

# 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*

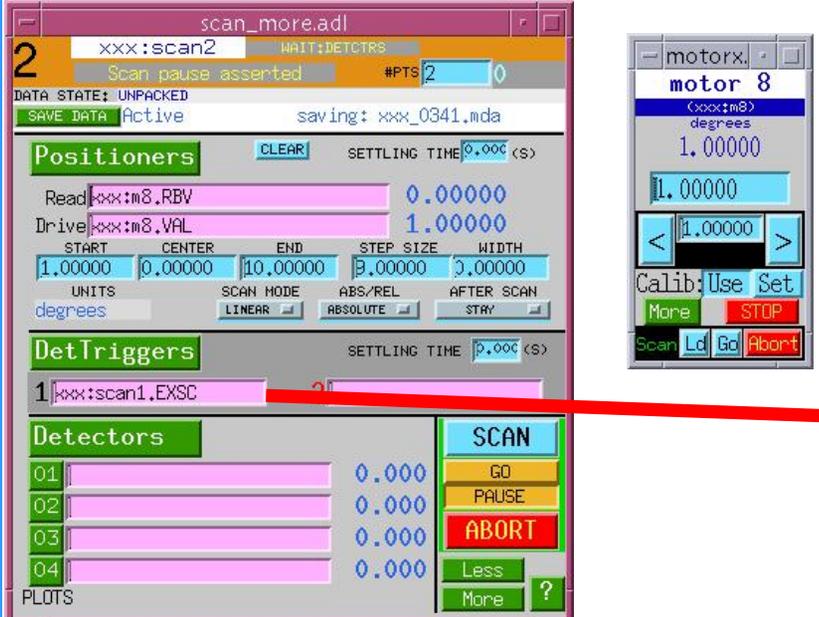
- Write data to disk

The screenshot displays the EPICS IOC software interface for a scan. The main window, 'scan\_more.adl', shows a scan configuration for 'xxx:scan1'. The scan is currently paused by the operator. Key parameters include #PTS 100 and 55. The 'Positioners' section shows the motor 7 position at 1.11111. The 'DetTriggers' section shows the scaler 1 count at 100000. The 'Detectors' section shows the scaler 1 count at 0.010. The 'motorx' window shows the motor 7 position at 1.11100 and 1.11111. The 'scaler.adl' window shows the 'Done' status and a count of 100000. Red circles and arrows highlight the connection between the scan parameters and the motor/scaler data.

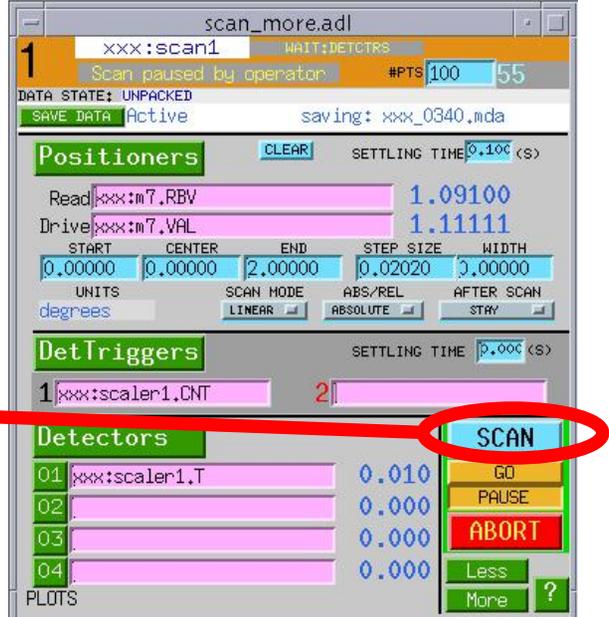
# ...Simple scans

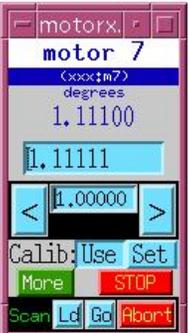
- **Multidimensional scan:**
  - Outer-loop scan's *detector trigger* executes inner-loop scan.
  - **saveData** monitors a set of **sscan** records, determines scan dimension when scan starts, and writes data as it is acquired.
  - No limit to the number of scan dimensions.

### 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

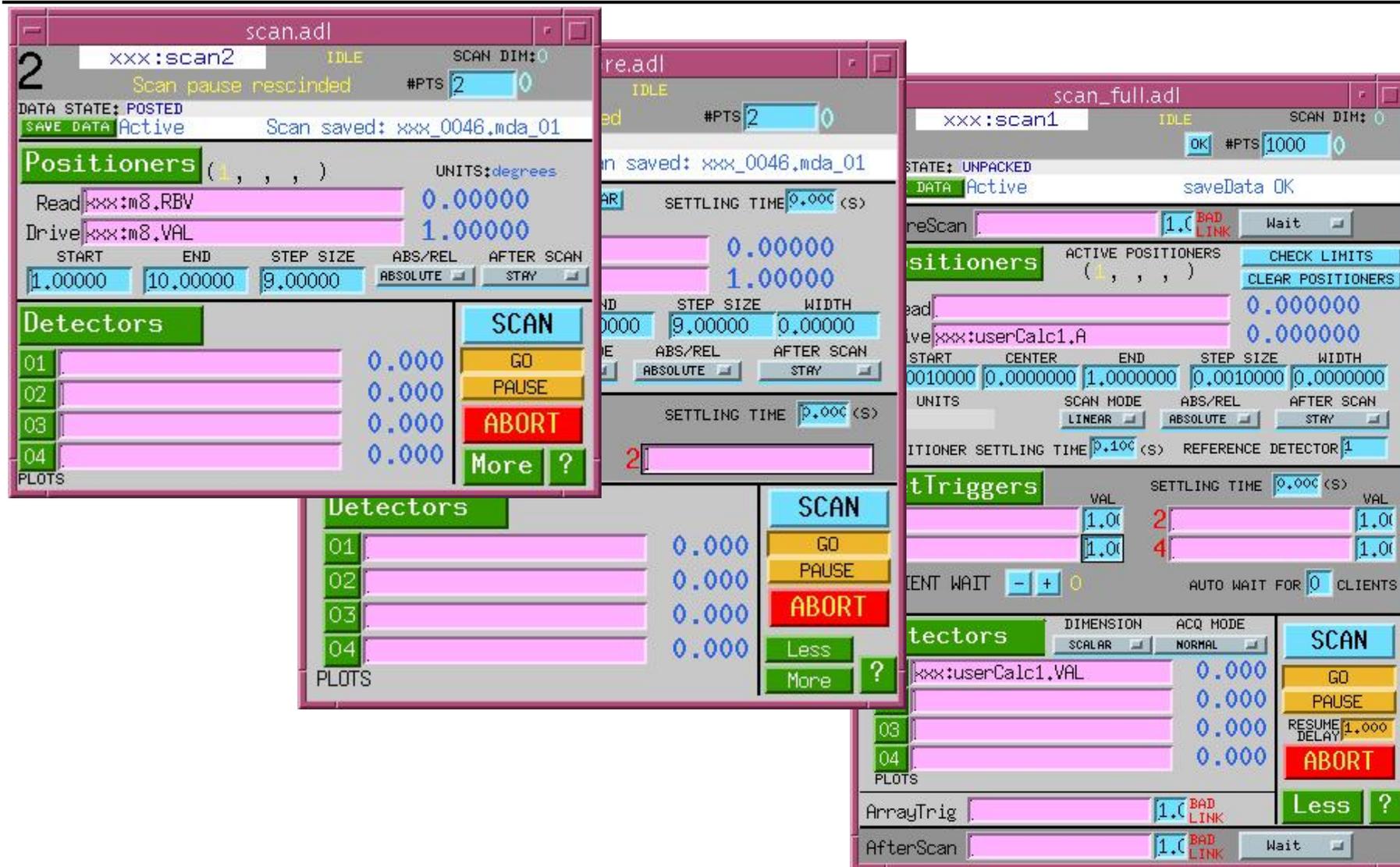
- Can write a constant value to any numeric or menu PV before the scan starts and/or after the scan ends.
- Can wait or not wait for completion of processing started by the write.
- If this sscan record is part of a multidimensional scan, links function on each iteration.
- Outer-loop sscan record can 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' control panel. At the top, it displays 'xxx:scan1' and 'IDLE' status. Below this, there are fields for 'BeforeScan' and 'AfterScan', both set to '1.0' and marked as 'BAD LINK'. The 'Positioners' section includes 'Read' and 'Drive' fields, with 'Drive' set to 'xxx:userCalc1.A'. A table of scan parameters is visible, including START, CENTER, END, STEP SIZE, and WIDTH. The 'DetTriggers' section shows four triggers, each with a 'VAL' field set to '1.0'. The 'Detectors' section lists four detectors (01-04) with 'VAL' fields set to '0.000'. On the right side, there are buttons for 'SCAN', 'GO', 'PAUSE', 'RESUME', 'DELAY', 'ABORT', 'Less', and '?'. At the bottom, there are 'ArrayTrig' and 'AfterScan' fields, both set to '1.0' and marked as 'BAD LINK'. A blue arrow points to the 'AfterScan' field at the bottom.



