# DISCOVERING PROCESS-VARIABLE-TO-SIGNAL RELATIONSHIPS IN EPICS 3.X AND 4.X *

N.D. Arnold, D. Dohan, A. Johnson, C. Saunders
*APS, Argonne National Laboratory, Argonne, IL, USA*

## ABSTRACT

For proper documentation and maintenance of a large EPICS-based control system, it is imperative to know the relationship between actual field signals and process variable (PV) names. Numerous attempts at providing such information in an accurate and exhaustive way have been made, yet due to the significant amount of human effort required, most schemes have fallen short (especially in large installations). Ideally, such information could be mined from configuration files using an automatic, exhaustive, and error-free scheme. This paper reviews the difficulty of discovering such information from EPICS 3.x applications and suggests requirements for EPICS Version 4.0 that could simplify this process.

## INTRODUCTION

At the Advanced Photon Source, an effort is underway to document the "as-built" control system (including components, IOCs, PVs, cabling, etc.) using relational database technology. The Integrated Relational Model of Installed Systems (IRMIS) will not only inventory the thousands of entities in the control system, but will also define relationships between them [1]. Because the control system is already in place, the content of the database is used to *describe* the installation, not *prescribe* or configure what will be installed. Therefore, as much data as possible is mined from software configuration files so as to avoid a gargantuan effort and minimize human entry error.

A key element in the IRMIS schema is the emphasis on *connections* or *relationships* between components. For physical devices, these relationships include how devices are housed (Where do I find it?), controlled (How does control data get to/from it?), and powered (Where does it get its power?). For cables, the connections are the relationships to the port on the device at each end of the cable. Since these relationships are stored in the database, extensive and exhaustive queries can be constructed that mine this data to answer operational and maintenance questions. Such queries could include tracing a fault to its root cause, predicting which applications might fail when a module is removed, or locating every instance of a particular device type that is powered by a specific power supply.

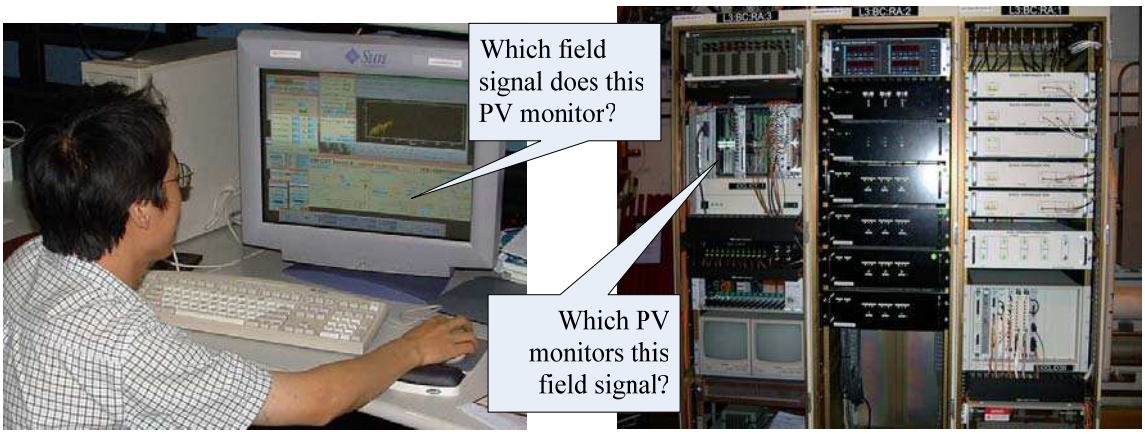


Figure 1. Process-Variable-to-Signal Relationships are Bidirectional.

An extremely useful relationship to track in an EPICS-based control system is the correlation between a PV (i.e., the tag name of a piece of information) and the field signal that this data represents (e.g., a thermocouple wire). This relationship should be traceable in a bidirectional way (Figure 1);

* Work supported by U.S. Department of Energy, Office of Basic Energy Sciences, under Contract No. W-31-109-ENG-38.

i.e., given the PV name, one should be able to determine the appropriate field signal and its connection to the control system, and conversely, one should be able to determine the PVs associated with any particular field signal.

