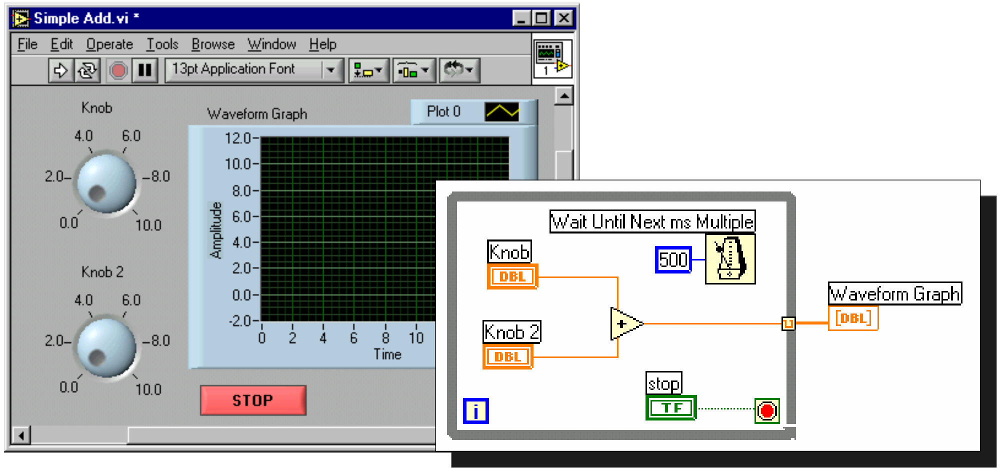
Making LabVIEW look like an IOC

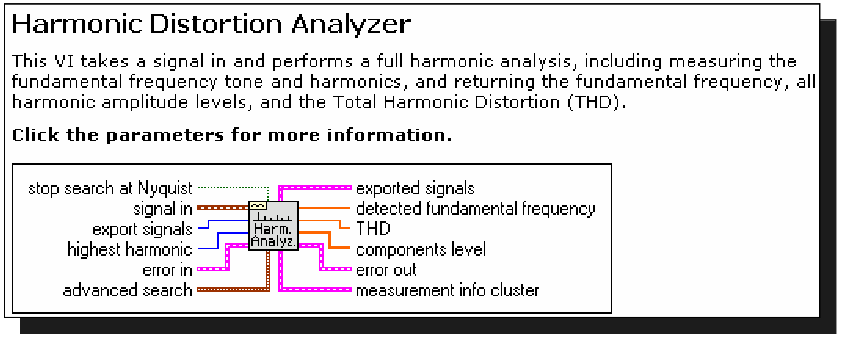
Kay-Uwe Kasemir, LANL

May 2002

National Instruments’ LabVIEW

- Integrated Graphical Programming Language and Operator Interface
  - Instructions and “Virtual Instruments” (Vis):
    Add, Wait, Loop, Open device, Read value, …
  - Many GUI elements: Knobs, Graphs, …
  - Supports huge number of GPIB, VXI, Serial as well as NI specific (PXI, FieldPoint) instruments
  - Library for Signal Analysis, …
- Widespread use for Test & Measurement



