The TINE Control System (an overview)

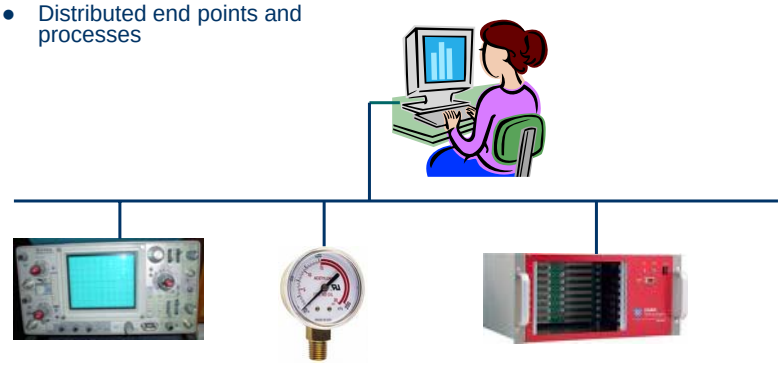
- Three-fold Integrated Networking Environment
- (This Is Not Epics)

Control Systems have …

- **Distributed** end points and processes
- **Central Services** (Archive, Alarm, Name Resolution, …)
- **Security** (who's allowed to do what from where?)
- **States** (Finite State Machines, Sequencing, Automation…)
- **Time synchronization** (time stamps, cycle ids, etc.)
- **Databases** (configuration, machine data, post-mortem data, …)
- **Statistics** (control system itself, operation, …)
- **Logging** (central, local, application, …)

