EPICS and the SDDS Toolkit

A contribution to the "Getting Started with EPICS" Lecture Series

Michael Borland
Operations Analysis Group
APS Operations Division
October 1, 2004

A U.S. Department of Energy
Office of Science Laboratory
Operated by The University of Chicago

Outline

- Brief history
- SDDS:
  - Concept
  - Advantages
  - Implementation
- Data analysis capabilities
- Data collection capabilities
- Process control capabilities
- Demos
- Application examples

A Brief History of SDDS

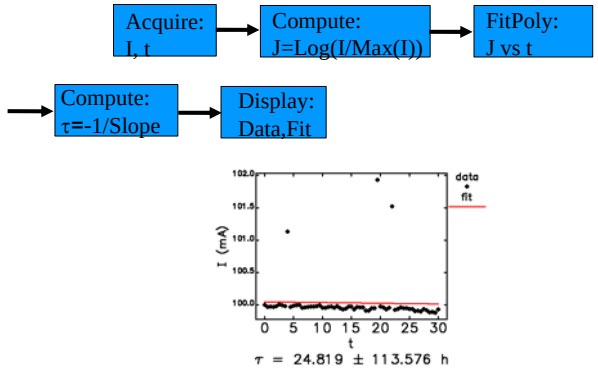
- In 1993, needed to find or create general-purpose software for APS commissioning
- Developed the Self Describing Data Sets (SDDS) file protocol and toolkits to meet this need
- Planned to later write traditional high-level applications based on algorithms developed with these tools
- Concept worked so well that it was used directly in operations
- SDDS used at APS, DESY, IPNS, BESSY II, RHIC, SLAC, ...

SDDS Concept is Unix-Inspired

	SDDS
• Everything is a file	• Everything is a self-describing file
• Programs are "filters" operating on ASCII streams	• Programs are operators that transform datasets
• Pipes allow sequencing filters arbitrarily	• Pipes allow sequencing operators arbitrarily
• Anyone can add a participating program	• Anyone can add a participating program
• New programs lead to new uses for old ones	• New programs lead to new uses for old ones

