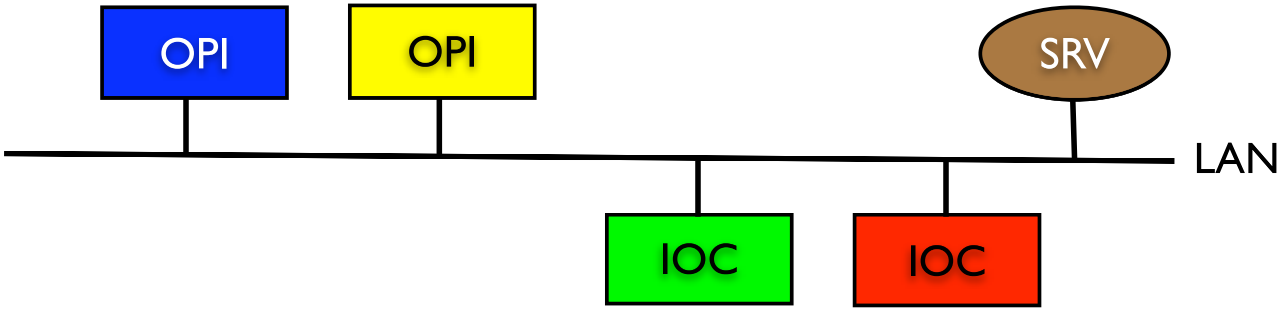
EPICS I

10 March 2008

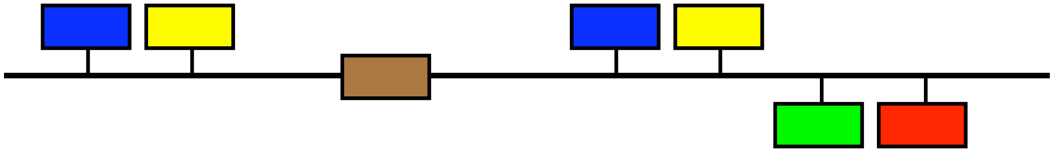
Basic Architecture

- IOC (Input Output Controller)
  - This is the *server*; at least one required
  - Real-time system that defines the Application
    - Traditionally a VME or cPCI crate (hard); can also be any PC/OS (soft)
- OPI (Operator Interface)
  - Workstation/PC with traditional OS (but could be diskless)
  - Runs EPICS *clients*
- SRV (Server)
  - Where Applications are built and loaded from
  - Can be file server for OPI clients
  - Can be archival repository



