Reliability Test System Using Data Acquisition and Instrument Control Toolbox

March 26, 2014
Presented by: Rob Danforth
Overview

• First experience with using MATLAB for DAQ
• Utilized DAQ and Instrument Control Toolboxes
• Compiled the application for multiple usage
• Tips for developing applications for others
  – Object Modeling
  – Object interaction with GUIs
Background

- Load Cycling
- Temperature Monitoring
- Power Monitoring
Background

- More Channels
- Faster Sampling Rate
- Specialized Performance Testing
- Report Automation
  - Load Cycling
  - Temperature Monitoring
  - Power Monitoring
Extend Current
Extend Current

MathWorks
Process

Writing for a single use

Learn (Play)  Test & Compile

Data Acquisition Toolbox
• No previous experience
• Developing functioning scripts was straightforward; completed in one day

Instrument Control Toolbox
• Straightforward and easy to use
• Honesty statement: Power Meter uses MODBUS protocol. Not straight forward!
<table>
<thead>
<tr>
<th>Object Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitronics (Power Meter)</td>
</tr>
<tr>
<td>NIDaq (TC, AI, DIO)</td>
</tr>
<tr>
<td>Loadbank (DIO)</td>
</tr>
<tr>
<td>TestStep</td>
</tr>
<tr>
<td>TestSequence</td>
</tr>
<tr>
<td>TestProfile</td>
</tr>
<tr>
<td>Channel</td>
</tr>
<tr>
<td>MetaData</td>
</tr>
<tr>
<td>DAQConfig</td>
</tr>
</tbody>
</table>
Object Definition

**NIDaq (TC, AI, DIO)**

What must this object do?
- Be Configured*
  - Setup Channels
  - Sampling Rate
- Connect to the Hardware
- Start collecting data
- Stop collecting data

*For ease of use and for distributed applications, I chose to use an XML file to initialize configuration of all objects.*
Object Definition

```xml
<?xml version="1.0" ?>

- <NIDaq>
  <SamplingTime>2.5</SamplingTime>
  <ID>DAQ8</ID>
- <ChannelList>
- - <Channel>
      <Name>ai0</Name>
      <Device>cDAQ9184-1911520Mod2</Device>
      <Type>TC</Type>
      <Description>Eng_Air_In</Description>
      <Display>On</Display>
      <Order />
      <RangeVolts />
      <SensorSerial />
      <SensorUnits>C</SensorUnits>
      <SensorGain>1</SensorGain>
      <SensorOffset>0</SensorOffset>
    </Channel>

function obj = importXML(FilePath)
    % importXML - Import XML file in
    obj=rDAQ.NIDaq();
    n=xml2struct(FilePath);
    obj.SamplingTime=str2double(n.NIDaq.SamplingTime.Text);
    obj.ID=n.NIDaq.ID.Text;
```
GUI Object Interaction

A few notes about GUI Lessons Learned:

• Trickiest part of building an application for others
• Couple key concepts are critical
  – Utilize a rigid structure, separating the data (object) from the GUI
  – Use listeners when possible to trigger GUI refresh
### Kohler Power Systems - Reliability

#### Test Steps

<table>
<thead>
<tr>
<th>Load Step (%)</th>
<th>Load (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Test Profile

<table>
<thead>
<tr>
<th>Profile</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Test Sequence

- **Volts**: 118.231
- **Amps**: 0
- **Freq**: 60.02
- **Power**: 0
- **PF**: 0

- **L-225 Load Bank**
  - 0.5 kW
  - 0.5 kW
  - 1 kW
  - 1 kW
  - 2 kW
  - 5 kW
  - 5 kW
  - 10 kW

- **Number of Test Sequence Loops**: 1

- **E-Stop**
Kohler Power Systems - Reliability

InitGui.m
RefreshGUI.m
UpdateGUI.m

Code inside InitGui.m

%% Events
events
PowerUpdate
derend

notify(obj,'PowerUpdate');

<table>
<thead>
<tr>
<th>Test Steps</th>
<th>Volts</th>
<th>Amps</th>
<th>Freq</th>
<th>Power</th>
<th>PF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Step (%)</td>
<td>Load (kW)</td>
<td>118.231</td>
<td>0</td>
<td>60.02</td>
<td>0</td>
</tr>
</tbody>
</table>
System Summary

Data Logging
- Interactive display
- Manual or Automated Mode
- Collects slow data
  - Export to CSV file
  - One file per day
- Collects fast data
  - Automated processing
  - Substantial improvement
- Future: Email notification
Summary

• Data Acquisition and Instrument Control Toolbox are cost effective solutions. They worked as advertised.

• You don’t need to be a professional programmer to make interactive and effective programs utilized by non-MATLAB users.

• Internal development allows us to customize without compromise.

• An all MathWorks approach allows us tight integration between acquisition and analysis in an efficient manner.

• We’ve gained a high confidence and are extending to other applications.
Questions?