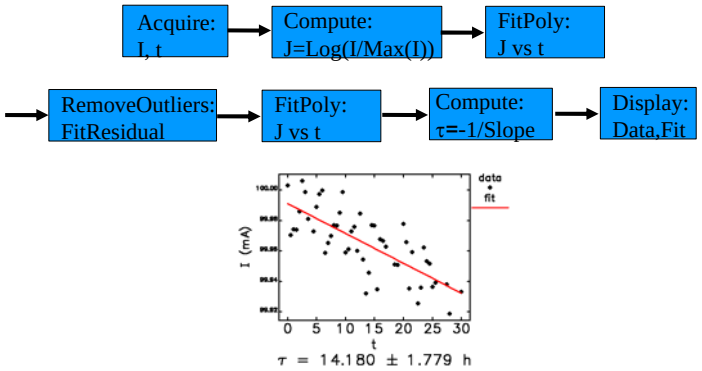
Example of the Concept

Modularized beam lifetime measurement

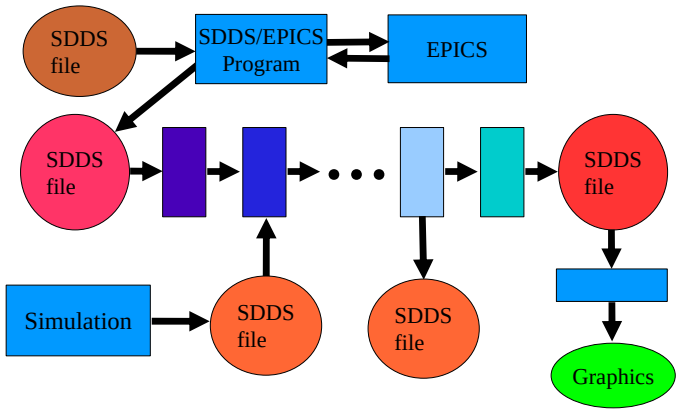


Example of the Concept

Improved modularized beam lifetime measurement



Concept Supports Very Complex Operations



SDDS

- Components
  - A stable, general-purpose, self-describing file protocol
  - Generic tools that operate on SDDS files
  - EPICS tools that are configured by SDDS files
  - Libraries for working with such files
- Multiplatform and open-source
  - Solaris, Linux, Windows, OS-X, VxWorks
- Supported languages
  - Shell commandline
  - C/C++, Tcl/Tk, Python, Java, IDL, MATLAB, FORTRAN

SDDS File Protocol

- Strictly self-describing data
  - Data is accessed by name only
  - Meta-data includes units, description, data type
- Data model
  - Complex enough to be useful, simple enough to be useable
  - File has a sequence of instances of a structure
  - Structure contains
    - Parameters (scalar values)
    - Table
    - Arbitrarily-dimensioned arrays (little-used)
- Options for binary, ASCII, and compressed storage

Why Use SDDS Files?

- Increased robustness and flexibility
  - Check existence, data-type, units of data instead of crashing
  - Respond appropriately to missing or wrong data
    - Exit and warn user
    - Apply units conversion
  - Old data doesn't become obsolete when program is upgraded
    - Use defaults for missing data
- Data sets can evolve without breaking applications
- Multiple uses for one data set are possible

Advantages of SDDS-Configured Programs

	Typical Non-SDDS
• Create configurations using scripts, SDDS tools, sddsedit, or text editor	• Create configurations using text editor or custom interface
• Configuration is data, shared among programs	• Configuration is text, specific to one program
• Use SDDS tools to combine, sort, merge, and select configuration data	• Write your own tools to manipulate configurations, or do it by hand
• Use SDDS tools and scripts to update configuration data	• Update configuration data by hand
• Use same SDDS tools to postprocess program output	• Use different tools to postprocess program output

SDDS File Example (Conceptual)

```
Column: CityName, type=string
Column: HighTemperature, type=double, units=F
Column: LowTemperature, type=double, units=F
Parameter: CountryName, type=string
Parameter: MaximumHighTemperature, type=double, units=F
Parameter: MinimumLowTemperature, type=double, units=F
Parameter: ColdestCity, type=string
Parameter: WarmestCity, type=string
```

- Header defines three columns and five parameters
- Typically parameters either
  - Relevant to context of column data (e.g., CountryName)
  - Abstracted from column data (e.g., ColdestCity)

SDDS File Example (Conceptual)

- Pages contain instances of data defined by header

Page 1

Parameter data

CountryName:	Minimum	Maximum	ColdestCity:	WarmestCity:
US	LowTemperature: 40	HighTemperature: 86	Chicago	Miami

Column data

CityName	HighTemperature	LowTemperature
Chicago	78	40
Dallas	82	63
Miami	86	67

Page 2

Parameter data

CountryName:	Minimum	Maximum	ColdestCity:	WarmestCity:
China	LowTemperature: 53	HighTemperature: 87	Beijing	Nanjing

Column data

CityName	HighTemperature	LowTemperature
Beijing	77	53
Nanjing	87	62
Shanghai	80	68
Chongching	86	68

SDDS and Tcl/Tk

- SDDS and Tcl/Tk complement each other
- Tcl/Tk is a good language for GUIs, but
  - Lacks data management
  - Not great for computation
- SDDS offers data management, analysis, and computation, but
  - Is not a programming language
  - Has commandline user interface
- Both are open source and multi-platform
- SDDS extensions available for other languages too

SDDS Quality Control

- We use SDDS software to operate the APS, including
  - Data logging
  - Closed-loop feedback and feedforward
  - Top-up injection control
- Downtime due to SDDS software is essentially nonexistent
- We also use the software on a daily basis for simulation work
- We perform regression testing prior to releasing changes

Why Not Use XYZ Instead?