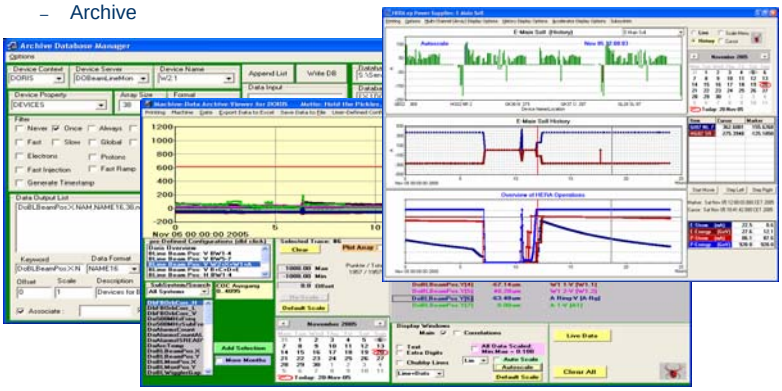
Control Systems have …

- Distributed end points and processes



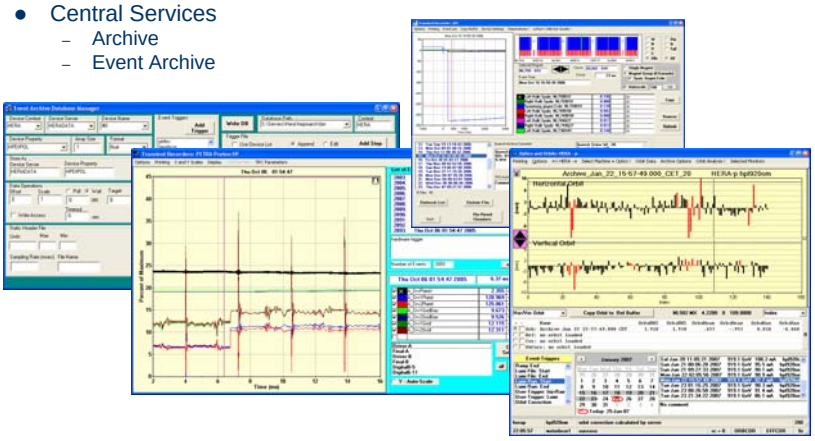
Control Systems have …

- Central Services
  - Archive



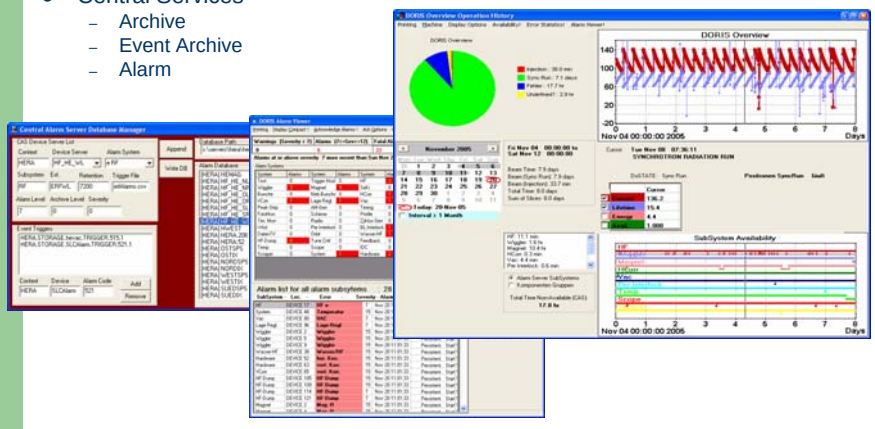
Control Systems have …

- Central Services
  - Archive
  - Event Archive



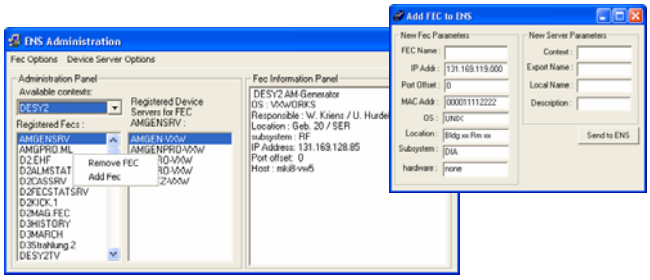
Control Systems have …

- Central Services
  - Archive
  - Event Archive
  - Alarm



Control Systems have …

- Central Services
  - Archive
  - Event Archive
  - Alarm
  - Name Resolution

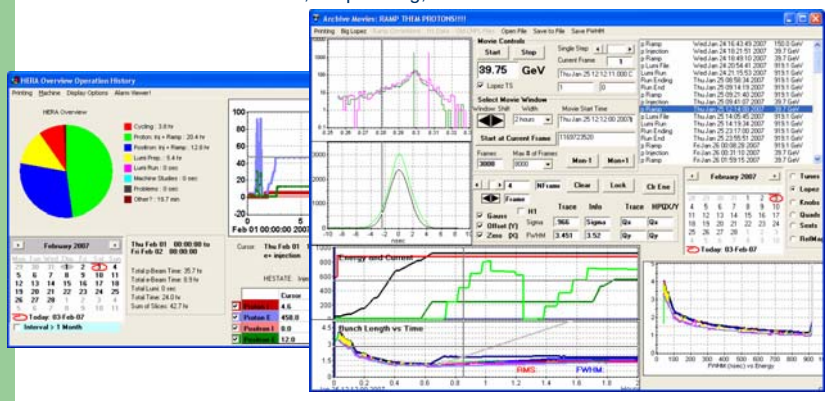


Control Systems have …

- Security
  - who's allowed to do what from where?
  - Username
    - Weak (easy to impersonate) but usually gets the job done
  - Network Address
    - Strong (much harder to impersonate)
  - Access Lock Token
    - Just the process that has the lock !

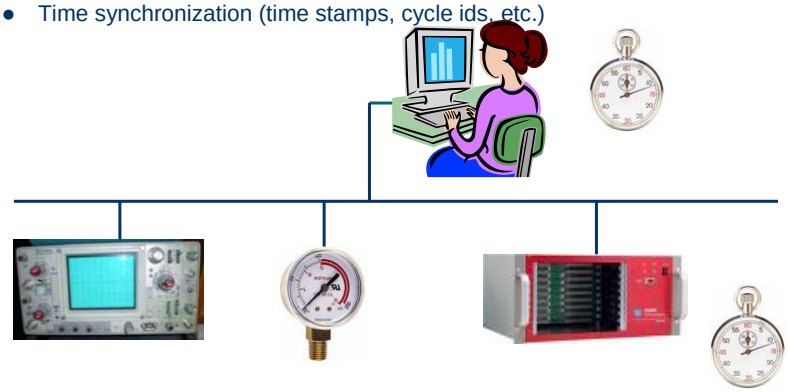
Control Systems have …

- States
  - Finite State Machines, Sequencing, Automation…



Control Systems have …

- Time synchronization (time stamps, cycle ids, etc.)



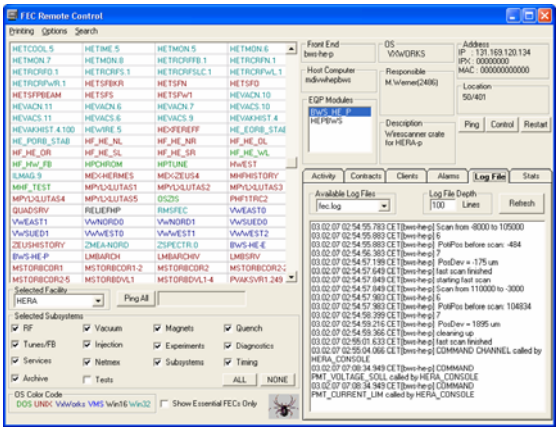
current Data timestamp: Sat Feb 3 16:50:45 2007 666 msec (UTC: 1170517845.666)

Control Systems have …

- Statistics

Control Systems have …

- Logging



TINE : multi-platform

- DOS
- Win16, Win32 (9x, NT, 2K, XP, …), Win64 ?
- Unix (Solaris, HP, OSF, SGI, Ultrix, …)
- Linux, FreeBSD (32 bit, 64 bit)
- MAC OS X
- VxWorks
- VMS (Vax, Alpha)
- LynxOS
- NIOS (plugs, single-threaded LWIP, …)
- Java

TINE: multi-protocol

- IPX (dead and no one has noticed?)
- UDP (most common)
- TCP/IP (upon request)
- Pipes (client-server on same UNIX machine)
- Windows Messages (client-server on same Windows machine)

TINE: multi-architecture

- Client-Server (classic)
  - Transaction based
  - Synchronous data access only
  - The “N-Client” Problem ?
  - ExecLink(“/HERA/BPM/WL167MX”,”ORBIT.X”, …)
- Publisher-Subscriber (nearly classic)
  - Connection Tables
  - Synchronous/Asynchronous data access
  - The “10N-Client” Problem ?
  - AttachLink(…, CM_REFRESH, 1000, linkCb)

TINE: multi-architecture

- Producer-Consumer
  - Asynchronous data messages (Multicast)
  - The “N-Producer” Problem ?
  - recvNetGlobal(“HPMAGEN”) or
  - AttachLink(“HPMAGEN”,…,CM_RECEIVE, 1000, linkCb)
- Publisher-Consumer (Producer-Subscriber?)
  - Like Publisher-Subscriber but:
  - Multicast group is a single connection Table entry
  - N = 1 !!!
  - AttachLink(…,CM_REFRESH|CM_NETWORK,1000, linkCb)

Client-Server Gedanken Experiment

What time is it?