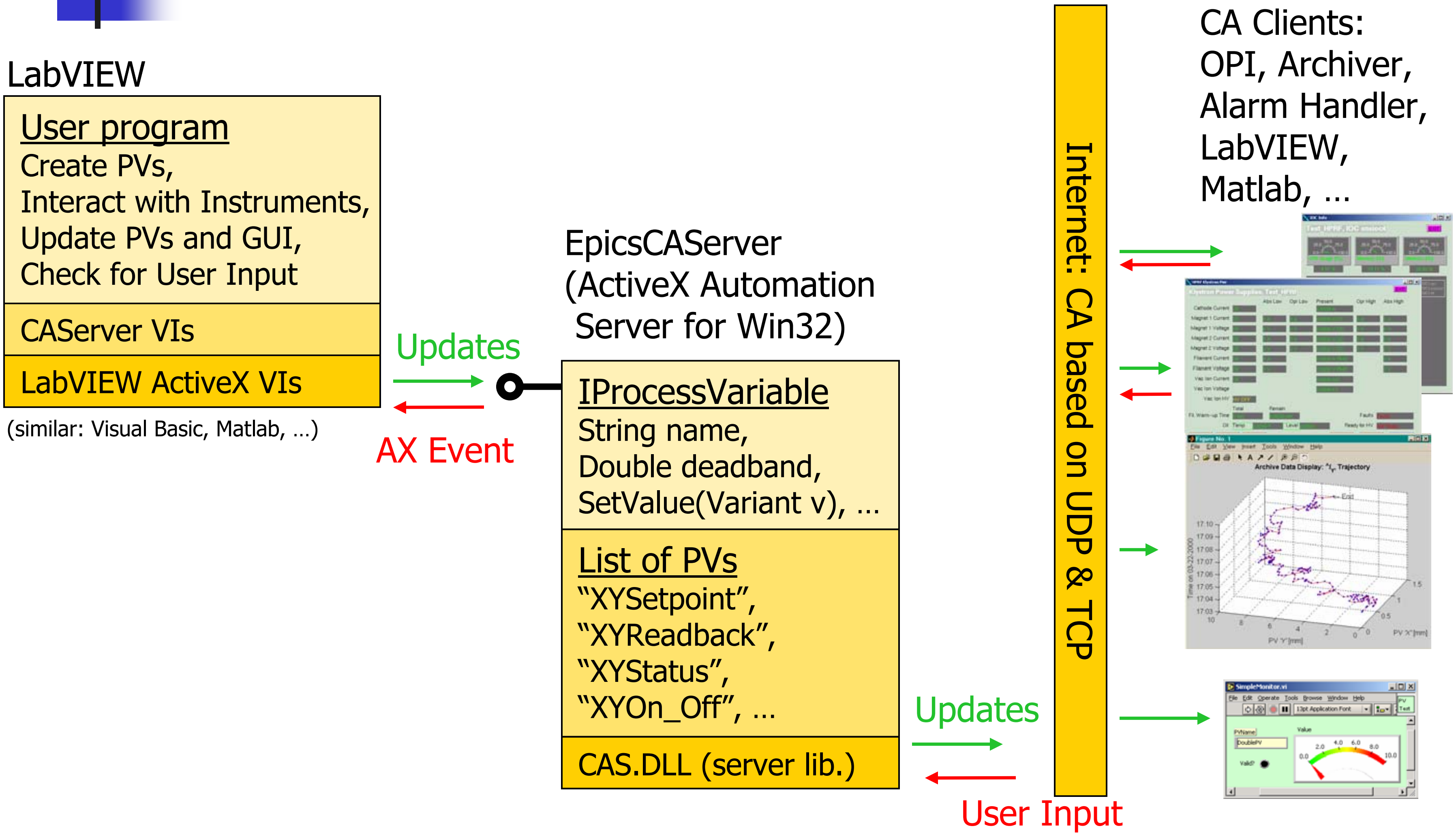


Harmonic Distortion Analyzer

This VI takes a signal in and performs a full harmonic analysis, including measuring the fundamental frequency tone and harmonics, and returning the fundamental frequency, all harmonic amplitude levels, and the Total Harmonic Distortion (THD).

Click the parameters for more information.

Integration via ChannelAccess, based on ActiveX



LabVIEW

User program
Create PVs,
Interact with Instruments,
Update PVs and GUI,
Check for User Input

CAServer VIs

LabVIEW ActiveX VIs

(similar: Visual Basic, Matlab, …)

Updates

AX Event

EpicsCAServer
(ActiveX Automation
Server for Win32)

IProcessVariable
String name,
Double deadband,
SetValue(Variant v), …

List of PVs
"XYSetpoint",
"XYReadback",
"XYStatus",
"XYOn_Off", …

CAS.DLL (server lib.)

