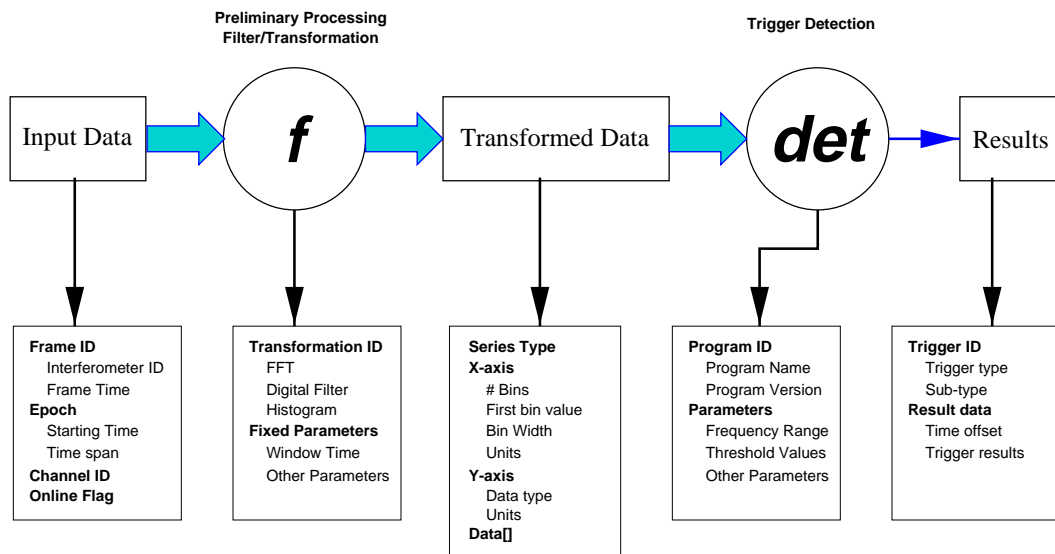
---

# Software: Trigger Management

## Trigger Definition/Requirements

- Triggers describe unusual events detected by an online monitor.
- Trigger records must fully describe the trigger source.
- Triggers must be routed to one/more handlers (operator console, meta-database, expert system).

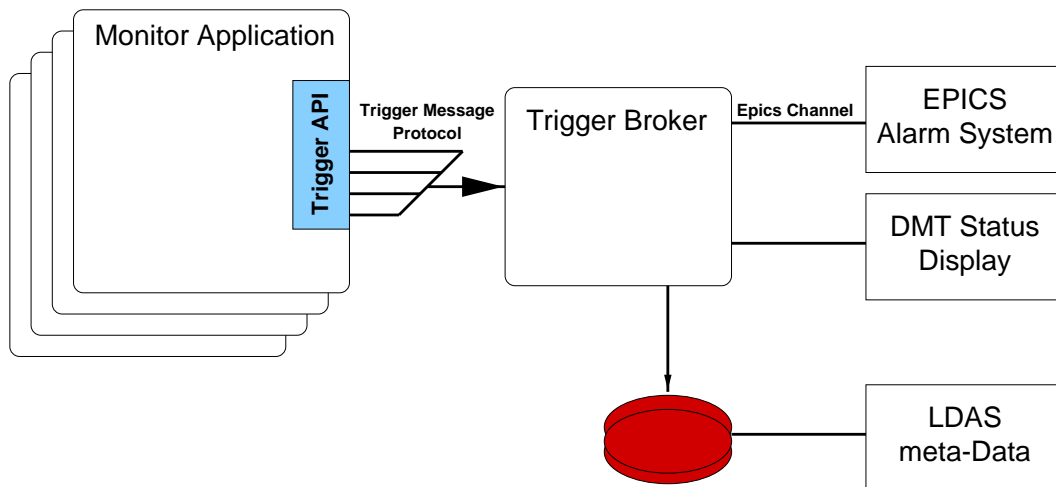
## Trigger Description Model:



---

## Software: Trigger Architecture

---



### Trigger Generation API:

- Classes correspond to trigger description stages.
- User constructs objects.
- Write to disk (prototype) or route to Broker.

