MET/CAL FLEXIBLE STANDARDS

This paper is divided into three major sections.

Section one covers a brief overview of the definition, purpose and implementation of Flexible Standards in MET/CAL.

Section two provides the information needed to get MET/CAL setup to run procedures that have been designed to use the Flexible Standards technique. New procedures provided by Fluke will use the Flexible Standards technique wherever it is appropriate, particularly in RF procedures where many different (but equivalent) instruments like counters and signal generators are in wide use.

Section three adds detail on how Flexible Standards is implemented. This information will be required if you choose to write your own procedures using the Flexible Standards technique. Understanding of this section will require some knowledge of the MET/CAL procedure language and the use of sub-procedures.

Section One

What is "Flexible Standards"?

Within a traditional MET/CAL procedure, each standard used to source or measure a tested parameter must be explicitly defined in the procedure file. This is generally done by including an appropriate Function Select Code (FSC) in the procedure at each test step requiring the standard. Most FSC's in MET/CAL are instrument model specific. If you want to use a 5720A for a test step you use the 5720 FSC in your MET/CAL procedure. Likewise, for a 5520A, use a 5520 FSC, etc.

But suppose you need a procedure that is written to use a PM6680 counter, but your lab owns an HP5334A instead. You're now faced with two problems. First the procedure will have to be modified to use the HP5334A. The second problem - which may be harder to overcome - is that MET/CAL doesn't provide an FSC to control a 5334A. This means that you are now responsible for both the metrology of the measurement and also for sending control strings to the 5334A from within the procedure.

Even if there is an FSC for the standard you need to substitute into your procedure, you are still required to edit the procedure file and replace the 6680 lines with the new replacement FSC. Your challenge in a nutshell...

- Is there someone around that knows how to edit a MET/CAL procedure?
- Can you find the time it takes to make the substitution?
- How will you manage the new procedure version?
- If there is no FSC for your standard, you must add control lines in your new procedure.

"Flexible Standards" (FS)provides a solution to this problem.

> ***Flexible Standards** is a MET/CAL technique that allows the operator to interchange any reference instrument with another, specially configured instrument, of the same functional class without necessitating procedure modification.*

When is the use of Flexible Standards Appropriate?

FS is best suited to those categories of remotely controllable standards which include many different models with essentially the same functionality. These are the types of instruments that have similar functional capabilities but each model possibly has different range points, different specifications and, most probably, has different control commands. Instruments like Signal Generators, Function Generators and Frequency Counters are prime candidates and are supported with the introduction of FS in MET/CAL 7.2.

Conceivably, any programmable standard can be configured in MET/CAL as a "Flexible Standard." But FS is best applied to simpler instruments because of the amount of work required to create and test the necessary instrument control files.

Limitations of Using Flexible Standards

Whenever you use Flexible Standards, you will give up some capabilities of using regular FSC's.

No Editor Based TUR Checking

When using the MET/CAL editor to write or modify procedures, you will not get TUR calculations for those test steps that use Flexible Standards. The reason, of course is because there is no way to know what instrument will be used on the final workstation when the procedure is executed.

Choosing an Adequate Standard

The standard that is actually used during the calibration process is determined by whoever configures the standards for the workstation. This moves the responsibility for choosing an instrument with sufficient performance to perform the calibration away from the procedure writer.

No State Checking

When an FSC is designed, it is common practice to determine the correct sequence of commands needed to transition the instrument between states. This is a typical requirement where function changes require interim commands. Since control of the standard is left to the procedure writer, it is up to you to add any intermediate commands, resets or delays to switch states of your calibrator properly.

How does Flexible Standards Work?

Overview

The new MET/CAL Flexible Standards feature is implemented with the use of sub-procedures and a special initialization file (user_config_instr.ini). Interaction between the main calibration procedure and a FS instrument is directed through a sub-procedure that has been designed to control a specific class of standards. Parameter values are passed between the main procedure and the driver sub-procedure using *Named Variables*. The actual command strings needed to control the FS instrument are stored in the initialization file with a section dedicated to each specific standard model. The driver sub-procedure will lookup the required control string from the initialization file and send it to the physical instrument as needed for the specific test.

How Does MET/CAL Know Which Model to Choose?