Updates

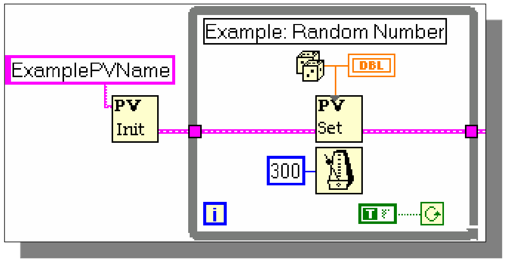
User Input

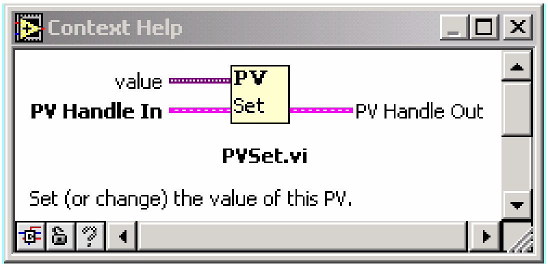
Internet: CA based on UDP & TCP

CA Clients:
OPI, Archiver,
Alarm Handler,
LabVIEW,
Matlab, …

Example LabVIEW Code

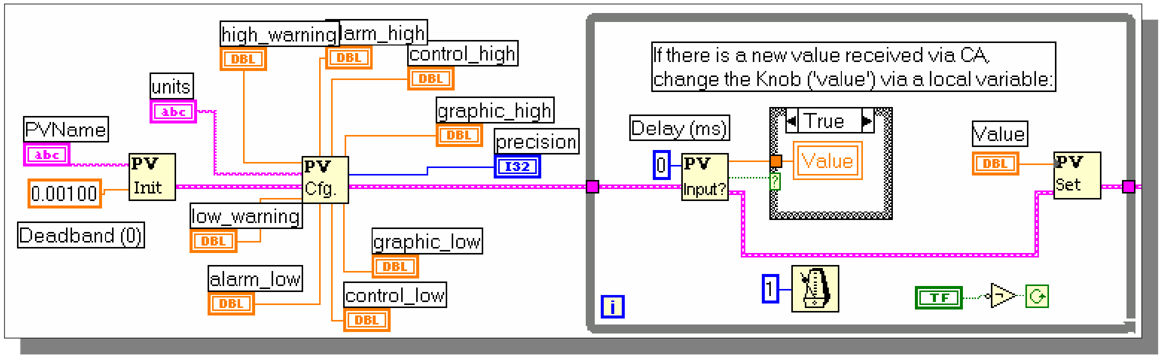
- 101: Publish PV and updates





Set (or change) the value of this PV.

- 102: Deadband, Config. Info, Reaction to remote input



If there is a new value received via CA, change the Knob ('value') via a local variable:

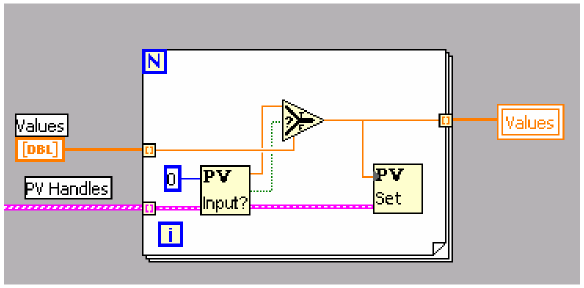
Performance

- COM call to update value of a PV

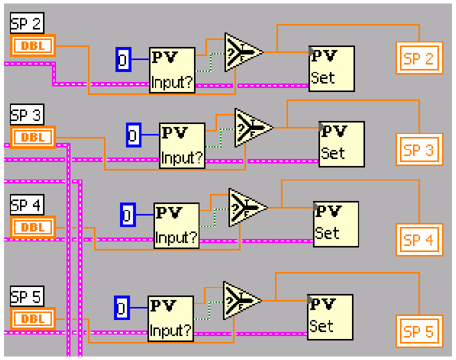
Data Served	LabVIEW	Visual Basic
Double	0.14 ms	0.08 ms
Double[100]	0.20 ms	0.16 ms
Double[500]	0.45 ms	0.40 ms
Double[1000]	0.75 ms	0.77 ms

(LabVIEW 7.I, 900MHz PC)

