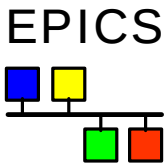
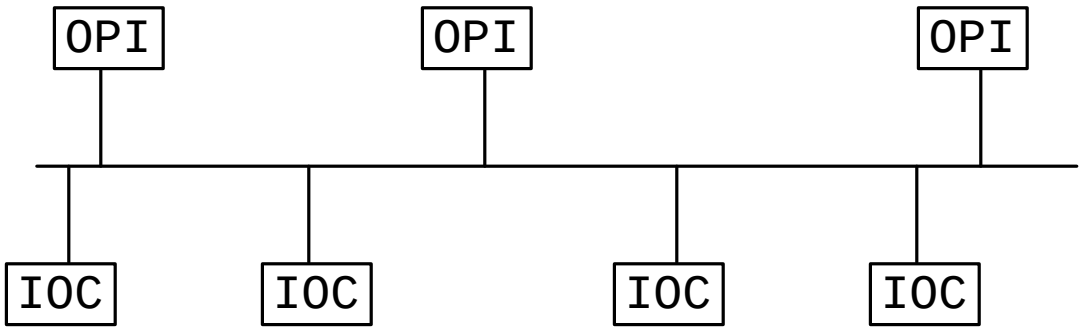


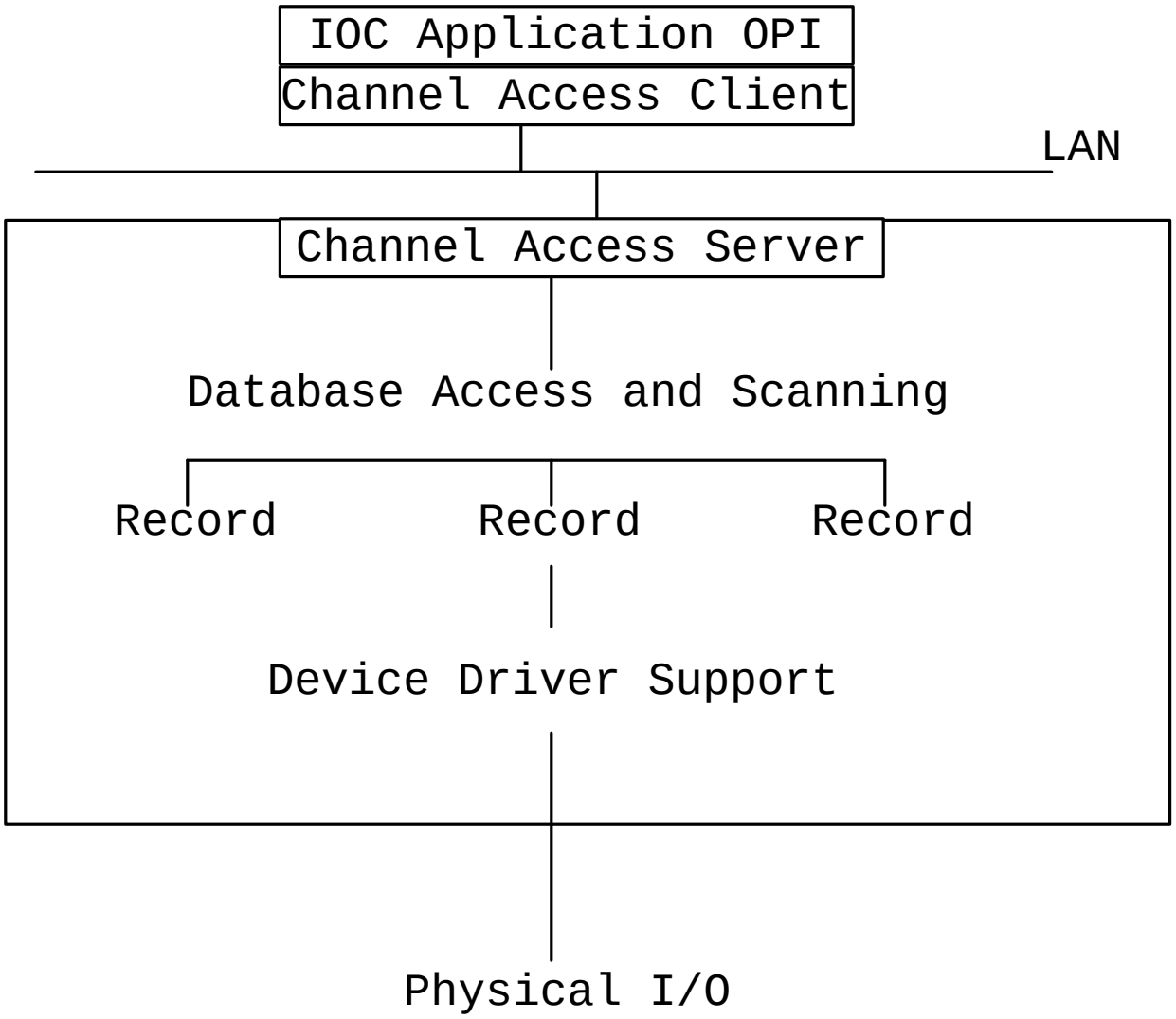

# *Flow of Control*

Marty Kraimer
APS

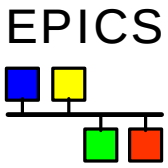
# *EPICS ARCHITECTURE PHYSICAL*



# *EPICS ARCHITECTURE LOGICAL*

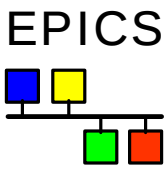


# *Summary*



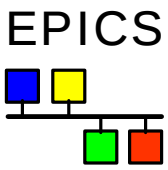

- IOC Initialization
  - Load various components
  - Connect to hardware
  - Start Channel Access Servers
- Record Processing
  - Hardware I/O monitor and control
  - Post events for CA clients
- Channel Access
  - Connect to channels
  - Get/Put requests
  - Monitor Requests
  - Connection Management

