

# **HVmon Discussion Session**

May 17, 1:00pm – 3:00pm

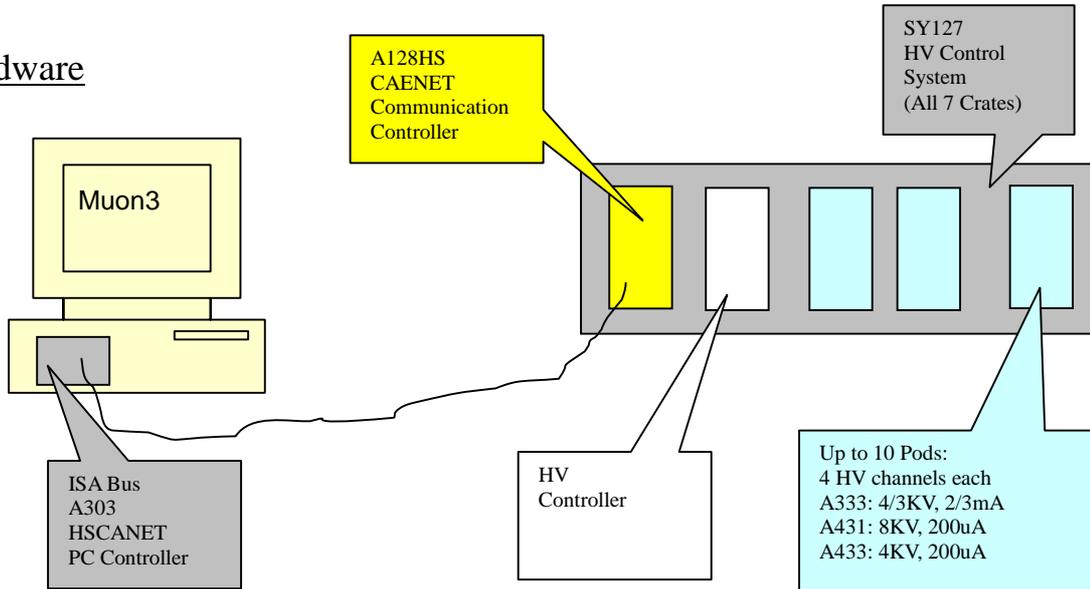
Pump Room

## Topics

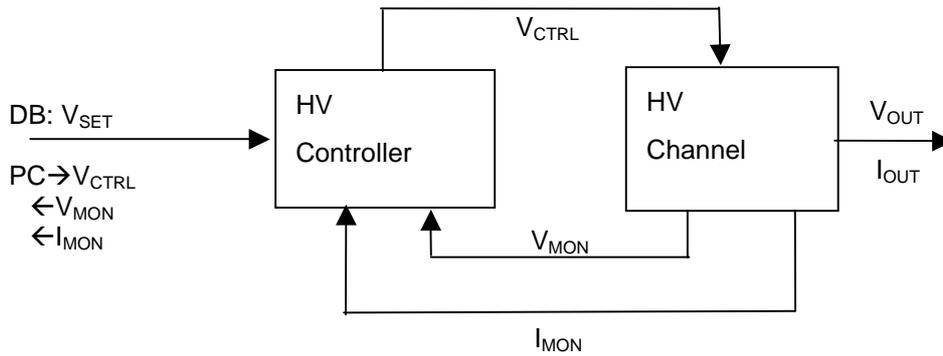
1. Introduction to HV supply and control system.
2. Progress up to date.
3. Current status of HVmon programs suite.
4. Reported problems, causes, and solutions.
5. Analysis of the 100% CPU usage problems.
6. Solutions for performance improvement.
7. Migration issues.
8. Upgrade plan.
9. Software voltage control scheme.
10. Software architecture.
11. Simulator.
12. Discussion, suggestions, and concerns – no slide.

# Introduction to HV Supply and Control System

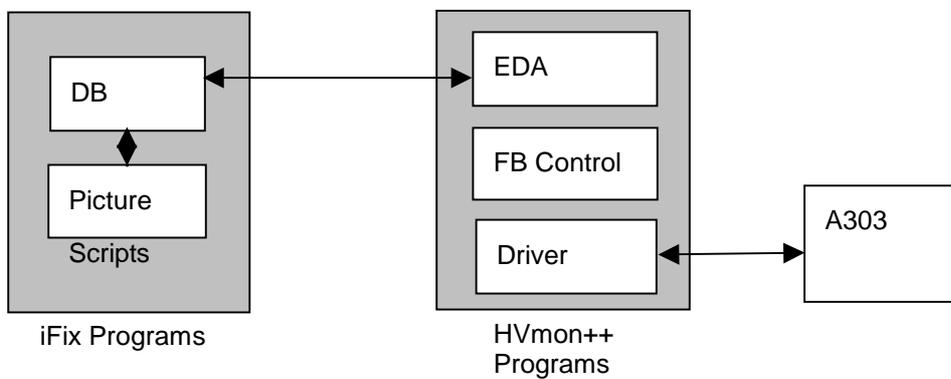
## Hardware



## HV Controller (Simplified view)



## Software: HVmon Program Suite



## **Progresses Up to Date**

### Learning

- Visual Basic for Application.
- Visual C++ 6.0.
- iFix.
- System programming in Windows system.