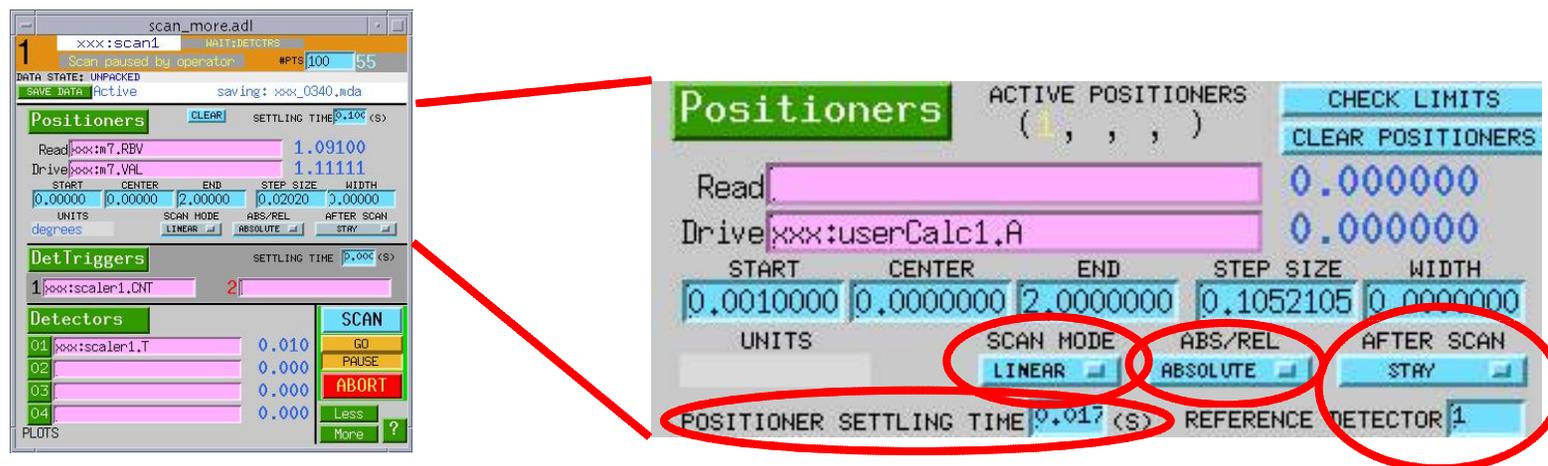
# MEDM user interface



The screenshot displays three overlapping windows from the MEDM user interface:

- scan.adl (left):** Shows a scan named 'xxx:scan2' in an 'IDLE' state. It includes a 'Positioners' section with fields for Read (xxx:m8.RBV) and Drive (xxx:m8.VAL), and a 'Detectors' section with four channels (01-04) all showing 0.000. Control buttons include SCAN, GO, PAUSE, ABORT, and More ?.
- readl (middle):** Shows a scan named 'xxx:scan1' in an 'IDLE' state. It includes a 'Positioners' section and a 'Detectors' section with four channels (01-04) all showing 0.000. Control buttons include SCAN, GO, PAUSE, ABORT, Less, and More ?.
- scan\_full.adl (right):** Shows a scan named 'xxx:scan1' in an 'IDLE' state. It includes a 'Positioners' section, a 'Triggers' section with two channels (2 and 4) showing 1.00, and a 'Detectors' section with four channels (01-04) all showing 0.000. Control buttons include SCAN, GO, PAUSE, RESUME DELAY, ABORT, Less, and More ?.

# Positioner options



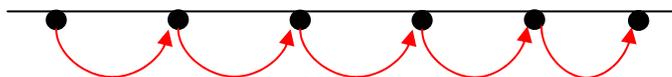
- **SCAN MODE (.PnSM - per positioner)**
  - Determines how and to where positioner moves
- **Absolute/Relative (.PnAR - per positioner)**
  - Determines how positioner locations are written
- **Positioner delay (.PDLY - affects all positioners)**
  - Delay while positioners are settling, after completing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

# ...Positioner options

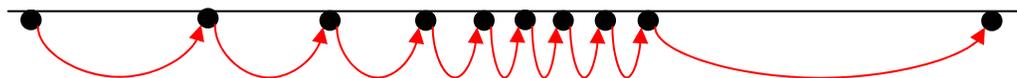
- **SCAN MODE (.PnSM - per positioner)**
  - **LINEAR** – Evenly spaced positions are calculated algorithmically
    - You specify positioner locations by setting any three of

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

- The sscan record reconciles unset parameters

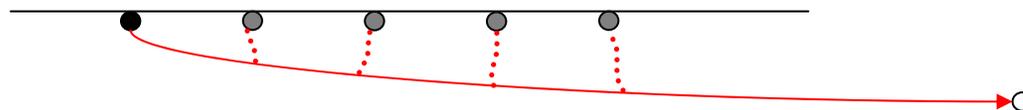


- **TABLE** – Positioner locations are contained in the *.PnPA* array
  - The array must contain at least **.NPTS** values
  - You must arrange for the array to contain the desired positions before starting the scan.
  - The *.PnPA* array is never overwritten by the sscan record



# ...Positioner options

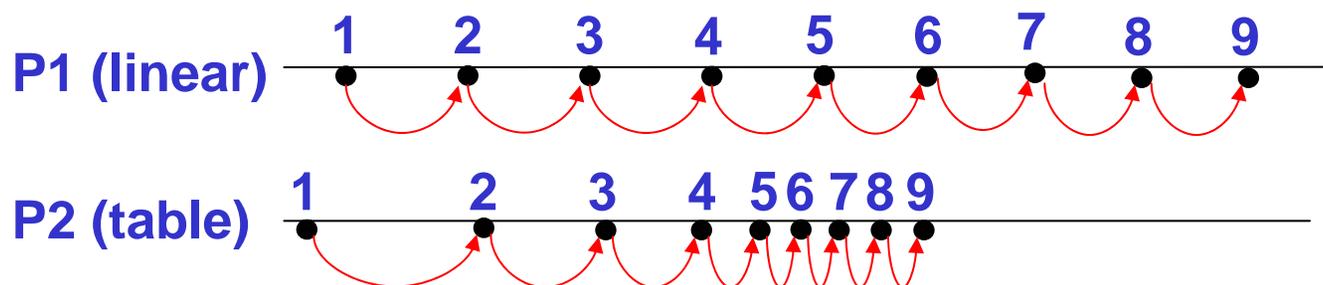
- ...SCAN MODE (.PnSM - per positioner)
  - FLY – data will be acquired *while* positioner moves
    - You specify positions at which data are acquired by setting *START*, *END*, positioner speed, and detector acquisition time.
    - The following algorithm is executed:
      - Positioner sent to *START*; 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.)