Many approaches have been attempted to document this relationship, but most involve a large amount of human data entry, which is time consuming, error prone, and very difficult to keep current. Figure 2 shows an attempt to document this data in a table format. This paper is interested in the association between the PV names in the far right column and the pin designator (Terminal) on this input module in the far left column. A well-designed cable database will track the wiring from this Terminal number to the field device, and a well-designed PV database will allow querying and viewing the PV definition and wherever it is used by application software. In this example, the relationship between the Terminal and the PV was configured by manually entering the appropriate data into the table. Ideally the correlation between I/O signals and PVs would be determined automatically, thereby eliminating the need to maintain this documentation by hand. This paper investigates what changes are necessary to mine this PV-to-signal relationship data from an EPICS-based control system, to ensure accurate and up-to-date documentation.

| Terminal | Input/Output | Signal Name | Field Connection | Wire ID | Signal | PV |
|---|---|---|---|---|---|---|
| 1 | Input 0 (Ch 1)+ | L2:MA:klyPos+ | Attenuator Control Panel : U2-5 | B2-02-RED | 0 | L2:MA:klyPosAI L2:IOCLID2:ABO:A1:statMI |
| 2 | Input 0 (Ch 1)- | L2:MA:klyPos- | Attenuator Control Panel : U2-6 | B2-02-BLK | | |
| 3 | Input 1 (Ch 2)+ | L2:MA:loDrvPos+ | Attenuator Control Panel : U2-7 | B2-03-RED | 1 | L2:MA:loDrvPosAI |
| 4 | Input 1 (Ch 2)- | L2:MA:loDrvPos- | Attenuator Control Panel : U2-8 | B2-03-BLK | | |
| 5 | | | | | | |
| 6 | Input 2 (Ch 3)+ | L2:LL:psP24V+ | Attenuator Control Panel : U3-1 | B2-01-PR1-RED | 2 | L2:LL:psP24VAI |
| 7 | Input 2 (Ch 3)- | L2:LL:psP24V- | Attenuator Control Panel : U3-2 | B2-01-PR1-BLK | | |

Figure 2. Typical Table for Associating a PV to Field Wiring.

## THE MISSING INFORMATION

In EPICS iocCore, a PV is the name given to a record within the IOC database (to be completely correct, the PV name comprises both the record name and a field name). A record is an instance of a type of functional block (e.g., analog-in, binary-out) that is usually designed to be independent of any particular hardware type (e.g., the analog input record type is used for many different devices). In order to retrieve or deliver the data from or to a particular device, the record support code calls a device support routine. This device support routine may be customized for a particular device, or it may call an underlying driver support routine that is customized for a particular device. Thus it is the device or driver support code that is specific to a device type, including I/O register maps, protocols, and association of signals to registers.

Figure 3 illustrates the basic flow of information to retrieve data from an input device. When the record processes, in order to read the data from the instrument, it looks up the relevant device support routine for that device in the Device Support Entry Table (DSET) that was located using the DTYP (Device Type) field at boot time. The record support then calls the device support routine. This device specific code uses the INP (Input) or OUT (Output) field to determine which device to talk to and what information to retrieve. Once it has fetched the required data, either directly or by calling a driver support routine, it places it in the appropriate fields of the record and returns control to the record processing routine for further manipulation as needed.

All the information shown in Figure 3 up to step 3 is currently mined by IRMIS and included in the process variable schema. Specifically, the “device(ai, VME_IO, devAiXX, ‘My XX’)” entry from the DBD (DataBase Definition) file is mined along with each record instance’s INP or OUT field. The DBD information must be obtained for each IOC independently, since the association of device support with record types can vary between IOCs.