### Maintenance

#### Performance Analysis

#### Planning for improvement

### Obstacles

- Complexity of programs.
  - About 10 thousands of iFix tags.
  - Thousands pages of codes.
- Access problems in
  - Windows NT 4.0 machine.
  - iFix program.
  - muon3.
  - CAEN hardware.
- This kind of meeting.

## Current Status of HVmon Program Suite

### Integrity of code

- Incredibly well written!
- Not much room for patch up to improve performance.
- Reported problems are mostly related to:
  - Mis-configured DB entry.
  - Manual operation on CAEN controller.
  - Poor network/system performance.
- Excuse for Randy.
  - Might have focused on “up & running.”
  - There was no performance estimation in the beginning.
  - Excellent job done from nothingness.
- Let's give thousands thanks to Randy.
  - A working program suite is available.
  - Overall performance estimation can be done.
  - Systematic planning for upgrade is possible.
  - Big advantage for the successor.

### Performance

- Questionable! Because:
  - iFix is junks of spaghetti codes.
  - HVmon++ is too bulky.
  - Design oversights in CAEN system.
    - ◆ Manual operation stops communication w/o error message
    - ◆ Non-negligible difference in  $V_{SET}$  and  $V_{MON}$

## Reported Problems, Causes, and Solutions

(No unexplainable problems found so far)

1. 100%-CPU usage and related problems – see next slide.
2. Non-Critical warnings (like “value out of range.”)
  - Causes: Zero-point errors in  $V_{MON}$  and  $I_{MON}$ .
  - Solutions:
    - Can be safely ignored for a while.
    - Dynamic offset control in next upgrade.
    - Remained warnings will be hidden in Pictures belong to non-expert.
3. Remote-Tag identification error.
  - Causes: Network slowdown.
  - Solutions:
    - Can be ignored safely for a while.
    - Next systematic upgrade.
4. Heart beat problem
  - Cause: C++ time read has automatic DLS time correction feature.
  - Walk-around: DLS time feature in OS is disabled.
  - Solution: Recompile HVmon++ with the DLS correction feature disabled, or with simple software correction.
5. Over/Under voltage problems (like “voltages raised until trips.”)
  - Cause:  $V_{SET}$  is considerably higher/lower than  $V_{MON}$ .
  - Hardware solution: Recalibrate the CAEN pod. (Difficult, but only solution with Randy’s programs.)
  - Software solution: Recalibrate the channel (available in upgrade.)
6. System slowness, lockups, and crashes.
  - Causes:
    - 100% CPU usage.
    - Network problems.
    - Manual control of CAEN system.
  - Solution: Redesign the structure of the programs.

## Analysis of the 100% CPU Usage Problems

- May be good for math problems in a non-multitasking environment.
- Bad or funny practice in controlling hardware in a multitasking environment.
  - like eating hamburgers and juggling two balls while running full speed.
- Villain is HVmon++ with the EDA iFix toolkit (over 95%)
  - keep checking CAEN hardware status for data communication.
  - keep adjusting  $V_{SET}$  for desired  $V_{MON}$ .
  - keep updating iFix DB (Garbage Can?)
- Why it is bad in multitasking environment?
  - CPU are optimized for single tread.
  - Switching requires lots of overheads in memory and CPU usage.
  - IF one tread or task is polling continuously, it will slow down all other processes.
  - Processes need more time to stay in memory.
  - Need more switching.
  - Further slow down and increase number of processes in the memory.
  - Make one or more of ill-written treads panic.
  - Eventually it will cause system lockups or crash.
- One Consequence
  - Page fault for TCP Stack: 6000 times per second.
    - ◆ Lack of RAM?
      - No, a half of 260Mbyte of RAM is available.
      - No hard drives can deliver 24 Mbytes per second bidirectional.
    - ◆ Refresh all tags for a remote node in every 2 sec?
      - No, opening two complex pictures in a remote node will lockup the computer.
    - ◆ Conclusion.
      - HVmon++ chocks up TCP stack.
- Getting Worse
  - Muon3 is going to be shared with CCU control.
  - Need more communication for oracle data base backup.
  - Need more tasks to run.