### Trigger Broker:

- Maintains routing and status tables.
- Routes triggers to appropriate destination.
- Prevents flooding of message channels or database.

---

## Software: Process Control

The process control subsystem must perform the following functions

- Start up DMT services and background monitors when the DMT platform is booted.
- Make sure all monitors continue to run.
- Make a list of current processes available to a user.
- Allow the user to start, stop or change the parameters of a background monitor or DMT service.

### Status:

- Not necessary until LIGO running starts.
  - Design, implementation will start 4Q1999.
-

---

## Background: C++ Environment

### Monitor structure

- Each monitor is a class derived from DMTBase
- Execution control, I/O handling in base class methods.
- Specific functionality in derived class virtual functions.
- Allows combining monitors into a single process.

### Monitor Specific (User Class) functions:

- **Class constructor:** Initialization.
- **ProcessFrame:** Frame data processing function.
- **Attention:** External interrupt handler.
- **Class destructor:** Termination, present results.

### Additional Classes & Methods

- **Par:** parameter access and storage.
- **Trigger classes:** Trigger generation and routing.
- **Others:** additional classes will be implemented as useful/necessary.

---

## Background: C Environment

### Overview

- C environment provided to encourage contributions from the widest possible base.
- Provide environment similar to that for C++ applications.
- Use simple interface layers (wrappers) when possible between C and C++ implementations.

### Implementation:

- Uses FrameL rather than FrameCPP.
  - Skeleton functions written to emulate the base class functionality.
  - C callable functions interface to DMT classes.
-

---

## Monitor Example: BitTest

**BitTest:** Monitor class

- Based on DMTBase class.
- Inherited Methods: constructor, destructor, ProcessFrame().
- Data member: list of “ChanBit” objects.

**ChanBit:** Channel data handling class

- Methods: constructor, destructor, Scan(), Print(), various accessors.
- Data members: Masks of bits stuck on/off, current & longest repeat counts, last data word, statistics.

**main()** Generated by EXECDMT(BitTest) macro

**BitTest()** Monitor constructor

- Read in a configuration file
- Create ChanBit object for each channel, add it to list.



---

## Monitor Example: BitTest (cont'd)

**BitTest::ProcessFrame()** Process one frames data

- Loop over ChanBit objects in list
- Find/unpack data for the appropriate channel.
- Invoke ChanBit::Scan() to process data.

**~BitTest()** Monitor destructor

- Invoke ChanBit::Print() for each ChanBit object.
- Delete the list.

**ChanBit()** Channel constructor

- Initialize data members.

**ChanBit::Scan()** Process data from one frame.

- Loop over data elements
- *OR* data into stuck off mask, *AND* data into stuck on mask.
- Increment repeat count if data word is the same as last, reset it if not.
- Collect statistics.

**ChanBit::Print()** Print data from one channel.

**~ChanBit()** Monitor destructor (NULL)

---

## Foreground: Overview

The foreground environment must have the following

- Scripting with a high level interpretive language.
- Efficient access to outside data.
- Graphical scientific data presentation capability.
- Programmable user interfaces.

ROOT was chosen as the *initial* foreground implementation because

- Designed for use in Physics experiments.
  - Good support for scientific graphics, GUIs.
  - It's free.
  - Uses a scripting language (C++) that can be used outside the ROOT environment.
  - Easily expandable data object and algorithm libraries.
-

---

## Foreground: Root User Classes

### Data Access

- \* **Dacc:** Get data from file or shared memory.
- \* **Channel:** Unpack Adc data to TSeries.

### Data Containers

- \* **FSeries:** Storage, manipulation of frequency series.
- \* **TSeries:** Storage, manipulation of time series.
- \* **DVector:** Generic data vector.
- \* **Time:** GPS time arithmetic and conversion.
- \* **Interval:** Time intervals.

### Mathematics

- \* **Complex:** Complex number arithmetic (template).
- \* **Poly:** Polynomial manipulation (template).

Note: \*= prototype version exists!