Figure 3 shows that it is not possible to derive automatically the association between a defined Process Variable and its port or pin on a specific device, using EPICS 3.x. The flow of information can be followed by an intelligent script until step 3, where the particular device support routine(s) called can be specific to the record type. The device and/or driver is also at liberty to make use of the address information found in the INP or OUT field however it sees fit, thus the mapping to a specific pin or signal on the device cannot be deduced. Therefore, it is necessary to maintain one or more additional tables in the relational database as well as suitable algorithms to perform the actual mapping. We believe it is entirely practical to maintain this missing information though, and when combined with the wealth of information already in the component control hierarchy, the actual process of mapping is straightforward. It is possible that additional reporting or annotation conventions in EPICS 4.x might allow us to come closer to automated discovery.

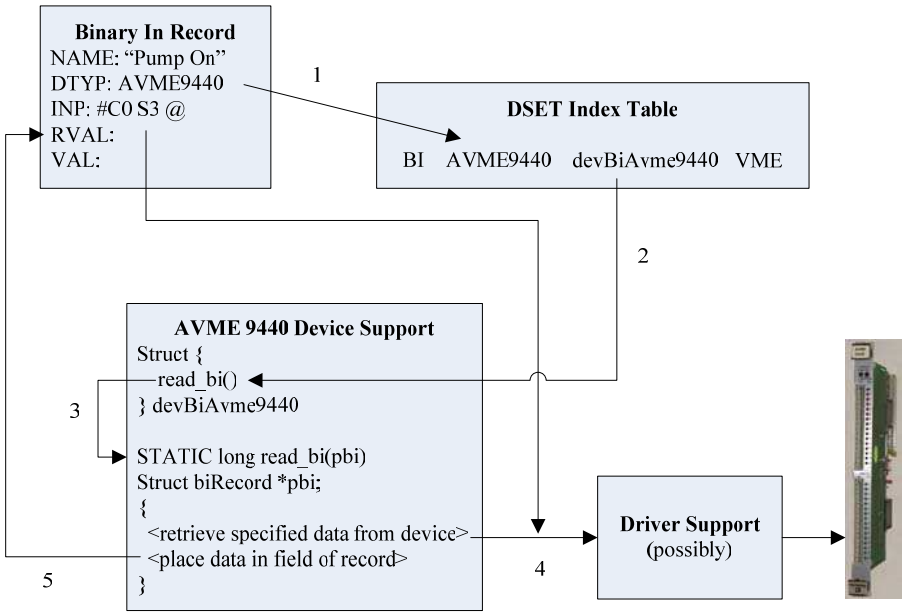


Figure 3. Tracing the Relationship Between a PV and the Hardware in EPICS.

For EPICS 3.x, a manually maintained table will be needed that keeps track of the device support inventory and hardware association(s) that are currently documented informally on the EPICS website. This will make explicit the current informal association between device support modules and the exact types of hardware (component types) they can control. A second manually maintained table is required to define how the INP/OUT link should be interpreted for a given device support module.

## THE DISCOVERY PROCESS

The IRMIS schema includes tables for defining every component type used in the system, a list of the ports (or connectors) on each component type, how these components are connected together both logically and physically, and all the PV definitions in every IOC [1]. The ultimate goal is to have a single database table that associates a PV record id with the appropriate signal id. We propose the outline of a process whose output is precisely this table (Figure 4).

First a minor distinction needs to be made, since we have thus far used the term signal and pin interchangeably. A signal is a named entity, associated with a particular component, that groups together zero, 1, or 2 component pins. A 0 pin signal corresponds to a purely software-generated value, such as a CPU board that reports free memory through a driver. A physical field signal consists of either one pin, as in a single-ended signal or fiber signal, or two pins, as in the case of a signal with ground or a differentially transmitted signal. Part of the task of the IRMIS user in defining a component instance is to define the groupings of pins into named signals. A PV is related to this abstract signal, from which the underlying pins, hardware, and cabling are readily derived.

