FRESCA
instrumentation & DAQ

2019-12-16

C. BARTH, A. GHRARIB & A. BALLARINO
Motivation

EXISTING: FRESCA instrumentation:
• Obsolete instruments (e.g. Keithley 182, 7001, 2000)
• Slow multi-channel DAQ (multiplexing)
• Data read into buffer (1024 points limit), no live data
• Low voltage DAQ only, separate transient recorder

EXISTING: FRESCA software:
• Unreliable (≥ 1 crash per day)
• Inflexible (fixed set of instruments, nearly impossible to add / exchange)
• Low programming quality (huge Labview ‘labyrinth’)

C. BARTH, A. GHARIB & A. BALLARINO
Motivation

GOALS: FRESCA instrumentation (low voltage DAQ):

- Focus on low noise & high acquisition speed:
  - low noise instruments
  - maximize data rate, synchronized parallel acquisition
  - live data, no limit in number of points

GOALS: FRESCA software:

- Focus on reliability:
  - instruments & GUI on separate computers & processes

- Focus on modularity, flexibility & upgradability
  - instruments & GUI in modules: class based approach
  - modules are separated & stand-alone: addition / modification does not affect others
  - any combination of modules, each module as n ‘clones’
Instruments

Available instruments:

• Nano-voltmeters
  • Keithley 2182A, Keithley 182
  • Keysight 34420A

• Multimeters
  • NI-DMM 7.5 digits (PXI-4071), 6.5 digits (PXI-4070)
  • Keysight 3458A, Keysight 3457A
  • Keithley 2001, Keithley 2000

Low voltage DAQ instrument selection:

• Qualitative tests on superconductor measurements
  • no digital / analog filtering
  • setting integration time to reach required noise levels
  • comparing data rate & price
Instruments

**Instrument performance:**

- **Minimal range**
  - 200 µV max signal → 1 mV range is sufficient
  - nano-voltmeters: 1 mV (34420A), 3 mV (182), 10 mV (2182A)
  - multimeters: 100 mV

- **Noise**
  - nano-voltmeters, 3458A and NI-DMMs can reach sufficiently low noise levels: ±50 nV

- **Data rate**
  - nano-voltmeters and 3458A provide significantly higher data rate than NI-DMMs at the same noise level
    - to reach 7.5 digits resolution
      - nano-voltmeters and 3458A: 1 - 2 NPLC integration
      - NI-DMM (PXI-4071): 7 NPLC integration
Instruments

Low voltage DAQ instruments:

• Nano-voltmeters
  - Keithley 2182A, Keithley 182
  - Keysight 34420A

• Multimeters
  - NI DMM 7.5 digits (PXI-4071), 6.5 digits (PXI-4070)
  - Keysight 3458A, Keysight 3457A
  - Keithley 2001, Keithley 2000

• Cannot reach required noise levels
• Obsolete
• Data rate too low
• Price too high
Instruments

Maximize data rate, no limit in number of points:
• Direct read-out, no buffer
• Integration end has to trigger read-out of last data point
• Independent of connection type or bus

Synchronize acquisition of multiple instruments:
• Simultaneous integration start on all instruments
• Wait for integration end of all instruments for next cycle
• Independent of connection type or bus

Common interface:
• Acquisition complete signal output: ‘VM complete’
• Start acquisition input: ‘trigger’
VM complete & trigger

**Fast readout: VM complete signal:**
- Send hardware interrupt → initiate reading
- 2 possibilities with NI hardware:
  - hardware timed single point clock → 1 per counter + 1 for all AI
  - value change event → 1 for all P0.x DI
- M-series DAQ: 4, X-series DAQ: 5, counter card: 9

**Synchronization: trigger signal:**
- Wait for all VM complete signals → send ‘trigger’ to start next acquisition cycle
- Any NI-FPGA can provide this functionality
- Why separate and not both functionality with FPGA?
  - no recompiling of FPGA code, modularity better with interrupt
  - ‘VM complete’ & ‘trigger’ on PXIe-408x only via bus
Instrumentation: low voltage DAQ