- SDDS is similar in capability to other systems
  - Mathematical capabities comparable to MATLAB or IDL
  - Data manipulation capabilities similar to a database
- SDDS was preferred by commissioning staff over commercial software
- SDDS advantages
  - APS uses and supports it
  - File-based system improves data management
  - It is free and open source
  - Extensions provided for other packages
  - People who learn it tend to really like it

SDDS Toolkit Capabilities

- Display
  - `sddsquery`: print out description of file contents
  - `sddsplot`: workhorse graphics program
    - X11, Windows, postscript, PNG, etc.
    - Plots (x,y) data or vector fields
    - Multi-panel, multi-axes plotting
    - Label plots with parameter data
  - `sddscontour`: contour and color-map plots
  - `sddsprintout`:
    - text and spreadsheet output
    - LaTeX table creation

SDDS Toolkit Capabilities

- Mathematical
  - `sddsprocess`: workhorse computational program
    - Analyze columns to create parameters
    - Create new columns and parameters using equations
    - Match and filter based on logic expressions for columns and parameters
  - `sddsinterp, sddsinterpset`: interpolate data
  - `sddsinteg, sddsderiv`: numerical integration and differentiation
  - `sddssmooth`: smooth and despike data
  - `sddspeakfind`: find peaks in data
  - `sddszerofind`: find zeros in data

SDDS Toolkit Capabilities

- Fitting:
  - `sddspfit, sddsmpfit`: polynomial fitting
  - `sddsexpfit, sddsgfit`: exponential and gaussian fitting
  - `sddsgenericfit`: fits a user-defined functional form
  - `sddsslopes`: used to create response matrices

SDDS Toolkit Capabilities

- Statistics
  - `sddscorrelate` and `sddsshiftcor`: correlation analysis
  - `sddshist`, `sddsmultihist`, `sddshist2d`: one- and two-dimensional histogramming
  - `sddsoutlier`: outlier analysis and removal
  - `sddsrunstats`: running statistics
  - `sddsrowstats`: statistics across columns

SDDS Toolkit Capabilities

- Digital signal processing
  - `sddsfft`: Fourier transforms and PSDs
  - `sddsnaff`: Numerical Analysis of Fundamental Frequencies
  - `sddsconvolve`: convolution, deconvolution, correlation
  - `sddsdigfilter, sddsfdfilter`: time- and frequency-domain digital filters
- Matrix operations:
  - `sddspseudoinverse`: invert matrix using SVD
  - `sddsmatrixop`: RPN matrix calculator for files

SDDS Toolkit Capabilities

- Data manipulation
  - `sddssort`: multi-field sort by columns or parameters
  - `sddscombine`: combine or merge datasets
  - `sddsxref`: match data between datasets and import columns, parameters, and arrays
  - `sdds2plaindata, plaindata2sdds`: convert to/from unadorned text or binary and SDDS
  - `sddscollapse`: create a new table from several pages of table parameters
  - `sddscollect`: collect data from like-named columns to create new columns indexed by the name prefix

`sddscollapse` Example

- Two page file prior to `sddscollapse`

Page 1

Parameter data

CountryName:	Minimum LowTemperature:	Maximum HighTemperature:	ColdestCity:	WarmestCity:
US	40	86	Chicago	Miami

Column data

CityName	HighTemperature	LowTemperature
Chicago	78	40
Dallas	82	63
Miami	86	67

Page 2

Parameter data

CountryName:	Minimum LowTemperature:	Maximum HighTemperature:	ColdestCity:	WarmestCity:
China	53	87	Beijing	Nanjing

Column data

CityName	HighTemperature	LowTemperature
Beijing	77	53
Nanjing	87	62
Shanghai	80	68
Chongching	86	68

`sddscollapse` Example

- Two page file after `sddscollapse`

```
% sddscollapse input output
```

Page 1

Column data

CountryName	Minimum LowTemperature	Maximum HighTemperature	ColdestCity	WarmestCity
US	40	86	Chicago	Miami
China	53	87	Beijing	Nanjing