---

## Root User Classes (cont'd)

### Science

- \* **Chirp:** Base class for template functions.
- \* **Inspiral:** Newtonian binary in-spiral template.
- **Other Templates:**
- **Optimal Filter:** Find template waveform in FSeries.

### Engineering

- **FIR Filter Design:** High-pass, low-pass, etc.
- **Filter Implementation:** In the time domain.

Note: \* = prototype version exists!

---

---

## Foreground: Root Macros

Interpretive scripts (macros) are used in the foreground to enter commonly used groups of commands or procedures.

### Session Control

- \* **sboxlogon:** Attach input data source, load useful macros.
- \* **sboxlogoff:** Release (close) data partition (file).

### Graphical Primitives

- \* **Bode:** Bode plot of an FSeries.
- **Spectrum:** Frequency spectrum.
- \* **THist:** Histogram a time series.
- \* **TPlot:** Plot a time series.

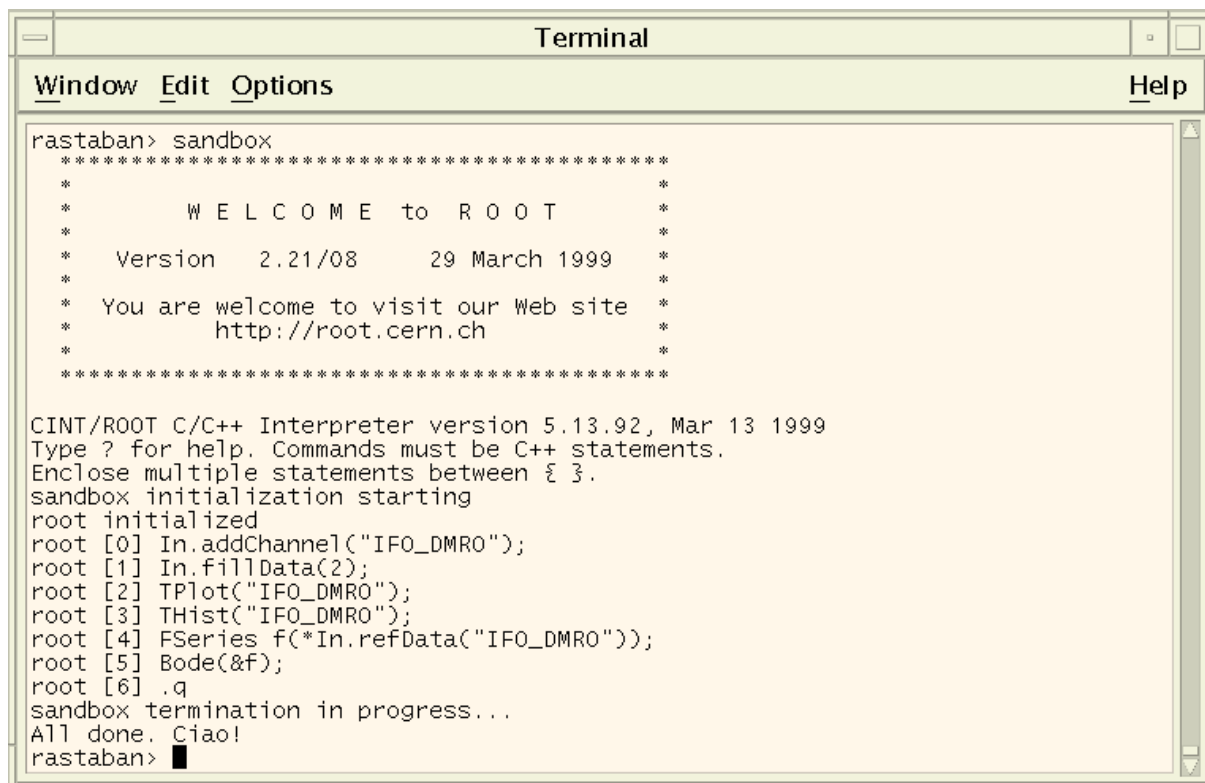
### Graphical User Interfaces

- Noise spectrum (best estimate).
- Volume of sensitivity vs. time.

Note: \*= prototype version exists!