- Hardware synchronization in 2 stages
  - 1\textsuperscript{st} stage: fetch last value upon VM complete
  - 2\textsuperscript{nd} stage: synchronization of multiple 2182A / DMMs

- **PXI:**
  - waits for hardware timed clock tick
  - reads last value with ‘fetch’

- **2182A:**
  - triggered

- **DAQ:**
  - PFI input
  - hardware timed single point clock
  - matched with counter

- **FPGA:**
  - waits for all VM complete signals during a cycle
  - configurable timeout
  - sends hardware trigger to all instruments to start a new cycle
Approach

Instrumentation: low voltage DAQ

• Hardware synchronization in 2 stages
  • 1\textsuperscript{st} stage: fetch last value upon VM complete
  • 2\textsuperscript{nd} stage: synchronization of multiple 2182A / DMMs
• Full speed down to 1 NPLC
• Live data without limit in number of points

→ Ideal for superconductor cable measurements
  • low voltage range of 3 – 5 μV → low noise instruments
  • supercond. transition can be < 1 sec → high data rate

→ Validated with demonstration DAQ
  • for 9 months operated in parallel
  • data used in both 11 T cable measurement campaigns
Approach

Software: general concept

- Transparent networked communication
  - $n$ sources – $m$ sinks
  - control, data & synchronization: any data type
- Class based & object oriented
  - standardized IO channels
  - inheritance of common features, including front-panel
- Fully modular
  - instruments & GUI elements as stand-alone modules
  - any instrument & GUI combination, creates ‘clones’ of modules
  - settings from text settings files, automation-conditions

→ Unified software for all superconductor test stations
  - wire test stations & LN$_2$ test station as subsets of FRESCA
Overview: FRESCA:

GUI PC #2 (Windows 10):
- plots
- data saving
- analysis

GUI PC #1 (Linux):
- plots
- data saving
- analysis
- measurement control
- instrument remotes

Gateway

PXIe #1 (RT OS):
- 2x NI-DMM
- 8x 2182A (via RS232)
- 2x FPGA
- 2x matrix 32x16
- 2x multifunc. DAQ
- 2x counter card

PXI #2 (RT OS / Linux):
- 2x matrix 32x16
- 4x transient recorder
- 1x multifunc. DAQ

PXI #3 (RT OS / Linux):
- 12x uQDS (via USB)
- 2x multifunc. DAQ

compactRIO (FPGA):
- 1x current input
- 1x current output

FGC (32 kA)
FGC (16 kA)
cryo published values
cryo PLC
20 mA

Ethernet
Overview: wire test station:

GUI PC #2 (Windows 10):
- plots
- data saving
- analysis

GUI PC #1 (Linux):
- plots
- data saving
- analysis
- measurement control
- instrument remotes

PXI #1 (RT OS):
- 2x NI-DMM
- 4x 2182A (via RS232)
- 1x FPGA
- 1x uQDS (via USB)
- 1x counter card
- 1x magnet (serial)

Gateway

Ethernet

FGC (6 kA)
cryo published values
Overview: LN$_2$ test station:

**GUI PC #1 (Linux):**
- plots
- data saving
- analysis
- measurement control
- instrument remotes

**PXI #1 (RT OS):**
- 1x 2000 (via RS232)
- 2x 182A (via GPIB)
- 1x FPGA
- 1x multiplex
- 1x multifunc. DAQ

Ethernet

Gateway
**Status**

**Instrumentation**

- Development in bottom-up approach
- LN$_2$ test station instrumentation rack
  - basic DAQ & analog current control only
  - done → in use as test platform for software development
- Wire test station instrumentation rack
  - exact FRESCA rack wiring (subset)
  - exact FRESCA interlock chain (subset)
  - within 2019 → FRESCA rack wiring demonstrator
    → impedance decoupling demonstrator
    → future software development platform
- FRESCA instrumentation racks
  - all internal wiring in PCB
  - impedance decoupling for all signals
  - 03/2020 → ready for FRESCA commissioning / operation
**Status**