- Former parameters are now columns
- There are no parameters in the new file

sddscollect Example

- Starting data file

Column data

Time	ChicagoTemp	SanFranTemp	MiamiTemp	etc
12:00:00 am	50	47	67	
01:00:00 am	51	46	65	
02:00:00 am	49	48	66	
12:00:00 am				

- Processed data file (`sddsprocess`)

```
% sddsprocess input output -process=*Temp,max,%sMax \
  -process=*Temp,min,%sMin -process=*Temp,ave,%sMean
```

Parameter data

ChicagoTempMax: 81	ChicagoTempMin: 42	ChicagoTempMean: 65
SanFranTempMax: 67	SanFranTempMin: 45	SanFranTempMean: 55
MiamiTempMax: 88	MiamiTempMin: 64	MiamiTempMean: 74
etc.		

Column data

Time	ChicagoTemp	SanFranTemp	MiamiTemp	etc
12:00:00 am	50	47	67	
01:00:00 am	51	46	65	
02:00:00 am	49	48	66	
12:00:00 am				

sddscollect Example

- Collapsed data file:

```
% sddscollapse input output
```

Column data

ChicagoTempMax	ChicagoTempMin	ChicagoTempMean	SanFranTempMax	etc
81	42	65	67	

- Collected collapsed data file:

```
% sddscollapse input -pipe=out | sddscollect -pipe=in output \
  -collect=suffix=TempMax -collect=suffix=TempMin \
  -collect=suffix=TempMean
```

Column data

Rootname	TempMax	TempMin	TempMean
Chicago	81	42	65
SanFran	67	45	55
Miami	88	64	66
etc.			

sddscollect Example

- Process again

```
% ... | sddsprocess -pipe=in output \
 -process=TempMax,max,TempMaxMax \
 -process=TempMax,max,WarmestCity,functionOf=Rootname,position
```

Parameter data

TempMaxMax:88	WarmestCity: Miami

Column data

Rootname	TempMax	TempMin	TempMean
Chicago	81	42	65
SanFran	67	45	55
Miami	88	64	66
etc.			

SDDS Toolkit Capabilities

- Miscellaneous
  - `sddsmakedataset`: make an SDDS dataset using data provided on commandline
  - `sddssampledist`: provide random or Halton-sequenced values to match a given probability distribution
  - `sddsdistest`: determine the probability that data is drawn from a specified distribution
  - `sddsspotanalysis`: analyze images of beam spots
  - `sddsimageprofiles`: make x and y profiles from an image
- Many more: about 80 total

SDDS/EPICS Toolkit Capabilities

- Scalar data collection
  - `sddsmonitor`
    - Venerable general-purpose time-series logger
    - Glitch-, trigger-, and alarm-initiated logging with a circular buffer
    - Conditional logging, log on-command
  - `sddslogger`
    - Time-series logging
    - Conditional logging, log on-command
    - PV-strobed logging
    - Multiple input and output files
  - `sddslogger` is APS's workhorse data logging program

SDDS/EPICS Toolkit Capabilities

- Statistics logging
  - `sddsstatmon`
    - Collects N samples at specified interval, then logs statistics:
      - Statistics are individual selectable
      - Mean, minimum, maximum, standard deviation, sigma, sample, sum
    - Conditional logging, log on-command

SDDS/EPICS Toolkit Capabilities

- Glitch data collection
  - Use circular buffers to record data at a high rate
  - Dump data to file when a predefined event occurs
  - `sddsmonitor`:
    - Quick and easy for simple triggers
    - Can't select what to record based on type of trigger
  - `sddsglitchlogger`
    - Multiple trigger sets
    - Record only information specific to triggered set
    - Multiple output files