---

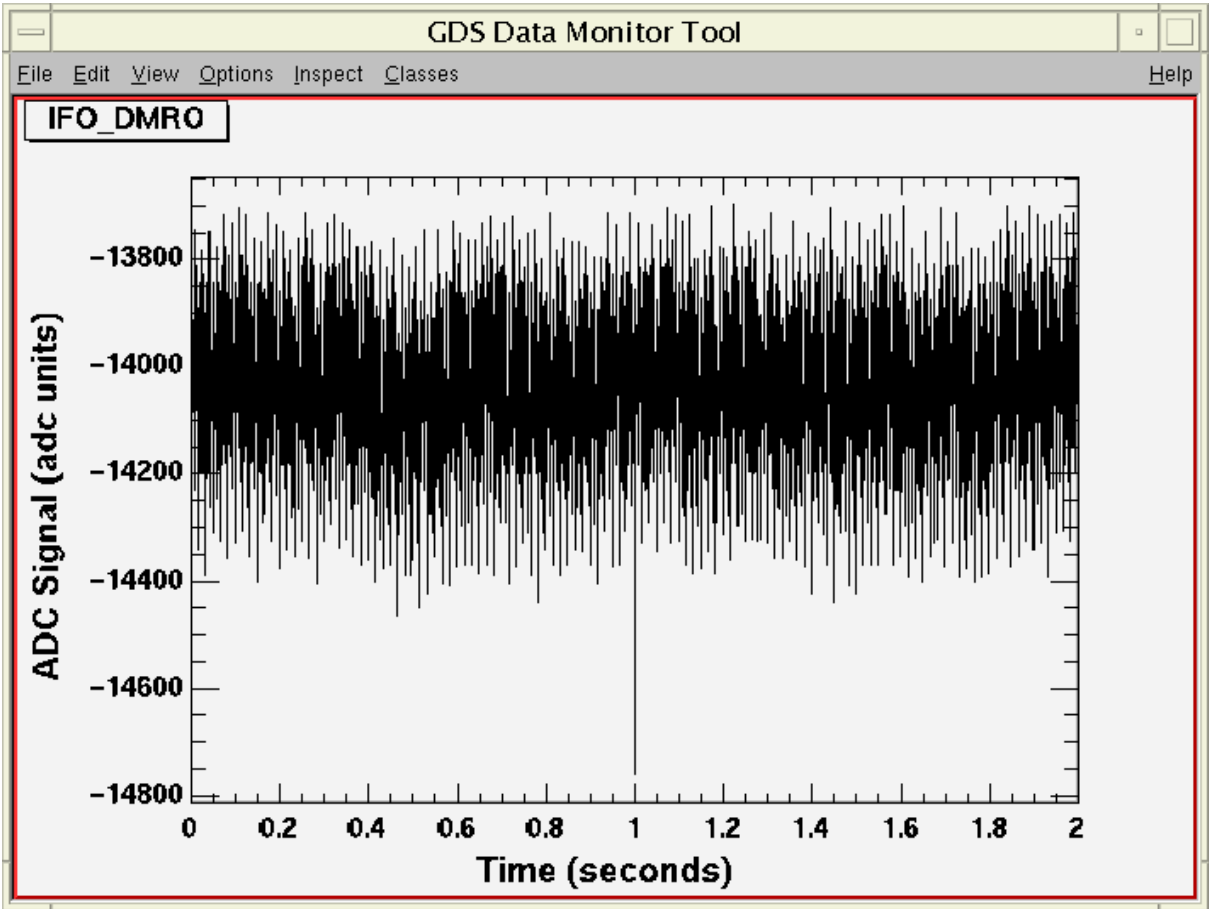
## Foreground: ROOT Session



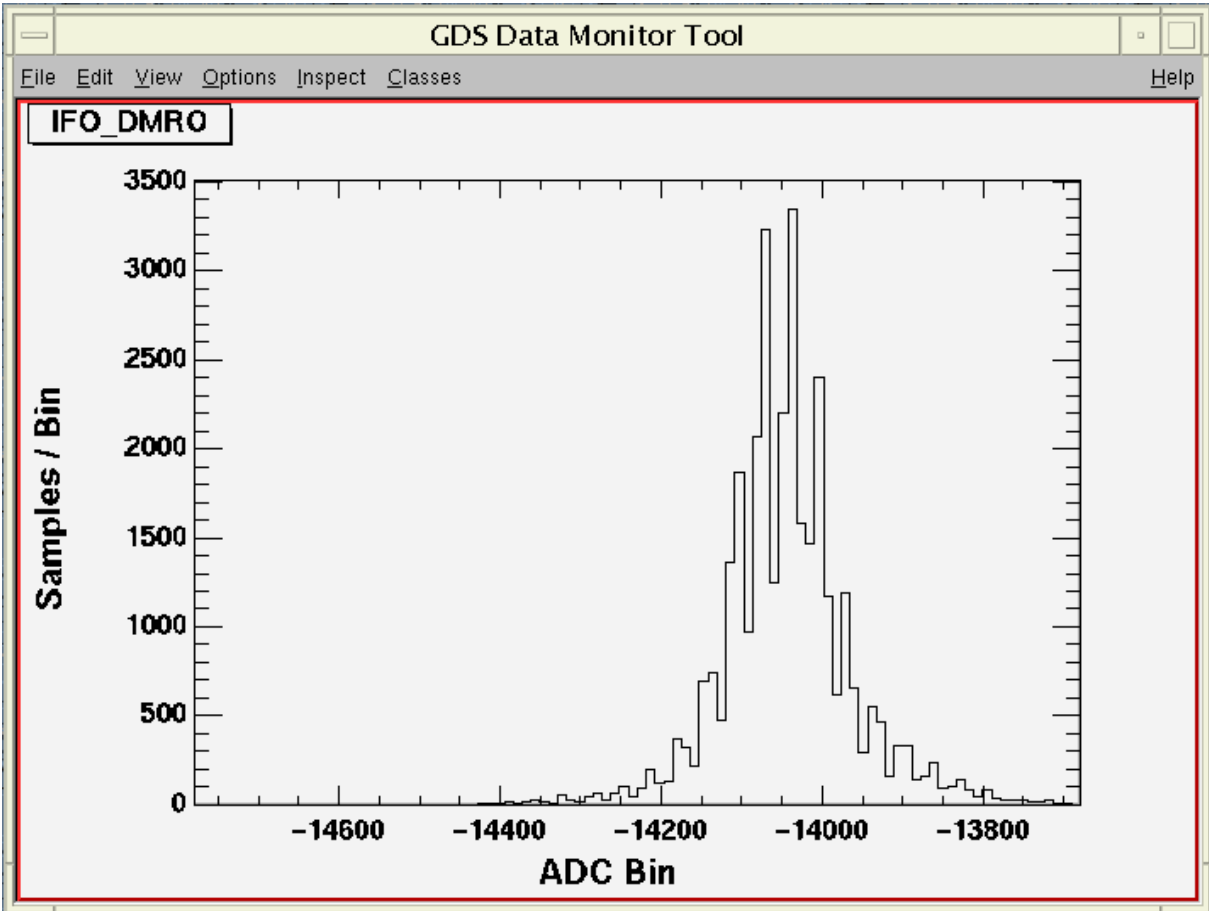
```
Terminal
Window Edit Options Help
rastaban> sandbox
*****
*
*      W E L C O M E to R O O T      *
*
*   Version  2.21/08    29 March 1999  *
*
*   You are welcome to visit our Web site *
*         http://root.cern.ch          *
*
*****

CINT/ROOT C/C++ Interpreter version 5.13.92, Mar 13 1999
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
sandbox initialization starting
root initialized
root [0] In.addChannel("IFO_DMRO");
root [1] In.fillData(2);
root [2] TPlot("IFO_DMRO");
root [3] THist("IFO_DMRO");
root [4] FSeries f(*In.refData("IFO_DMRO"));
root [5] Bode(&f);
root [6] .q
sandbox termination in progress...
All done. Ciao!
rastaban> █
```