**Software**

- Development in bottom-up approach

**Milestone 0**

- **general**: communication, module classes & error handling
- **instruments**: basic DAQ & analog current control
- within 2019 → validation on LN$_2$ measurements

**Milestone 1**

- **instruments**: uQDS, FGC power converter control
- **UI**: cryo published values, plotting & saving with automation
- 01/2020 → sufficient for FRESCA commissioning

**Milestone 2**

- **instruments**: matrix
- **UI**: instrument remotes, measurement sequencer
- 02/2020 → sufficient for nominal FRESCA operation
Software package utilization

Development languages:

• NI Labview (control & DAQ)
  • + ‘perfect’ support of NI hardware (including FPGA)
  • + many instrument drivers & communication frameworks
  • + good for GUI
  • reputation of being ‘messy’ with many instruments
    • 1 instrument or GUI element per module
  • reputation of being unreliable & slow
    • separation of instruments & GUI, all modules in individual processes
  • reputation of being difficult to maintain & upgrade
    • new instruments added in new modules, no affect on existing ones
    • class based approach with inheritance to minimize development effort
    • standard packages / frameworks & provide good documentation

• Python (analysis)
Software package utilization

Commercial software licenses:

• Instrument side
  • full Labview license
    • core, Labview RT, Labview FPGA, fuzzy logic & control
  • NI RT OS licenses

• GUI side
  • full Labview license
    • core, fuzzy logic & control
  • GUI agnostic to operating system
    • will work under Windows 10 or under Linux
    • Windows 10 license is not mandatory

→ we have minimized the reliance on commercial software packages
Conclusions

- Instrumentation optimized for $I_c$ measurements
  - low noise instruments, maximize data rate: Keithley 2182A
  - paralleled & synchronized acquisition, high speed read-out
  - approach independent of instrument, connection & bus
  - validated with demonstration DAQ for 9 months

- Software optimized for stability
  - instrument & GUI separation, individual processes

- Software optimized for modularity & upgradability
  - any instruments & GUI combination
  - unified software for all superconductor test stations

- Development in bottom-up approach
  - increasing complexity to simplify debugging
  - clear milestones to resume FRESCA operation in 03/2020
Proposal

Demonstration in SM18 after SC-link DEMO2 test:

• Add synchronization to MSC-TF PXle chassis
• Cost neutral

GUI PC #1 (Linux / Windows):
• plots
• data saving
• measurement control
• instrument remotes

PXle #1 (RT OS):
• 9x NI-DMM
• 1x counter card

PXle #2 (RT OS):
• 9x NI-DMM
• 1x counter card

PXle #3 (RT OS):
• 9x NI-DMM
• 1x counter card

PXle #4 (RT OS):
• 9x NI-DMM
• 1x counter card

Gateway

PXI #5 (RT OS):
• 4x Keithley 2182A
• 2x NI-DMM
• 1x counter card

PXI #6 (RT OS):
• 2x Keithley 182
• 1x Keithley 2000
• 1x multifunc. DAQ

Ethernet

cryo published values

FGC (20 kA)
Optional: demonstration video (2:30 min)
Demonstration program

Basic DAQ implemented:

- xy-plot clone #3
- xy-plot clone #2
- xy-plot clone #1

Hardware & GUI on same PXI:

- main loader
- NI DMM
- FPGA sync.
- 2182A clone #1
- 2182A clone #1
- min / max values
- data saver

networked communication: 127.0.0.1
Demonstration program

Basic DAQ implemented:
**Vision**

### Text based settings files:

- Main ‘loader’ settings file determines instrument / data sink combination
  - specifies which instruments / data sinks to load how many times and which settings file to use for each clone
  - no limitations in terms of clone numbers / combinations