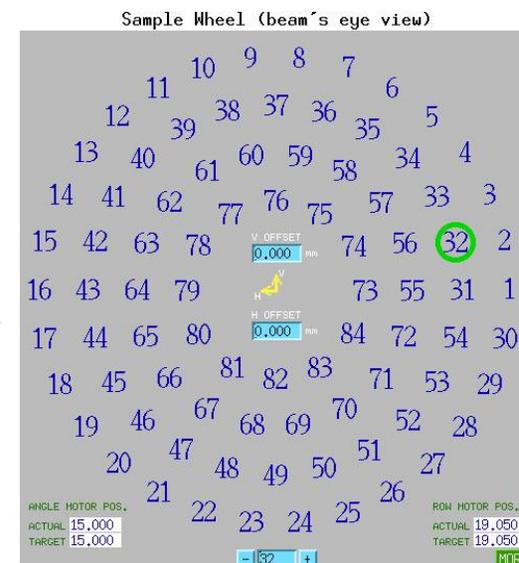
# ...Positioner options

- ...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

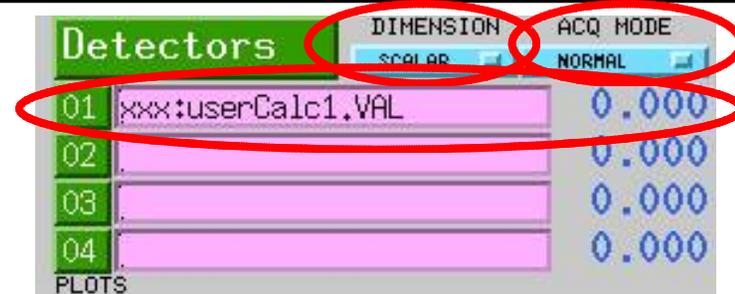
# Detector triggers



- 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.



# Scan user documentation

## MEDM displays

This display controls a one-dimensional scan, or one dimension of a multi-dimensional scan. In multi-dimensional scans, inner scans act as detectors (triggered and waited for, though usually not read) for outer scans.

In a one-dimensional scan, the following sequence of actions occurs:

- 1) Write to before-scan PV, and (option) wait for completion.
- 2) Generally, NPTS iterations of:
  - Write to positioners.
  - Wait for positioners to declare themselves 'done'.
  - Wait for positioner-settling time.
  - Write to detector triggers.
  - Wait for triggers to declare 'done'.
  - Wait until client-wait count is zero.
  - Wait for detector-settling time.
  - Read positioner readbacks and detectors. If acquisition mode is 'ACCUMULATE' or 'ADD-TO-PREV', new data is added to last scan's data.
- 3) Write to array-read trigger, wait for completion, and read any array-valued detector signals.
- 4) Wait for data-storage client to finish writing last scan's data.

See also: [About Display Fields](#)

Mostly status and identifying info. The number of data points is also specified here.

State of data according to scan record 'POSTED' means it's been sent to saveData

Info about saveData -- the program that monitors scans and writes data files.

Command that executes before any positioners move. Usually, a write causes some processing to occur, and you choose whether the scan should wait for that processing to complete before proceeding to the next step.

Positioners set conditions under which data will be acquired. For example, you can move motors, set amplifier gains, etc. After all positioner commands have been sent, the scan waits for positioners to declare themselves done, and then waits for any programmed settling time, before triggering detectors. [MORE](#)

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. Because some data-acquisition is done by clients that can't declare completion in the normal way, a client-wait field allows clients to declare completion by writing to a PV. If clients aren't quick enough to rescind completion, the software can do it for them. [MORE](#)

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'. [MORE](#))

Command that executes after the scan is finished, data has been posted, and any positioner-after-scan motions have completed.

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 readable PV can be named as a detector-trigger, but if the scan is to wait for the triggered action to complete, then either the PV must be implemented so that its 'done' callback really does signify completion of that action, or some other wait mechanism must be used.

The possibility is the detector-settling time. If no better scheme 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 scheme is for the data-acquisition client to declare completion by writing '0' to the scan's WAIT field. (The CLIENT-WAIT button labels) -- does exactly this.) The WAIT mechanism is a tolerant handshake, and the client normally asserts that it's busy by writing '1' to WAIT (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'. [MORE](#))

It's ok to have a scan with no detector triggers, no data-acquisition clients, no detectors, or any combination of the above. If the scan doesn't have to wait for trigger callbacks or data-acquisition clients, then it won't delay for the detector settling time either.

The 'reference detector' is the detector whose data will be used to direct positioner after-scan actions. (For example, positioners can be sent to the peak after a scan is finished.)

Data can be saved over several scan iterations. To start such an accumulation beginning with the next scan, set the acquisition mode to 'ACCUMULATE'. To include already acquired data, set the acquisition mode to 'ADD-TO-PREV'.

Command that executes after all scalar data acquisition is done. Intended to cause any array hardware to post data.

Command that executes after all scalar acquisition is done. Intended to cause any array hardware to post data.

NPTS - The number of data points to be acquired in one loop of this scan (dimension). NPTS applies to all positioners.

ABS/REL - Controls whether positions generated by algorithms are to be applied as generated, or added to their pre-scan positions. Each positioner has its own ABS/REL parameter.

AFTER-SCAN - Controls the values written to positioners after the last data point has been acquired. AFTER-SCAN applies to all positioners. The choices:

- STOP - No values are written.
- START POS - Positioners are sent to their first-data-point positions.
- PREV POS - Positioners are sent to their pre-scan positions.
- PEAK POS - Data acquired from the reference detector is acquired, and positioners are sent to positions at which the peak value occurred.
- VALLEY POS - The positions at which the smallest data value was acquired.
- EDGE - Peak of the derivative of reference-detector data.
- CODE - Valley of the derivative.

In FLY mode, the timing of data acquisition is controlled entirely by detector dwell times, and it is your job to make sure that positioners move at speeds commensurate with dwell times. Scan software will acquire positioner-readback data just before it acquires detector-signal data, but otherwise it regards flymode positioners as latched analogies.

It's ok for different positioners to have different scan codes.

If any positioners are in LINEAR or FLY mode, the scan waits after all positioners have declared themselves done, for the positioner-settling time to elapse, before triggering detectors. If no positioners are in LINEAR or FLY mode, the positioner-settling time is zero.

It's ok to have a scan in which no positioners are specified at all. In this case, none of the positioner parameters matter. If positioner readbacks are specified, they will still be read.

