

# Getting Started with EPICS

## *Applications / Special Topics*

### Scans

*Tim Mooney*  
3/15/2005

### **Argonne National Laboratory**



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



# The synApps SSCAN module

---

- **Where is it?**
  - <http://www.aps.anl.gov/aod/bcda/synApps/sscan.html>
- **What's in it?**
  - Code
    - the sscan record
    - the busy record
    - the recDynLink library
    - the saveData data-storage client
    - the scanparm record
  - EPICS databases
    - scan databases
    - scanParms and alignParms databases
  - MEDM displays
    - scan\*.adl
    - scan\*\_help.adl

# Simple scans

- A one-dimensional scan:

- Do NPTS times:

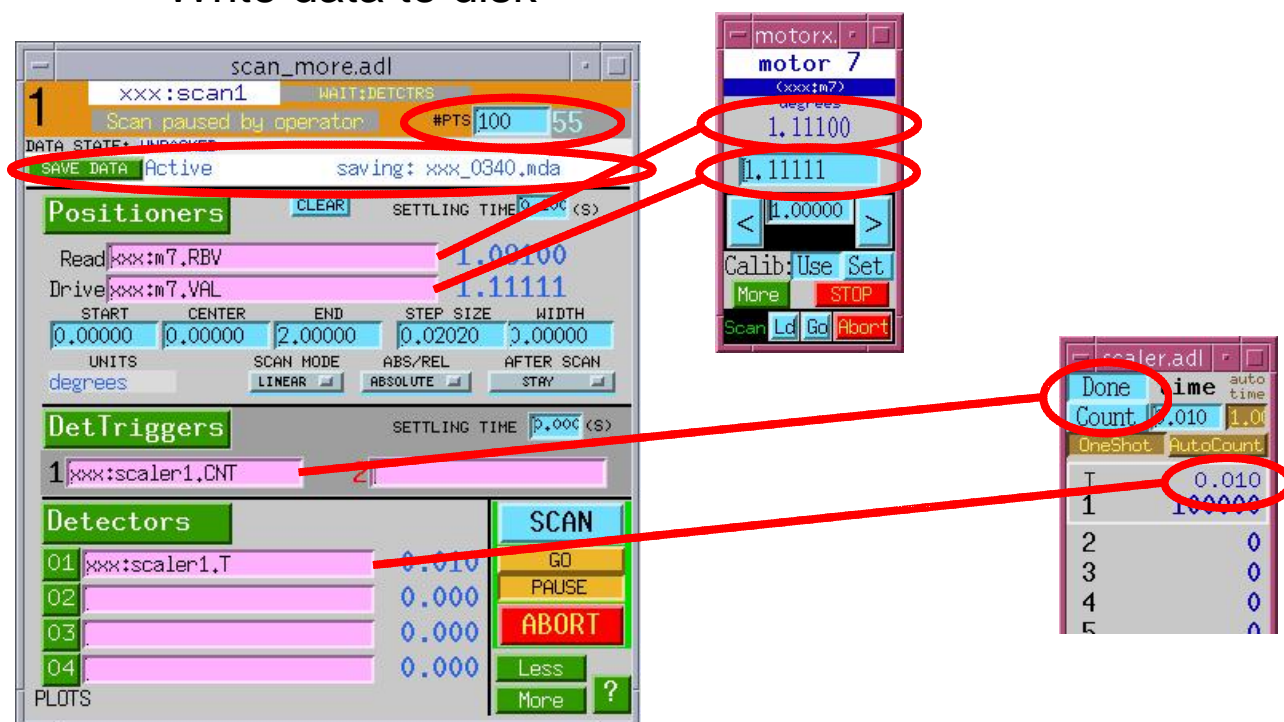
- Set conditions
    - Trigger detectors
    - Acquire data

*e.g., move motors; wait for completion*

*e.g., start scaler; wait for completion*

*read detector signals; store in arrays*

- Write data to disk



The main window shows the following configuration:

- scan\_more.adl**: #PTS 100, 55. Scan paused by operator. SAVE DATA Active. saving: xxx\_0340.mda.
- Positioners**: Read xxx:m7.RBV (1.00100), Drive xxx:m7.VAL (1.11111). START 0.00000, CENTER 0.00000, END 2.00000, STEP SIZE 0.02020, WIDTH 0.00000. UNITS: degrees. SCAN MODE: LINEAR. ABS/REL: ABSOLUTE. AFTER SCAN: STAY.
- DetTriggers**: 1 xxx:scaler1.CNT.
- Detectors**: 01 xxx:scaler1.T (0.010), 02 (0.000), 03 (0.000), 04 (0.000).


The **motorx** sub-window shows motor 7 with a position of 1.11100 and a velocity of 1.11111.

The **scaler.adl** sub-window shows a Done time of 0.010 and a Count of 1.00000.

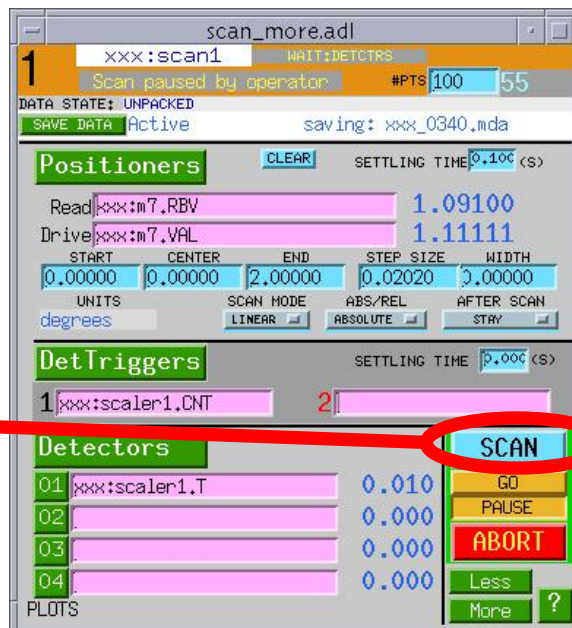
# ...Simple scans

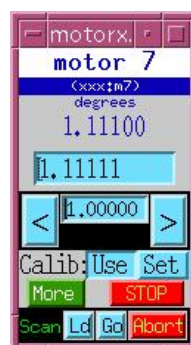
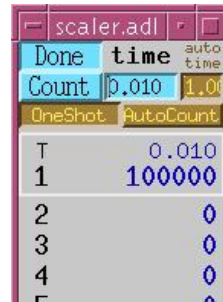
- **Multidimensional scan:**
  - Outer-loop scan's *detector trigger* executes inner-loop scan.
  - **saveData** monitors a set of **sscan** records, determines scan dimension when scan starts, and writes data as it is acquired.
  - No limit to the number of scan dimensions.

### outer-loop scan



### inner-loop scan



# Scan features

---

- **0-4 positioners, 0-4 detector triggers, 0-70 detector signals**
  - Positioner and readback values are of type `double`
  - Detector values are of type `float`
- **Acquisition from scalar and/or array PV's**
  - Array PV's acquire .NPTS elements
- **Number of data points limited only by IOC memory**
  - Standard max. is 2000  $(x_i, y_i)$  points per scan dimension
  - Can increase to  $\sim \text{EPICS\_CA\_MAX\_ARRAY\_BYTES} / 8$
- **Detector/client wait, data-storage wait**
  - Can wait for multiple data-acquisition clients
  - Only one data-storage client
- **Pause/resume, abort**
  - Data from aborted scans are written to disk
- **Double buffered: writes 1D acquired data after the scan is finished**
  - Can write during next 1D scan



## ...Scan features

---

- ***saveData* writes XDR-format (".mda") files to disk.**
  - Files can be read on any type of computer
- **A positioner can have private scan parameters (scanparm record).**
  - Load preset scan parameters with one mouse click
  - Useful for alignment
- **After-scan actions include move to peak, valley, +/-edge.**
  - Can, e.g., track a moving peak through a series of scans
- **scanparm record + after-scan action = automated 1-D alignment.**

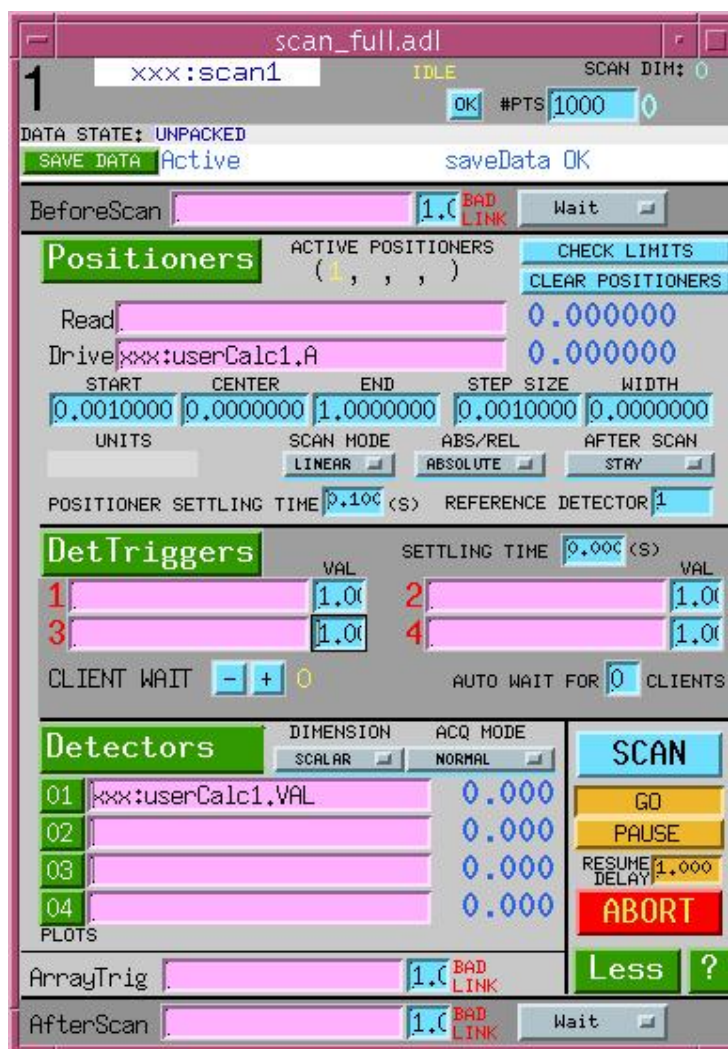
# Scan implementation

---

- **The sscan record is a channel-access client**
  - scanned PV's can be hosted by any ioc
  - uses recDynLink library to manage connections with PV's
  - uses ca\_put\_callback() to set conditions, trigger detectors, and await completion
  - uses ca\_get\_callback() before acquiring data
- **saveData is a channel-access client**
  - in principle, saveData can monitor sscan records hosted by a different ioc
  - in practice, don't do this if you can avoid it
- **Scan acquisition/storage can run on vxWorks, Linux, or Solaris.**
  - New in synApps 5.1 (EPICS 3.14)
- **The sscan record can be driven by any channel-access client.**
  - manual operation, via MEDM, is one option
  - can simplify user-written scan-control software