It's 10:30 a.m.

Synchronous!

Client-Server Gedanken Experiment

What time is it?






+ . . .

It's 10:31 a.m.

Synchronous!

The "N-Client" problem

Publish-Subscribe Gedanken Experiment

Please put me on your “What time is it?” list!

+ . . .

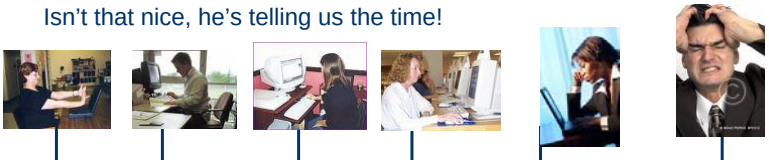
It’s 10:32 a.m.

Asynchronous (much better)!

Producer-Consumer Gedanken Experiment

Shut up!

Isn’t that nice, he’s telling us the time!



+ . . .

It’s 10:33 a.m.

Asynchronous!

Producer-Consumer Gedanken Experiment

Did somebody just mention the time ?

The “N-Producer problem

Asynchronous!

It’s 10:33 a.m.

It’s 7 degrees

It’s still raining

The DAX is up 2%

The wind is South-West

+ . . .

Publisher-Consumer Gedanken Experiment

I want to receive the “What time is it?” notification!

+ . . .

It’s 10:33 a.m.

Scheduled asynchronous notification (best)!

TINE Servers
(Publishers, Producers)

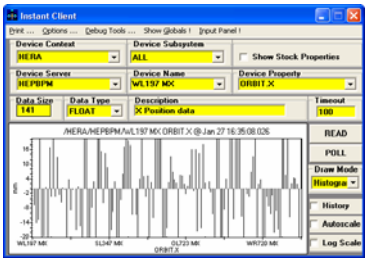
- A Front End Controller (FEC) :
  - Has one or more **Equipment Modules** (device servers)
    - *Handles requests* for data and commands
    - Has a *well-known identity*
    - Has a unique *export name* for each running instance
    - Is *always running*
    - Can be a member of a device *Group*

TINE Clients
(Subscribers, Consumers)

- *anonymous* -> i.e. Not known to the system
- *Not unique*
- Can *come and go*

TINE Naming Convention

- Hierarchical
- Device is specified by :
  - **Device Context** (*Facility* in DOOCS)
  - **Device Server** (or **Group**) (*Device* in DOOCS)
  - **Device Name** (*Location* in DOOCS)
- Data endpoint is specified by:
  - **Property** (*Property* in DOOCS)
    - Are really 'methods' or 'calls'



TINE ENS: Plug and Play

I want to be known to the system as "BPM.P" in the context "HERA"

Does BPM.P already exist for HERA?

-> **Yes** : Is same address as already registered?

-> **Yes**: **Accept** and increment reboot count

-> **No**:

Is the currently listed BPM.P for HERA running ?

Yes : **Refuse** and send "in-use" message

No : **Accept** and update database

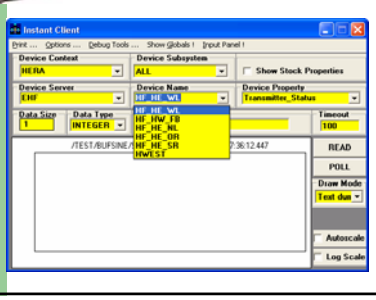
-> **No** : **Accept** and update database

Equipment Name Server (ENS)

• Forward accepted requests to secondary name servers

Plug and Play (joining a group)

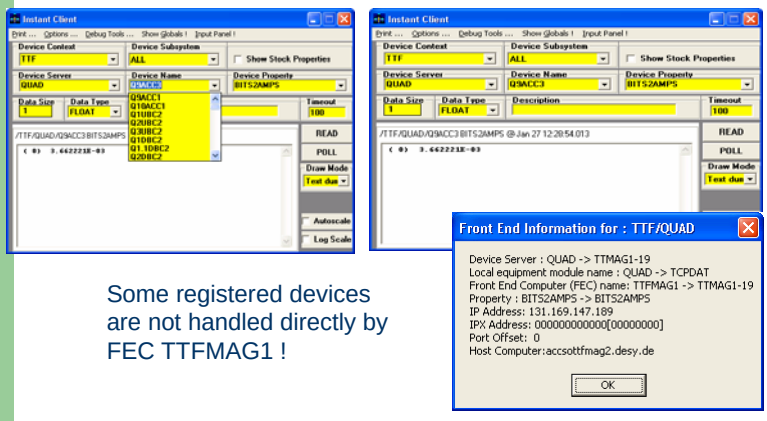
I want to be known to the system as “ERF.WL” in the context “HERA” and join the group “EHF”



- ERF.WL registers with ENS as before
- ERF.WL registers group EHF with GENS
  - Does group EHF exist ?
    - **Yes**:
      - Is ERF.WL a member?
        - **Yes**: Update device list if different
        - **No**: join group
    - **No**:
      - Create Group and register Group as Server in Context HERA with the ENS

- Clients see a “Server” called “EHF”
- Selected Device is redirected to the appropriate physical server.

Address Redirection



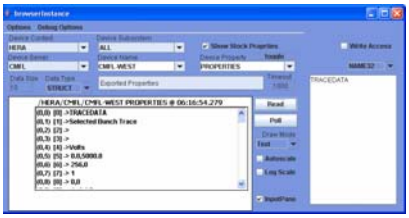
Some registered devices are not handled directly by FEC TTFMAG1 !