## Solutions for Performance Improvement

### ◆ Upgrade to Windows 2000?

- Good for stability.
- A little performance improvement, but not enough.
- Current iFix program will not work in Windows 2000.

### ◆ Hardware upgrade?

- Pentium III, 1.3 GHz with DDR RAM: 20~30% improvement.
- Pentium IV, 1.5 GHz with RDRAM: no worth.
- Upgrade to Giga-Base network: too costly.

### ◆ Software Upgrade: Only practical solutions

#### ■ Bad news:

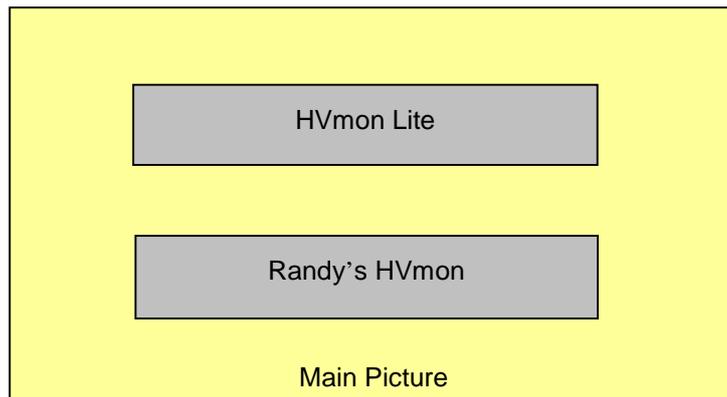
- ◆ Need to cut down Tags considerably.
- ◆ Get rid of EDA.
- ◆ Simplify voltage control.
- ◆ To overcome iFix stupidity, need system programming.
- ◆ Programs have to be restructured from grounds up.

#### ■ Good news:

- ◆ Working pieces of codes are available.
- ◆ Lots of codes are reusable.

## Migration Issues

- ◆ New program will be tested in an isolated computer with the simulator, or in the Test/Calibration setup.
- ◆ Integration test with the simulator.
- ◆ Integration test with the CAEN system.
- ◆ Integration with Randy's HVmon as a backup.



- ◆ Remove worse one eventually.

# Upgrade Plan

## Performance Improvement

- ◆ Reduce tags and compress.
  - Simplify Pictures for non-expert users.
  - Add more functionality to Pictures for experts.
  - 10000 Tags → 600 Tags and 10000 simple variables.
- ◆ Get rid of EDA tool kits.
  - Use Active X control instead.
- ◆ Non-feed back control.

## Design Considerations

- COM technology
  - Easy management and better reusability for codes.
- Simulator
  - A little effort in the beginning will save a lot of labor later.

## Added functionality

- More startup options.
- Automatic recover from a trip.
- HV trip summary report.
- Exhaustive status reports for debugging.
- Software lockout/tag-out.
- Redundant data integrity checking.

## Software Voltage Control Scheme

### Purpose

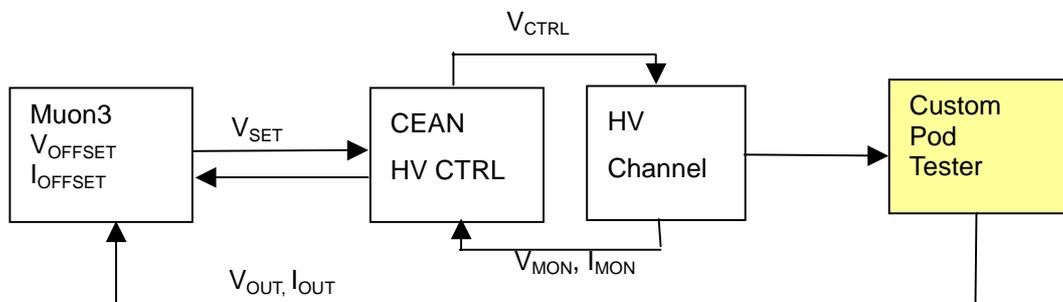
- Improve voltage precision by software control.

### Feed-back Control(Randy's Method)

- Continuously update  $V_{CTRL}$  until  $V_{SET} = V_{MON}$ .
- Problems
  - No guarantee that  $V_{MON} = V_{OUT}$ .
  - Could not overcome inaccuracy in  $V_{MON}$  reading.
  - Too much traffic. → Slowdown system performance.
  - Possibility of instability. → possibly cause trips.
    - ◆ External source such as discharges in detectors.
    - ◆ Internal source associated with digitizing error or control delay.
  - Regular hardware calibration is required.
  - Need offset control for  $V_{MON}$  and  $I_{MON}$ .