# Before-scan / after-scan links

- Can write a constant value to any numeric or menu PV before the scan starts and/or after the scan ends.
- Can wait or not wait for completion of processing started by the write.
- If this sscan record is part of a multidimensional scan, links function on each iteration.
- Outer-loop sscan record can write to these links, and to the values they write.
- These links cannot write to their own sscan record's START, etc. fields



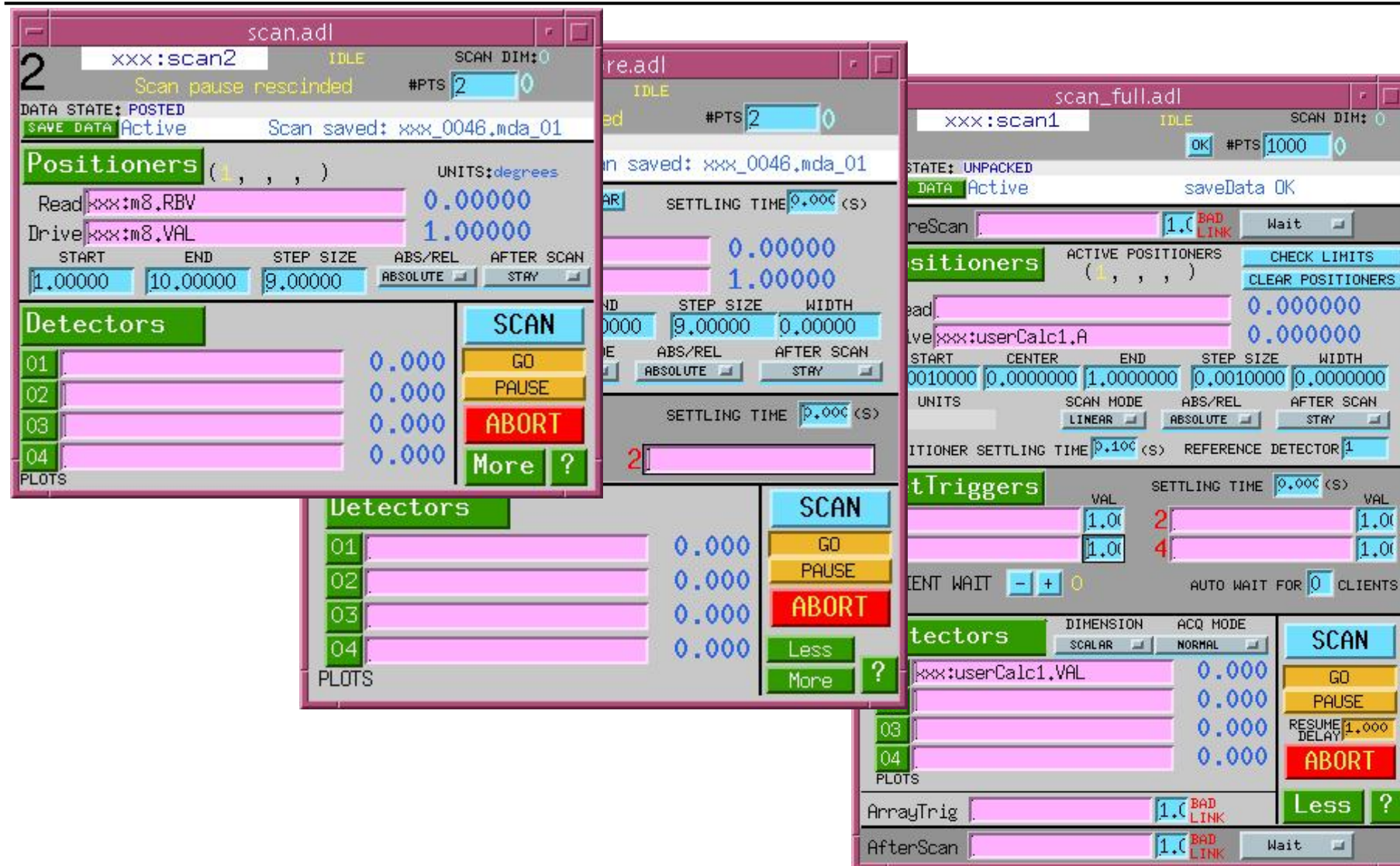
The screenshot shows the 'scan\_full.adl' window with the following sections:

- Header:** Title 'xxx:scan1', IDLE status, #PTS 1000, and a 'Wait' button.
- Positioners:** Includes 'ACTIVE POSITIONERS' (1, , ,), 'CHECK LIMITS', 'CLEAR POSITIONERS', and fields for 'Read' (0.000000) and 'Drive' (xxx:userCalc1.A, 0.000000). It also has a table for scan parameters: START (0.0010000), CENTER (0.0000000), END (1.0000000), STEP SIZE (0.0010000), and WIDTH (0.0000000). Below this are 'UNITS', 'SCAN MODE' (LINEAR), 'ABS/REL' (ABSOLUTE), and 'AFTER SCAN' (STAY).
- DetTriggers:** Includes 'POSITIONER SETTLING TIME' (0.100 <S>) and 'REFERENCE DETECTOR' (1). It has four rows for triggers with 'VAL' and 'SETTLING TIME' (0.000 <S>) fields.
- Detectors:** Includes 'DIMENSION' (SCALAR), 'ACQ MODE' (NORMAL), and a 'SCAN' button. It has four rows for detectors with 'VAL' fields (0.000).
- Bottom Section:** Includes 'CLIENT WAIT' (- + 0), 'AUTO WAIT FOR' (0 CLIENTS), 'PLOTS', 'ArrayTrig' (1.0 BAD LINK), and 'AfterScan' (1.0 BAD LINK) with a 'Wait' button.

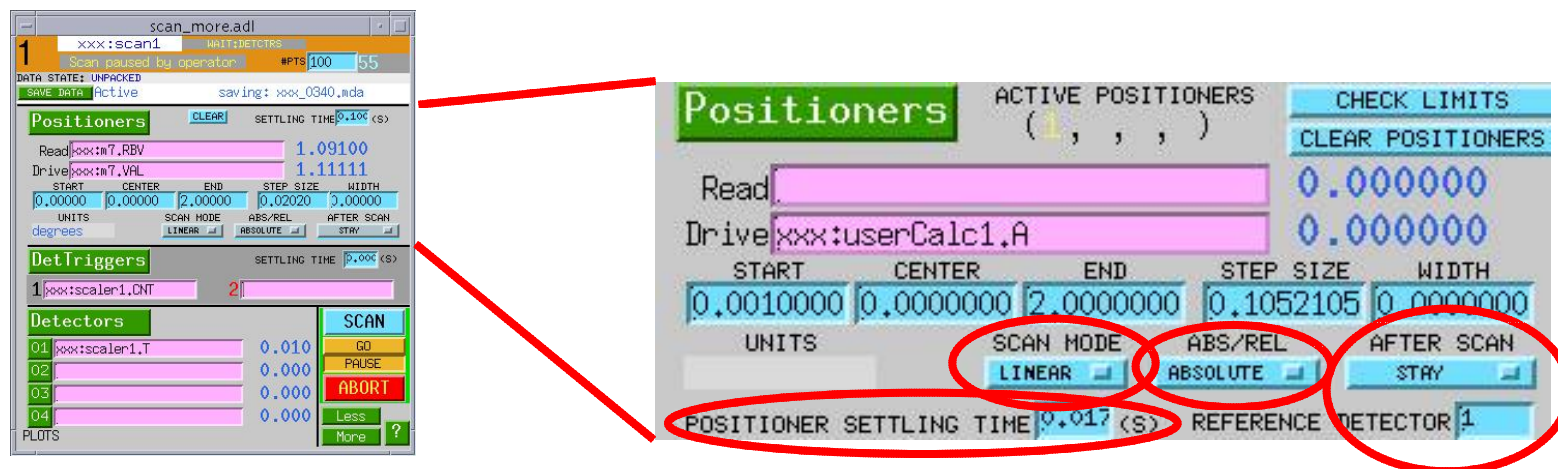
Blue arrows point from the text in the list to the 'BeforeScan' and 'AfterScan' fields in the interface.



# MEDM user interface



# Positioner options



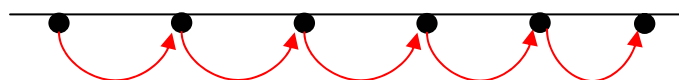
- **SCAN MODE (.PnSM - per positioner)**
  - Determines how and to where positioner moves
- **Absolute/Relative (.PnAR - per positioner)**
  - Determines how positioner locations are written
- **Positioner delay (.PDLY - affects all positioners)**
  - Delay while positioners are settling, after completing their moves
- **After-scan motion (.PASM - affects all positioners)**
  - Determines what, if anything, is done with positioners when scan is finished

# ...Positioner options

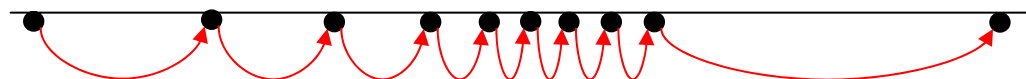
- **SCAN MODE (.PnSM - per positioner)**
  - **LINEAR** – Evenly spaced positions are calculated algorithmically
    - You specify positioner locations by setting any three of

<i>START</i>	<i>CENTER</i>	<i>END</i>	<i>WIDTH</i>	<i>STEP SIZE</i>	<i># POINTS</i>
<b>.PnSP</b>	<b>.PnCP</b>	<b>.PnEP</b>	<b>.PnWD</b>	<b>.PnSI</b>	<b>.NPTS</b>

- The sscan record reconciles unset parameters

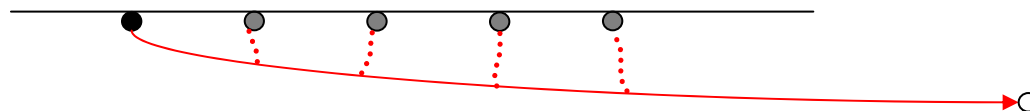


- **TABLE** – Positioner locations are contained in the **.PnPA** array
  - The array must contain at least **.NPTS** values
  - You must arrange for the array to contain the desired positions before starting the scan.
  - The **.PnPA** array is never overwritten by the sscan record



