**CPC600 Automated Calibration**

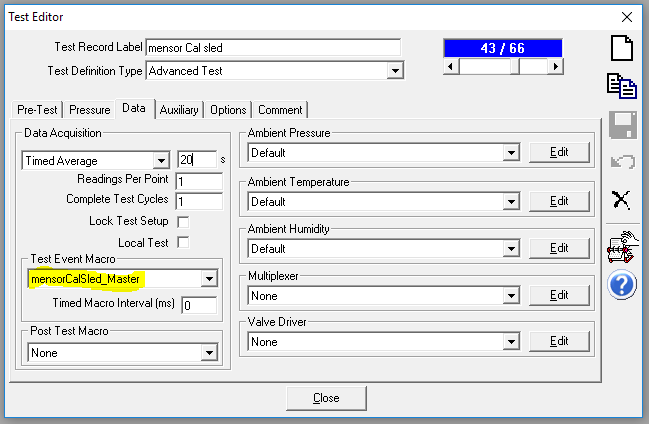
16 August 2019

The calibration and adjustment of a mensor CPC600 module is automated by using these two macros:

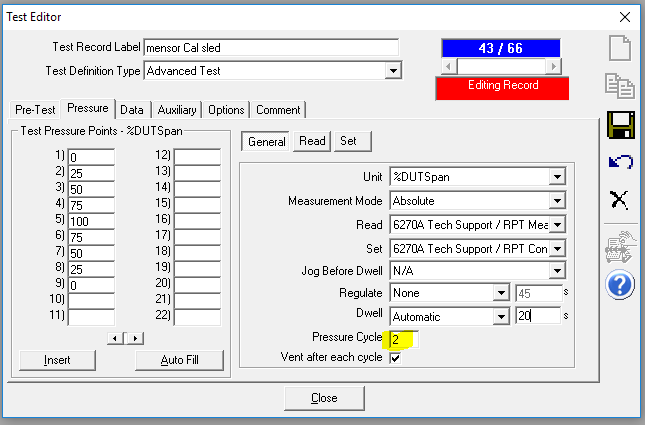
* “mensorCalSled\_Master”
* “mensorCalSled\_AbsGauge”

The CPC600 has a primary range and secondary range, and automated adjustment for both ranges is supported. The DUT definition contains the setup information and initialization commands required to change between the two ranges.

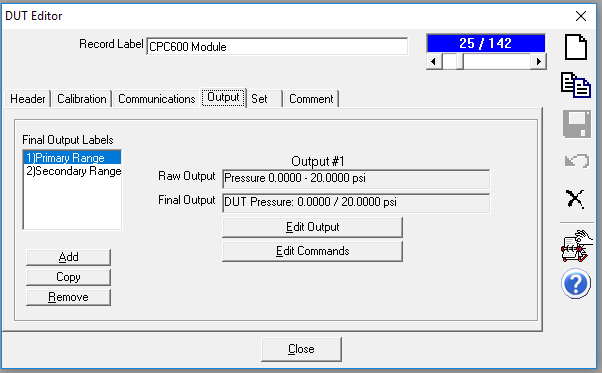
The Test macro, “mensorCalSled\_Master”, is called as the Test Event Macro in the Test Definition.

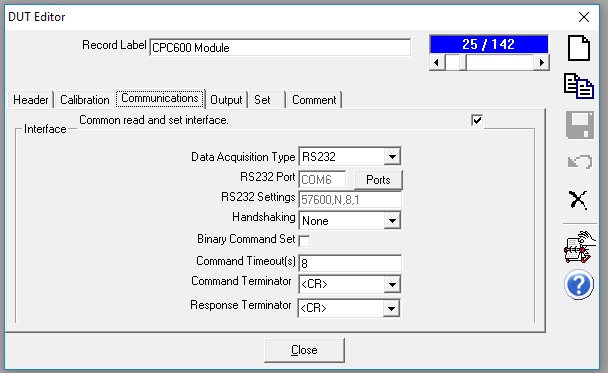


Two Pressure Cycles are required – the first cycle collects As Found data, and if required the second cycle collects the As Left verification data. At the end of the first pressure cycle the As Found data is analyzed for being in or out of tolerance. A “Tolerance Factor” is used to determine if the data is well inside of the specification or marginally inside. If it’s marginally inside then an adjustment is applied.