### New Control Scheme(No control actually)

- Software calibration by using a custom pod tester(added hardware).

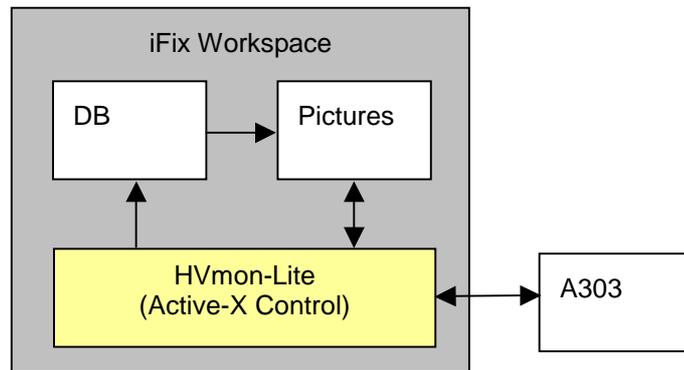


- Control:  $V_{CTRL} = K_{CTRL} * V_{SET}$
- Monitor:  $V_{OUT} = K_{OUT} * (V_{MON} - V_{OFFSET})$
- Dynamic offset control.
- “Set a value and forget it.”
- Benefits:
  - $V_{OUT} = V_{SET}$ , guaranteed in a recalibration period.
  - Reduced CPU usage.
  - No need for hardware calibration.

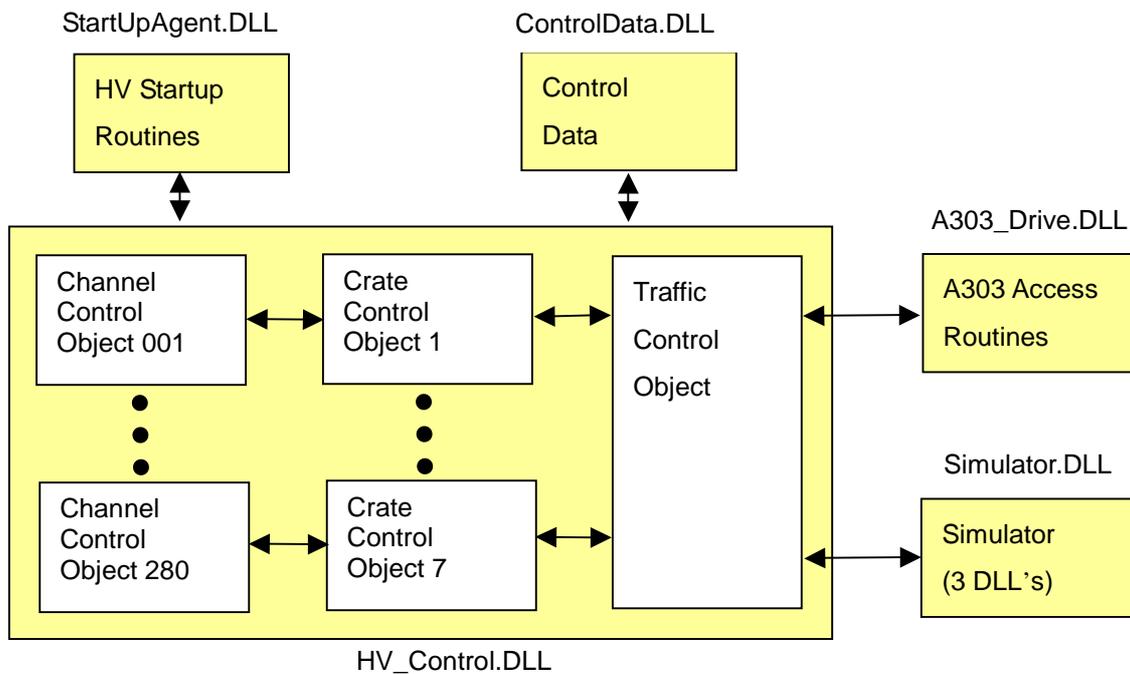
# Software Architecture

HVmon as an Active-X control will be a part of the iFix program

- No need for EDA toolkit
- Reduced tags



Components of HVmon-Lite



# Simulator

## Purpose

- Run HVmon suite without CAEN hardware

## Benefit

- Software development without stopping running codes
- Exhaustive testing and debugging is possible.
- Detect misbehaving control without destroying pods or detectors
- Can be reusable as a good tutorial for a new-comer.

## Implementation