# ...Positioner options

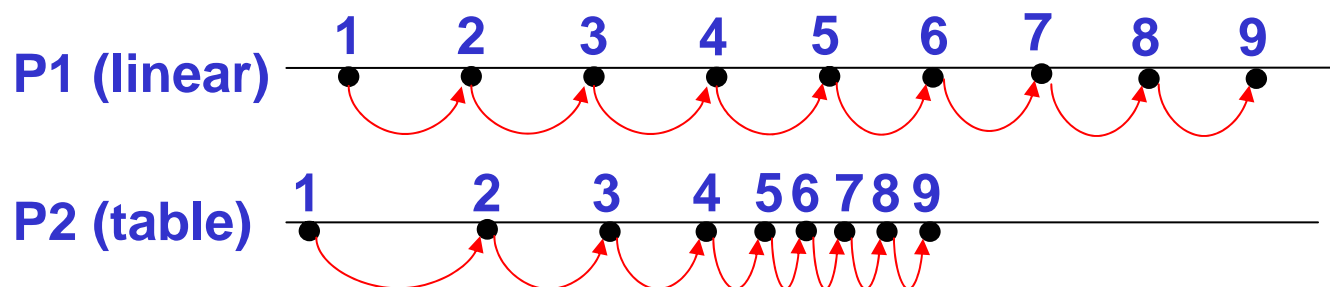
- **...SCAN MODE (.PnSM - per positioner)**
  - **FLY** – data will be acquired *while* positioner moves
    - You specify positions at which data are acquired by setting *START*, *END*, positioner speed, and detector acquisition time.
    - The following algorithm is executed:
      - Positioner sent to *START*; reports completion
      - Detector triggered; reports completion
      - First data point acquired
      - Positioner sent to *END*
      - *NPTS*-1 iterations of
        - Detector triggered; reports completion
        - Data point acquired
    - The timing of data points is controlled by the detector's acquisition time.
    - Fly-mode positioners do not report completion. (The positioner may still be moving after the scan ends.)



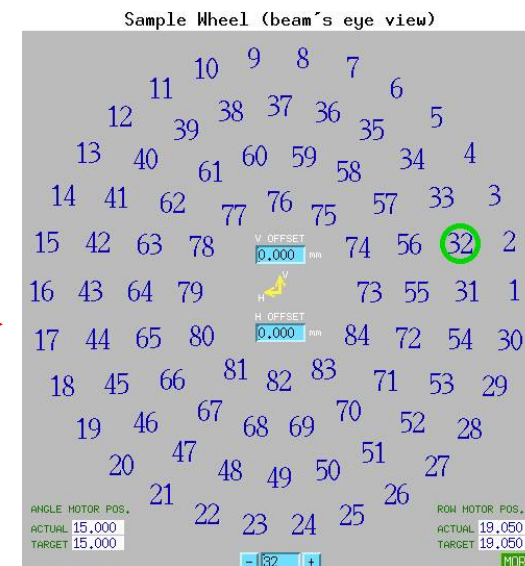
# ...Positioner options

- ...SCAN MODE (.PnSM - per positioner)

- OK to mix scan modes:



- Don't be limited by existing positioner modes
  - A positioner is *anything* you can write to
  - Can specify positions algorithmically, using calcout or transform
    - E.g., sample-wheel
  - Can write to positioner through interpolation table
    - Use a spare positioner readback to get actual positions into the data file





## ...Positioner options

---

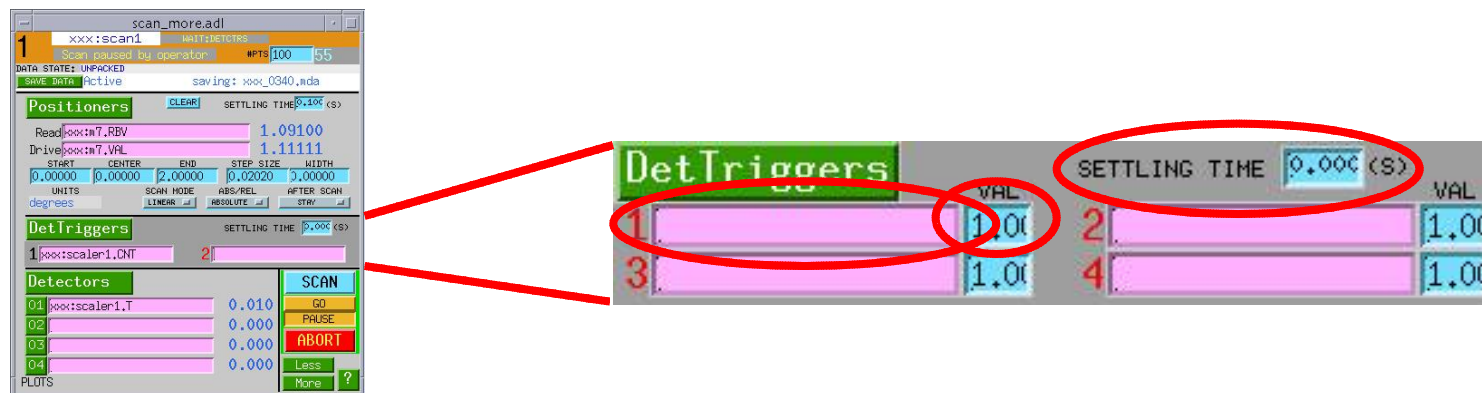
- **Absolute/Relative (.PnAR - per positioner)**
  - If **.PnAR** == “ABSOLUTE” (0), positions are sent exactly as given.
  - If **.PnAR** == “RELATIVE” (1), positions are added to pre-scan position before being sent to positioner.
  
- **Settling time (.PDLY - affects all positioners)**
  - If any positioner PV is specified, then after all positioners report completion, the sscan record waits for **.PDLY** seconds before moving to next phase of sscan.
    - Useful for positioners that “ring” after move is completed
    - Useful work-around for positioners that cannot report completion
  - If no positioners, then settling time is ignored.
  - Settling time is adjusted to nearest multiple of system-clock period (typically 1/60Hz).

# ...Positioner options

---

- **After-scan motion (.PASM - affects all positioners)**
  - STAY – positioners are simply left where they ended up
  - START POS – positioners are sent to their *START* positions
  - PRIOR POS – positioners are sent to their pre-scan positions
  - PEAK POS – data from the reference detector (number given by the **.REFD** field, in range [1..70]) is examined. If a peak is found, positioners are sent to where it was acquired.
  - VALLEY POS – similar, but valley instead of peak
  - +EDGE POS – peak of derivative of reference data
  - -EDGE POS – valley of derivative of reference data

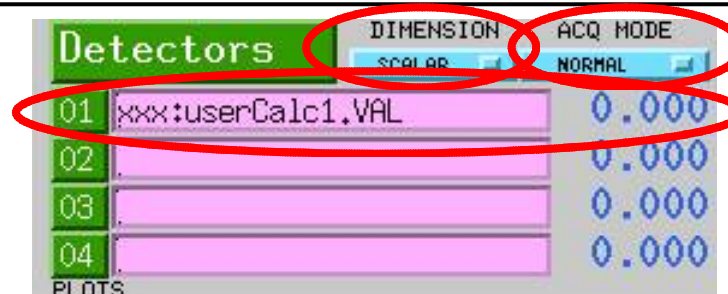
# Detector triggers



- 0-4 detector triggers (.TnPV), intended to start data-acquisition
- Similar to positioners, but value sent (.TnCD) is constant
- Triggers execute after all positioners have completed, and after any positioner settling time has elapsed.
- Detector settling time begins after all detector triggers have reported completion.
- If no triggers, then settling time is ignored.

# Detectors

- PV's to be acquired during scan
- 0-70 detectors (.D01PV - .D70PV)
- Detector options
  - Acquisition type (**.ACQT**)
    - SCALAR
      - scalar PV's acquired at each positioner location
      - Array PV's (**.NPTS** elements) acquired at end of scan
    - 1D ARRAY
      - use this mode only if ALL detectors are array valued
      - Positioners are only sent to their START positions.
      - In the future, array-valued positioners may be supported.
  - Acquisition mode (**.ACQM**)
    - NORMAL – store values as acquired
    - ACCUMULATE – add detector values, starting with next scan
    - ADD TO PREV – same, but starting with previous scan



# Scan controls

- **SCAN**
  - Writing '1' starts this sscan record
  - Writing '0' stops this sscan record. (But with the supplied database, always use the 'ABORT' button to stop.)
- **GO/PAUSE**
  - Pause is immediate, Go occurs after delay
- **ABORT**
  - Writes '1' to 'xxx:allstop.VAL', which should stop motors
  - Sends "stop" message to *all* sscan records in the supplied database
    - First 'Abort' attempt ends scan after outstanding completion callbacks have come in, and data-storage client has released the previous scan's data arrays.
    - Second 'Abort' attempt waits only for data-storage client.
    - Third successive 'Abort' attempt kills scan with no regard for consequences.





finish writing last scan's data.

scan\_detector\_help.pdf

Detector triggers are like positioners, but they are intended to start data acquisition, and the values written to them do not vary during a scan. Any writable PV can be named as a detector trigger, but if the scan is to wait for the trigger's action to complete, then either the PV must be implemented so that it's done 'callback' really does signify completion of that action, or some other wait mechanism must be used.

One possibility is the detector-settling law. If no better mechanism is handy, you can just make the settling time long enough to ensure that acquisition is done before the scan reads the data.

A better mechanism is for the data-acquisition client to declare completion by writing '0' to the scan's WAIT field. (The CLIENT-WAIT button labelled '+' does exactly this.) The WAIT mechanism is a two-part handshake, and the client normally asserts that it's busy by writing '1' into the '+' button (does this) and then asserts 'done' by writing '0'.

If the client can assert 'done', but can't assert 'busy' (or might not do it quickly enough), the scan can assert 'busy' on the client's behalf to arrange this, simply set the AUTO-WAIT count to the number of clients that will assert 'done' but will not assert 'busy'. Just make sure the scan and the client don't BOTH assert 'busy' for the same operation.

Detectors are signals read after all triggers and data-acquisition clients have completed. Any readable, numeric PV can be named as a detector. If an array-valued PV is named, NPTS elements will be acquired. (If all PV's are array valued, you can set the acquisition type ('DIMENSION') to '1D ARRAY'.)