TINE Data Types

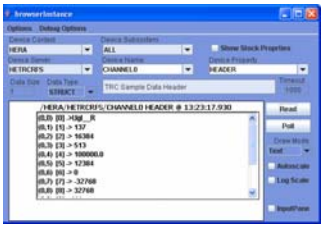
- Primitives (byte, short, int*, long*, float, double, char*)
- Fixed-length Strings (“NAME8”, “NAME16”, “NAME32”, …)
- Doublets (FLTINT, “INTINT”, “DBLDBL”, “NAME32INT”, …)
- Triplets (“FLTINTINT”, “NAME16FLTINT”, …)
- Quadruplets (“FLTINTINTINT”, “INTINTINTINT”, …)
- Specials (“USTRING”, “UNAME”, SPECTRUM, …)
- **User-defined “Tagged Structures”**
  - Structure registered at both ends (client, server)

*Platform dependent

TINE Tagged Structures

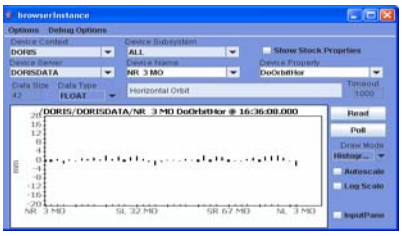


Some Stock Properties use Tagged Structures.

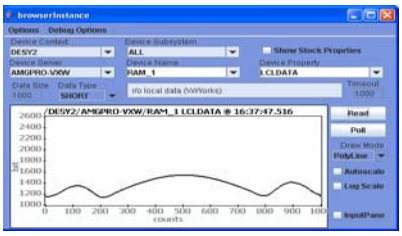


Transient Recorder Servers use Tagged structures for headers.

TINE Array Type Examples



- "DoOrbitHor" gives multi-channel Array
- Preferred display mode : Histogram
- x-Axis : monitor names

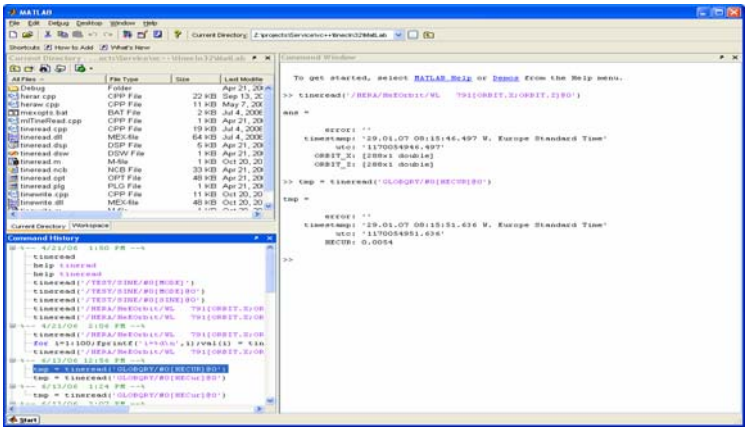


- "LCLDATA" gives spectrum array
- Preferred display mode : PolyLine
- x-Axis : units, range from property query !

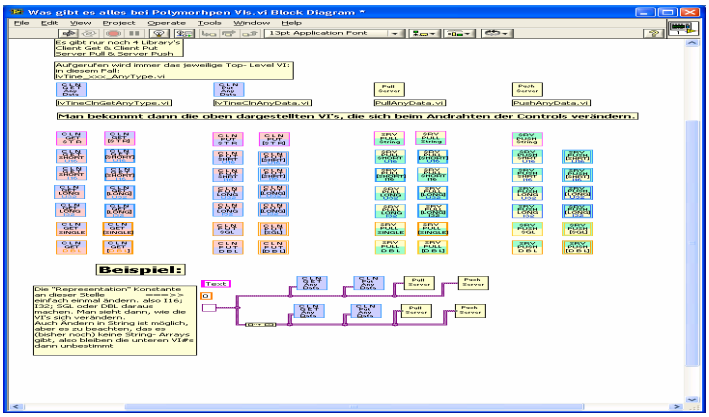
TINE API (Application Programmer's Interface)

- C, C++
- VB
- Java
- C# (coming soon)
- Lazarus (Visual Pascal) anybody interested?
- Command line scripts
- Plus …