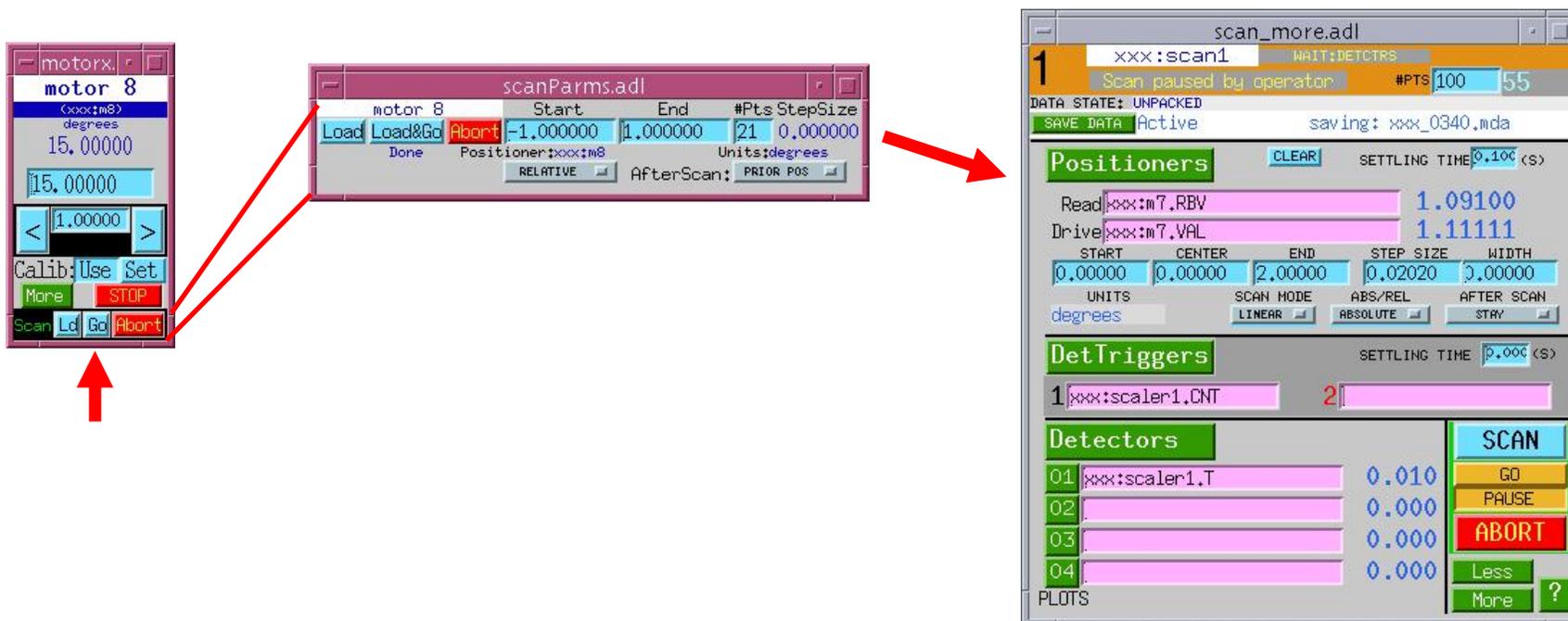
See description of the AFTER-SCAN code

'Rounded to the nearest clock 'tick' Typically, a 60-Hz clock is used.