POLOTS

Let's try

1g CPV TO WRITE TO: VAL TO BAD WRITE LINK

2g CPV TO WRITE TO: VAL TO BAD WRITE LINK

3g CPV TO WRITE TO: VAL TO BAD WRITE LINK

4g CPV TO WRITE TO: VAL TO BAD WRITE LINK

5g CPV TO WRITE TO: VAL TO BAD WRITE LINK

6g CPV TO WRITE TO: VAL TO BAD WRITE LINK

7g CPV TO WRITE TO: VAL TO BAD WRITE LINK

8g CPV TO WRITE TO: VAL TO BAD WRITE LINK

9g CPV TO WRITE TO: VAL TO BAD WRITE LINK

10g CPV TO WRITE TO: VAL TO BAD WRITE LINK

11g CPV TO WRITE TO: VAL TO BAD WRITE LINK

12g CPV TO WRITE TO: VAL TO BAD WRITE LINK

13g CPV TO WRITE TO: VAL TO BAD WRITE LINK

14g CPV TO WRITE TO: VAL TO BAD WRITE LINK

15g CPV TO WRITE TO: VAL TO BAD WRITE LINK

16g CPV TO WRITE TO: VAL TO BAD WRITE LINK

17g CPV TO WRITE TO: VAL TO BAD WRITE LINK

18g CPV TO WRITE TO: VAL TO BAD WRITE LINK

19g CPV TO WRITE TO: VAL TO BAD WRITE LINK

20g CPV TO WRITE TO: VAL TO BAD WRITE LINK

21g CPV TO WRITE TO: VAL TO BAD WRITE LINK

22g CPV TO WRITE TO: VAL TO BAD WRITE LINK

23g CPV TO WRITE TO: VAL TO BAD WRITE LINK

24g CPV TO WRITE TO: VAL TO BAD WRITE LINK

25g CPV TO WRITE TO: VAL TO BAD WRITE LINK

26g CPV TO WRITE TO: VAL TO BAD WRITE LINK

27g CPV TO WRITE TO: VAL TO BAD WRITE LINK

28g CPV TO WRITE TO: VAL TO BAD WRITE LINK

29g CPV TO WRITE TO: VAL TO BAD WRITE LINK

30g CPV TO WRITE TO: VAL TO BAD WRITE LINK

31g CPV TO WRITE TO: VAL TO BAD WRITE LINK

32g CPV TO WRITE TO: VAL TO BAD WRITE LINK

33g CPV TO WRITE TO: VAL TO BAD WRITE LINK

34g CPV TO WRITE TO: VAL TO BAD WRITE LINK

35g CPV TO WRITE TO: VAL TO BAD WRITE LINK

36g CPV TO WRITE TO: VAL TO BAD WRITE LINK

37g CPV TO WRITE TO: VAL TO BAD WRITE LINK

38g CPV TO WRITE TO: VAL TO BAD WRITE LINK

39g CPV TO WRITE TO: VAL TO BAD WRITE LINK

40g CPV TO WRITE TO: VAL TO BAD WRITE LINK

41g CPV TO WRITE TO: VAL TO BAD WRITE LINK

42g CPV TO WRITE TO: VAL TO BAD WRITE LINK

43g CPV TO WRITE TO: VAL TO BAD WRITE LINK

44g CPV TO WRITE TO: VAL TO BAD WRITE LINK

45g CPV TO WRITE TO: VAL TO BAD WRITE LINK

46g CPV TO WRITE TO: VAL TO BAD WRITE LINK

47g CPV TO WRITE TO: VAL TO BAD WRITE LINK

48g CPV TO WRITE TO: VAL TO BAD WRITE LINK

49g CPV TO WRITE TO: VAL TO BAD WRITE LINK

50g CPV TO WRITE TO: VAL TO BAD WRITE LINK

51g CPV TO WRITE TO: VAL TO BAD WRITE LINK

52g CPV TO WRITE TO: VAL TO BAD WRITE LINK

53g CPV TO WRITE TO: VAL TO BAD WRITE LINK

54g CPV TO WRITE TO: VAL TO BAD WRITE LINK

55g CPV TO WRITE TO: VAL TO BAD WRITE LINK

56g CPV TO WRITE TO: VAL TO BAD WRITE LINK

57g CPV TO WRITE TO: VAL TO BAD WRITE LINK

58g CPV TO WRITE TO: VAL TO BAD WRITE LINK

59g CPV TO WRITE TO: VAL TO BAD WRITE LINK

60g CPV TO WRITE TO: VAL TO BAD WRITE LINK

61g CPV TO WRITE TO: VAL TO BAD WRITE LINK

62g CPV TO WRITE TO: VAL TO BAD WRITE LINK

63g CPV TO WRITE TO: VAL TO BAD WRITE LINK

64g CPV TO WRITE TO: VAL TO BAD WRITE LINK

65g CPV TO WRITE TO: VAL TO BAD WRITE LINK

66g CPV TO WRITE TO: VAL TO BAD WRITE LINK

67g CPV TO WRITE TO: VAL TO BAD WRITE LINK

68g CPV TO WRITE TO: VAL TO BAD WRITE LINK

69g CPV TO WRITE TO: VAL TO BAD WRITE LINK

70g CPV TO WRITE TO: VAL TO BAD WRITE LINK

71g CPV TO WRITE TO: VAL TO BAD WRITE LINK

72g CPV TO WRITE TO: VAL TO BAD WRITE LINK

73g CPV TO WRITE TO: VAL TO BAD WRITE LINK

74g CPV TO WRITE TO: VAL TO BAD WRITE LINK

75g CPV TO WRITE TO: VAL TO BAD WRITE LINK

76g CPV TO WRITE TO: VAL TO BAD WRITE LINK

77g CPV TO WRITE TO: VAL TO BAD WRITE LINK

78g CPV TO WRITE TO: VAL TO BAD WRITE LINK

79g CPV TO WRITE TO: VAL TO BAD WRITE LINK

80g CPV TO WRITE TO: VAL TO BAD WRITE LINK

81g CPV TO WRITE TO: VAL TO BAD WRITE LINK

82g CPV TO WRITE TO: VAL TO BAD WRITE LINK

83g CPV TO WRITE TO: VAL TO BAD WRITE LINK

84g CPV TO WRITE TO: VAL TO BAD WRITE LINK

85g CPV TO WRITE TO: VAL TO BAD WRITE LINK

86g CPV TO WRITE TO: VAL TO BAD WRITE LINK

87g CPV TO WRITE TO: VAL TO BAD WRITE LINK

88g CPV TO WRITE TO: VAL TO BAD WRITE LINK

89g CPV TO WRITE TO: VAL TO BAD WRITE LINK

90g CPV TO WRITE TO: VAL TO BAD WRITE LINK

91g CPV TO WRITE TO: VAL TO BAD WRITE LINK

92g CPV TO WRITE TO: VAL TO BAD WRITE LINK

93g CPV TO WRITE TO: VAL TO BAD WRITE LINK

94g CPV TO WRITE TO: VAL TO BAD WRITE LINK

95g CPV TO WRITE TO: VAL TO BAD WRITE LINK

96g CPV TO WRITE TO: VAL TO BAD WRITE LINK

97g CPV TO WRITE TO: VAL TO BAD WRITE LINK

98g CPV TO WRITE TO: VAL TO BAD WRITE LINK

99g CPV TO WRITE TO: VAL TO BAD WRITE LINK

100g CPV TO WRITE TO: VAL TO BAD WRITE LINK

101g CPV TO WRITE TO: VAL TO BAD WRITE LINK

102g CPV TO WRITE TO: VAL TO BAD WRITE LINK

103g CPV TO WRITE TO: VAL TO BAD WRITE LINK

104g CPV TO WRITE TO: VAL TO BAD WRITE LINK

105g CPV TO WRITE TO: VAL TO BAD WRITE LINK

106g CPV TO WRITE TO: VAL TO BAD WRITE LINK

107g CPV TO WRITE TO: VAL TO BAD WRITE LINK

108g CPV TO WRITE TO: VAL TO BAD WRITE LINK