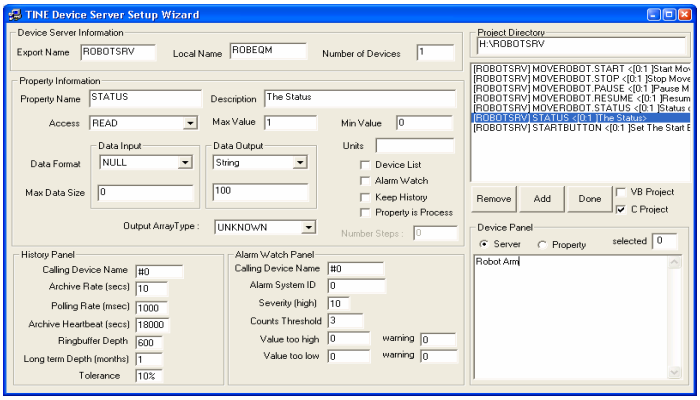
TINE and MatLab …



TINE and LabView API



TINE Server Wizard

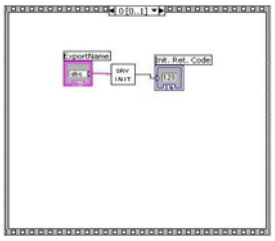


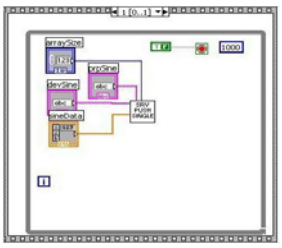
TINE Server Wizard
(generated C code)

```
short g_MOVEROBOTProcessLevel;
short g_MOVEROBOTNumProcessSteps = 1;

void robeqm_bkg(void)
{
  int i;
  /* TODO  put your IO or other background activity here          */

  /* Clear alarms at start of IO task (see if they come back)     */
  ClearAlarm(ROBEQM_TAG,-1);

  /* TODO  replace SimulateStringData() with your own data acquisition */
  /*       and manipulation routines                              */
  strncpy(g_statusBuffer.SimulateStringData(PRP_STATUS_SIZE).PRP_STATUS_SIZE);

  /* call process function  */
  MOVEROBOTProcessFunction();

  /* make use of SetAlarm() as needed :

  SetAlarm(ROBEQM_TAG.devNr,almCode,almFlag);

  */
}
int MOVEROBOTProcessFunction(void)
{
  static int duration = 0;
  static time_t start = 0;
  time_t interim = time(NULL);

  if (g_MOVEROBOTIdleCondition) goto MOVEROBOTProcessFunctionExit;
  /* TODO:  fill in the activities to be performed at each step     */
  /*        The following example code simply waits for 3 seconds at */
  /*        each process level.  You may want to start IO readouts  */
  /*        at one level and poll for a ready flag in another, etc.  */
  if (g_MOVEROBOTProcessLevel > 0 && duration == 0)
  {
    start = interim;
    duration = 3;
  }
  switch (g_MOVEROBOTProcessLevel)
  {
    case 1:
    if (interim < start + duration) return 0;
    duration = 0;
      break;
    default:
      g_MOVEROBOTProcessLevel = 0;
      break;
  }
```

TINE Server Wizard
(using with LabView)

- Make use of generated database files + 'buffered server API' for LabView.





TINE Device Layer

- Middle layer Servers acquire data from other Servers (not connected to hardware)
- "Do it yourself" + your hardware API
- EPICS IOCs (asyn drivers) + Epics2Tine
- LabView IVIs + TINE LabView
- DOOCS + (turn on that TINE thread!)

- **CDI** (Common Device Interface) !!!

CDI Examples

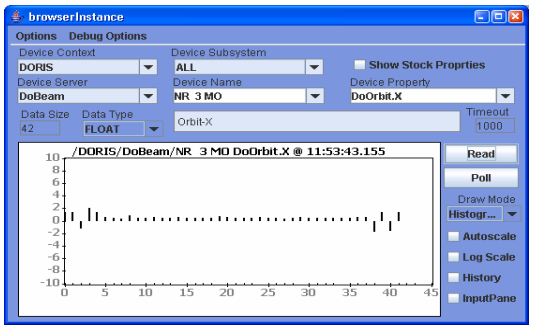
```
dout.dArrayLength = 1;
dout.dFormat = CF_UINT16;
dout.data.sptr = rbPressData;
AttachLink("/localhost/cdi/pump1", "RECV.CLBR", &dout, NULL,1000, cbPressData)
```

Reads device pump1, calibrates the raw data, fills in rbPressData and calls the callback cbPressData() at 1 Hz.

```
din.dArrayLength = 1;
din.dFormat = CF_UINT16;
din.data.sptr = &setValue;
dout.dArrayLength = 1;
dout.dFormat = CF_UINT16;
dout.data.sptr = &rbValue;
ExecLink ("/localhost/cdi/motor1.soll", "SEND.RECV.ATOM", &dout, &din,1000)
```

Sends 'setValue' to device #1 and reads rbValue from device #1 atomically.

Testing your server …

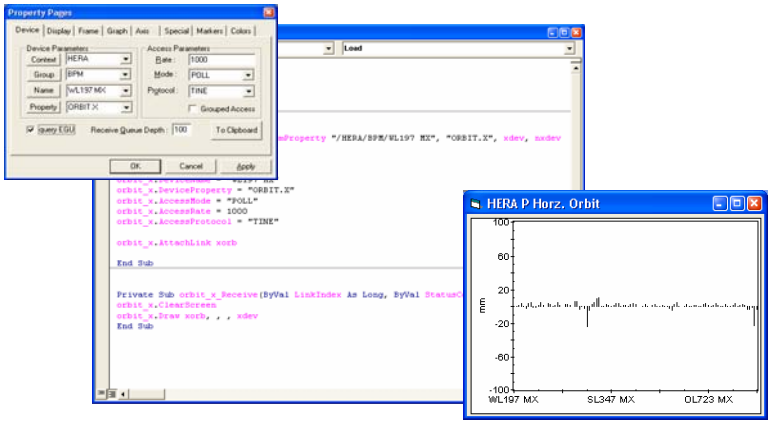


Writing GUI Applications (rich clients)

- Use what you've got + TINE Client API for your platform, or …
- VC++ or VB plus ACOP* ActiveX control (HPVee, LabView), or direct TINE library calls
- Java + ACOP* bean (eclipse, net beans), or direct TINE calls
- LabView + TINE client VIs
- MatLab + TINE Client API
- .NET is on the way …

*Advanced Component Oriented Programming

Rich client programming in VB

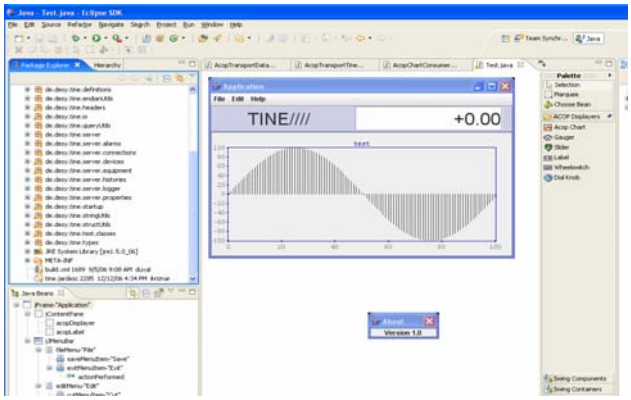


Configuring GUI Clients (simple clients)