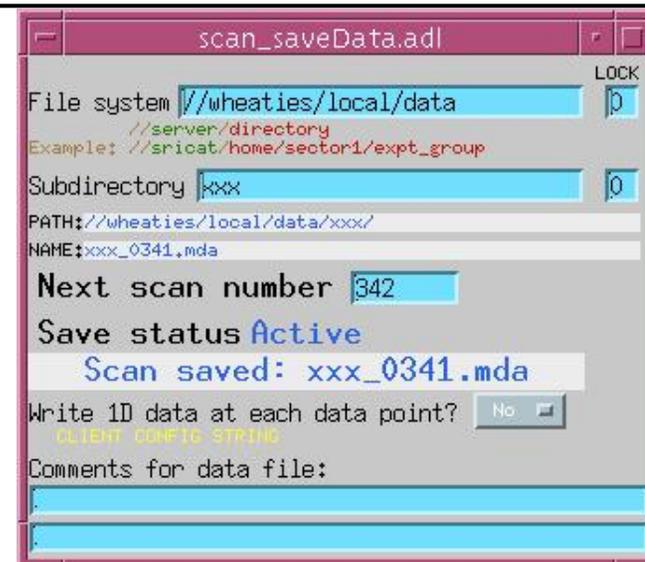
# One-click scans

- 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



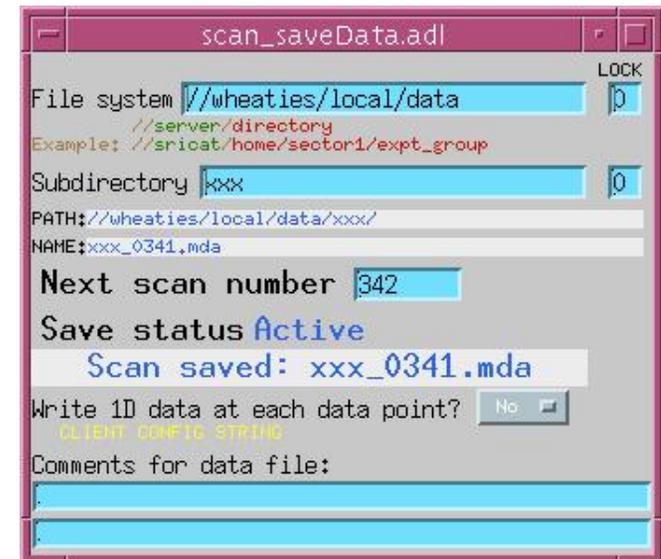
# Data storage

- 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**



# 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"
...

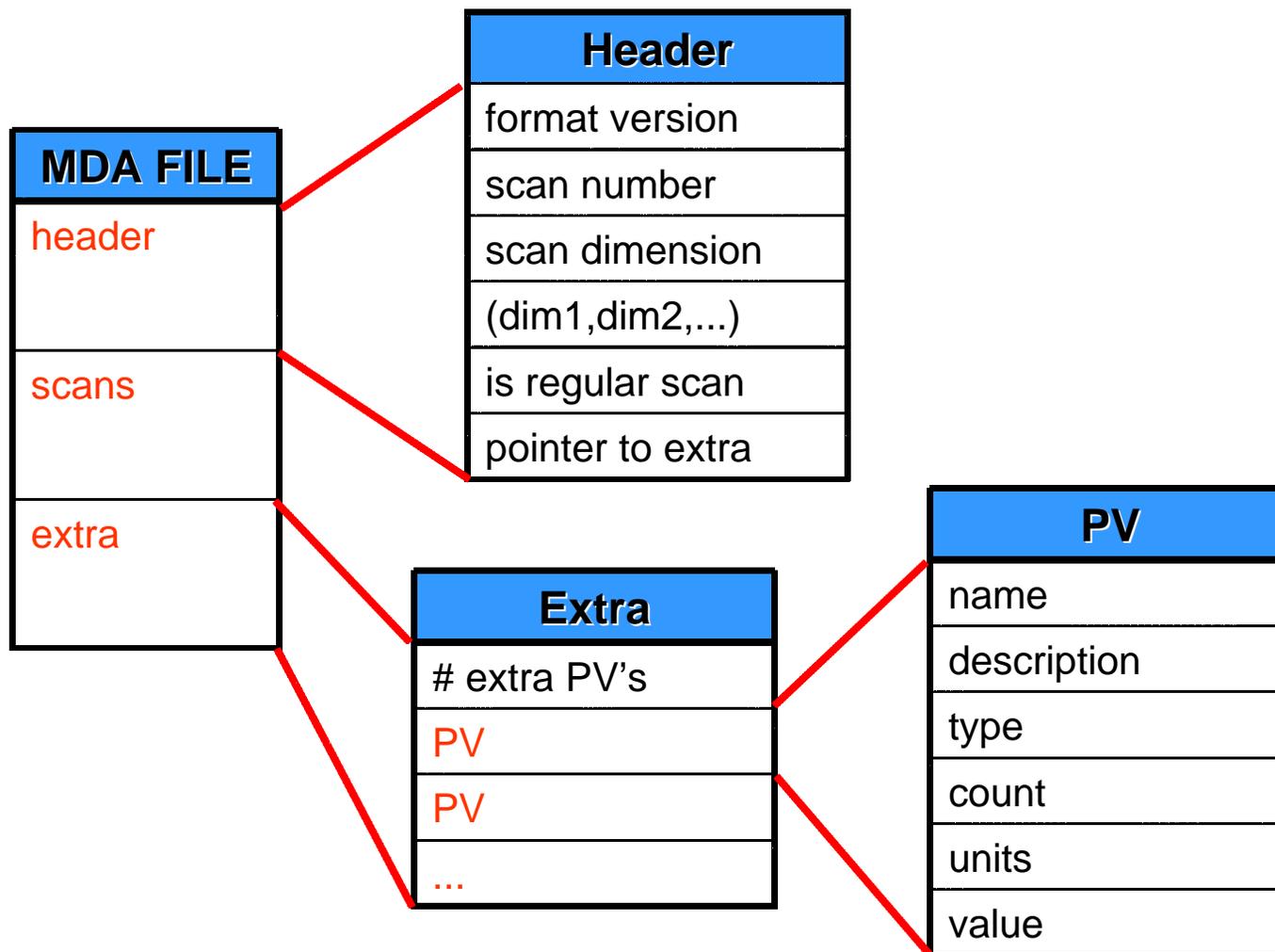
```

Section head

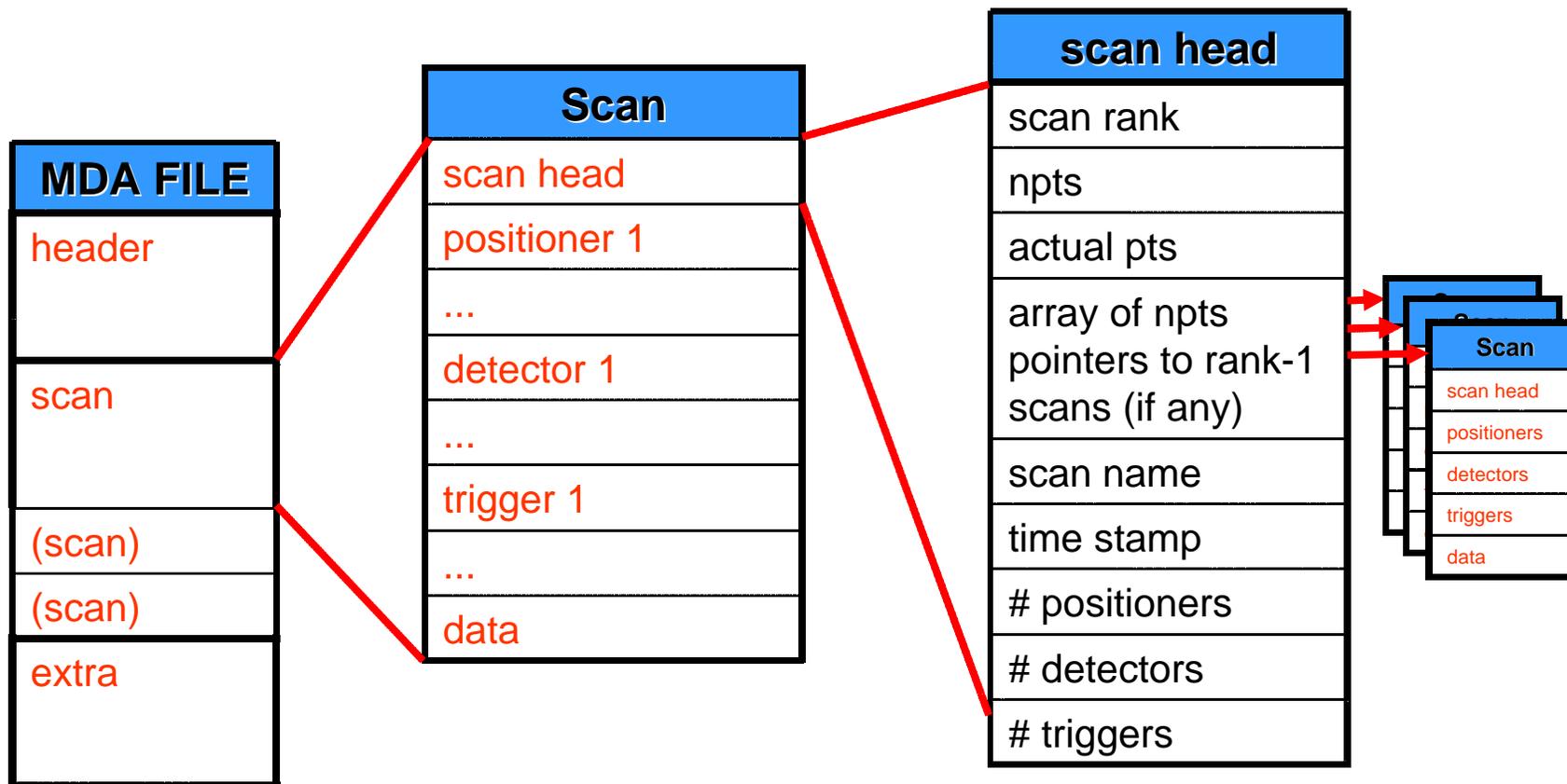