In each MET/CAL workstation, instruments that need to be used as Flexible Standards will be configured just like all the other standards used in the system, but with one addition, the Alias name will contain the Flexible Standards *Class Name*. The MET/CAL procedure will contain the Alias name at each test step that requires the Flexible Standard instrument. MET/CAL's Run Time application will be able to associate the Alias name with an actual instrument model.

What is a Flexible Standards Class?

The instruments used as flexible standards are grouped into like functionality or *classes*. A driver sub-procedure(s) is created for each class. MET/CAL 7.2 includes sub-procedure drivers for the following pre-defined flexible standards classes that you can use right away. For efficiency reasons, the included driver sub-procedures have been created as a pair of procedure files; more information is provided about that later.

The Flexible Standard classes included in MET/CAL 7.2 are:

MET/CAL 7.2 Flexible Standard Classes

NAME	CLASS TYPE	DRIVER PROC FILES
LFCTR	Low Frequency Counter	sub_driver_lfctr.txt sub_send_cmd_lfctr.txt
HFCTR	High Frequency Counter	sub_driver_hfctr.txt sub_send_cmd_hfctr.txt
UWCTR	Microwave Frequency Counter	sub_driver_uwctr.txt sub_send_cmd_uwctr.txt
DMM	Digital Multimeter	sub_driver_dmm.txt sub_send_cmd_dmm.txt
FGEN	Function Generator	sub_driver_fgen.txt sub_send_cmd_fgen.txt
LFSG	Low Frequency Signal Generator	sub_driver_lfsg.txt sub_send_cmd_lfsg.txt
HFSG	High Frequency Signal Generator	sub_driver_hfsg.txt sub_send_cmd_hfsg.txt
UWSG	Microwave Signal Generator	sub_get_options_uwsg.txt
LVLG	Level Generator	sub_driver_lvlg.txt sub_send_cmd_lvlg.txt

MET/CAL 7.2 Flexible Standard Classes		
SWPG	Sweep Generator	sub_driver_swpg.txt sub_send_cmd_swpg.txt
LO	Local Oscillator	sub_driver_lo.txt sub_send_cmd_lo.txt

Section Two

How Do I Prepare MET/CAL to Use Flexible Standards?

Many of the newly developed MET/CAL procedures coming from Fluke use the new FS technique. To use these new procedures, perform the following steps:

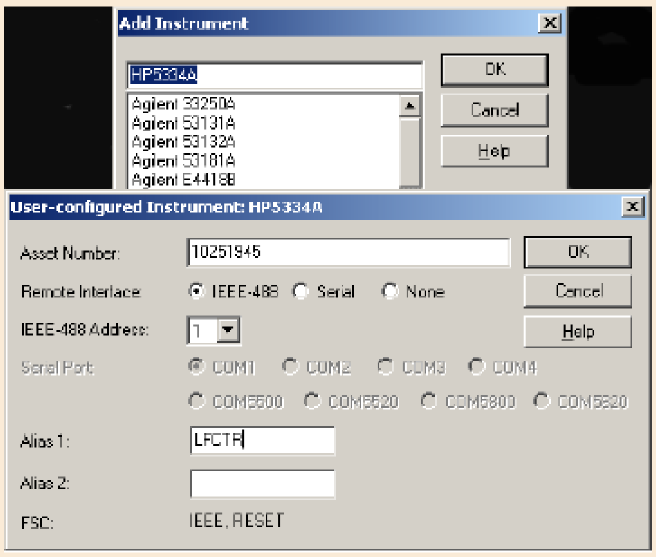
1. ### Determine if the instrument you want to use is supported.

   You can check Appendix A of this document for a list of instruments you can use with MET/CAL v7.2.

   If a Fluke provided procedure uses a model that is not listed, then a new FS initialization file is available. If the new file is not provided with the procedure, it will be made available for download on Fluke's WEB site.

2. ### Add your instrument to MET/CAL as "User Configured".

   In order for MET/CAL to make the connection between the standards class name used in the procedure and the actual instrument you want to use, you must configure your workstation by adding your standard. This is done using the Run Time or Editor application.

