Calibrating a UUT on a Remote Computer Using Fluke MET/CAL®

Michael L. Schwartz
Cal Lab Solutions

“ACHIEVING COMPETITIVE ADVANTAGE THROUGH MEASUREMENT INNOVATION”
INTRODUCTION

• Current and next generation test equipment presents challenge for calibration labs

• Technologies can be designed to work together

• Fluke MET/CAL® procedure and Metrology.NET
  • Basic design patterns of remote computing
  • Command interface for non-message based instrument
  • Remotely communicate with the instrument
THE PROBLEM

• Labs may not have resources to retool in order to support manufacturers’ software solutions in maintaining PXI & PXIE instruments

• Customer required a solution to support National Instruments PXI-5122.

• Manufacturer solution required a Fluke 9500, but customer can’t justify purchase.
THE PROBLEM DOMAIN

• The calibration lab needs a way to support the PXI-5122 in-house
• They do not have a Fluke 9500
• They have a Fluke 5520
• Testing them manually is not an option
OUR SOLUTION

• Starting point is with Fluke MET/CAL®
• Software based instruments do not always run on every operating system
• Life expectancy
• Decouple the UUT code from the standard’s code
  • Text command interface for the UUT
  • Create a service
  • Create a client messaging app
  • Write the procedure and test
It's about Decoupling

Metrology Services Bus

ANY Language
ANY Platform

- Test Point(s)
- Test Process
- UUT Config
- STD Config
- Measurement(s)
- Uncertainty Calculations
- PASS/FAIL
- Results
• Metrology Service Bus Layer
  • Language agnostic & platform independent

• Measurement Process Driver
  • Any language
  • Focus is quality measurement

• Command Base Driver
  • Not all instruments are command based
  • IEEE SCPI calls and RS-232 programming

• Function Calls

• Low Level Instrument Control
## Creating a Command Set

<table>
<thead>
<tr>
<th>Command</th>
<th>Function Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDN:</td>
<td></td>
</tr>
<tr>
<td>Reset:</td>
<td>niScope_init</td>
</tr>
<tr>
<td>SelfCal:</td>
<td></td>
</tr>
<tr>
<td>SelfTest:</td>
<td></td>
</tr>
<tr>
<td>ConfigureChanCharacteristics:</td>
<td>niScope_ConfigureChanCharacteristics</td>
</tr>
<tr>
<td>Channel=</td>
<td></td>
</tr>
<tr>
<td>Impedance=</td>
<td></td>
</tr>
<tr>
<td>Bandwidth=</td>
<td></td>
</tr>
<tr>
<td>ConfigureVertical:</td>
<td>niScope_ConfigureVertical</td>
</tr>
<tr>
<td>Channel=</td>
<td></td>
</tr>
<tr>
<td>Coupling=</td>
<td></td>
</tr>
<tr>
<td>Attenuation=</td>
<td></td>
</tr>
<tr>
<td>Range=</td>
<td></td>
</tr>
<tr>
<td>Offset=</td>
<td></td>
</tr>
<tr>
<td>ConfigureHorizontalTiming:</td>
<td>niScope_ConfigureHorizontalTiming</td>
</tr>
<tr>
<td>SampleRate=</td>
<td></td>
</tr>
<tr>
<td>Position=</td>
<td></td>
</tr>
<tr>
<td>Points=</td>
<td></td>
</tr>
<tr>
<td>ConfigureEdgeTrigger:</td>
<td>niScope_ConfigureTrigger</td>
</tr>
<tr>
<td>Channel=</td>
<td></td>
</tr>
<tr>
<td>Slope=</td>
<td></td>
</tr>
<tr>
<td>Coupling=</td>
<td></td>
</tr>
<tr>
<td>Level=</td>
<td></td>
</tr>
<tr>
<td>ConfigureImmediateTrigger:</td>
<td>niScope_Initiate</td>
</tr>
<tr>
<td>Commit:</td>
<td>niScope_Commit</td>
</tr>
<tr>
<td>Measure:</td>
<td>niScope_Fetch</td>
</tr>
<tr>
<td>Channel=</td>
<td></td>
</tr>
<tr>
<td>NumberOfAverages=</td>
<td></td>
</tr>
<tr>
<td>Measurement=</td>
<td></td>
</tr>
</tbody>
</table>

More often, newer software based instruments do not support a command based language

**Function calls**

**Command processor**

- Define the command language and write a string parser

Example

ConfigureVertical: Channel= 1, Coupling= DC, Attenuation= 0, Range= 10, Offset= 0
Creating a Command Processor

Public Overrides Function Command(ByVal CMD As String) As String

    If UCase(CMD).Contains("IDN:“.ToUpper) Then
        Return myScope.Identity.InstrumentModel
        Exit Function
    End If

    If UCase(CMD).Contains("Reset:“.ToUpper) Then
        If Me.Reset() = 0 Then
            Return "Success"
        Else
            Return "ERROR!"
        End If
        Exit Function
    End If