109g CPV TO WRITE TO: VAL TO BAD WRITE LINK

110g CPV TO WRITE TO: VAL TO BAD WRITE LINK

111g CPV TO WRITE TO: VAL TO BAD WRITE LINK

112g CPV TO WRITE TO: VAL TO BAD WRITE LINK

113g CPV TO WRITE TO: VAL TO BAD WRITE LINK

114g CPV TO WRITE TO: VAL TO BAD WRITE LINK

115g CPV TO WRITE TO: VAL TO BAD WRITE LINK

116g CPV TO WRITE TO: VAL TO BAD WRITE LINK

117g CPV TO WRITE TO: VAL TO BAD WRITE LINK

118g CPV TO WRITE TO: VAL TO BAD WRITE LINK

119g CPV TO WRITE TO: VAL TO BAD WRITE LINK

120g CPV TO WRITE TO: VAL TO BAD WRITE LINK

121g CPV TO WRITE TO: VAL TO BAD WRITE LINK

122g CPV TO WRITE TO: VAL TO BAD WRITE LINK

123g CPV TO WRITE TO: VAL TO BAD WRITE LINK

124g CPV TO WRITE TO: VAL TO BAD WRITE LINK

125g CPV TO WRITE TO: VAL TO BAD WRITE LINK

126g CPV TO WRITE TO: VAL TO BAD WRITE LINK

127g CPV TO WRITE TO: VAL TO BAD WRITE LINK

128g CPV TO WRITE TO: VAL TO BAD WRITE LINK

129g CPV TO WRITE TO: VAL TO BAD WRITE LINK

130g CPV TO WRITE TO: VAL TO BAD WRITE LINK

131g CPV TO WRITE TO: VAL TO BAD WRITE LINK

132g CPV TO WRITE TO: VAL TO BAD WRITE LINK

133g CPV TO WRITE TO: VAL TO BAD WRITE LINK

134g CPV TO WRITE TO: VAL TO BAD WRITE LINK

135g CPV TO WRITE TO: VAL TO BAD WRITE LINK

136g CPV TO WRITE TO: VAL TO BAD WRITE LINK

137g CPV TO WRITE TO: VAL TO BAD WRITE LINK

138g CPV TO WRITE TO: VAL TO BAD WRITE LINK

139g CPV TO WRITE TO: VAL TO BAD WRITE LINK

140g CPV TO WRITE TO: VAL TO BAD WRITE LINK

141g CPV TO WRITE TO: VAL TO BAD WRITE LINK

142g CPV TO WRITE TO: VAL TO BAD WRITE LINK

143g CPV TO WRITE TO: VAL TO BAD WRITE LINK

144g CPV TO WRITE TO: VAL TO BAD WRITE LINK

145g CPV TO WRITE TO: VAL TO BAD WRITE LINK

146g CPV TO WRITE TO: VAL TO BAD WRITE LINK

147g CPV TO WRITE TO: VAL TO BAD WRITE LINK

148g CPV TO WRITE TO: VAL TO BAD WRITE LINK

149g CPV TO WRITE TO: VAL TO BAD WRITE LINK

150g CPV TO WRITE TO: VAL TO BAD WRITE LINK

151g CPV TO WRITE TO: VAL TO BAD WRITE LINK

152g CPV TO WRITE TO: VAL TO BAD WRITE LINK

153g CPV TO WRITE TO: VAL TO BAD WRITE LINK

154g CPV TO WRITE TO: VAL TO BAD WRITE LINK

155g CPV TO WRITE TO: VAL TO BAD WRITE LINK

156g CPV TO WRITE TO: VAL TO BAD WRITE LINK

157g CPV TO WRITE TO: VAL TO BAD WRITE LINK

158g CPV TO WRITE TO: VAL TO BAD WRITE LINK

159g CPV TO WRITE TO: VAL TO BAD WRITE LINK

160g CPV TO WRITE TO: VAL TO BAD WRITE LINK

161g CPV TO WRITE TO: VAL TO BAD WRITE LINK

162g CPV TO WRITE TO: VAL TO BAD WRITE LINK

163g CPV TO WRITE TO: VAL TO BAD WRITE LINK

164g CPV TO WRITE TO: VAL TO BAD WRITE LINK

165g CPV TO WRITE TO: VAL TO BAD WRITE LINK

166g CPV TO WRITE TO: VAL TO BAD WRITE LINK

167g CPV TO WRITE TO: VAL TO BAD WRITE LINK

168g CPV TO WRITE TO: VAL TO BAD WRITE LINK

169g CPV TO WRITE TO: VAL TO BAD WRITE LINK

170g CPV TO WRITE TO: VAL TO BAD WRITE LINK

171g CPV TO WRITE TO: VAL TO BAD WRITE LINK

172g CPV TO WRITE TO: VAL TO BAD WRITE LINK

173g CPV TO WRITE TO: VAL TO BAD WRITE LINK

174g CPV TO WRITE TO: VAL TO BAD WRITE LINK

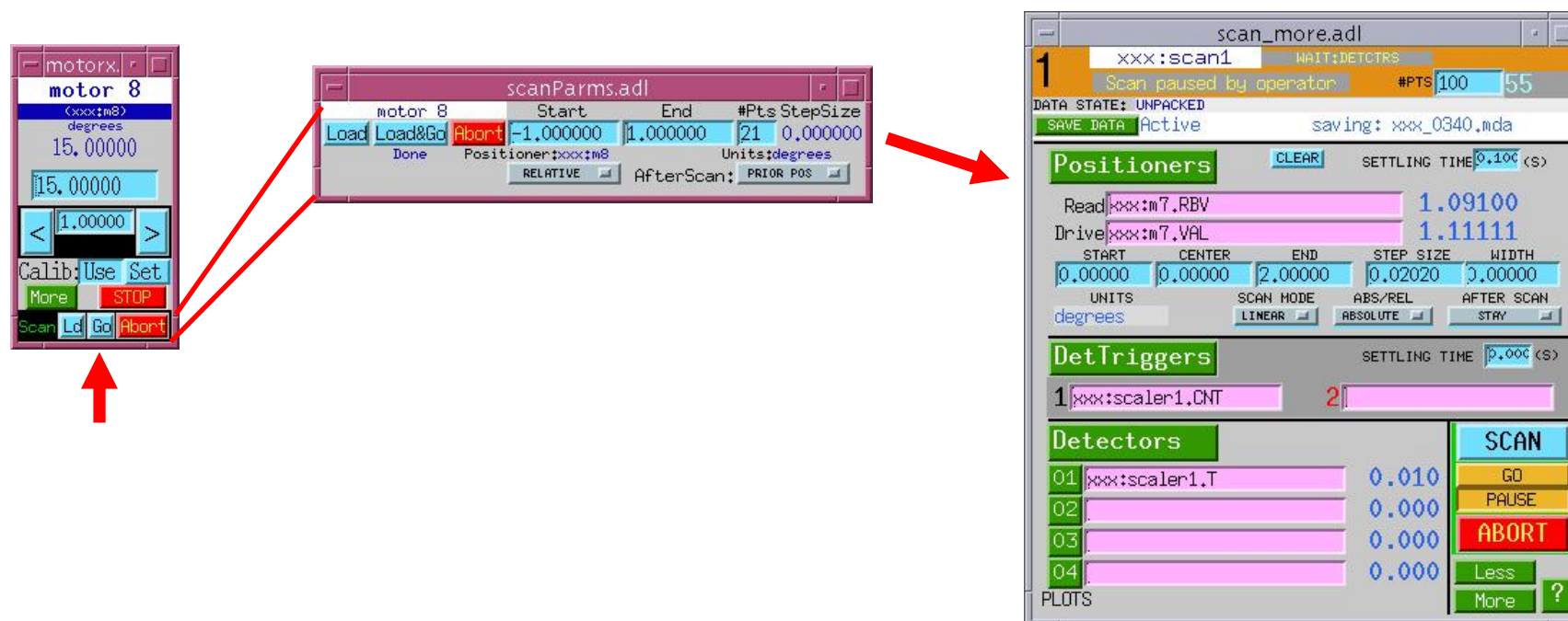
175g CPV TO WRITE TO: VAL TO BAD WRITE LINK

176g CPV TO WRITE TO: VAL TO BAD WRITE LINK

177g CPV TO WRITE TO: VAL TO BAD WRITE LINK</

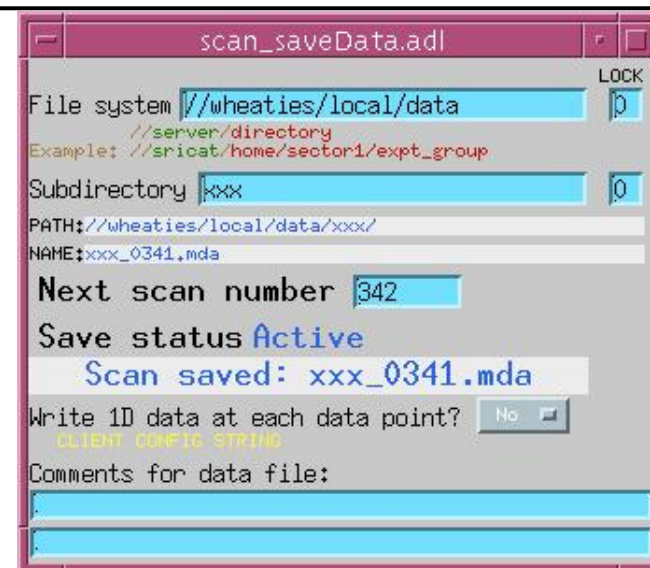
# One-click scans

- The scanparm record executes preprogrammed *linear* scans
  - Holds scan parameters for a positioner
  - Writes parameters to a particular **sscan** record
  - Optionally executes the **sscan** record
  - Useful for alignment



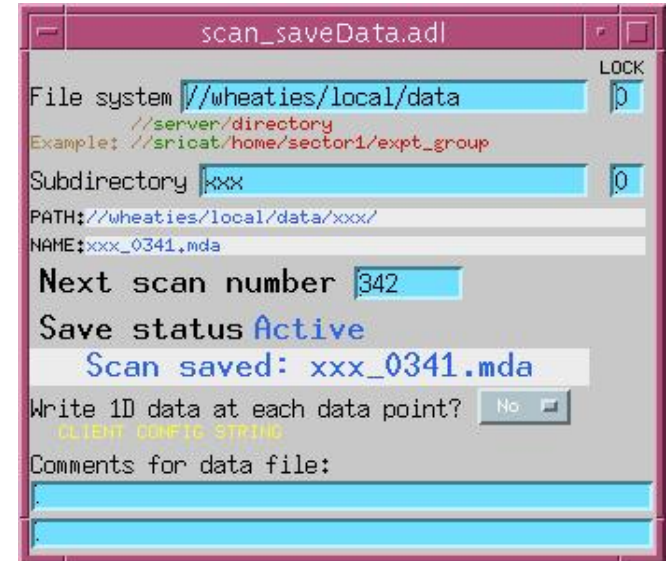
# Data storage

- saveData monitors sscan records and writes their data to numbered files.
- Handshake permits pipelined operation.
- saveData's boot-time init can specify list of PV's to write with every scan's data
- saveData writes "MDA" files
  - MultiDimensional Archive
  - Binary, cross-platform (XDR) format
  - Format is optimized for run-time access.
  - Format permits file to be closed after each set of writes.
- Automatic file numbering
  - e.g., 'xxx\_0123.mda', 'xxx\_0124.mda'
  - overlap is handled: 'xxx\_0123.mda\_01'



# ...Data storage

- **Location of data files**
  - 'File system' + 'subdirectory'
  - vxWorks:
    - File system is NFS-mount point
    - '//<hostname>' is required
  - Linux, Solaris:
    - saveData doesn't mount the file system (system administrator does this)
    - '//<hostname>', if present, is ignored
- **Cannot write to 'File system' or 'subdirectory' while a scan is in progress. (See 'LOCK' PV.)**
- **Don't delete or rename the directory saveData is writing to.**
- **Comment PV's saved only if they are named in saveData.req**



scan\_saveData.adl

File system  LOCK ☐

Example: //server/directory

Subdirectory  ☐

PATH: //wheaties/local/data/xxx/

NAME: xxx\_0341.mda

Next scan number

Save status **Active**

Scan saved: xxx\_0341.mda

Write 1D data at each data point?