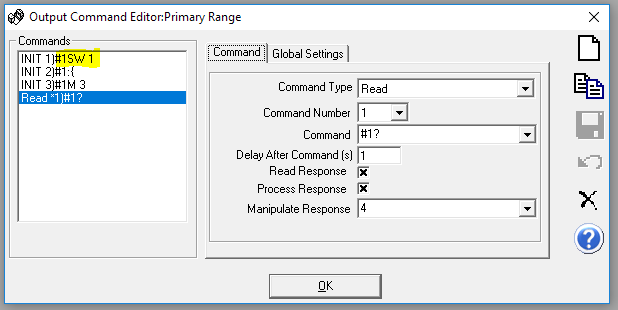
The DUT Definition is configured as an Advanced Device as there will be two outputs – one for the Primary and Secondary ranges. The Cal Sled communications are fixed at a baud rate of 57600 with <CR> only as the Command and Response terminator. A regular straight through RS232 cable is required.





The remote commands configuration is nearly identical for both the Primary and Secondary ranges with the key difference being the range change command:

* **#1SW 1** = Primary Range
* **#1SW 2** = Secondary Range



Note: the Read command requires a “Manipulate Response” value of “4” to strip away the first four leading characters.

**mensorCalSled\_Master code:**

*'This is the main TestEvent Macro to use with a mensor CPC600 module calibration.*

*'It's used to call "mensorCalSled\_AbsGauge" test macro. As of August 2019*

*'only gauge mode and absolute mode modules are supported. It's likely that*

*'differential mode would be but this has not been tested.*

*'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**Function** **mensorCalSledCalibrationAdjust**(iT, iL, iC, iP, cTest, cConfig)

*' Determine which macro to run.*

**Select Case** cTest.TestPrsMeasMode

**Case** 0:

**Call** **mensorCalSledGauge**(iT, iL, iC, iP, cTest, cConfig)

**Case** 1:

**Call** **mensorCalSledGauge**(iT, iL, iC, iP, cTest, cConfig)

**Case** 3:

**Call** mensorCalSledDifferential(iT, iL, iC, iP, cTest, cConfig)

**End Select**

**End Function**

**Function** **mensorCalSled\_Dwell**(dwell) *'"dwell" declared as a parameter of the function.*

cDebug.LogStatus "Dwell for " & dwell & "s"

tStart = timer

**Do**

cCOMPASS.**TimeDelay** 1

td = **CInt**(dwell-**time\_Difference**(tStart))

cCOMPASS.StatusDisplay "Dwell..." & td

**If** cCOMPASS.SystemAbort **Then** **Exit** **Function**

**If** td <= 0 **Then** **Exit** **Do**

**Loop** **Until** **False**

**End Function**

**Function** **mensorCalSled\_WaitForReady**(isZero, timeout)

cDebug.LogStatus "Wait for Ready: timeout=" & timeout

tStart = Timer

*'The Do Loop tells COMPASS to look at the logic output of the UseReady method. The logic True / False*

*'can be queried by using the .Ready parameter. The return from cConfig.SetPres(1).Ready is either TRUE or FALSE.*

**Do**

cCOMPASS.**TimeDelay** 2 *'delay controller*

cCOMPASS.StatusDisplay process & ": Waiting for Ready.........."