    If UCase(CMD).Contains("ConfigureChanCharacteristics:“.ToUpper) Then
        If Me.ConfigureChanCharacteristics(CMD) = 0 Then
            Return "Done"
        Else
            Return "ERROR!"
        End If
    End If
Exposing the Command Processor

Overrides in the function call

Public Overrides Function Command(ByVal CMD As String) As String

Operating Contract and WebGet

<OperationContract()>
<WebGet(ResponseFormat:=WebMessageFormat.Xml,
BodyStyle:=WebMessageBodyStyle.Bare)>
Public MustOverride Function Command(ByVal CMD As String) As String

Creating a web interface

' Create New host
Dim host = New WebServiceHost(handler, New Uri("http://" & Me.IP & ":" & Me.Port))
Dim EP = host.AddServiceEndpoint(GetType(iTxtCommand), New WebHttpBinding(), Name)
host.Open()
CREATING THE MCNETCOMM.EXE

- Next step: link to MET/CAL ®
- McNetComm.exe
  - Supports MET/CAL ® versions 5.0 – 8.x
  - COM visible
The MET/CAL ® Procedure

Calling the Default Test Configuration resetting the global variables:

3.001 LABEL Default
# Channel Settings
3.002 MATH @Channel = 1
3.003 MATH @Impedance = 1e6
3.004 MATH @Bandwidth = 100e6
3.005 MATH @Coupl = "'DC'"
3.006 MATH @Atten = 1
3.007 MATH @Range = 4
3.008 MATH @Offset = 0
3.016 MATH @AVG = 8
# Horizontal Settings
3.009 MATH @SampleRate = 10e6
3.010 MATH @Position = 50
3.011 MATH @Points = 100e3
#Trigger Settings
3.012 MATH @TChannel = 1
3.013 MATH @Slope = "'POS'"
3.014 MATH @TCoupl = "'DC'"
3.015 MATH @Level = 0.00125

With each test group we would set the Test Channel:

3.002 MATH @Channel = <Test Channel>

And every point we set the required variables and execute the test:

#-----------------------------
10.005 MATH @Volts=0.09*1
10.006 MATH @Range=0.2*1
10.007 VSET UUT_Res = .001
10.008 IF Find(S[23],"EnableRepeatability",1)>0
10.009 VSET U3 = 0
10.010 ENDIF
10.011 CALL NI 51xx Sub Test Routines-Conf
10.012 MATH L[9]=Fld(S[31],2,"Unc=")/1
10.013 ACC 0.000%_ L9U
10.014 IF 1==0
10.015 TARGET -m
10.016 CALL NI 51xx Sub Test Routines-Meas
10.017 ENDIF
10.018 MATH MEM=Fld(S[31],2,"Value=")/1
10.019 MEMCX 0.2 _ 0.65U

The test routines would configure the UUT using the following Sub Tools Calls:

# Set up the Channel
3.023 MATH S[30]="ConfChanChar"
3.024 CALL NI 51xx Sub Tools
3.025 MATH S[30]="ConfVert"
3.026 CALL NI 51xx Sub Tools

The Sub Tools then passes the commands to the UUT as follows:

#=====================================================================
7.001 LABEL ConfChanChar
7.002 MATH MEM2 = "ConfigureChanCharacteristics:
7.003 MATH MEM2=MEM2& " Channel= " & @Channel
7.004 MATH MEM2=MEM2& ",Impedance= " & @Impedance
7.005 MATH MEM2=MEM2& ",Bandwidth= " & @Bandwidth
7.006 DOS C:\CLS\McNetComm.exe Query UUT
7.007 IF Find(MEM2,"Configure",1)
7.008 DISP Communication Error Command Not Executed
7.009 ENDIF
7.010 END
#=====================================================================
8.001 LABEL ConfVert
8.002 MATH MEM2 = "ConfigureVertical: 
8.003 MATH MEM2=MEM2& " Channel= " & @Channel
8.004 MATH MEM2=MEM2& ",Coupling= " & @Coupl
8.005 MATH MEM2=MEM2& ",Attenuation= " & @Atten
8.006 MATH MEM2=MEM2& ",Range= " & @Range
8.007 MATH MEM2=MEM2& ",Offset= " & @Offset
8.008 DOS C:\CLS\McNetComm.exe Query UUT
8.009 IF Find(MEM2,"Configure",1)
8.010 DISP Communication Error Command Not Executed
8.011 ENDIF
8.012 END
CONCLUSION

• Why do all this work?
  • The software is now Decoupled
  • You can Cal the UUT in the Mainframe
  • You don’t have to reboot your workstation every time you change a UUT card.
  • BECAUSE IT’s COOL
Questions? / Comments

Michael L. Schwartz
Cal Lab Solutions
mschwartz@callabsolutions.com