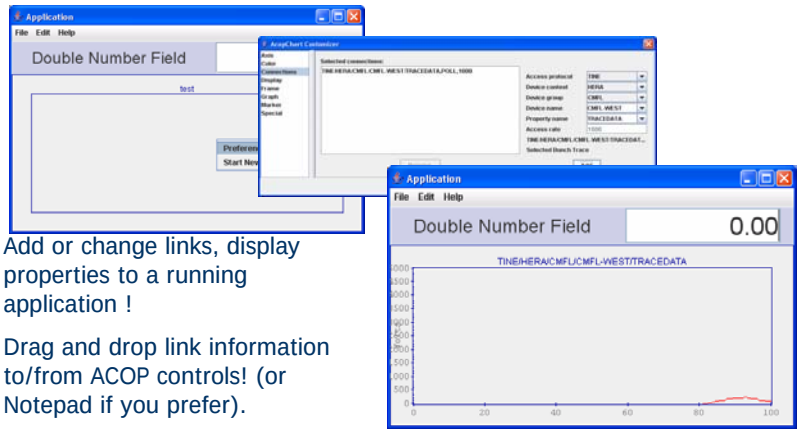
- JoiMint + TINE
- DOOCS DDD + TINE
- ACOP Family of beans + TINE *

*More later ….

ACOP beans …



ACOP Simple clients …



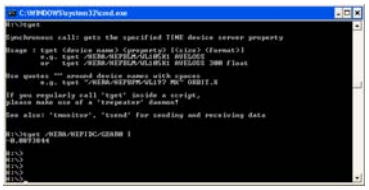
Add or change links, display properties to a running application !

Drag and drop link information to/from ACOP controls! (or Notepad if you prefer).

TINE command-line tools

- tservers (queries the ENS for registered servers)
- tdevlist (queries a server for its devices)
- tproplist (queries a server for its properties)
- tinfo (queries a server for property information)
- tget (synchronous read-only call to server)
- tsend (synchronous write/read call to server)
- tmonitor (asynchronous read-only poll to server)
- thistory (queries the archive server)

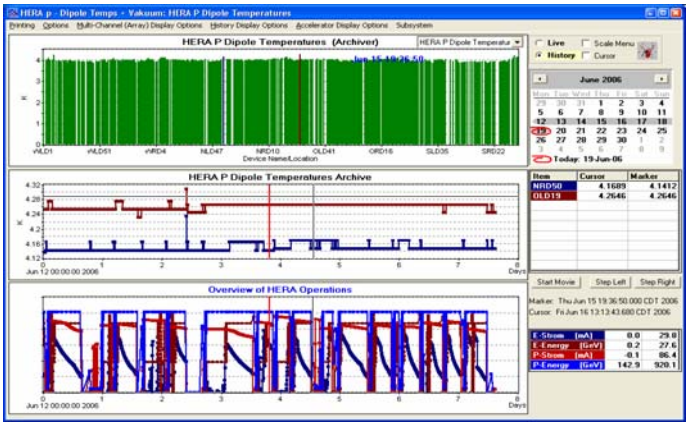
With ‘tget’ in tcl scripts
PLEASE use a tineRepeater !



Connectivity to other systems

- Already embedded in DOOCS
- Epics2Tine runs on any EPICS ioc
- Connect to STARS/COACK via STARS bridge (Japan)
- Anything else needs a gateway

EPICS to TINE @ HERA



TINE – Adoption at PITZ

Talk Overview

- Introduction to PITZ
- PITZ control system
  - Past
  - Introduction
  - Historical layout
- TINE at PITZ
  - Introducing
  - Advantages
  - Use cases
  - PITZ Video System
  - History and Outlook

TINE – Adoption at PITZ

Introduction to PITZ

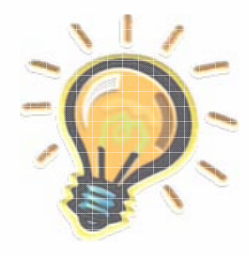
- **P**hoto **I**njector
  Test Facility **Z**euthen
  - test, condition and optimize sources of high brightness electron beams for future free electron lasers and linear colliders
  - goal: intense electron-beam with very small transverse emittance and reasonably small longitudinal emittance
  - goal is requirement for FEL operation

"The challenge of PITZ is the production of such beams with very high quality by applying the most advanced techniques in combination with key parameters of projects based on TESLA technology like the FLASH, the European XFEL, and the proposed BESSY-FEL."