SDDS/EPICS Toolkit Capabilities

- Monitor-based data collection
  - `sddslogonchange`
    - May specify dead-bands to limit logging of small changes
    - Space-efficient coded format
    - Conditional logging supported
    - Used to log “all” changes to accelerator setpoints for review and rollback
  - `sddsalarmlog`
    - Logs alarms in a space-efficient coded format
    - Can log related PV when alarm occurs
    - Used for archival alarm logging and post-mortem analysis

SDDS/EPICS Toolkit Capabilities

- Synchronized collection: `sddssynchlog`
  - Does time-stamp alignment of high-rate data
  - Supports scalars and waveforms
  - Optionally collects related, unsynchronized data
  - Used for on-demand investigation of correlations

SDDS/EPICS Toolkit Capabilities

- Waveform collection:
  - `sddswmonitor`:
    - Log waveforms at intervals, or when changed
    - Specify waveform PVs in file or on commandline
    - Simultaneous collection of scalar values
  - `sddswget/sddswput`
    - Get waveform snapshot and write to file
    - Write saved waveform from file to PV
    - SDDS toolkit to make transformations in between

SDDS/EPICS Toolkit Capabilities

- Experiment execution (`sddsexperiment`)
  - N-dimensional experiments
  - Data and statistics collection
  - Validity testing
  - Subprocess execution
- Applications include
  - BPM offset measurement
  - Measuring response matrices for feedback
  - Characterizing ID x-ray BPMs to allow feedforward
- Great as an experiment execution engine for scripts
- `ExperimentDesigner` better for interactive use

SDDS/EPICS Toolkit Capabilities

- Feedback: **`sddscontrollaw`**
  - Generic proportional or integral feedback
  - Validity testing, change limits, deadbands, logging
  - PV controls include locking semaphores, gain control
  - Will run under VxWorks
- Applications include
  - Storage ring orbit control
  - Beamline steering
  - Linac energy and trajectory control
  - Quick one-parameter feedback GUI

SDDS/EPICS Toolkit Capabilities

- **`sddsfeedforward`**
  - Generic feedforward program
  - Multiple input and output PVs
  - Locking semaphores
  - Will run under VxWorks
- Applications
  - X-ray BPM gap-dependence compensation
  - Rf BPM intensity-dependence compensation
  - EMW switching correction
  - Septum magnet temperature drift compensation

SDDS/EPICS Toolkit Capabilities

- Generic optimization (**`sddsoptimize`**)
  - Simplex or successive 1D scan methods
  - Validity testing
  - Script option for setting conditions
  - Script option for computing penalty function
- Applications include
  - Kicker bump matching
  - Coupling optimization
  - Injector efficiency optimization
  - Optimization of simulation results

SDDS/EPICS Toolkit Capabilities

- Save/restore
  - Venerable `burtrb`/`burtwb` pair are (mostly) SDDS-compliant
  - New **`sddscasr`** program is completely compliant
    - Faster than `burtrb`/`burtwb`
    - Server mode with PV controls is faster yet
    - Waveform save/restore
  - Program **`sddscaramp`** ramps through a sequence of snapshots

SDDS/EPICS Toolkit Capabilities

- PV creation
  - PVs can be created on-the-fly with `sddspcas`
  - SDDS-configured by a file that can also double as
    - Data logger input file
    - Save/restore input file
  - Creates scalar and waveform PVs
  - Checks for existence of PVs before creating
  - Handy for development work

Related Capabilities

- Message logging
  - `logDaemon`
    - Server for multi-log message logging
  - `logMessage`
    - Utility for sending messages to `logDaemon`
  - Used for logging script activity on APS control system

Setting up a feedback process

- Collect names of error PVs (to be regulated)
- Collect names of actuator PVs (used for control)
- For each actuator
  - Use `sddsexperiment` to vary actuator and record error readbacks
  - Use `sddsslopes` and `sddscollect` to make response vector file
- Use `sddsxref` to combine response vector files into a response matrix
- Use `sddspseudoinverse` to invert the response matrix
- Use `sddscontrollaw` to run the feedback with the inverse matrix