   - Click on [Configure]
   - Select [Add]
   - Enter a name for this instrument that matches one of the model names provided in Appendix A, exactly.
   - Fill out the details including
     - Asset Number
     - type of remote interface
     - If IEEE-488, provide address on the bus
     - Provide an "Alias" value corresponding to a class name appropriate for this flexible standard. See Appendix A.

   Note:
   - *Many instruments used as Flexible Standards can fulfill the requirements of more than one FS Class. When this is the case, you can use that instrument in both capacities by using the second Alias name to define the second FS Class.*
   - *If your selected instrument is already configured (as "user configured") in your system, you may retain the Alias name already defined and use the second Alias name to configure it as a Flexible Standard.*

3. ### *Additions to metcal.ini*

   In the [startup] section of the metcal.ini file, verify that the following line exists:

   ```
   rinfdir            = C:\metcal
   ```

   This line specifies where the flexible standards initialization file (user_config_instr.ini) is located. The actual directory may be different on your system, particularly if you are using a host with multiple workstations connected. The flexible standards initialization file should be placed where the main MET/CAL program files are located.

Section Three

Details

If you only intend to use FS in procedures supplied from Fluke, you do not need to delve into the implementation details presented in the following topics.

Typical Usage

As an example of how FS is typically used, we will explore the LFCTR class. There are 5 main actions needed to use instruments in this class. These are: Initialize, Reset, Measure, Setup and Read. Note that all of the driver sub-procedures exist in one procedure file. This is possible because each procedure file can have up to 6 Instrument names. If more than 6 actions are needed for a particular FS model, the required sub-procedures can be written in as many files as are required. The only important thing is that the procedure code for each action is contained in its own sub-procedure name.

For the LFCTR class, the relevant sub-procedure names are:

INSTRUMENT:	*Sub Initialize /LFCTR*
INSTRUMENT:	*Sub Reset /LFCTR*
INSTRUMENT:	*Sub Measure /LFCTR*
INSTRUMENT:	*Sub Setup /LFCTR*
INSTRUMENT:	*Sub Read /LFCTR*

Initialize

The first action to be accomplished is Initialize. This action is primarily used to set named memory variables to an initial state in preparation for controlling the flexible standard and capturing a reading to be used in a subsequent test evaluation.

Your mainline procedure will call the appropriate sub-procedure to complete the initialization action:
`CALL Sub Initialize /LFCTR`

Below is the initialization section of the driver sub-procedure Sub Initialize /LFCTR

Code Review for Sub Initialize /LFCTR

Line 2.002 MET/CAL will return the actual model name you have configured in your system with the alias LFCTR.

Lines 2.003-2.004 Store the section name in the initialization file for the configured LFCTR in memory variable LFCTR_ProgSecName then transfer that name to MEM2.

Line 2.005 Determine the control type of the configured LFCTR instrument by looking up the values (IEEE, IEEE2 or SCPI) in the initialization file. Store the value in memory variable LFCTR_FSC.

Lines 2.006 - 2.010 Get the names of terminals on the configured LFCTR instrument from the initialization file and store those values in named memory variables. This will allow the main procedure to display accurate connection messages to the user that match the actual instrument used.

Lines 2.011-2.027 Initialize named memory variables for each of the setup parameters required by the configured LFCTR to enable its operation.