CLIENT CONFIG STRING

Comments for data file:



# saveData.req init file

```
[prefix] ←
$(P)

[status]
$(P)saveData_status
...

[scanRecord] ←
$(P)scanH
$(P)scan1
$(P)scan2
$(P)scan3
$(P)scan4

[extraPV] ←
#<PV name> <description>
$(P)scaler1.TP "scaler preset (s)"
$(P)scaler1.NM1 "scaler chan 1 desc"
...
```

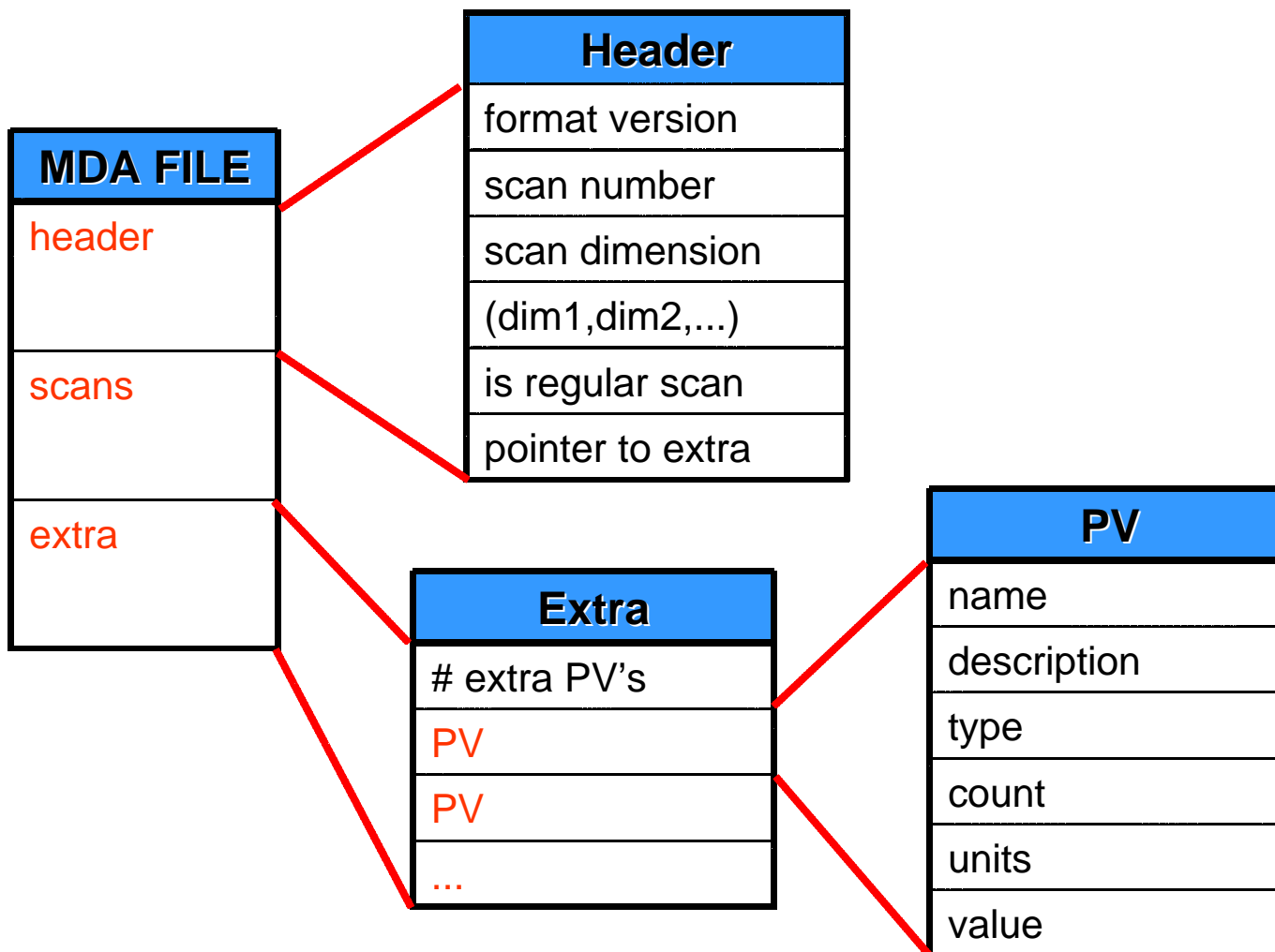
Section head

List of sscan records to monitor:

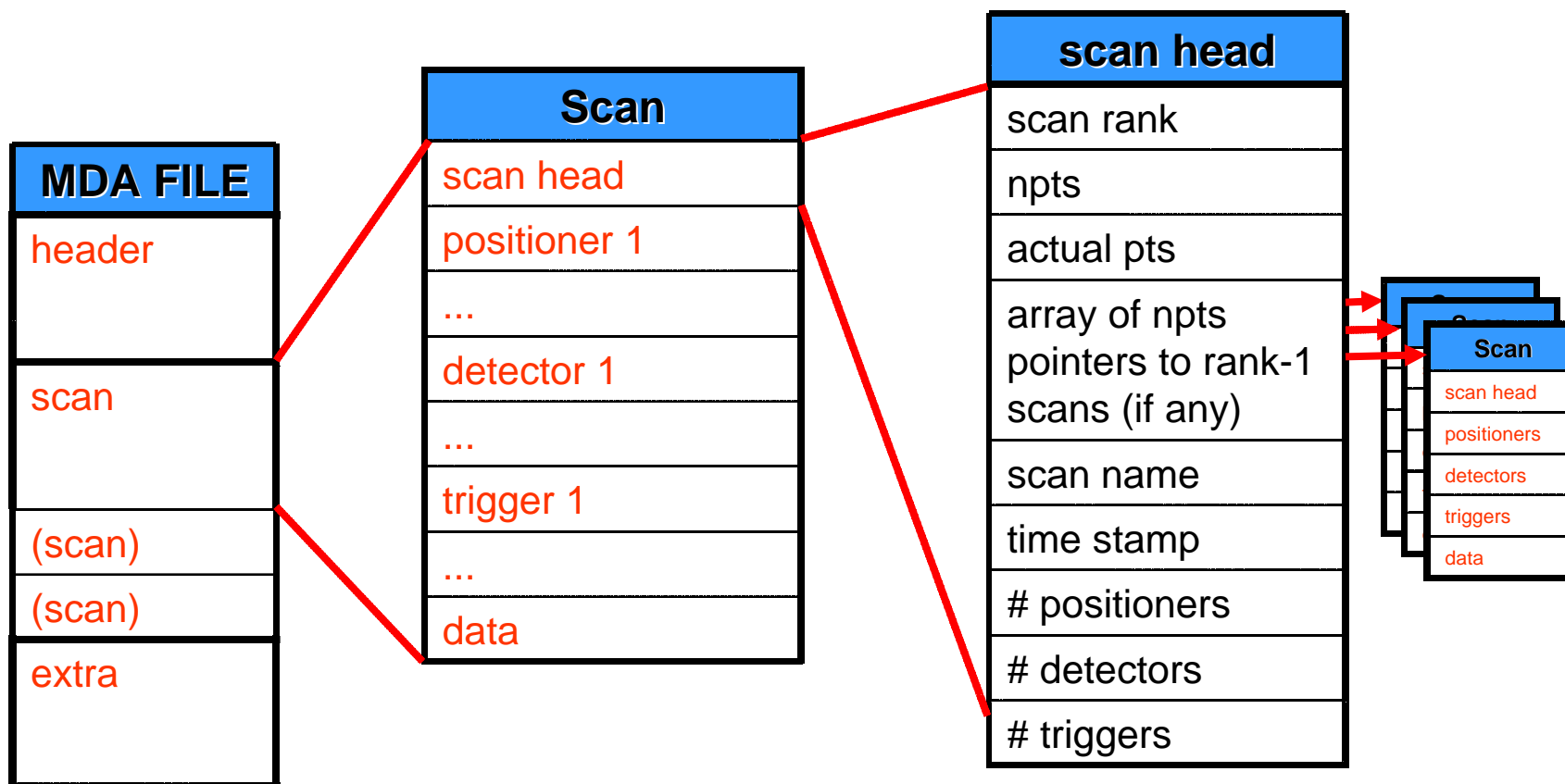
List of PV's to be saved with every scan (Normally, this is the only section you modify.)