# Foreground: ROOT Time Plot

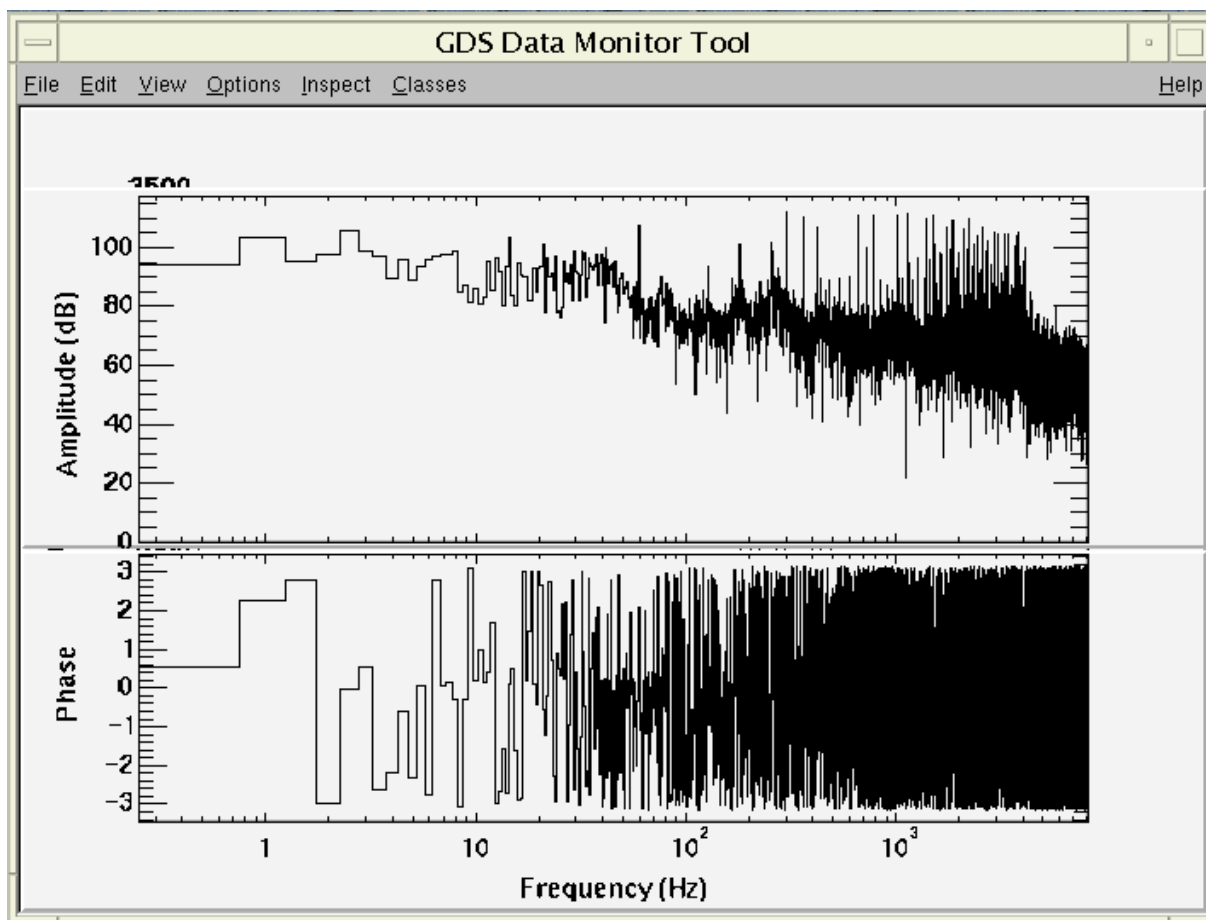


# Foreground: ROOT ADC Histogram





# Foreground: ROOT Bode Plot



# COSTS

Pos	Qty.	Item	Unit	Ext.
excitation engine (LVEA, 1 / ifo)				
1	1	VME crate	3200	3200
2	2	Baja4700E-200-64, CPU	8500	19000
3	1	VMIC 5588, reflective memory	7400	7400
4	1	VMIC 5591, optical bypass switch	1200	1200
5	1	Brandywine syncclock32, GPS slave	1200	1200
6	1	CDS timing board	1000	1000
7	1	ICS 115, 32 channel DAC	9595	9595
8	1	CDS AA filters, 32 chn	6000	6000
				48595
excitation engine (mid/end, 1 / ifo)				
9	2	ICS 115, 4 chn	3595	7190
10	2	CDS timing board	1000	2000
11	2	CDS AA filters, 4 chn	1000	2000
				11190