Lines 2.028-2.035 Lookup the proper command to reset the configured LFCTR instrument from the initialization file.

```
# ================================ Initialize ===================================

  2.001  LABEL        INITIALIZE
# Get and store device name.
  2.002  MATH         @LFCTR_DevName = INSTR("LFCTR")

# Get and store programming section name.
  2.003  MATH         MEM2 = RINFE(@LFCTR_DevName, "ProgSecName")
  2.004  MATH         @LFCTR_ProgSecName = MEM2

# Get and store FSC.
  2.005  MATH         @LFCTR_FSC = RINFE(@LFCTR_ProgSecName, "FSC")

# Get and store terminal names.
  2.006  MATH         @LFCTR_Ch1    = RINFE(@LFCTR_ProgSecName, "Ch1")
# Use RINF instead of RINFE because some counters have only one channel.
  2.007  MATH         @LFCTR_Ch2    = RINF(@LFCTR_ProgSecName, "Ch2")
  2.008  MATH         @LFCTR_RefIn  = RINFE(@LFCTR_ProgSecName, "RefIn")
  2.009  MATH         @LFCTR_RefOut = RINFE(@LFCTR_ProgSecName, "RefOut")
  2.010  MATH         @LFCTR_ExtArm = RINFE(@LFCTR_ProgSecName, "ExtArm")

# Initialize parameters to the empty string (unset).
  2.011  MATH         @LFCTR_Func     = ""
  2.012  MATH         @LFCTR_Ch1Attn  = ""
  2.013  MATH         @LFCTR_Ch2Attn  = ""
  2.014  MATH         @LFCTR_Ch1Cpl   = ""
  2.015  MATH         @LFCTR_Ch2Cpl   = ""
  2.016  MATH         @LFCTR_Ch1Slope = ""
```

```
  2.017  MATH        @LFCTR_Ch2Slope = ""
  2.018  MATH        @LFCTR_Ch1Lvl   = ""
  2.019  MATH        @LFCTR_Ch2Lvl   = ""
  2.020  MATH        @LFCTR_Ch1Hyst  = ""
  2.021  MATH        @LFCTR_Ch2Hyst  = ""
  2.022  MATH        @LFCTR_Ch1Imp   = ""
  2.023  MATH        @LFCTR_Ch2Imp   = ""
  2.024  MATH        @LFCTR_Ch1Lpf   = ""
  2.025  MATH        @LFCTR_COM      = ""
  2.026  MATH        @LFCTR_MeasTime = ""
  2.027  MATH        @LFCTR_ROSC     = ""

# Get programming string for RESET FSC.
  2.028  MATH        ResetCmd = RINF(@LFCTR_ProgSecName, "ResetFSC")

# If ResetFSC is defined, establish the RESET FSC.
  2.029  IF          NOT(EMPTY(ResetCmd))

  2.030  IF          ZCMPI(ResetCmd, "[SDC]")
  2.031  RESET         [@LFCTR][SDC]
  2.032  ELSE
  2.033  RESET         [@LFCTR][V ResetCmd]
  2.034  ENDIF

  2.035  ENDIF

# See if input termination other than EOI is specified.
  2.036  MATH        InputTerm = RINF(@LFCTR_ProgSecName, "TERM")

# See CR or LF termination was specified...
  2.037  IF          ZCMPI(InputTerm, "CR")
  2.038  IEEE          [@LFCTR][TERM CR]
  2.039  ELSEIF        ZCMPI(InputTerm, "LF")
  2.040  IEEE          [@LFCTR][TERM LF]
  2.041  ENDIF

  2.042  END
```

Reset

To be sure that the configured LFCTR instrument is in a known reset state, your main procedure will call the reset driver sub-procedure:

```
CALL Sub Reset /LFCTR
```

Below is the reset section of the driver sub-procedure Sub Reset /LFCTR

Code Review for Sub Reset /LFCTR

Lines 3.001-3.005 Store the complete reset command string in named memory variable LFCTR_Cmd, then call the Sub Send Command /LFCTR to send the reset command to the configured LFCTR instrument. Noticed this is done by another sub-procedure Sub Send Command /LFCTR. The actual interaction with the LFCTR instrument has been broken out into its own sub-procedure to allow this code to be reused in multiple driver sub-procedures without duplicating these procedure steps.

```
# ================================= Reset
===================================

  3.001  LABEL       RESET
  3.002  MATH        @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, "Reset")
  3.003  CALL        Sub Send Command /LFCTR
  3.004  END
```

Getting Ready for a Measurement

Now we have all of the named variables loaded with the setup strings needed to send a complete connection message to the operator in preparation for making a measurement.