- Reaction to user input (Check AX Event, post new value)
  - 2 COM transactions, ~0.28ms
- LabVIEW Implementations: 10 PVs, handle user input
  - Loop: 7ms



  - Parallel: 0.5ms



LANL Experience

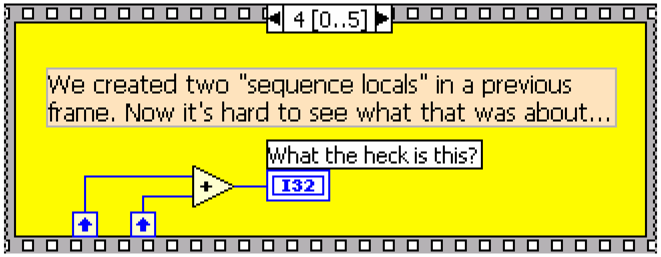
- Terrific for small systems
  - Signal generator, GPIB, new PC: handled in one afternoon
  - No need for vxWorks, IOC, boot host, display computer
  - Compare: 3 days for EPICS IOC, GPIB Lan Box, EDM.
    (Not considered: Advantages of EPICS driver & DB and Lan box)
- Faster initial development cycle
  - No recompile & reboot (until turned into distributed system)
- Handled up to ~500 PVs on one PC

LabVIEW: Limits of Visual Coding

- Diagram gets too big, “wires” tangled
  - Hardware engineers switched to e.g. VHDL for this reason
- No ASCII import/export
  - no real CVS support, no comparable NI tool
  - no script-generated code from signal list, RDB, Capfast, ...
- What you see is all you get
  - IOC’s scanning mechanisms and reaction to remote input have to be re-implemented with fundamental instructions (loops, delays, …)

LabVIEW Oddities we ran into

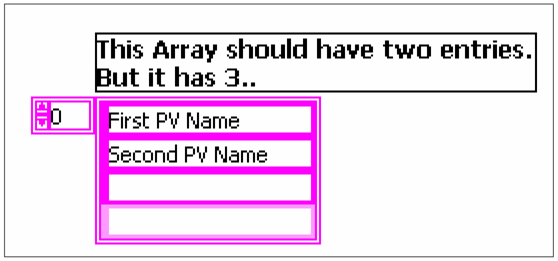
- Movie-Style “Sequence” is a cute but only shows one frame at a time



We created two "sequence locals" in a previous frame. Now it's hard to see what that was about...

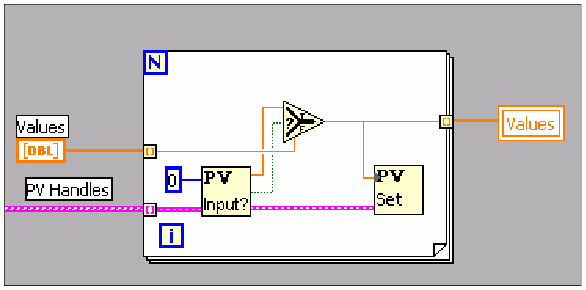
What the heck is this?

- Array handling:
  - Array constants easy to accidentally extend



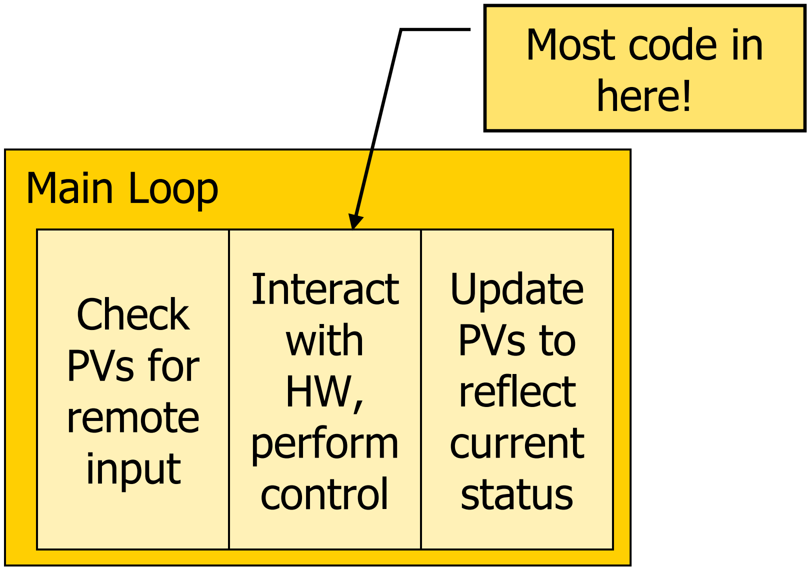
This Array should have two entries. But it has 3..