Setting up a feedforward process

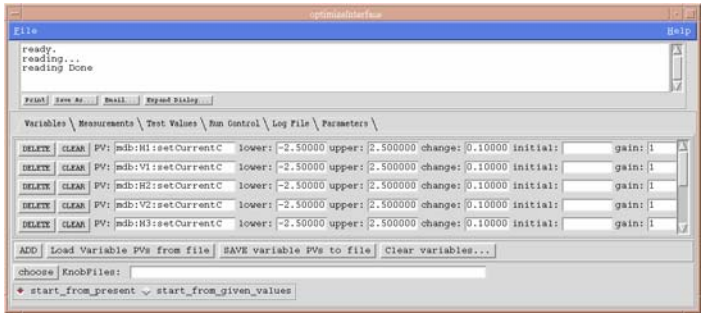
- Set up and run feedback process
- For each perturbation source
  - Use `sddsexperiment` to vary the perturbation while reading the relevant feedback actuators
  - For each actuator
    - Use `sddsprocess` to extract table of pertubation values and actuator values
    - Optionally use, e.g., `sddspfit` to get a smooth fit
  - Use `sddscombine` to merge these into a multipage file, one page per actuator
- Use `sddscombine` to merge per-perturbation files into a single multipage file
- Use this file with `sddsfeedforward`

Demonstrations

- Demonstrations can be downloaded from Web
- Tested on Linux only
- Presently work with Base 3.13.X
- After downloading, see the README file for help getting started
- Includes
  - Data acquistion and processing
  - Response matrix measurement and feedback

Example: Generic EPICS Optimization GUI

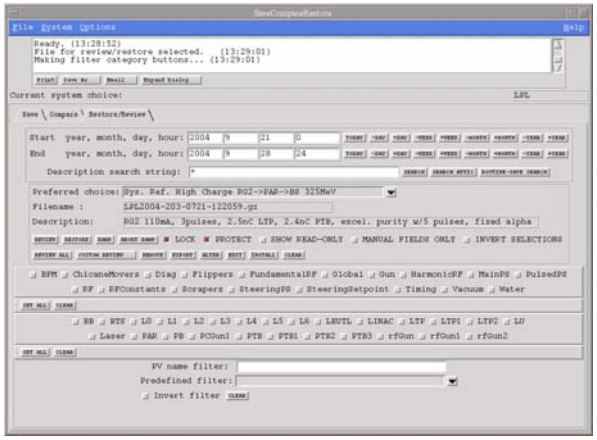
- Configuration data saved/restored from SDDS files
- Uses sddsoptimize to perform optimization
- Uses sddsplot to display progress



Example: Save/Compare/Restore System

- SDDS request and snapshot files have PV meta-data
  - System, subsystem
  - Data type (numerical, enumerated)
  - Access mode (read-only, protected, manual-only)
  - Tolerance
- Uses SDDS files to keep lists of "reference" configuration names and permissions
- Activity logs in SDDS files can be used to see exactly what was done when by whom.
- Uses `sddsprocess` for subset selection
- Uses `sddscasr` and `sddscaramp` for save/restore