# COSTS (2)

Pos	Qty.	Item	Unit	Ext.
data monitoring tool (MSR, 1 / ifo)				
12	1	SUN450, 4x400MHz, 4MB cache, 1GB RAM, 3x9GB disk, 17" monitor	50000	50000
13	1	ATM OC12, ForeRunner HE622 & NM-1/622MMSCLC	6000	6000
				56000
network data server for DMT (MSR, 1 / site)				
14	1	SUN Ultra10, 330MHz, 512MB RAM, 9GB disk	4000	4000
15	1	VMIC 5588, reflective memory, PCI	9000	9000
16	1	VMIC 5591, optical bypass switch	1200	1200
17	1	ATM OC12, ForeRunner HE622 & NM-1/622MMSCLC	6000	6000
				20200
hardware total (3 ifos)				
				<b>387755</b>



# COSTS (3)

---

Pos	Qty.	Item	Unit	Ext.
software support				
18	1	CDS 1 FTE, 6 month	80000	80000
total				
		allocated		<b>180000</b>
		estimate		<b>467755</b>
		difference (CCB)		<b>-287755</b>



# WORK LOAD SOFTWARE DEVELOPMENT

---

Activity	Personnel	FTE (month)	
		to date	needed
<b>Diagnostics tests</b>			
Excitation engine	D. Sigg/M. Pratt	6	0
Diagnostics test kernel	D. Sigg	8	3
Analysis algorithms	P. Fritschel/E. Daw	2	2
DAC hardware driver	C. Patton	0.5	0.5
GUI	D. Barker/C. Patton		3.5
<b>Data monitor tool</b>			
Application environment GDS services	J. Zweizig	5	8
Frame builder extensions	R. Bork/A. Ivanov		2
LDAS DB tables	B. Kratochwill	0.5	0.5



# SCHEDULE

Task	LHO 2K	
	start	stop
<b>excitation engine</b>		
software development	1/1/98	6/30/99
hardware installation	5/1/99	7/31/99
<b>test point manager</b>		
software development	9/1/98	4/28/99
integration test with DAQS and ISC	5/1/99	6/30/99
<b>diagnostics kernel</b>		
test software	8/1/98	7/31/99
command line interface	1/1/99	5/1/99
GUI	4/1/99	12/31/99
<b>data monitoring tool</b>		
data distribution	1/1/99	5/31/99
trigger manager	6/1/99	10/1/99
signal processing library	3/1/99	12/31/99
application environment	3/1/99	10/1/99
prototype at LHO	5/1/99	6/1/99
final hardware	8/1/99	12/31/99



# PDR ACTION ITEMS

---

1. GDS should coordinate with all subsystems test input/output points.

*ISC/IOO done. GDS will provide temporary test inputs/outputs.*

2. Determine the processor/workstation allotment for GDS. Identify the computing components that are CDS, DAQS, LDAS and the components that GDS must provide.

*Done. GDS: excitation engine, DMT workstations & DMT frame builder.*

3. GDS should coordinate with CDS any extended bandwidth requirements, resulting from data transfers such as FFTs, on the CDS local area network.

*Done. Front-end compute engine has been eliminated. The DMT will have its own frame-builder.*

4. Define the interfaces that will be used to access the GDS database. This database should have commons interface and data formats with other LIGO databases being access.

*Done. GDS doesn't have a db. Diagnostics test use LIGO LW XML; DMT uses event tables in the LDAS db.*

5. The “Test Template Result Record” contains a “detector state vector” sub-component. This detector state vector definition should be specified and reviewed by a broader audience.