  - Auto-indexing misleading for different sized input arrays



LabVIEW: Polling

- Remote input via ChannelAccess
  - ActiveX Event is sent with <new value>,
  - LabVIEW has to catch event, check the value and - if accepted - update PV to that new value
- While e.g. VisualBasic can handle events async., LabVIEW can only “WaitForEvent”
  - Response delayed until LabVIEW’s main loop comes around to service the PV changes
  - In principle, LabVIEW offers threads & semaphores, but is that still “easy”? Not debuggable, even some Nat.Inst. VIs are incompatible.



Most code in here!

Main Loop

Check PVs for remote input

Interact with HW, perform control

Update PVs to reflect current status

Conclusion (still)

Win32-Program (with not too much) data worth serving, e.g. LabVIEW

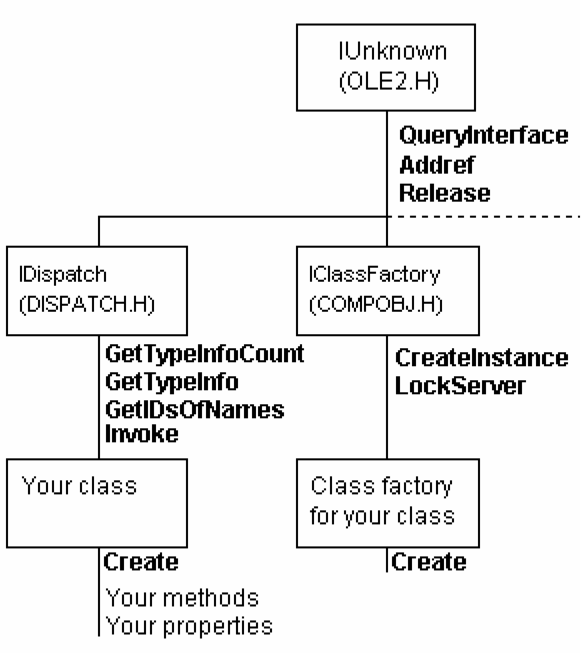
+

ActiveX CA Server

⇨ ‘EPICS’ data

ActiveX Automation Server

- COM (Component Object Model) allows programs to share objects
  - COM Classes identified via CLSID
  - Win32 handles creation and messaging
  - All objects implement the IUnknown Interface: *AddRef, Release, QueryInterface*
  - Programs agree on custom interfaces so that they can use each other's objects.
- ActiveX Automation:
  - Additional well-known interfaces, allowing IDEs and interpreted languages to
    - "browse" properties, methods and events
    - "late" as well as "early" binding
  - LabVIEW: required IProvideClassInfo in addition to standard skeleton created by MS Visual C++ ATL wizard



IUnknown (OLE2.H)

QueryInterface
Addref
Release

IDispatch (DISPATCH.H)

GetTypeInfoCount
GetTypeInfo
GetIDsOfNames
Invoke

IClassFactory (COMPOBJ.H)

CreateInstance
LockServer

Your class

Create
Your methods
Your properties

Class factory for your class

Create

EpicsCAServer.IProcessVariable

- String name
- Double deadband
- String enum_string(Long index)
- String units, Long precision,
  Double low_warning, high_warning, ...
- SetValue(Variant new_value),
  SetEnumValue(Long new_value),
  SetValueAndTime(Variant new_value, …)
- Event Changed(Variant value_received)

EpicsCAClient.IProcessVariable

- String name
- Long is_connected
- Variant Value
- String units, ...
- Event NewValue(Long is_connected, Variant value)