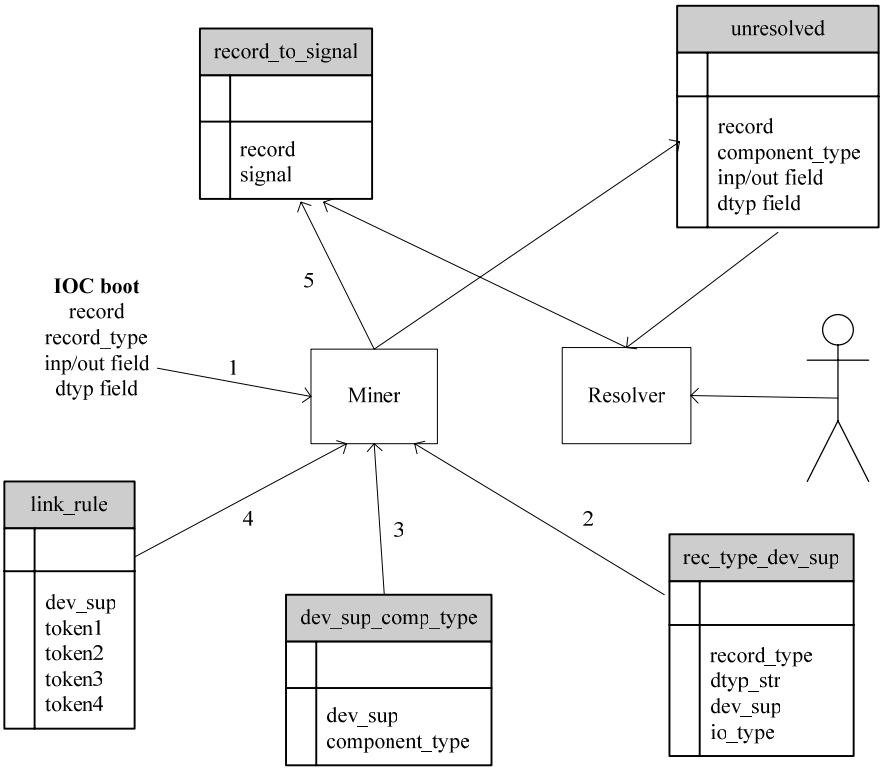


Figure 4. Proposed Scheme for Discovering the PV-to-Signal Relationship.

The discovery process begins with the reboot of an IOC (step 1). The PV crawler runs at this point, scanning in a new set of PVs into the process variable tables. The miner process shown above is run shortly afterwards, with the goal of making a best attempt at associating hardware PVs with components. Some will not be associated, either because the miner was unable to determine the precise component or because the component type has not yet been entered into the IRMIS component schema. In both cases the record and its possible associations are entered into the "unresolved" table, which must be manually resolved.

The miner process begins by iterating on every record in the IOC boot that contains a non-soft DTYP field. For each record, it looks up the device support DSET name and I/O type from the rec_type_dev_sup table (step 2). Recall that this information is automatically mined by the PV crawler and maintained in the PV schema. There will be only one matching entry here, since the DTYP field resolves to only one DSET for a given record type on a given IOC.

The miner then looks up potential component types for the given DSET name (step 3). Knowledge of the supported component types and the IOC in which the records are located dramatically reduces the search space of which component instances could fulfill the "address" given in the INP/OUT field of the record. Some component instances are readily located as they are physically in the same chassis as the CPU component representing the IOC. Some component instances are physically distributed, as in the case of GPIB instruments or a field bus such as Bitbus. Fortunately, the control hierarchy maintained within IRMIS provides a uniform means to locate all components that fall under the control of a given CPU (IOC).

At this point (after step 3) the miner has a list of potential components of the correct type obtained from the control hierarchy (Figure 5). The miner also knows the INP/OUT field address of the record. It reads the appropriate rule from the link_rule table (Figure 4, step 4), which defines what each token in the INP/OUT field means. For an address such as "#C0 S3", the first token is a card number, the second token is the signal number on that card. For a field such as "@asyn(ABC,3,…)", the token is an asyn specifier, and would be further parsed by the miner using business logic that knows how to interpret an asyn address.