<table>
<thead>
<tr>
<th>Clone #1 of xy-plot settings: FRESCA_UI_plot.xy_plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone #2 of xy-plot settings: FRESCA_It_plot.xy_plot</td>
</tr>
<tr>
<td>Clone #1 of Keithley 2182A settings: FRESCA_NV1.2182A</td>
</tr>
<tr>
<td>Clone #2 of Keithley 2182A settings: FRESCA_NV2.2182A</td>
</tr>
</tbody>
</table>
Text based settings files:

- All settings parsed from text based settings files
  - human readable, support comments
  - restricted to limited number of data types: string, multi-line string, numeric & bool
  - automatic check for validity upon loading

```plaintext
$NETWORKED_MEASUREMENT_SYSTEM_SETTINGS_FILE:
$ the file is read line by line. Each line corresponds to the settings of a control with matching name. Command
$ 'g' --> comment char --> everything behind this char is ignored
$ ':' --> delimiter --> used to separate the control name from the control value. There can be only one deli
$ 's' --> new line char --> used to add a newline to multi-line strings. This is e.g. used to define several c

$MODTYPE 2182A
$PRESSET 1
$IP 127.0.0.1
$NAME NV1
$UNIT V
$SCALING_FACTOR 1
$CONNECTION COM1
$RANGE 1
$INTEGRATION_TIME 1
$ANALOG_FILTER 0
$DIGITAL_FILTER 0
$AUTOZERO 0
```

$Type of the module, this has to match the module
$Preset for window secondary settings (e.g. window position): 0 = i
$IP address of the data socket server. A local data socket server is s
$Name of the clone, this is used for identification of the running clon
$Name of the data read from the Nanovolt meter. The name cannot
$Unit of the measured signal. Default is V.
$Factor used to scale the data measured by the instrument. Default
$Connection of the instrument, can be serial port or GPIB address. If
$Autorange, 1 = 10 mV, 2 = 100 mV, 3 = 1 V, 4 = 10 V and 5 =
$Integration time in NPLC: from 0.01 to 50, default is 5
$S = disabled and 1 = enabled
$Preset of digital filter: 0 = disabled, 1 = preset 1, 2 = preset 2, 3:
$Controls the auto zero of the internal amplifiers: 0 = auto zero in t
### Automation and synchronization of sinks:

- Channels values read upon update of clock channel
  - synchronization with data source
- Start / stop can be automated
  - comparison expression(s) with channel value
  - **true** → trigger start / stop
  - from text based settings & adjustable during runtime

<table>
<thead>
<tr>
<th>No.</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>§§§§</td>
<td>module type: xy plot</td>
</tr>
<tr>
<td>10</td>
<td>§§§§</td>
<td>preset: 1</td>
</tr>
<tr>
<td>11</td>
<td>§§§§</td>
<td>ip address: 127.0.0.1</td>
</tr>
<tr>
<td>12</td>
<td>§§§§</td>
<td>module name: UI_plot</td>
</tr>
<tr>
<td>13</td>
<td>§§§§</td>
<td>clock delay: 10</td>
</tr>
<tr>
<td>14</td>
<td>§§§§</td>
<td>clock channel: 127.0.0.1/NV1/data/U</td>
</tr>
<tr>
<td>15</td>
<td>§§§§</td>
<td>x channel: 127.0.0.1/DMM/data/I</td>
</tr>
<tr>
<td>16</td>
<td>§§§§</td>
<td>y channel(s): 127.0.0.1/NV1/data/U $ 127.0.0.1/NV2/data/U</td>
</tr>
<tr>
<td>17</td>
<td>§§§§</td>
<td>start conditions(s): '127.0.0.1/DMM/data/I'</td>
</tr>
<tr>
<td>18</td>
<td>§§§§</td>
<td>stop conditions(s): '127.0.0.1/DMM/data/I'</td>
</tr>
</tbody>
</table>

- ‘clock’ channel for synchronization
  - **example:** voltage read by 2182A #1

- automatic plot start condition
  - **example:** current |>| 3 kA

- automatic plot stop condition
  - **example:** current |<| 0.1 kA