Example Main Line procedure

```
  3.007  DISP          Make the following connections:
  3.007  DISP
  3.007  DISP          [32]   UUT (rear panel)     to     [V @LFCTR_DevName]
  3.007  DISP          [32]  REF FREQUENCY OUT -------> [V @LFCTR_RefIn]
  3.007  DISP
  3.007  DISP          [32]  UUT (9640A-50)      to     [V @LFCTR_DevName]
  3.007  DISP          [32]   Leveling Head -------> [V @LFCTR_Ch1]
```

Now measurement parameters will be set in the main line procedure for the type of measurement *action we want the configured LFCTR instrument to perform:*

Example Main Line procedure

```
3.010  MATH        @LFCTR_ROSC = "Ext"
3.011  MATH        @LFCTR_MeasTime = "2s"
3.012  MATH        @LFCTR_Func = "FreqCh1"
3.013  MATH        @LFCTR_Ch1Imp = "LoZ"
```

Measure

Your mainline procedure will call the appropriate sub-procedure to complete the measure action: `CALL Sub Measure /LFCTR`

Below is the measure section of the driver sub-procedure Sub Measure /LFCTR

Code Review for Sub Measure /LFCTR

Lines 4.002-4.139 This section examines the values of each setup variable and for non-empty strings, sends these setup values to the configured LFCTR instrument.