**If** cCOMPASS.SystemAbort **Then** **Exit** **Function**

**If** **time\_Difference**(tStart) > timeout **Then**

cDebug.LogStatus "\*\*\*\*\*\*\*TIMEOUT WAITING for Ready\*\*\*\*"

cCOMPASS.StatusDisplay process & ": ............TIMEOUT Waiting for Ready.........."

cCOMPASS.**TimeDelay** 5 *'delay to show above message*

**If** (cCOMPASS.cConfig.DUTPrs(1).RangeMain.MeasMode = 0) And (isZero = 1) **Then**

cDebug.LogStatus "Timeout for absolute 0, as good as it will get"

**Exit** **Do** *'absolute 0, as good as possible*

**End If**

**Exit** **Function**

**End If**

*'If the SetPrs device supports the UseReady concept then loop until the .Ready parameter becomes TRUE or,*

*'If the SetPrs device doesn't support UseReady then look at if the RefPrs device does. The presumption*

*'is that either the controller or the reference device will support the UseReady method. Exit the loop*

*'once one of these two devices indicates Ready.*

**If** cConfig.SetPrs(1).RangeMain.Useready **Then**

cDebug.LogStatus "Wait for ready...SetPrs.Ready=" & cConfig.SetPrs(1).Ready

**If** cConfig.SetPrs(1).Ready **Then** **Exit** **Do**

**Else**

cDebug.LogStatus "Wait for ready...RefPrs.Ready=" & cConfig.RefPrs(1).Ready

**If** cConfig.RefPrs(1).Ready **Then** **Exit** **Do**

**End If**

**Loop**

**mensorCalSled\_WaitForReady** = **True**

**End Function**

**Function** **mensorCalSled\_Date**()

yy = right(Year(**Date**),2)

dd = Right(**String**(2, "0") & Day(**Date**), 2)

mm = Right(**String**(2, "0") & Month(**Date**), 2)

**mensorCalSled\_Date** = mm & dd & yy

**End Function**

**mensorCalSled\_AbsGauge code:**

*'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*'Used to calibrate and adjust gauge and absolute mode CPR6000 modules using the mensor*

*'cal sled. Called by the master test macro "mensorCalSled\_Master".*

*'*

*'This version includes logic to test the As Found data for being inside of 70%*

*'of Tolerance. If it is then the test ends; if not then adjustments are made*

*'and the test runs a second cycle.*

*'*

*'This version include range querying information. This macro works for both*

*'Range 1 and Range 2.*

*'*

*'The Test Definition must be configured for two pressure cycles, and the macro*

*'is called as a Test Event macro.*

*'16 Aug 2019: edited to include the Exp numberic conversion.*

*'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**Function** **mensorCalSledGauge**(iT, iL, iC, iP, cTest, cConfig)

**Dim** SC\_Removed\_Data()

**Dim** Reference\_Data()

**Dim** SC\_Corrected\_Data()

**Dim** res(4,7)

ZC = 0

SC = 0

*'<currentteststep 1010 - record AF coefs.>*

**If** cCOMPASS.CurrentTestStep = 1010 **Then** *' Test Definition data files were created for each DUT.*

cDebug.LogStatus "Capture As Found Coeffs"

**For** i = 1 **To** cConfig.DUTPrs.Count

cDebug.LogStatus "DUT: " & i

*' Read module range info*

Rnge = cConfig.DUTPrs(**CInt**(i)).IoSendCommand("#1B?", **False**)

RngeP = cConfig.DUTPrs(**CInt**(i)).IoSendCommand("#1R+?", **False**)

cDebug.LogStatus "Module Range being calibrated: " & Rnge

cDebug.LogStatus "Module Range FS (psi): " & RngeP

*' Read A/F ZC from module; write to calcoef1:*

ZCraw = cConfig.DUTPrs(**CInt**(i)).IoSendCommand("#1ZC?", **False**)

ZC = **Mid**(ZCraw,6,10)

cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient1 = ZC

cDebug.LogStatus "Command #1ZC?: " & ZC

cDebug.LogStatus "ZC Coefficient: " & cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient1

*' Read A/F SC from module; write to calcoef2*

SCraw = cConfig.DUTPrs(**CInt**(i)).IoSendCommand("#1SC?", **False**)

SC = **Mid**(SCraw,6,10)

cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient2 = SC

cDebug.LogStatus "Command #1SC?: " & SC

cDebug.LogStatus "SC Coefficient: " & cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient2