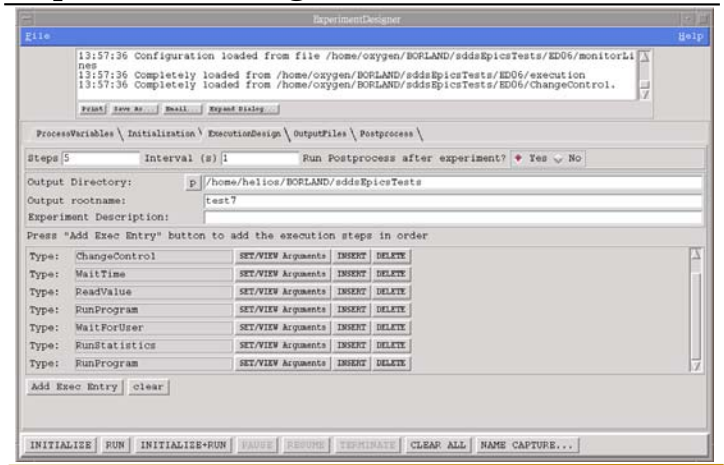
SaveCompareRestore GUI



Example: ExperimentDesigner

- This is a GUI application for performing complex experiments
- Uses SDDS files for saving and loading application configuration
- Uses SDDS/EPICS Toolkit for data collection
  - sddsstatmon
  - sddslogger
  - sddswmonitor

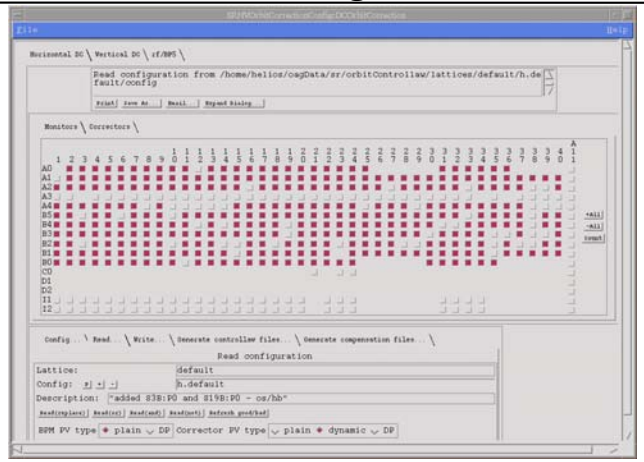
Experiment Designer



Example: Orbit Correction Suite

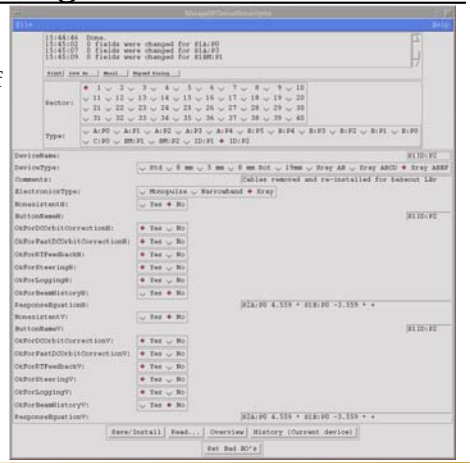
- SDDS-linked Tcl/Tk GUIs for
  - Component status tracking (e.g., "bad BPMs")
  - Correction configuration management
  - Starting and monitoring processes
- SDDS-configured processes include
  - In-IOC or workstation-based feedback
  - Permission-to-run testers
  - Feedforward for x-ray BPM correction and fault tolerance
- All data storage and preparation uses SDDS, including
  - Simulation data (response matrix)
  - Measurements (feedforward data)
  - Configurations and configuration history

Orbit Correction Configuration GUI



BPM Status Management GUI

- Database fields are stored in an SDDS file, allowing multiple use of the GUI
- Database instances also in SDDS files



Resources

- Software at http://www.aps.anl.gov/asd/oag/oagPackages.shtml
  - Source code
  - Binaries for Linux, OS-X, and Windows
  - Installation guides
  - Demo scripts
- Documentation
  - Manuals http://www.aps.anl.gov/asd/oag/oagSoftware.shtml
  - SDDS information http://www.aps.anl.gov/asd/oag/SDDSInfo.shtml

Conclusion

- SDDS is a powerful open source software system
  - Unix-inspired concept
  - Sophisticated data analysis and display
  - EPICS data collection and process control
- This software is used to
  - Automate APS accelerator operations and experiments
  - Perform data logging, analysis, and display
  - Perform feedback, feedforward, and optimization
  - Pre- and post-process simulation data
- Software is generic and highly configurable