List of sscan records to monitor:

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

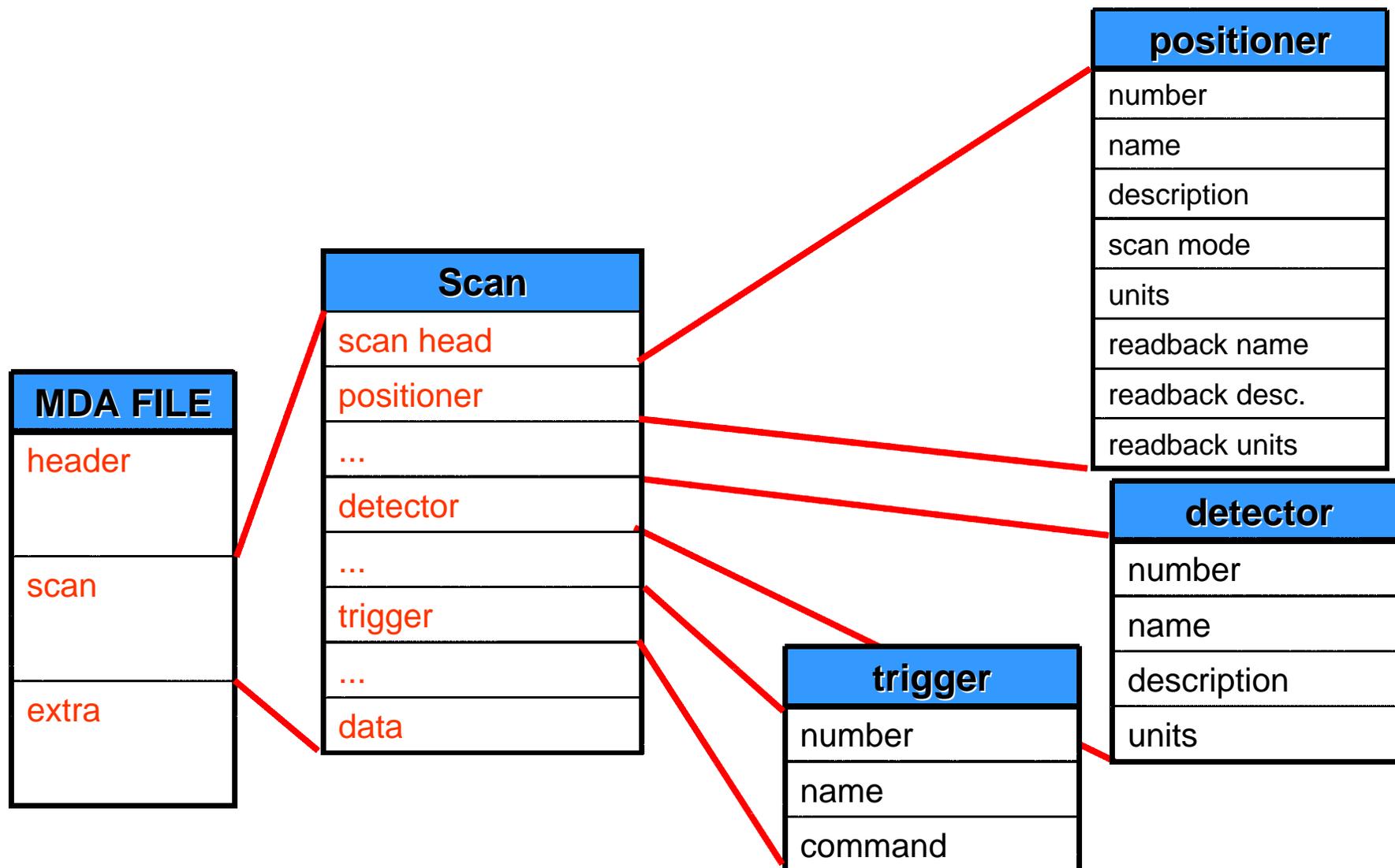
# MDA file format



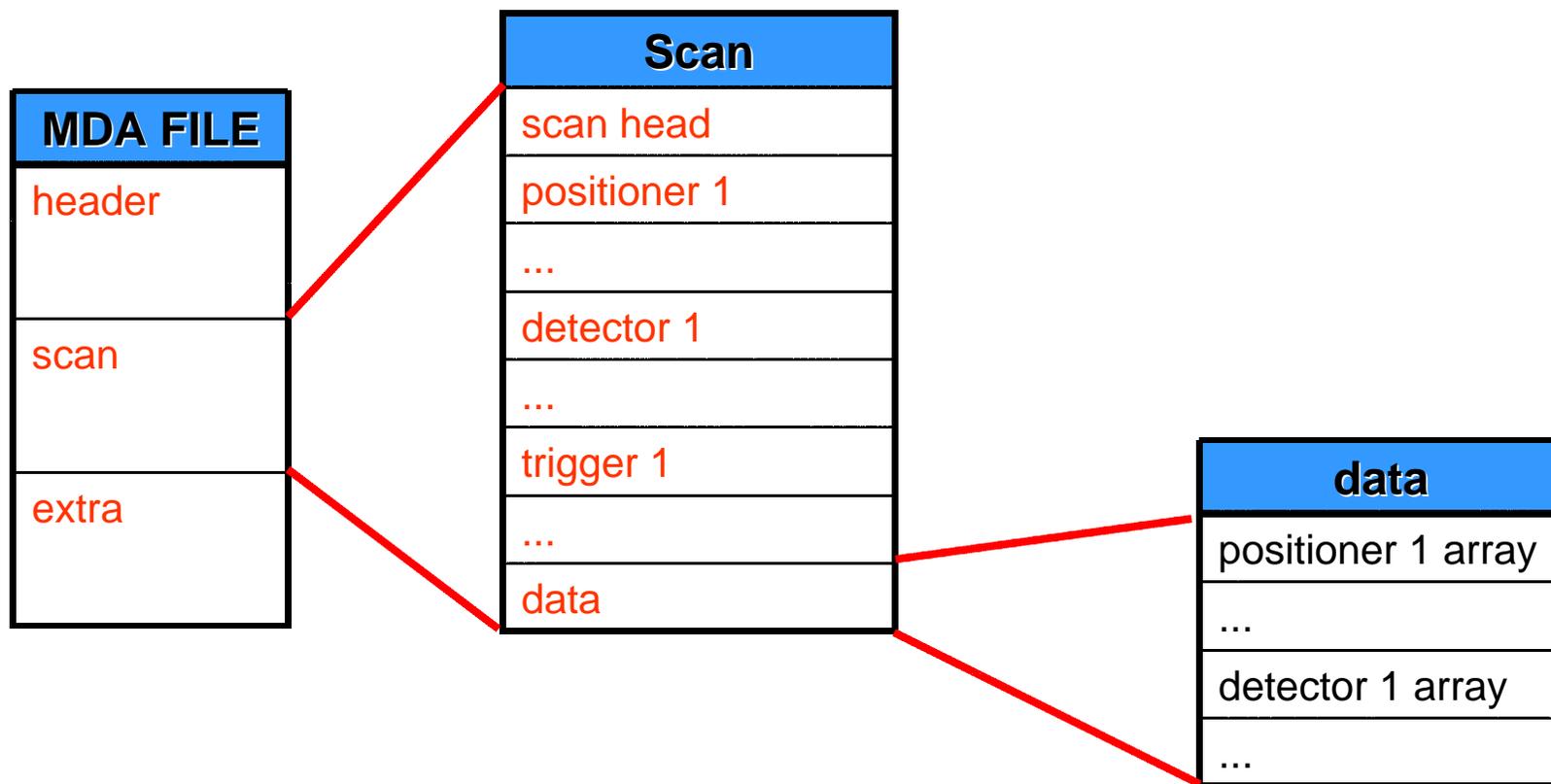
# ...MDA file format



# ...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.

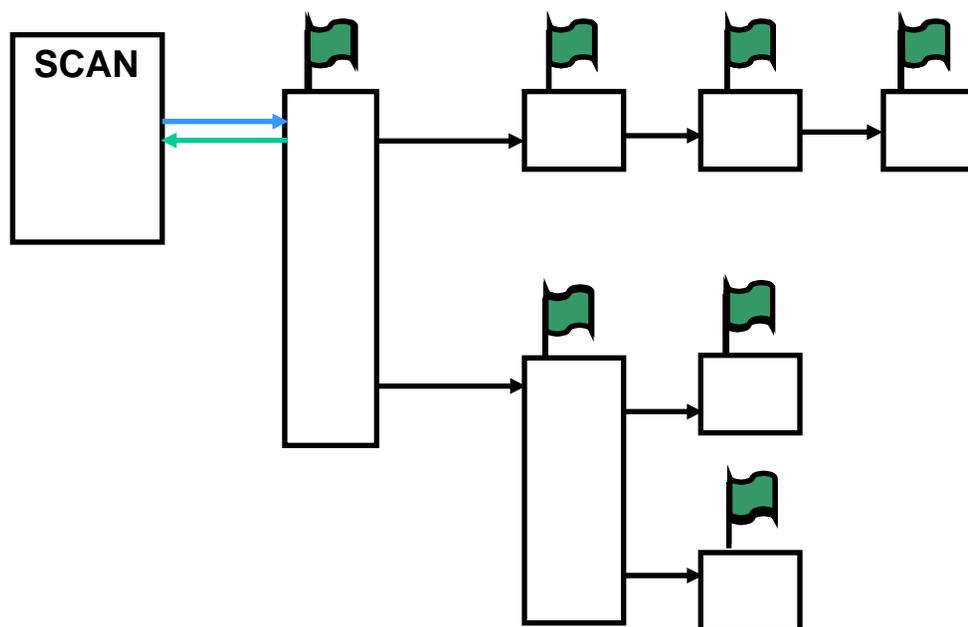
# Completion reporting

---

- **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