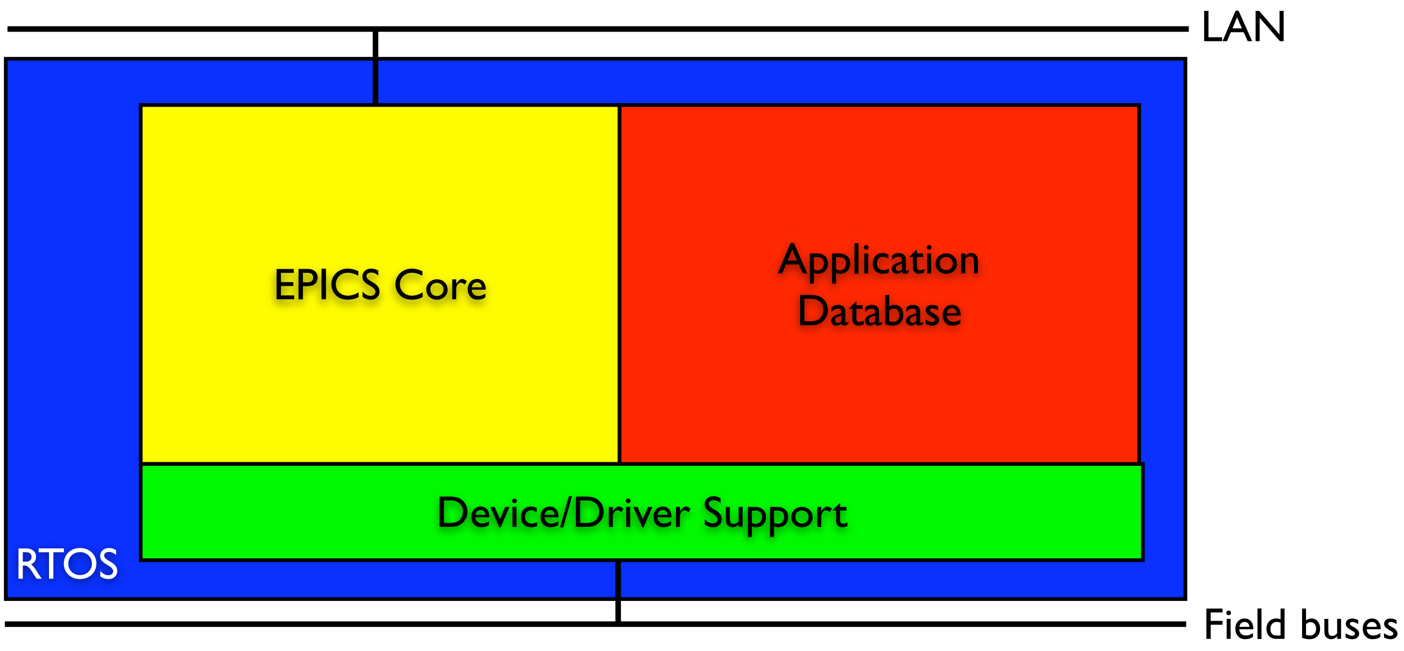
Basic Architecture

- Architecture is ‘Flat’
  - Data moves peer-to-peer (no copy/no relay)
  - No central services; uses a ‘discovery’ protocol
  - No single-point-of-failure
- All entities are independent
  - Clients and servers can be started and stopped in any order
    - Minor versions and be mixed (3.x with 3.y)
- Network
  - Basic system runs on LAN
  - Gateways, switches, and routers can join LANs over WANs