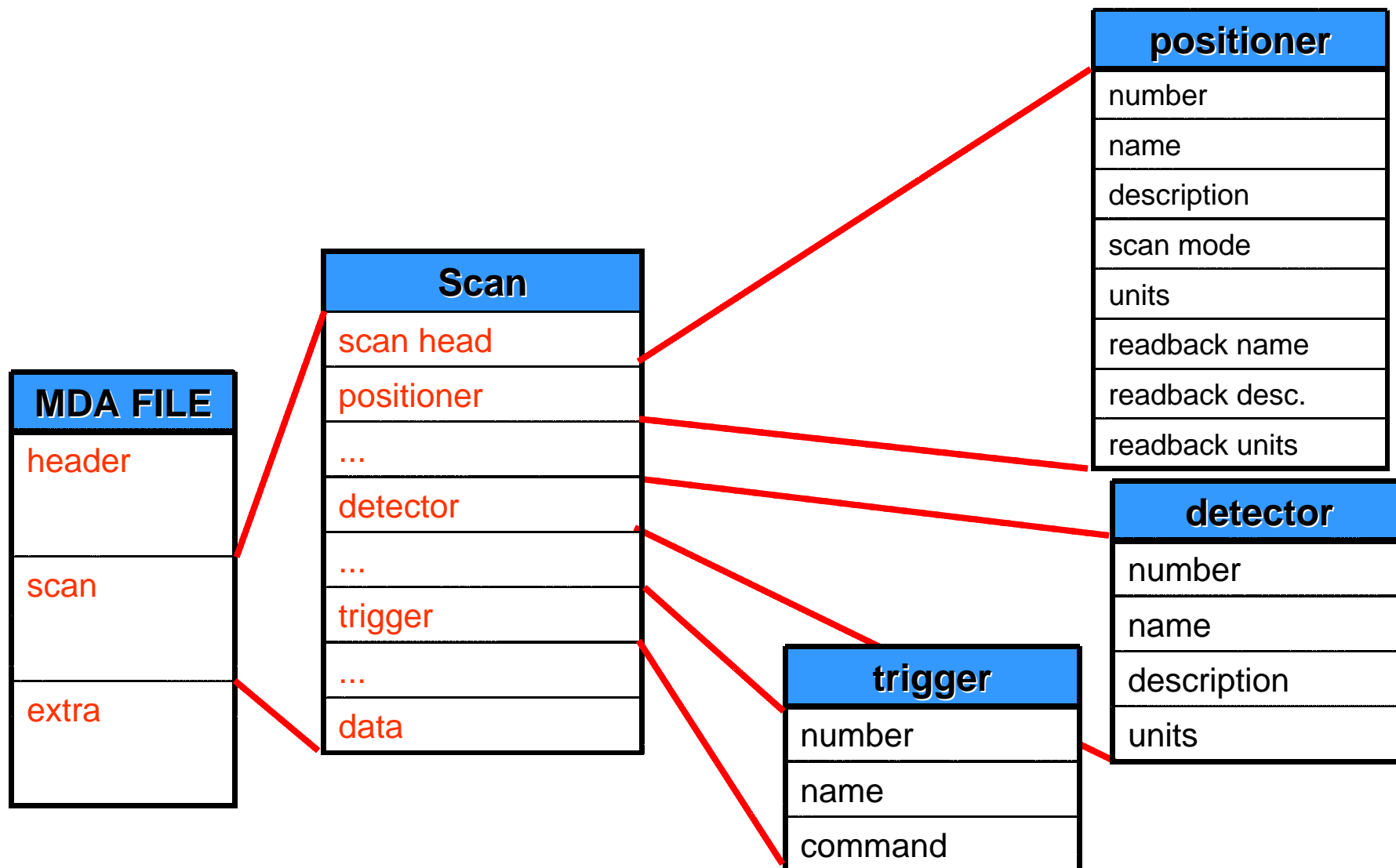
# MDA file format



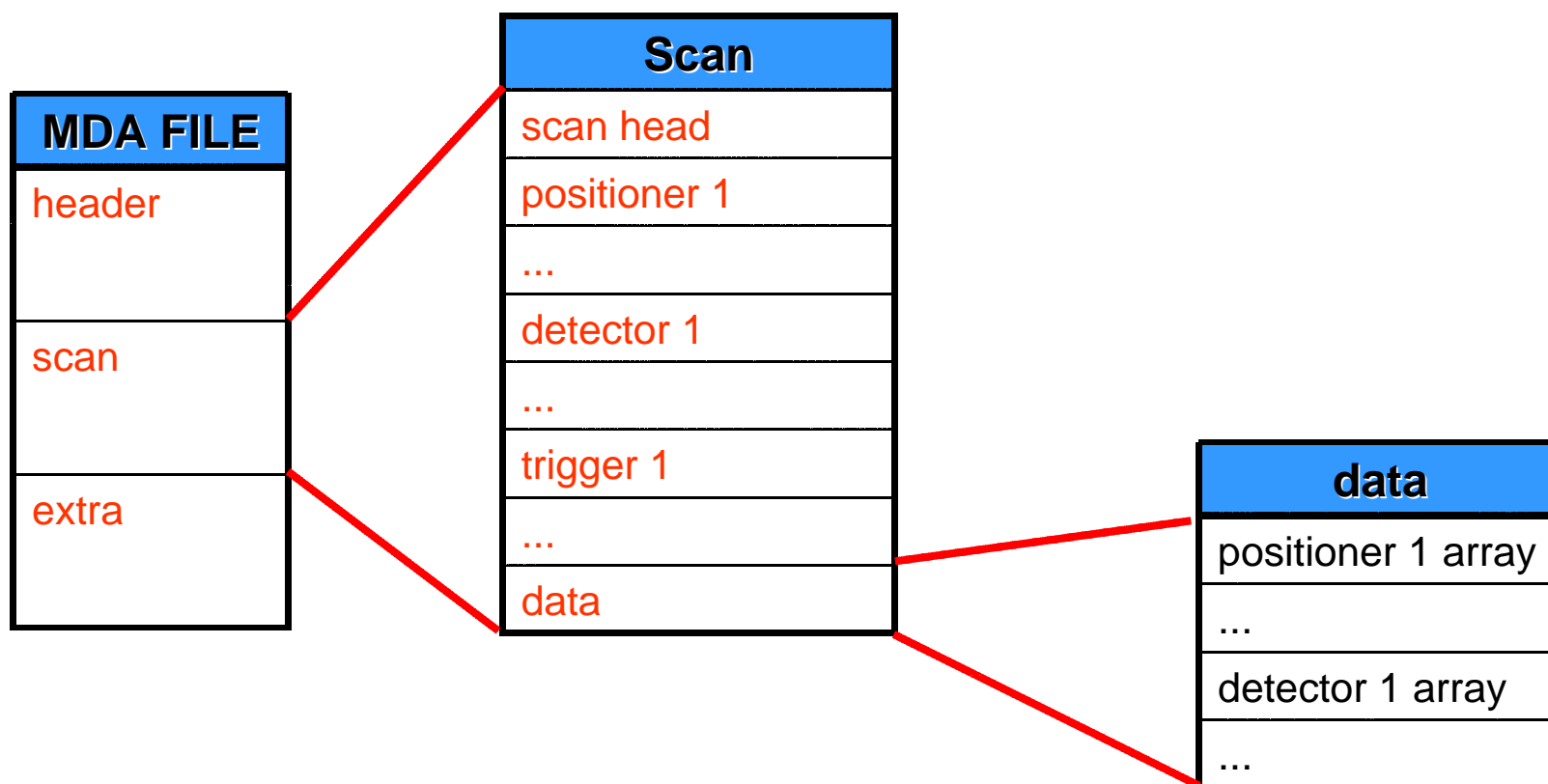
# ...MDA file format



# ...MDA file format



# ...MDA file format



# Other data-acquisition-related software

---

- **Data-visualization tools for use with synApps**
  - Run-time look at scan data
  - Offline tools for data-file manipulation
  - Supports 1-3 dimensional data
  - Distributed independently of ioc software
  - See lecture “*Data Visualization.*”
  
- **CCD data-acquisition tools**
  - 1) CCD module (see lecture “*Detectors and Feedback*”)
  - 2) Portable CA Server based CCD support, and related software
    - <http://www.aps.anl.gov/aod/bcda/dataAcq/index.php>
  - Both of these solutions allow an EPICS CA client to drive data acquisition.
  - Both support `ca_put_callback()`, as needed by the **sscan** record.



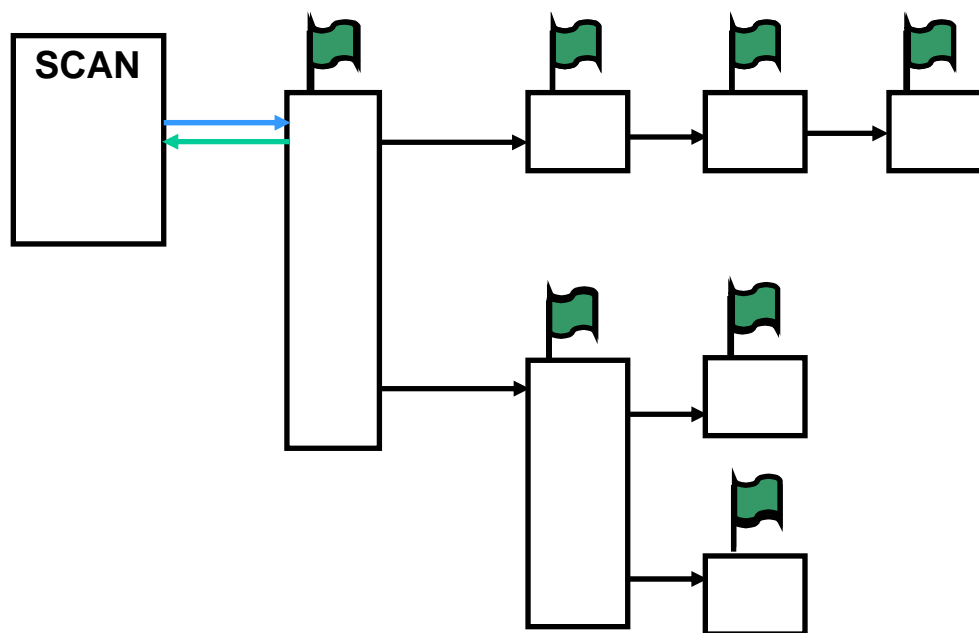
# Completion reporting

---

- **Simple prescription for databases contained within a single ioc:**
  - Use only PP links and forward links in execution chain.
- **Database operations spanning more than one ioc:**
  - Use records with put\_callback links to span iocs:
    - **calcout** with asynchronous device support
    - **sscan**, **swait** (i.e., a synApps “userCalc”)
    - **sseq** or **sCalcout** (with .WAIT\* = “Wait”)
- **Cases in which a CA client performs part of the operation:**
  - 1) Database sets a **busy** record via PP or put\_callback link.
  - 2) CA client clears the **busy** record when operation is done.
- **Cases in which part of the operation is driven by a CP link:**
  - Not different from above; a CP link is a CA client

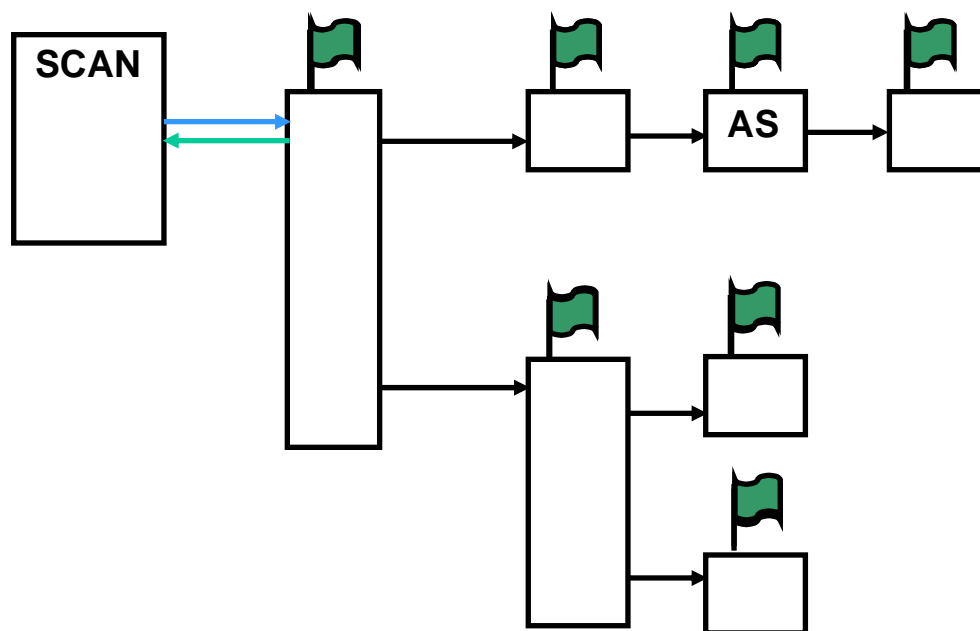
# ...Completion reporting

- Use only PP links and forward links in execution chain.