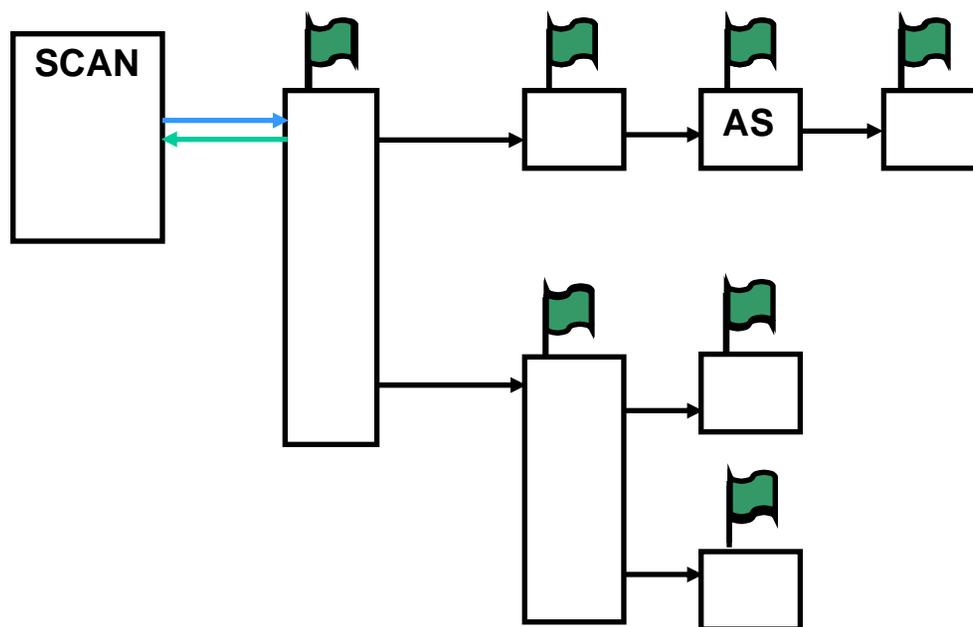
# ...Completion reporting

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



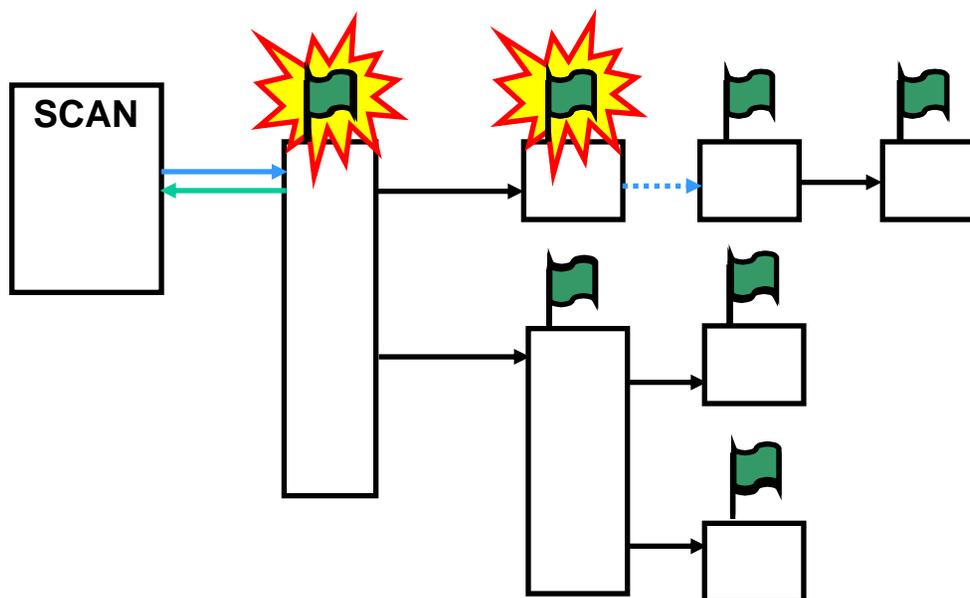
# ...Completion reporting

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



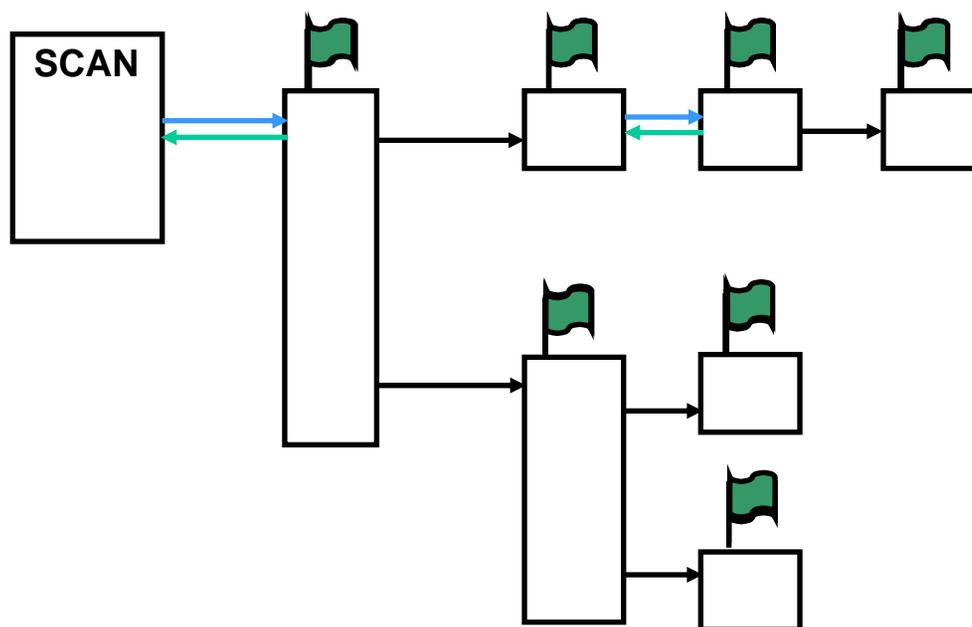
# ...Completion reporting

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