*' There is no Zoffset used with the CPR6000*

zOffset = 0

calDate = cConfig.DUTPrs(**CInt**(i)).IoSendCommand("#1DC?", **False**)

cCOMPASS.DataCollection(i).DUT.LastCalDate = calDate

cDebug.LogStatus "Command #1DC?: " & calDate

cDebug.LogStatus "LastCalDate: " & cCOMPASS.DataCollection(i).DUT.LastCalDate

**Next**

cdebug.LogStatus "End of CurrentTestStep 1010"

**ElseIf** cCOMPASS.CurrentTestStep = 1150 And iC = 1 **Then**

cCOMPASS.StatusDisplay "Calculating Adjustment..."

cCOMPASS.**TimeDelay** 3

cDebug.LogStatus "First Cycle is complete. Starting CurrentTestStep 1150 - Evaluating data"

cDebug.LogStatus "#Data files: " & cCOMPASS.DataCollection.Count

*'Evaluate A/F data for comparison to 70% of DUT Tolerance:*

*'70% DATA EVALUATION STAGE Evaluates if A/F data is within 70% of DUT Tolerance. If it is, then end the test.*

TOLFACT = .70 *'70% DUT Tolerance. User definable.*

**For** i = 1 **To** cCOMPASS.DataCollection(1).NumberofPressurePoints

**For** j = 1 **To** cConfig.DUTPrs.Count

Ref = cCOMPASS.DataCollection(j).DataPointRef(1,1,1,**CInt**(i)).RefPressure

DUT = cCOMPASS.DataCollection(j).DataPointRef(1,1,1,**CInt**(i)).DUTPressure

Tol = cCOMPASS.DataCollection(j).DataPointRef(1,1,1,**CInt**(i)).Tolerance

ST = cCOMPASS.DataCollection(j).DataPointRef(1,1,1,**CInt**(i)).Status

cDebug.LogStatus "DUT Tolerance is: "& Tol

**If** abs(DUT-Ref) > Tol \* TOLFACT **Then**

FAIL = **True**

cDebug.Logstatus "Test point error(s) were outside 70% of Tolerance."

cCOMPASS.Statusdisplay "Test point error(s) were greater than 70% of Tolerance. Preparing To apply adjustments and run an As Left verification."

cCOMPASS.**TimeDelay** 6

**Exit** **For**

**Else**

cDebug.Logstatus "Test point agreement is inside 70% of Tolerance. A/F data = A/L data. "

cCOMPASS.StatusDisplay "Test point agreement is inside of 70% of tolerance. No adjustments will be made. Preparing to finish calibration."

cCOMPASS.**TimeDelay** 6

**End If**

**Next**

**If** fail = **True** **Then** **Exit** **For**

cDebug.LogStatus "Ending test after 1 cycle. Disregard the next 1 second of MacStat entries - no edits to coefs were applied. "

cCOMPASS.AbortTest **True**

**Next**

*'END OF 70% TOL CHECK.*

*'If outside of 70% Tolerance then begin the adjustment based on As Found data.*

*'Back out the A/F coefficients...*

**For** i = 1 **To** cConfig.DUTPrs.Count

pressurePoints = cCOMPASS.DataCollection(i).NumberofPressurePoints

cDebug.LogStatus "DUT: " & i & " - Pressure Points: " & pressurePoints

**Redim** Reference\_Data(pressurePoints-1)

**Redim** SC\_Removed\_Data(pressurePoints-1)

**Redim** SC\_Corrected\_Data(pressurePoints-1)

*'populate module A/F coefficients from memory:*

ZC = cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient1

SC = cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient2

zOffset = 0

cDebug.LogStatus "ZC: " & ZC

cDebug.LogStatus "SC: " & SC

dutUnit = cConfig.DUTPrs(i).RangeMain.UnitFinal

cDebug.LogStatus "DUT UnitFinal: " & dutUnit

cDebug.LogStatus "DUT UnitFinalText: " & cConfig.DUTPrs(i).RangeMain.UnitFinalText

refUnit = cConfig.RefPrs(i).RangeMain.UnitFinal

cDebug.LogStatus "Ref UnitFinal: " & refUnit

cDebug.LogStatus "Ref UnitFinal: " & cConfig.RefPrs(i).RangeMain.UnitFinalText

*' Iterate through each pressure point*

**For** j = 1 **To** pressurePoints

ix = **CInt**(j)-1

cDebug.LogStatus "mensorAdjust, DUTPressurePoint(" & j & ")"