*DMT provides infrastructure to compute state vector. Diagnostics tests do not save the state vector anymore. Definition → sys.*



# PDR ACTION ITEMS (2)

---

6. The “Standard Result Record” should have a standardized format in common with LDAS.

*Done. Diagnostics tests are saved in LIGO LW XML.*

7. GDS should coordinate “event” object definition with LDAS which has a similar data object of “lightweight format”.

*Done. DMT events go into LDAS db.*

8. GDS should coordinate with CDS and LDAS on issues of databases, formats, APIs, and other common components.

*Database: Done (all LDAS).*

*Formats: Done (LIGO LW XML).*

*APIs (reflectivce memory APIs & NDS): Done.*

9. The GDS database should have the capability to archive and access LSC measurements used in calibration and auto-regression of dataset.

*GDS doesn't maintain its own db. Calibration measurements will be saved as XML files.*

10. Implement a “state checking” capability into the “doing test” functions allowing determination of correct state of instrument for each particular test.

*Open. Waits for the definition of the state vector. Will be implemented on request.*





# PDR ACTION ITEMS (3)

---

11. Implement an “acquiring data” capability into each test verifying that needed channels and bandwidth (rate) are available to each particular test.

*Done. Checks done automatically before starting test.*

12. The 1/16th of a second delivery of data could tax components in the GDS and other sub-systems.

*Diagnostics tests: Turned out to be no problem.*

*DMT: now works at a rate of 1 Hz; has its own frame builder, networking and hardware.*

Provide the capability to decimate data rates to slower rates where and when appropriate to diagnostics.

*Diagnostics tests: data rates don't seem to be a problem.*

*DMT: uses frames of full data sets to simplify NDS.*

13. Identify the method and technology for providing synchronized video frames to the 16 Hz data delivery, allowing triggers to capture video where and when needed.

*Video is analog only. A video time insertion module using an IRIG-B signal has been identified.*



# PDR ACTION ITEMS (4)

---

14. Make identification of “dead channels” a lower priority in the sense that accurate identification of a dead channel is expected to be very difficult. Consider moving detection of dead channels over to the RTDD where it could handle “broken channel” triggers on time scales of minutes without driving scope of real-time components.

*Will be done. Also, “broken channel” will cycle through channels, rather than work on them concurrently.*

15. Add manual reset to “broken channel” trigger specification.

*Will use EPICS alarm system.*

16. Provide a method for tracking (identifying) “portable channels” such as those associated with digitizing oscilloscopes, thereby removing possibility to misidentify a signal’s source.

*Rely on iLog.*

17. Coordinate with 40 meter personnel to identify a baseline of useful and necessary visual display (viewing) tools.

*GDS uses CDS (40m) data viewer.*

18. Provide baseline requirement for viewing tools, e.g., number of channels, sample rates, etc. Separate out view tool requirements from pre-viewing number crunching methods, possibly running on another processor.

*Descoped. No preprocessing except FFTs through XMgr.*



# PDR ACTION ITEMS (5)

---

19. GDS stimulus component should be able to drive actuation points at above Nyquist frequencies to test for aliasing and other down conversion paths.

*Done. Support of SR DS340 with 15 MHz bandwidth.*

20. Estimate on-line computing needs of the GDS. Show current planned suite of calculations assumed and provide estimates for each calculation, e.g., FFTs, trend estimates, RMS, peak-to-peak, amplitude probability distributions, frequency vs. time distributions, etc.

*Real-time number cruncher has been eliminated.*

*Trend data: part of frame builder.*

*Diagnostics tests: not a problem (control room workstation).*

*DMT: open.*

21. Identify on-line (or pre-diagnostics testing) calibrations that must be available for diagnostics and the interface to be used in accessing these calibrations.

*Calibration corrections will be off-line only.*

22. Provide reading and writing methods for parameters used in particular diagnostic tests via the database, allowing easy insertion/extraction of previous parameters sets in a later (possibly repeated) test.

*Database: Parameters stored in XML LIGO LW.*

*Insertion/extraction: Supported through partial save/restore.*



# PDR ACTION ITEMS (6)

---

23. Coordinate with the instrumental characterization group recently formed within the LSC to better utilize interested man power outside of LIGO.

*Done, but slow.*