# *IOC Initialization*



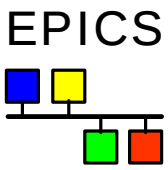

- `ld`
  - `iocCore`
  - `xxxApp` - device,driver, record support
- `dbLoadDatabase`
  Load database definition files for menus, recordtypes, devices, and drivers
- `dbLoadRecords, dbLoadTemplates`
  Load record instance files
- `iocInit`
  - Dynamically link to record, device, and driver support
  - Start general purpose tasks
    log server, task watchdog, callback, dbCa, etc
  - Initialize drivers
  - Initialize records (two passes) and devices
  - Start scan tasks
  - Start access security
  - Start Channel Access Servers

# *Record Processing: Common Part*



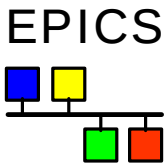

- Database Scanning calls dbProcess
  - Periodic
  - I/O Interrupt
  - Event
  - Passive
- Skip if already active
  - A lock set may have loops
  - The record may be asynchronous
- Skip if the record is disabled
- Call record support process routine
- Linked records (input, output, forward) that are process passive are processed recursively

# *Process: Synchronous I/O*



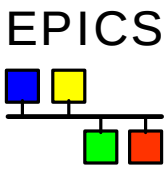

- Set PACT true (processing active)
- Perform I/O operation
  - For hardware link call device support
  - For database link
    - linked records (except scalar NPP NMS input) are in a common lock set.
    - recursive processing may happen
      - inputs processed before data transfer
      - outputs processed after data transfer
  - For channel access link
    - inputs are monitored
    - outputs are handled via a separate task
- Check for record specific alarm conditions
- Call db_post_event for any fields that change because of record processing
- Request processing of forward link
- Set PACT false
- return

# *Process: Asynchronous I/O*



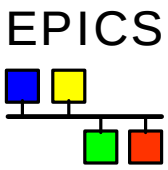

- Asynchronous Start (Called by dbProcess)
  - Set PACT True
  - Start I/O operation (normally via device support)
  - return leaving PACT true
- Asynchronous Completion
  Record processing routine is called again
  - Complete I/O operation
  - Check for record specific alarm conditions
  - Call db_post_event for any fields that change because of record processing
  - Request processing of forward link
  - Set PACT false
  - return

# *Channel Access*



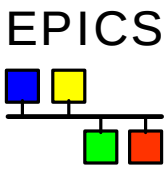

- Search
  - UDP broadcast PVNAME on local subnet (buffered)
  - Each IOC has a CA broadcast listener
  - IOC that has PV sends UDP message back to client
  - TCP connection between each client/server
  - For each client IOC establishes following tasks
    - Server task (all communication except monitors)
    - Event task which handles monitors, I.e. calls to db_post_event
- Get
  - Issue requests
  - Wait for replies
- Put
  - Issue put. No response except for errors
  - May cause record processing

# *Channel Access continued*



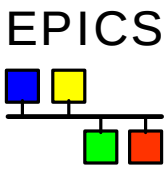

- Add Event
  - Ask for monitor on PV (record.field)
  - Monitors happen because of calls to db_post_event
- Put Notify
  - Issue put and wait for completion
  - Completion when all records complete processing
- Connection Management
  Automatic reconnect
- Access Security

# *Example CA Put Notify*



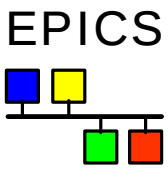

- Client issues put notify and flush
- IOC CA server accepts request
- dbPutNotify is called
- dbScanLock is called - Entire lock set is locked
- Value is written to record
- Record is processed (if passive and PP true in dbd file)
  A set of records may start and some complete processing
- dbScanUnlock is called
- When all records complete processing the CA server is notified
- The CA server sends a message back to the client

# *Example Asynchronous Devices*




- Serial and GPIB Devices
  - Are typically slow
  - Device support can NOT be synchronous
- Solution
  - Create separate task to manage requests
  - Device support is asynchronous
    - Async start queues a request to task
    - When task completes request it notifys device support which causes asynchronous completion.
  - Generic GPIB support is available
  - Several Serial support options are available
- Now lets look at CA Put Notify example
  - Database has series of linked records reading gpib / serial values.
    - rec1 -> rec2 -> rec3->…
  - What does CA Put Notify do?

# *CA put notify*




- The following is sequence of events
  - Client issues put notify request
  - Server receives request
  - dbPutNotify is called
    - dbScanLock is called
    - Value is written to record rec1
    - rec1 starts processing
      - Device support queuesrequest to separate task
      - Record left with PACT true
    - dbscanUnlock is called
    - dbPutNotify waits
  - Sometime later
    - Task finishes request
    - rec1 has asynchronous completion
    - rec2 is processed and starts asynchronous phase
    - … until all records in chain complete
  - dbPutNotify completes
  - CA sends message back to client