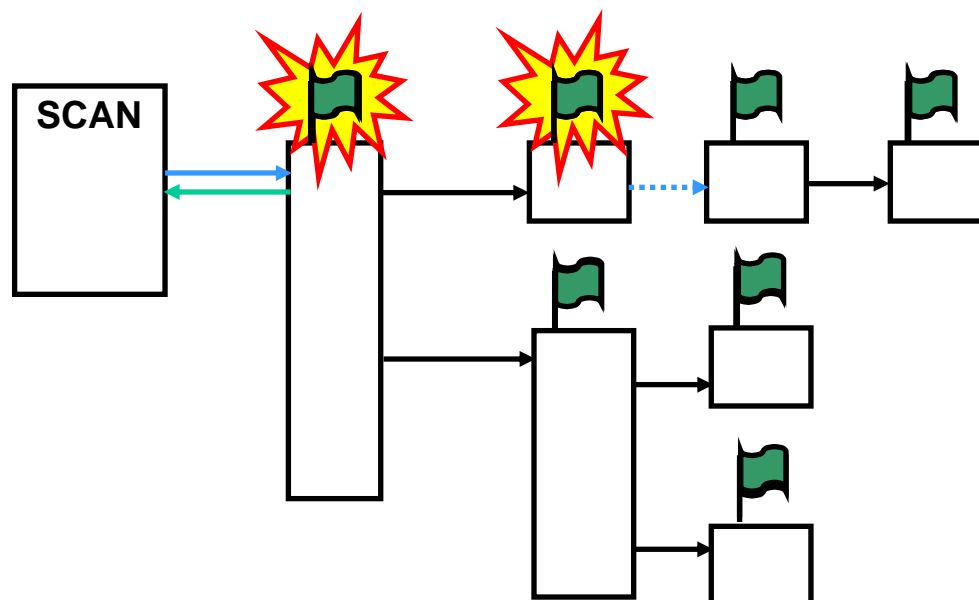
## ...Completion reporting

- Same as before, but with an *asynchronous* record



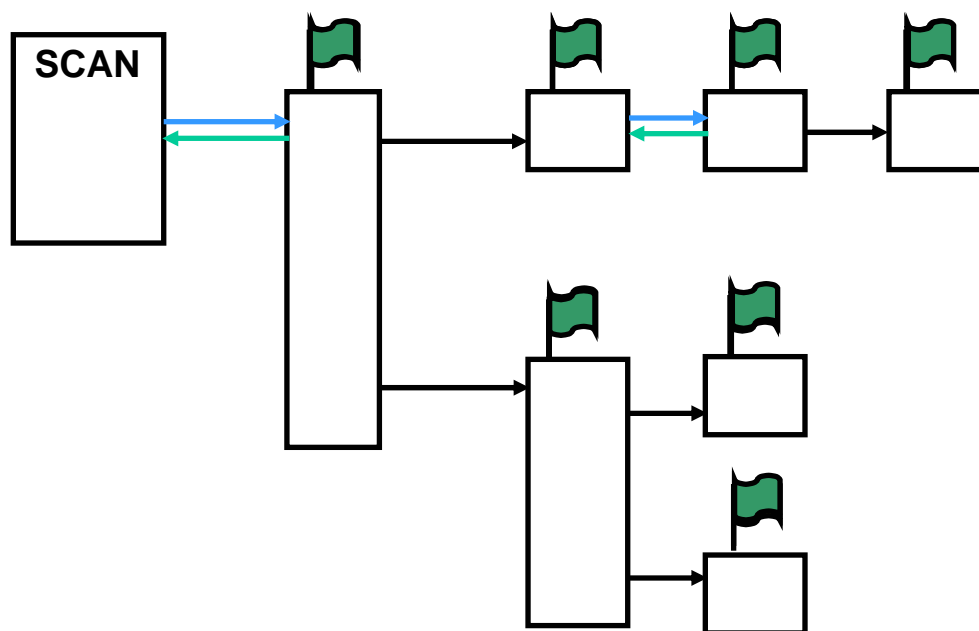
# ...Completion reporting

- Premature “DONE” report, because CA-link execution is not traced



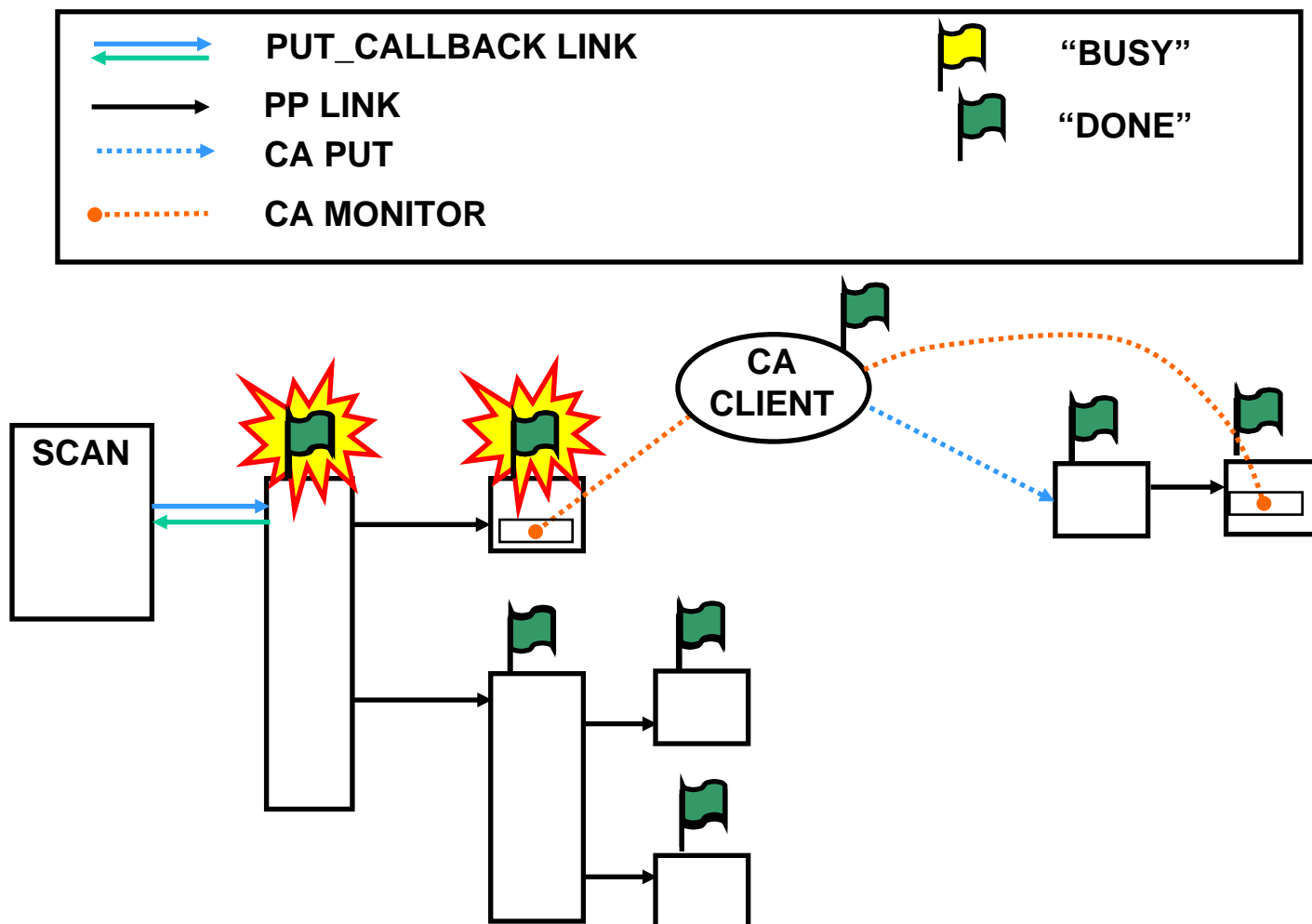
# ...Completion reporting

- Premature-DONE problem fixed with a PUT\_CALLBACK link



# ...Completion reporting

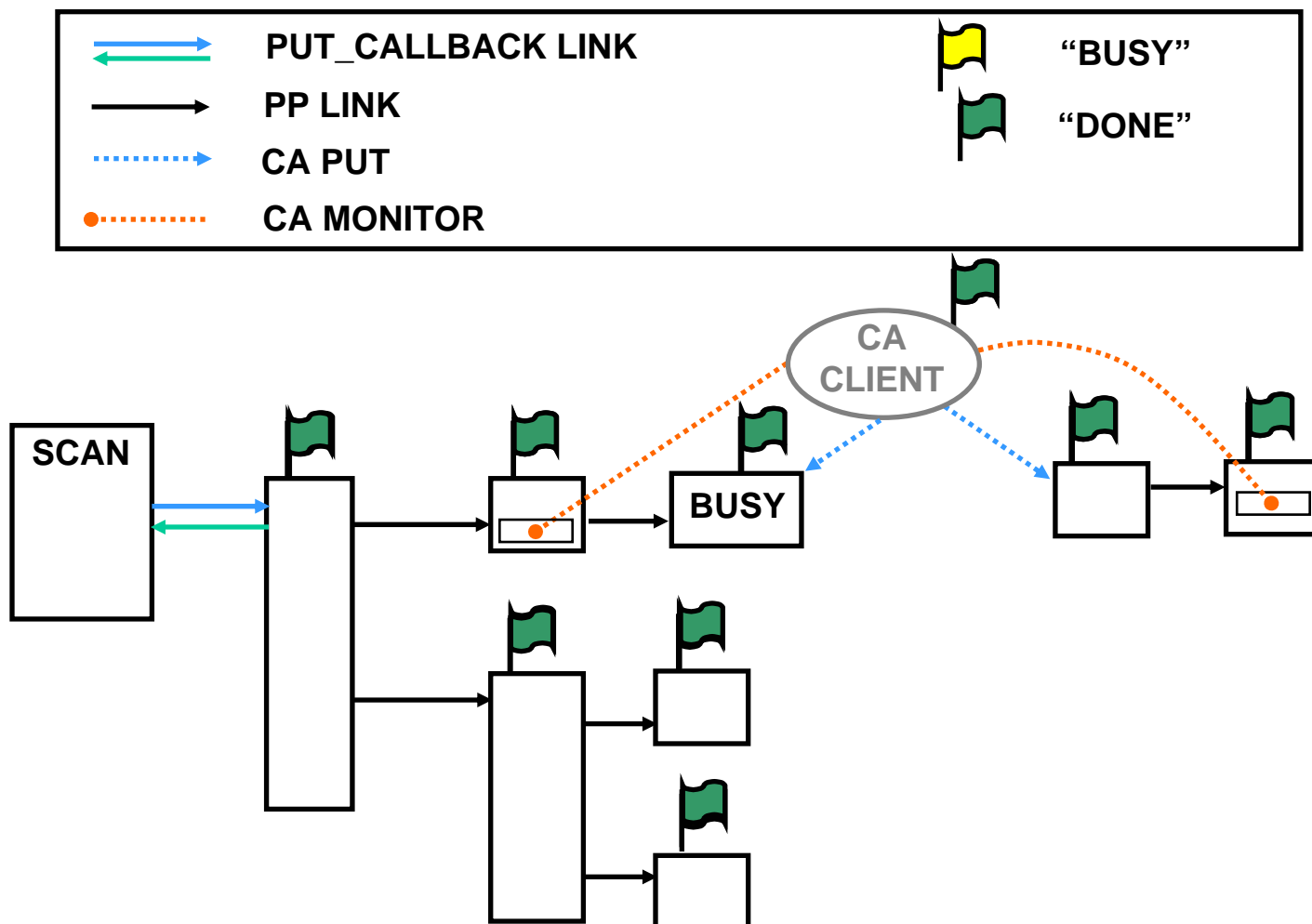
- Premature “DONE” because CA-client processing is not traced

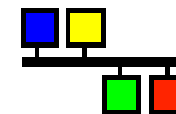




# ...Completion reporting

- Premature “DONE” problem fixed with a ‘BUSY’ record





# ...Completion reporting

---