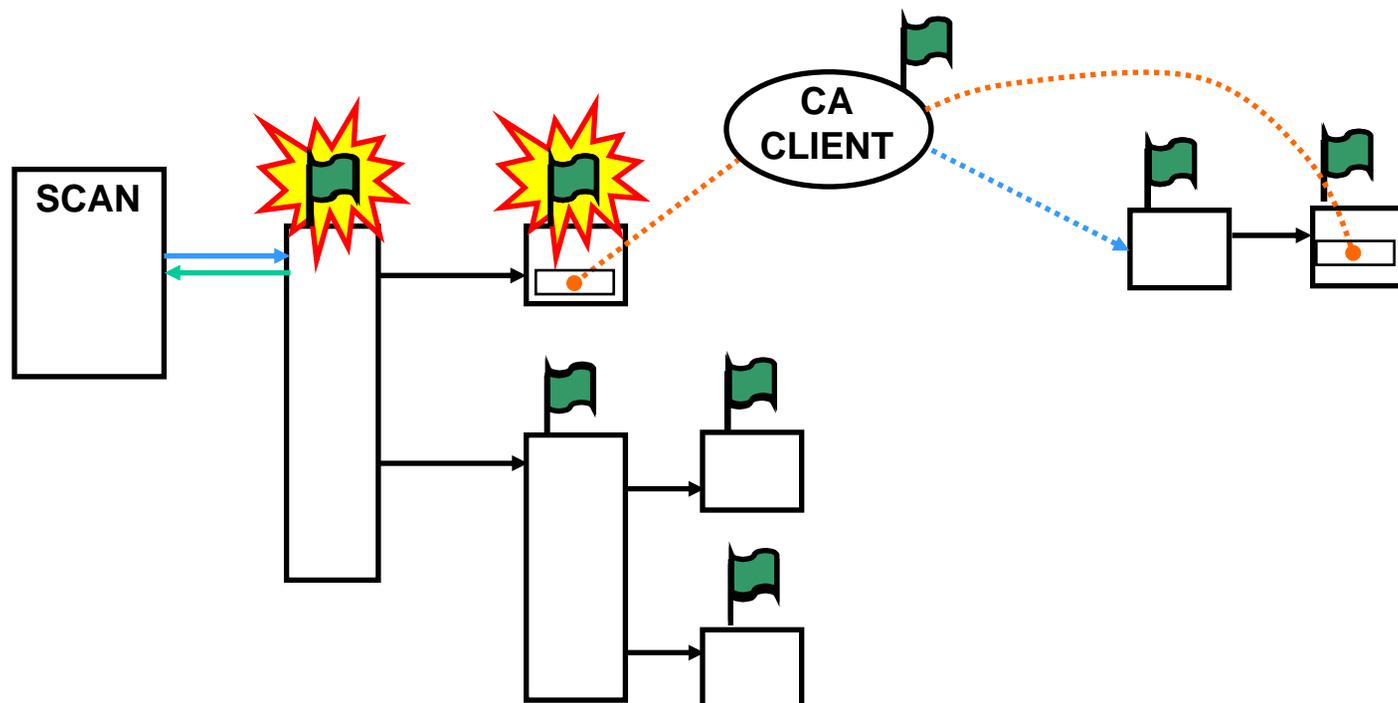
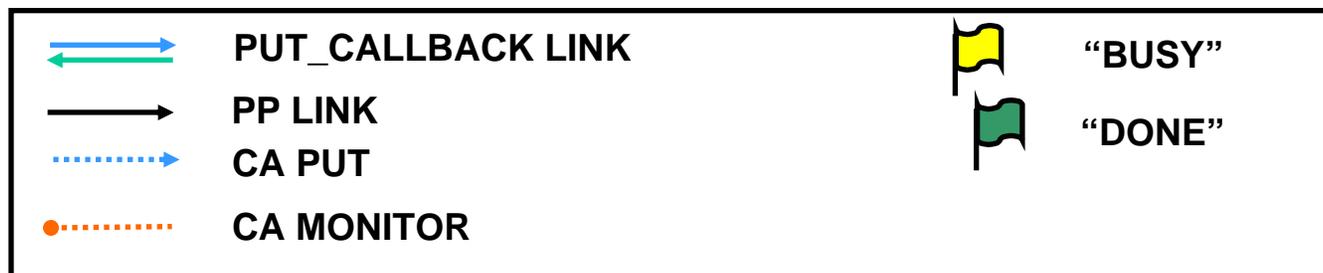
# ...Completion reporting

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



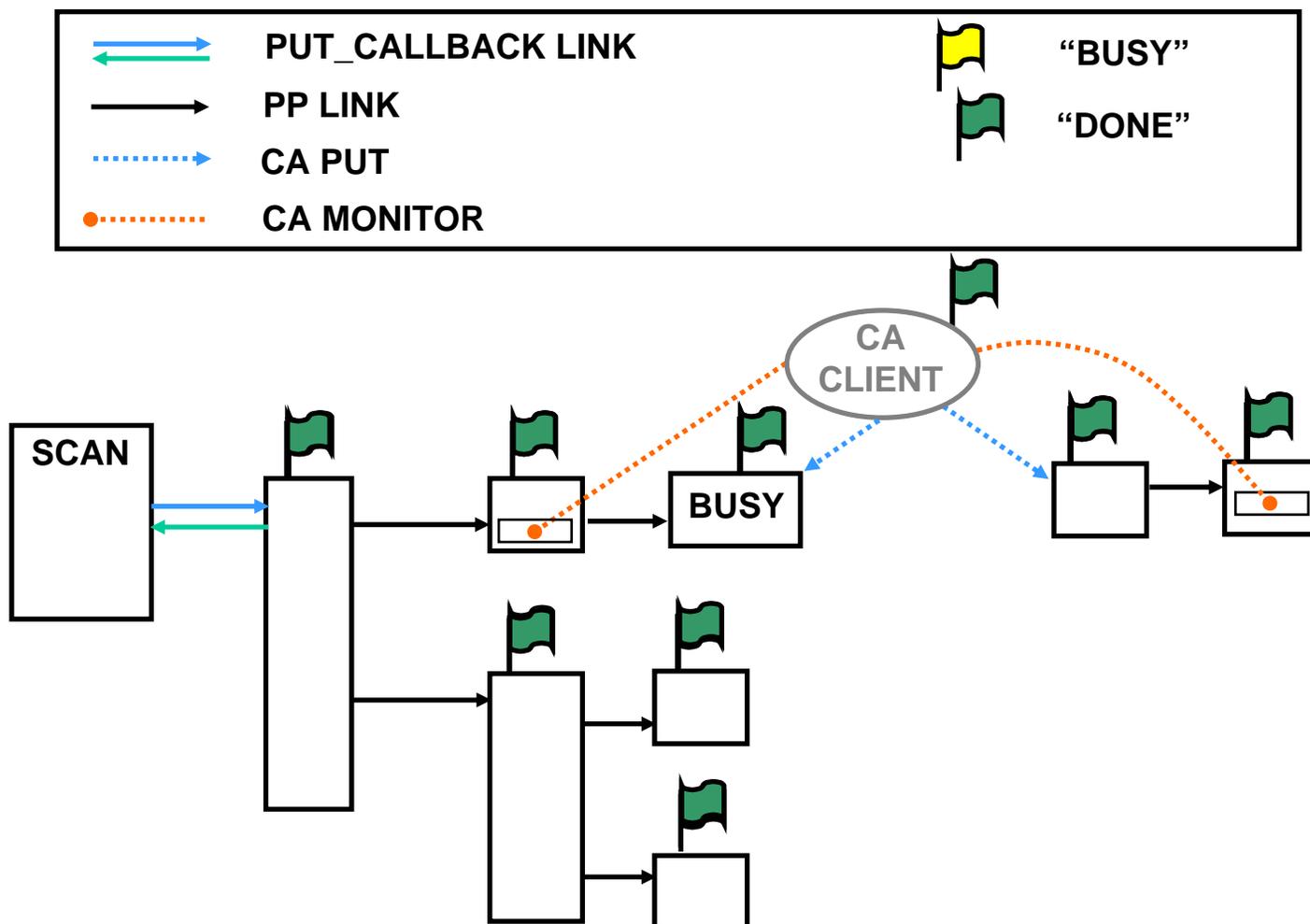
# ...Completion reporting

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



# ...Completion reporting

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



# ...Completion reporting

---