TINE – Adoption at PITZ

Terminology and Theory

- synchronous vs. asynchronous calls
- polling vs. event notification
- multicast vs. unicast
- property model
- DOOCS
- FEC



TINE – Adoption at PITZ

PITZ Control System: Past

- PITZ operation started in January 2002 (first applied machine physics in Zeuthen for a long time)
- got help from MVP control system group in Hamburg to install a control system at Zeuthen: DOOCS architecture
  - based on control system from TTF
- collaboration with other institutes (BESSY, MBI) requires integration of other components and control system architectures into PITZ control syste
  - challenge for Control System group and IT services

TINE – Adoption at PITZ

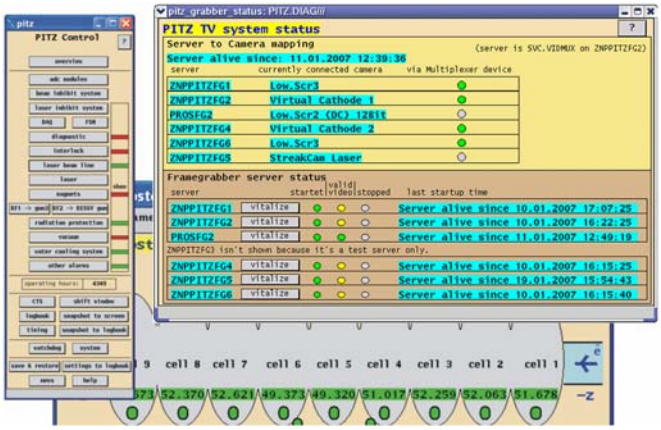
Introduction to PITZ Control System

- Distributed Control System (see en.wikipedia.org)
  - power plants, airplanes, industrial facilities, traffic signals, environmental control, space ship, TV studio, …
  - co-operation of operator terminals, network wires, electronics, diagnostic sensors, control hardware, intermediate machines
  - control, detect, read-out, set, switch, dim, check, block, open, close, move, rotate, …

http://en.wikipedia.org/wiki/Distributed_Control_System

TINE – Adoption at PITZ

Introduction to PITZ Control System



Technical Seminar February 6, 2007 Stefan Weiße, Philip Duval

TINE – Adoption at PITZ

PITZ Distributed Control System Structure


Klystron hall


Rack room

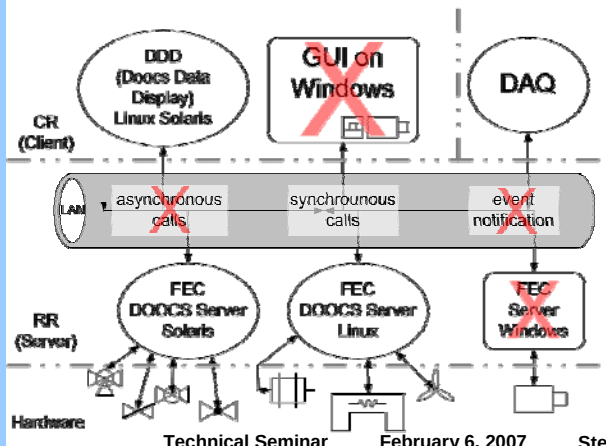

PITZ tunnel

Laser room


Control room

TINE – Adoption at PITZ

PITZ Control System: Historical Layout

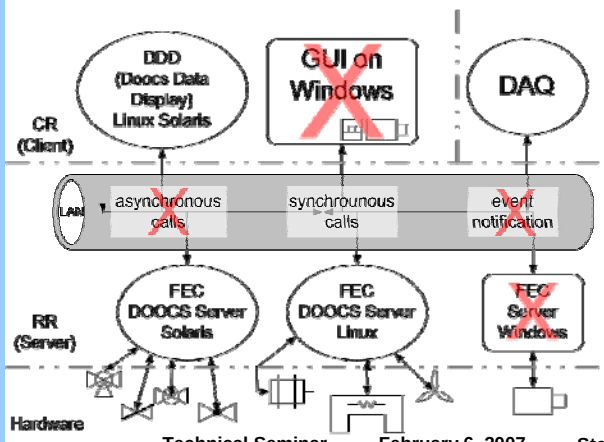


DOOCS solution (homogeneous)

- DDD as GUI
- DOOCS on front end servers (FEC)
- synchronous RPC transport
- DOOCS-based DAQ

TINE – Adoption at PITZ

PITZ Control System: Historical Layout



Drawbacks

- Solaris+Linux only, support on other platforms difficult
- OS interoperability defeated
- depending on DOOCS
- only **synchronous** RPC calls

TINE – Adoption at PITZ

Introducing TINE at PITZ

- design and implementation of PITZ Video System originated 2002
- former PITZ laser beamline control
- gateway-server to control magnets of PITZ (by BESSY, controlled by EPICS control system)

TINE – Adoption at PITZ

Advantages of using TINE at PITZ

- multicasting possiblity
  - fundamental for huge data streams distributed across the network (video frames)
- asynchronous protocol features: e.g. subscription, scheduled transfer
- interoperability and free platform choice
  - supported and maintaned on DOS, HPUX, JAVA, Labview, Linux, Matlab, MacOS X, Solaris, VMS, VXWorks, Windows (to be continued…)
- transparent design

TINE – Adoption at PITZ

Use Cases

- enable TINE thread in any DOOCS server
  - increase versatility
  - ability to talk to DOOCS from e.g. Microsoft OS world
  - benefit instantaneously of asynchronous transfers
  - ability to avoid synchronous calls if reasonable or necessary
  - at present: TINE thread is integrated into any DOOCS server as second transport choice

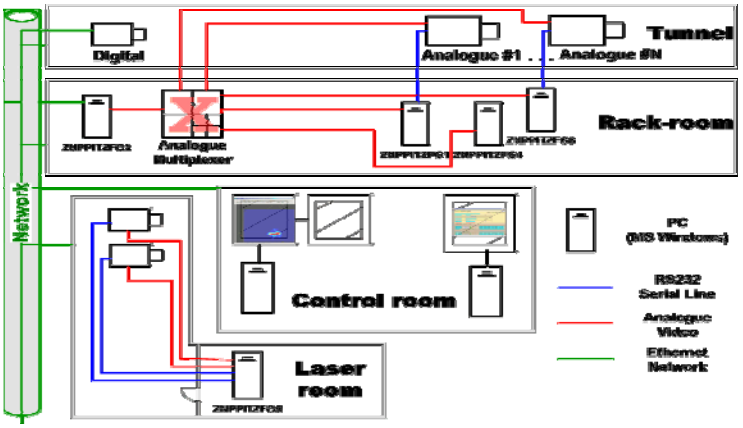
TINE – Adoption at PITZ

Use Cases (continued)