The controls hierarchy provides the means to locate the component of interest, either by card number or symbolic name. The miner must now find the correct signal on that component. The physical ports

and pins of that component are ordered and named such that the additional tokens of the INP/OUT field will uniquely identify them. From that, the correct signal can be located by finding the one that contains the discovered pin. This process does depend upon accurate entry of component ports and pins, as well as some amount of component type specific logic. This final address resolution to the signal level will always make use of manually entered data about the ports, pins, and signals of every component. There is no data in an EPICS control system that can provide this final mapping. If the miner fails to resolve the signal, it records the information it does have in the unresolved table for manual intervention.

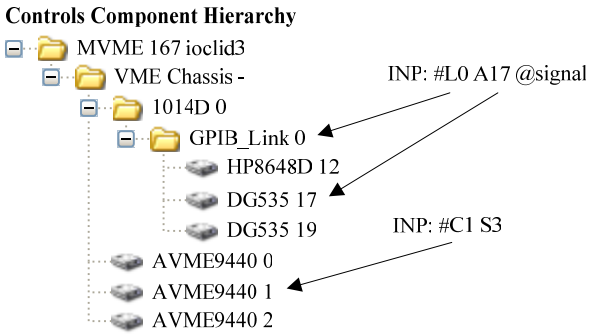


Figure 5. Logical Connections Between Installed Components.

The final step of the discovery process is the entry of a new row in the record_to_signal table. This table answers the bidirectional question posed in Figure 1: which field signal does this PV monitor, or which PV monitors this field signal?

The discovery process described depends on the controls component hierarchy shown in Figure 5 in order to complete the resolution of the INP/OUT address to a single component. However, this same process can also be used to aid the generation of an initial hierarchy [2]. With an empty or minimal hierarchy, the miner process may not be able to find an existing component for a given PV. In cases where a device support module has only one type of associated hardware, the miner can automatically enter this component in the hierarchy. In cases where a device support module has several possible associated hardware types, the miner process will enter this set of candidates in the unresolved table. Manual intervention using the resolver process is then necessary to choose from this candidate set when building each path in the hierarchy. This set of candidates is informed by all the same information the full discovery process utilizes, so the search space of new component types is reduced substantially.

## EPICS 4.X

We have described a process for discovering the signals associated with PVs that relies substantially on the accuracy of data in the controls component hierarchy. We rely on this hierarchy, which must be created through manual data entry, due to the lack of rigid reporting mechanisms in EPICS 3.x. There are hooks for both driver and device support modules to report their configuration at run-time, but these are optional, often incomplete, and intended for debugging purposes, not automatic configuration gathering. The dbior and dbhcr reports do not tie their output back to the address space of the INP/OUT string as defined by the device support module.

We believe that a more organized approach to device reporting or documentation in EPICS 4.x will benefit the controls community by providing data to make the crucial connection between PVs and their hardware accurate and automated. The component controls hierarchy could be generated and maintained automatically. A potential alternative to imposing coding requirements on device support developers would be an annotation convention for the IOC startup file; an agreed-upon format for entering comment text could be utilized by an automated crawler.

## CONCLUSION

The discovery process proposed is not yet part of the IRMIS codebase, although scripts implementing similar ideas have been used to prepopulate the IRMIS component database at APS.

Faced with the task of bootstrapping our component database, it became clear that a best effort to prepopulate all suspected components based on guidance from the PV database was going to speed the data entry greatly. The discovery process will fulfill both goals; bootstrapping the component database, and accurately mapping the PVs to their component signals.

The IRMIS project has a complete PV database, and is close to completing the component and cable databases. This paper addresses the final integration between the PVs and components, after which IRMIS will truly become a fully integrated model of installed systems. The multitude of naming conventions used to label components and signals (a "feature" of drawing and spreadsheet-based approaches) will be normalized into one location in the IRMIS database.

## REFERENCES

[1] D. A. Dohan and N. D. Arnold, "Integrated Relational Modeling of Software, Hardware, and Cable Databases at the APS," Proceedings of ICALEPCS2003, Gyeongju, Korea, October 2003, pp. 365-367 (2004).

[2] N. D. Arnold and D. A. Dohan, "Connection-Oriented Relational Database of the APS Control System Hardware," Proceedings of PAC2003, Portland, OR, pp. 2327-2329 (2003).