Lines 4.131-4.145 These lines command the configured LFCTR instrument to return a measurement reading. The "[I]" syntax causes the measurement value is returned to the main line procedure in the MEM variable.

```
# ============================== Setup or Measure ============================

  4.001  LABEL       SETUP

# -----  Function

  4.002  MATH        @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, @LFCTR_Func)
  4.003  CALL        Sub Send Command /LFCTR

# -----  Measurement Time

  4.004  IF          NOT(EMPTY(@LFCTR_MeasTime))
  4.005  MATH        Cmd = RINFE(@LFCTR_ProgSecName, "MeasTime")
# Convert to base units and insert in programming string.
  4.006  MATH        @LFCTR_Cmd = REPL("<val>", BASE(@LFCTR_MeasTime), Cmd)
  4.007  CALL        Sub Send Command /LFCTR
  4.008  ENDIF

# -----  Reference Oscillator

  4.009  IF          NOT(EMPTY(@LFCTR_ROSC))

  4.010  IF          ZCMPI(@LFCTR_ROSC, "Int")
  4.011  MATH        RefOsc = "RefOscInt"
  4.012  ELSE
  4.013  MATH        RefOsc = "RefOscExt"
  4.014  ENDIF

  4.015  MATH        @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, RefOsc)
  4.016  CALL        Sub Send Command /LFCTR
  4.017  ENDIF

# -----  Channel 1 Input Impedance

  4.018  IF          NOT(EMPTY(@LFCTR_Ch1Imp))

  4.019  IF          ZCMPI(@LFCTR_Ch1Imp, "LoZ")
  4.020  MATH        Imp = "Ch1Imp50_Ohm"
  4.021  ELSE
  4.022  MATH        Imp = "Ch1Imp1_MOhm"
  4.023  ENDIF

  4.024  MATH        @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Imp)
  4.025  CALL        Sub Send Command /LFCTR
  4.026  ENDIF

# -----  Channel 1 Input Coupling

  4.027  IF          NOT(EMPTY(@LFCTR_Ch1Cpl))

  4.028  IF          ZCMPI(@LFCTR_Ch1Cpl, "AC")
  4.029  MATH        Cpl = "Ch1CplAC"
  4.030  ELSE
  4.031  MATH        Cpl = "Ch1CplDC"
  4.032  ENDIF

  4.033  MATH        @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Cpl)
  4.034  CALL        Sub Send Command /LFCTR
  4.035  ENDIF

# -----  Channel 1 Input Attenuation
```

```
 4.036  IF          NOT(EMPTY(@LFCTR_Ch1Attn))

 4.037  IF          ZCMPI(@LFCTR_Ch1Attn, "x10")
 4.038  MATH          Attn = "Ch1Attn_x10"
 4.039  ELSE
 4.040  MATH          Attn = "Ch1Attn_x1"
 4.041  ENDIF

 4.042  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Attn)
 4.043  CALL         Sub Send Command /LFCTR
 4.044  ENDIF

# -----  Channel 1 Low-pass Filter

 4.045  IF          NOT(EMPTY(@LFCTR_Ch1Lpf))

 4.046  IF          ZCMPI(@LFCTR_Ch1Lpf, "On")
 4.047  MATH          Lpf = "Ch1LpfOn"
 4.048  ELSE
 4.049  MATH          Lpf = "Ch1LpfOff"
 4.050  ENDIF

 4.051  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Lpf)
 4.052  CALL         Sub Send Command /LFCTR
 4.053  ENDIF

# -----  Channel 1 Trigger Slope

 4.054  IF          NOT(EMPTY(@LFCTR_Ch1Slope))

 4.055  IF          ZCMPI(@LFCTR_Ch1Slope, "Pos")
 4.056  MATH          Slope = "Ch1SlopePos"
 4.057  ELSE
 4.058  MATH          Slope = "Ch1SlopeNeg"
 4.059  ENDIF

 4.060  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Slope)
 4.061  CALL         Sub Send Command /LFCTR
 4.062  ENDIF

# -----  Channel 1 Trigger Level

 4.063  IF          NOT(EMPTY(@LFCTR_Ch1Lvl))
 4.064  MATH          Cmd = RINFE(@LFCTR_ProgSecName, "Ch1TrigLevel")
 4.065  MATH          @LFCTR_Cmd = REPL("<val>", BASE(@LFCTR_Ch1Lvl), Cmd)
 4.066  CALL         Sub Send Command /LFCTR
 4.067  ENDIF

# -----  Channel 1 Trigger Hysteresis

 4.068  IF          NOT(EMPTY(@LFCTR_Ch1Hyst))
 4.069  MATH          Cmd = RINFE(@LFCTR_ProgSecName, "Ch1TrigHyst")
 4.070  MATH          @LFCTR_Cmd = REPL("<val>", BASE(@LFCTR_Ch1Hyst), Cmd)
 4.071  CALL         Sub Send Command /LFCTR
 4.072  ENDIF

# -----  Channel 2 Input Impedance

 4.073  IF          NOT(EMPTY(@LFCTR_Ch2Imp))

 4.074  IF          ZCMPI(@LFCTR_Ch2Imp, "LoZ")
 4.075  MATH          Imp = "Ch2Imp50_Ohm"
 4.076  ELSE
 4.077  MATH          Imp = "Ch2Imp1_MOhm"
 4.078  ENDIF

 4.079  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Imp)
 4.080  CALL         Sub Send Command /LFCTR
 4.081  ENDIF

# -----  Channel 2 Input Coupling

 4.082  IF          NOT(EMPTY(@LFCTR_Ch2Cpl))

 4.083  IF          ZCMPI(@LFCTR_Ch2Cpl, "AC")
 4.084  MATH          Cpl = "Ch2CplAC"
 4.085  ELSE
 4.086  MATH          Cpl = "Ch2CplDC"
 4.087  ENDIF

 4.088  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Cpl)
 4.089  CALL         Sub Send Command /LFCTR
 4.090  ENDIF

# -----  Channel 2 Input Attenuation

 4.091  IF          NOT(EMPTY(@LFCTR_Ch2Attn))

 4.092  IF          ZCMPI(@LFCTR_Ch2Attn, "x10")
 4.093  MATH          Attn = "Ch2Attn_x10"
 4.094  ELSE
 4.095  MATH          Attn = "Ch2Attn_x1"
 4.096  ENDIF

 4.097  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Attn)
 4.098  CALL         Sub Send Command /LFCTR
```

```
  4.099  ENDIF

# -----  Channel 2 Trigger Slope

  4.100  IF          NOT(EMPTY(@LFCTR_Ch2Slope))

  4.101  IF          ZCMPI(@LFCTR_Ch2Slope, "Pos")
  4.102  MATH          Slope = "Ch2SlopePos"
  4.103  ELSE
  4.104  MATH          Slope = "Ch2SlopeNeg"
  4.105  ENDIF

  4.106  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Slope)
  4.107  CALL         Sub Send Command /LFCTR
  4.108  ENDIF

# -----  Channel 2 Trigger Level

  4.109  IF          NOT(EMPTY(@LFCTR_Ch2Lvl))
  4.110  MATH          Cmd = RINFE(@LFCTR_ProgSecName, "Ch2TrigLevel")
  4.111  MATH          @LFCTR_Cmd = REPL("<val>", BASE(@LFCTR_Ch2Lvl), Cmd)
  4.112  CALL         Sub Send Command /LFCTR
  4.113  ENDIF

# -----  Channel 2 Trigger Hysteresis

  4.114  IF          NOT(EMPTY(@LFCTR_Ch2Hyst))
  4.115  MATH          Cmd = RINFE(@LFCTR_ProgSecName, "Ch2TrigHyst")
  4.116  MATH          @LFCTR_Cmd = REPL("<val>", BASE(@LFCTR_Ch2Hyst), Cmd)
  4.117  CALL         Sub Send Command /LFCTR
  4.118  ENDIF

# -----  Channel 2 COM (2 via 1)

  4.119  IF          NOT(EMPTY(@LFCTR_COM))

  4.120  IF          ZCMPI(@LFCTR_COM, "Off")
  4.121  MATH          Com = "Ch2ComOff"
  4.122  ELSE
  4.123  MATH          Com = "Ch2ComOn"
  4.124  ENDIF

  4.125  MATH          @LFCTR_Cmd = RINFE(@LFCTR_ProgSecName, Com)
  4.126  CALL         Sub Send Command /LFCTR
  4.127  ENDIF

# Exit here if Setup.
  4.128  IF          PSUBI("Setup")
  4.129  END
  4.130  ENDIF

# Drop through for Measure.

# ============================== Read or Measure ============================

  4.131  LABEL         READ
# See if there is an initiate command.
  4.132  MATH          @LFCTR_Cmd = RINF(@LFCTR_ProgSecName, "Initiate")

# If there is an initiate command, send it.
  4.133  IF          NOT(EMPTY(@LFCTR_Cmd))
  4.134  CALL         Sub Send Command /LFCTR
  4.135  ENDIF

# See if there is a fetch command.
  4.136  MATH          @LFCTR_Cmd = RINF(@LFCTR_ProgSecName, "Fetch")

# If there is no fetch command simply get the reading.
  4.137  IF          EMPTY(@LFCTR_Cmd)
  4.138  IEEE         [@LFCTR][I]
# Otherwise send the fetch command and get the reading.
  4.139  ELSEIF        ZCMPI(@LFCTR_FSC, "SCPI")
  4.140  SCPI         [@LFCTR][V @LFCTR_Cmd][I]
  4.141  ELSEIF        ZCMPI(@LFCTR_FSC, "IEEE2")
  4.142  IEEE2        [@LFCTR][V @LFCTR_Cmd][I]
  4.143  ELSE
  4.144  IEEE         [@LFCTR][V @LFCTR_Cmd][I]
  4.145  ENDIF

  4.146  END
```