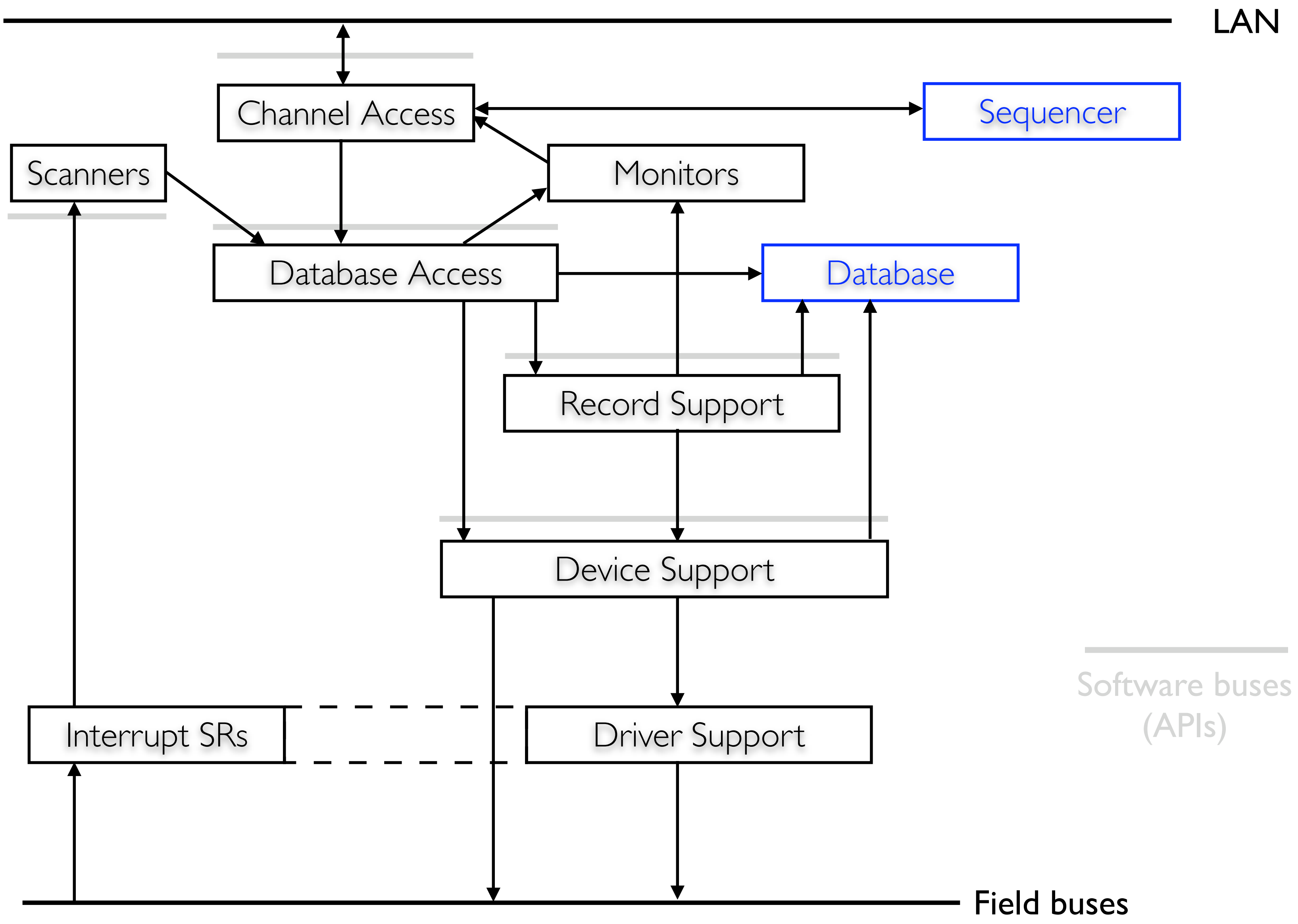


IOC: Basic Components

- EPICS Core (‘iocCore’; ‘base’)
  - Shared (re-entrant) code for records
  - Channel Access server and client
- Application Database
  - Instances of record types, possibly linked
- Device/Driver Support
  - Device- and bus-specific code



IOC: Components



IOC: Components

- The Database and the Sequencer *are* the Application
  - Sequencer is a true *client*
- Channel Access (CA) is the *only* external entry point
- Record-/Device-/Driver-Support/Scanners provided to Application Developer

The Database is the Heart of EPICS

Channel Access is the Backbone of EPICS

The IOC Developer's Guide is the primary reference

IOC: Components

- Database (DB)
  - Memory-resident collection of ‘function-blocks’ (*aka* ‘records’)
  - Composed/Aggregated (‘linked’) to form combinatorial explosion of new functionality
    - Linkage mechanism ‘transparent’
  - Deterministic; runs synchronously or asynchronously; periodically or even-driven
  - Has fine-grained Access Control
  - Provides Simulation and Tracing
  - Has textual and graphical representation (using VDCT)
    - This is main effort for Application Developer
    - Supports a rich variety of instantiation and macro substitution facilities

The fundamental job of the Application Developer is to instantiate and link records into the right processing ‘chains’

IOC: Components

- Record Support
  - Provides functionality for record types
    - Processing is what records ‘do’
    - I/O records use Device Support for hardware access
  - I/O records perform scaling, smoothing, masking, shifting, linearization, *etc*
  - Performs limit checking and raises alarms
  - Triggers monitors (callbacks)
  - ‘Pulls’ or ‘pushes’ (or monitors) data via *Links*
    - Can link to records on other IOCs via Channel Access (*ie*, an IOC is also a *client*)

IOC: Components

- Record Types
  - I/O types
    - Analog IN/OUT (ADC, DAC)
    - Binary IN/OUT (TTL, relay, ...)
    - Long IN/OUT (Counter, Timer, Scalar)
    - Motor
    - String IN/OUT (TCP/IP, RS-232, GPIB, ...)
    - Waveform (Digitizer, camera, ...)
  - Secondary processing types
    - Calculation
    - Escape-to-C subroutine
    - Proportional-Integral-Derivative (feedback)
    - Transform

IOC: Components

- Record Types
  - Data Storage types
    - Compression
    - Histogram
    - SubArray
    - State
  - Control
    - Data Fanout
    - Control Fanout
    - Event
    - Select
    - Sequence
    - Scan
    - Wait

The Record Reference Manual and other documents provide details

IOC: Components

- Record Fields

A record's attributes are held in its *fields*. A field name is a 3- or 4-character abbreviation. Each field of a record (also called a *Process Variable*, or PV) is a *Channel.*

All records 'inherit' a core set of fields:

- `NAME`—the only way CA clients can find it
- `DESC`—a free-form description
- `VAL` —the quantity ('value') of interest
- `FLNK`—forward processing to another record

Other records have more specific fields:

- `DTYP`—device type for I/O records
- `INP` —input parameters for I/O records
- `OUT` —output parameters for I/O records

IOC: Components

- Device Support
  - Isolates Record Support from hardware details
    - New devices use can old records
    - Optionally uses Driver Support
  - ‘Soft’ types (available for most records) provide place-holders for simulation, client persistence, ‘global’ items, *etc*
- Driver Support
  - Used typically for non-trivial low-level bus I/O, wire protocols, *etc*
  - *Not EPICS-specific (but usually bus-specific and often OS-specific)*

A large repertoire of Device and Driver routines
are shared by the EPICS community

IOC: Components

- Scanners
  - These are the active threads that call the working records' code
  - Every record has a one of these scanning types specified:
    - Periodic
      - A (modifiable) selection of rates, typically 10 Hz to 1 minute
    - Event
      - One of 255 'soft' events, via other records or Channel Access
      - Via hardware interrupt (I/O complete)
        - Supports asynchronous I/O with appropriate Driver Support
    - Passive
      - Invoked by 'push', 'pull', or 'forward' link from another record
      - Invoked by Channel Access 'put'

Selecting the best scanning option for each record is the 'art' of configuring an EPICS database; and often many equivalent solutions exist

IOC: Components

- Channel Access (CA)
  - It is the only portal between external entities and the database
    - Even a co-resident Sequencer is a ‘pure’ CA client
  - It is fundamentally a publish/subscribe paradigm
  - Based on TCP/IP
    - TCP for data transport
    - UDP for connection management
  - Basic Operations
    - Search and Connect to a Channel*
    - Write to that Channel (‘put’)
    - Read from that Channel (‘get’)
    - Monitor (await callback from) that Channel
    - Disconnect from that Channel

**A Channel is defined as:*

“<record_name>.<field_name>”

IOC: Components

- Channel Access...
  - Read and Monitor return:
    - Value requested
    - Time-stamp
    - Status (read/write/access/undefined/...)
    - Severity (normal/minor/major/undefined)
  - Hard and Soft IOCs contain both client and server
  - OPI tools use only client

IOC: other

- Autosave
  - Implements ‘warm reboot’
    - Saves changed values (‘set-points’) back to server
    - Restores them after reboot
    - Developer supplies a list of PVs
- Console access
  - ‘Debug’ serial port on all IOCs wired to Terminal Server
    - Can watch start-up script
    - Can run CA, Sequencer, DB diagnostics
- VME Remote crate control
  - Control/Monitor power, voltages, temperatures
    - ‘Hard’ and ‘Soft’ reset when even Console access fails
- IOC self-monitoring
  - heart-beat, time-of-day, resource loading

Sequencer

Implements a true Finite-State Machine (FSM), with some Harel extensions. Source code is written in the State Notation Language (SNL) which is compiled into C by the EPICS build system. (Channel Access and Connection Management is part of SNL.) A Sequencer program is a collection of communicating ‘state sets’, each of which has states and transitions.

Whereas the Database is optimum for combinatorial solutions, the Sequencer is best for time-dependent solutions.

State transitions are triggered by any combination of: elapsed time; channel change; channel value; and software event.

SNL code is re-entrant and supports multiple instantiation with macro substitution. Inspection of running sequences is provided. It runs directly on an IOC and also on any OPI or SRV host.

```
State A{
       when( X ) {
          do Y;
       } State B;
       ...
}
...
```

Client Tools

- All of these allow drag-and-drop of PV names between them
- `caget` and `caput` from command line
  - Quick sanity check on all of the following...
- Probe
  - Single PV GUI-style diagnostic; handy monitor/adjust functions
- Extensible Display Manager (EDM)
  - Implements 'soft' control panels for typical devices
  - Drag-and-drop from palette of appropriate widgets
  - Only PV names required
  - Excellent macro substitution facility
  - Pre-built screens for all known devices
- StripTool
  - Multi-channel emulation of paper strip-chart recorder
- Alarm Handler (ALH)
  - Provides a hierarchy to drill down to 'first-fault'
  - Can invoke EDM screens, dial pagers, call processes, give help
  - Excellent macro substitution facility

Client Tools

- Channel Access is available as library or plug-in for
  - Matlab
  - C/C++
  - Python
  - Mathematica
  - Java
  - Perl
  - LabView
  - Unix/Linux shells