DUTPressure = cCOMPASS.DataCollection(i).DataPointRef(1, 1, 1, **CInt**(j)).DUTPressure

cDebug.LogStatus "mensorAdjust, DUTPressure: " & DUTPressure

DUTPsiPres = cCOMPASS.UnitConversion(**CDbl**(DUTPressure), 9, **CInt**(dutUnit), 0)

cDebug.LogStatus "mensorAdjust, DUTPsiPres: " & DUTPsiPres

*'this is our Pressure with A/F coefs applied. In y=mx+b it is our "y"*

refPres = cCOMPASS.DataCollection(i).DataPointRef(1, 1, 1, **CInt**(j)).RefPressure

cDebug.LogStatus "mensorAdjust, RefPres: " & refPres

Reference\_Data(ix) = cCOMPASS.UnitConversion(**CDbl**(refPres), 9, **CInt**(refUnit), 0)

cDebug.LogStatus "mensorAdjust, RefPsiPres: " & Reference\_Data(ix)

*'Determine raw zero and slope conditions. The goal is to determine the raw "x".*

*'NOTE: the CPC600 modules don't use a zoffset coefficient. Therefore this value is defined as 0 for the math.*

zOff\_Removed = **CDbl**(DUTPsiPres) - 0

ZC\_Removed = zOff\_Removed - ZC

*'backs out the offset*

*'(y-b) = mx*

cDebug.LogStatus "mensorAdjust, ZC\_Removed: " & ZC\_Removed

SC\_Removed\_Data(ix) = ZC\_Removed / SC

*'the result is our raw "x" pressure*

*'(y-b)/m = x*

cDebug.LogStatus "mensorAdjust, SC\_Removed\_Data: " & SC\_Removed\_Data(ix)

**Next**

*'Determine new A/L Zero and Slope...*

*'Poly-Fit is a COMPASS global code module for performing a linear regression on a data collection.*

**Call** **Poly\_Fit**(Reference\_Data, SC\_Removed\_Data, res, 1)

slope = res(0,1)

cDebug.LogStatus "mensorAdjust, slope: " & slope

**If** slope = 0 **Then**

new\_SC = 0

**Else**

new\_SC = 1 / slope

**End If**

cDebug.LogStatus "calculated new\_SC: " & new\_SC

*'Calculate the SC\_corrected for each point:*

**For** j = 0 **To** pressurePoints - 1

SC\_Corrected\_Data(**CInt**(j)) = SC\_Removed\_Data(**CInt**(j)) \* new\_SC

cDebug.LogStatus "mensorAdjust, SC\_Corrected\_DataPt(" & j & "): " & SC\_Corrected\_Data(**CInt**(j))

**Next**

**Call** **Poly\_Fit**(Reference\_Data, SC\_Corrected\_Data, res, 1)

new\_ZC = 0 - res(0,0)

cDebug.LogStatus "variable new\_ZC is: " & new\_ZC

*'Data type conversion:*

dval = **CDbl**(new\_ZC)

sval = FormatNumber(new\_ZC,6)

sval = trim(sval) & "0000000000"

sval = left(sval,10)

new\_ZC = sval

cmd = "#1ZC " & sval

*'Save new coefs to memory AND write to the module but not yet saved to the module:*

cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient4 = new\_ZC

cConfig.DUTPrs(**CInt**(i)).IoSendCommand "#1CALU",**True**

cCOMPASS.**TimeDelay** 1

**Call** cConfig.DUTPrs(**CInt**(i)).IoSendCommand(**CStr**(cmd), **True**)

cCOMPASS.**TimeDelay** 2

cDebug.LogStatus "CalibrationCoefficient4: " & cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient4

cDebug.LogStatus "Sent command: " & cmd

cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient5 = new\_SC

*'Data type conversion:*

dval = **CDbl**(new\_SC)