- PITZ Video System
  - used to live monitor laser and electron beam
    - what you steer is what you get
    - *„it's like watching TV for physicists"*
  - tool to measure beam size, emittance (slice/transverse), momentum (distribution, spread)
  - beam alignment is done with it
  - intuitive, versatile part of core diagnostics and measurement tools
  - **high bandwidth**, **realtime** and **loss-less** demands are heavy tasks for any control system

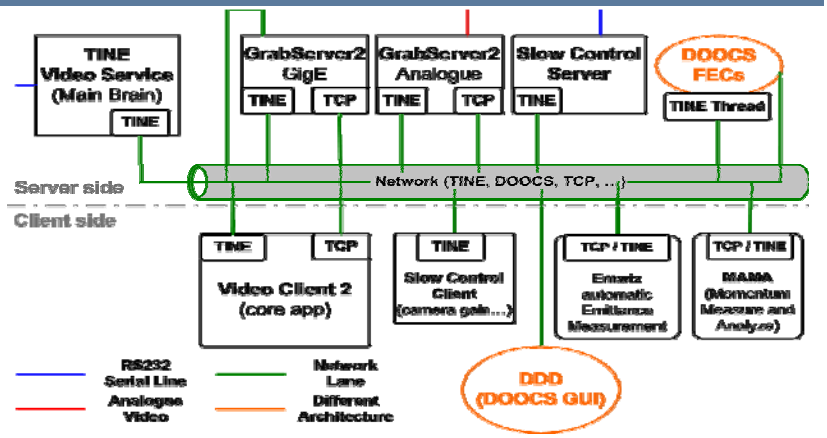
TINE – Adoption at PITZ

PITZ Video System: Hardware Structure



TINE – Adoption at PITZ

PITZ Video System: Software Structure



TINE – Adoption at PITZ

PITZ Video System: Outline Use

- **Online-DAQ inside Video Client 2**
  - acquire relevant experiment parameters at each image shot time (requesting up to 40 single properties up to 10 times per second)
  - asynchronous requests important
- **Video Frame Delivery**
  - TINE protocol based: transparent choice between multicasting, unicasting, shared memory, …
- **Video Control Connections**
  - slow control for setting camera parameters
  - communication between Video Service, clients and various grabber servers
  - compatibility to DOOCS clients maintained by property design

TINE – Adoption at PITZ

History and Outlook

- Introduction to PITZ
- PITZ control system
  - Past
  - Introduction
  - Historical layout
- TINE at PITZ
  - Introducing
  - Advantages
  - Use cases
  - PITZ Video System
  - **History and Outlook**

TINE – Adoption at PITZ

PITZ TV System History

- PITZ Video System
  - originated 2002, initial revision created on Windows platform
    - API and drivers for hardware available only for Windows
    - client uses MS Windows NT OS for fast drawing capability
      - Solaris tests at that time far inferior
    - DOOCS protocol not available on Windows NT platform (2002)
    - TINE protocol was initally chosen to support control connections and interoperability to Control System at PITZ
    - initially streaming sockets (TCP) to deliver video frames to multiple clients (as many as the network is capable of)
  - work partnership of TINE ← → Video System to push things further and react on demands
  - camera slow control
    - uses TINE to communicate between server, client and DOOCS
    - Windows GUI client, adjustment can also be done via DOOCS

TINE – Adoption at PITZ

PITZ TV System Evolution

- implemented TINE frame transfer (multicasting preferred)
  - keep TCP as inferior fallback solution
    (will feed redundant data to network though)
- implemented Online DAQ functionality
  - record experiment parameters together with video frame analysis to textfile for each frame
  - rough DOOCS library available in 2003, lead to instabilities based on synchronous calls
  - as TINE thread was possible, all functionality was moved to TINE transport
    -> now working fine
- used notification mechanism
  - event-based delivery of video frames, no polling any more!
- kept TINE servers compatible to DOOCS clients, e.g. parts of the TV system is available in DDD (GUI) world

TINE – Adoption at PITZ

Outlook

- **transport layer for Video System v3 (initiated at PITZ)**
  - v3: component based, well-defined specification of interoperability
  - requirement of video data exchange protocol between different parts
  - transparent transport option choice (multicast, shared memory, …)
  - drastically enhance versatility, reuse and exchange software between different Video Systems used at various accelerators and facilities
  - TINE is proven to be fit for this purpose and passed all tests in the past
- **TINE Archive Server and Post Mortem analysis**
  - event-based dump gives experts detailed data at certain crash events
  - track down and debug behaviour (hardware and software) of control system and accelerator facility issues
  - ability to browse online history of facility in near-realtime (DAQ is not capable of)

TINE – Adoption at PITZ

Outlook (2)

- **continue enabling TINE thread on any DOOCS server**
  - to have full-choice of interoperability
  - ability to interconnect with TINE Archive Server and Post Mortem Archiving System
- **brief idea to restructure parts of DAQ system**
  - enhance stability and minimize maintenance time
  - thoughts to use TINE as transport layer to asynchronously collect property data throughout the facility

TINE – Adoption at PITZ

Thank you for listening.

Questions? Comments? Additions?

Technical Seminar February 6, 2007 Stefan Weiße, Philip Duval