sval = FormatNumber(new\_SC,6)

sval = trim(sval) & "0000000000"

sval = left(sval,10)

new\_SC = sval

cmd = "#1SC " & sval

cConfig.DUTPrs(**CInt**(i)).IoSendCommand "#1CALU",**True**

cCOMPASS.**TimeDelay** 1

**Call** cConfig.DUTPrs(**CInt**(i)).IoSendCommand(**CStr**(cmd), **True**)

cCOMPASS.**TimeDelay** 2

cDebug.LogStatus "CalibrationCoefficient5: " & cCOMPASS.DataCollection(i).DUT.CalibrationCoefficient5

cDebug.LogStatus "Sent command: " & cmd

*'Determine new date of calibration, save to memory AND write to module:*

new\_date = **mensorCalSled\_Date**

cCOMPASS.DataCollection(i).DUT.CalDueDate = **CDate**(new\_date)

cmd = "#1DC " & new\_date

cConfig.DUTPrs(**CInt**(i)).IoSendCommand "#1CALU",**True**

**Call** cConfig.DUTPrs(**CInt**(i)).IoSendCommand(**CStr**(cmd), **True**)

cDebug.LogStatus "CalDueDate: " & cCOMPASS.DataCollection(i).DUT.CalDueDate

cDebug.LogStatus "Sent command: " & cmd

**Next**

**Set** obj = **Nothing**

cdebug.LogStatus "Finished with CurrentTestStep 1150"

*'<currentteststep 2000 - test is complete, A/L data has been collected. Save the coefs>*

*' Test complete. The post test options have not displayed and the user notes have not been entered.*

**ElseIf** cCOMPASS.CurrentTestStep = 2000 **Then**

cDebug.LogStatus "Test complete. CurrentTestStep 2000."

*'OPTIONAL GUARD BANDING FEATURE: This block of code can be commented out if guardbanding evaluation is not desired.*

TOLFACT = .50 *'50% gaurdband. User definable.*

**For** i = 1 **To** cCOMPASS.DataCollection(1).NumberofPressurePoints

**For** j = 1 **To** cConfig.DUTPrs.Count

Ref = cCOMPASS.DataCollection(j).DataPointRef(1,1,2,**CInt**(i)).RefPressure

DUT = cCOMPASS.DataCollection(j).DataPointRef(1,1,2,**CInt**(i)).DUTPressure

Tol = cCOMPASS.DataCollection(j).DataPointRef(1,1,2,**CInt**(i)).Tolerance

ST = cCOMPASS.DataCollection(j).DataPointRef(1,1,2,**CInt**(i)).Status

**If** abs(DUT-Ref) > Tol \* TOLFACT **Then**

FAIL = **True**

cCOMPASS.DataCollection(j).DataPointRef(1,1,2,**CInt**(i)).Status = ST & "T"

*'Generate Failure message for user*

GB = msgbox("The As Left results did not fall inside the guardbanded tolerance. Click OK to continue.",0+48+4096, "Guard Band Warning")

**Exit** **For**

**End If**

**Next**

**If** fail = **True** **Then** **Exit** **For**

**Next**

*'END OF GUARD BANDING CODE.*

msg = "The test is complete. The coefficients are active in module memory but "

msg = msg & "have not been written to permanent memory. Would you like to activate "

msg = msg & "the calibration?"

**If** msgbox(msg,vbquestion + vbYesNo + vbSystemModal,"Activate Calibration") = vbNO **Then**

msg = "Cycling the power of the module will "

msg = msg & "remove the coefficients from the memory and reset the module."

msgbox msg,vbSystemModal,"Activate Calibration"

**Exit** **Function**

**Else**

**For** i = 1 **To** cConfig.DUTPrs.Count

cmd = "#1SAVE"

**TimeDelay** 2

**Call** cConfig.DUTPrs(**CInt**(i)).IoSendCommand(**CStr**(cmd), **True**)

cDebug.LogStatus "Sent command: " & cmd

cDebug.LogStatus "End of Adjustment macro operations."

**Next**

**End If**

**End If**

**End Function**