Appendix A - Instruments Supported as Flexible Standards

Flexible Standard Instruments in MET/CAL 7.2	
Model	**FS Class**
Agilent 33220A	FGEN
Agilent 33250A	FGEN
Agilent 53131A	LFCTR
Agilent 53132A	LFCTR

Flexible Standard Instruments in MET/CAL 7.2	
Agilent 53181A	LFCTR HFCTR
Agilent E4400A	HFSG
Agilent E4400B	HFSG
Agilent E4420A	HFSG
Agilent E4420B	HFSG
Agilent E4421A	HFSG
Agilent E4421B	HFSG
Agilent E4422A	HFSG
Agilent E4422B	HFSG
Agilent E4423B	HFSG
Agilent E4424B	HFSG
Agilent E4425B	HFSG
Agilent E4426B	HFSG
Agilent E4430B	HFSG
Agilent E4431B	HFSG
Agilent E4432B	HFSG
Agilent E4433B	HFSG
Agilent E4434B	HFSG
Agilent E4435B	HFSG
Agilent E4436B	HFSG
Agilent E4437B	HFSG
Agilent E4438C	HFSG
Agilent E8247C	UWSC
Agilent E8257C	UWSC
Agilent E8257D	UWSC
Agilent E8267C	UWSC
Agilent E8267D	UWSC
Fluke 45	DMM
Fluke 80	FGEN
Fluke 81	FGEN
Fluke 271	FGEN
Fluke 281	FGEN
Fluke 282	FGEN
Fluke 6060A	HFSG
Fluke 6060A/AN	HFSG
Fluke 6060B	HFSG
Fluke 6061A	HFSG
Fluke 6062A	HFSG
Fluke 6080A	HFSG
Fluke 6082A	HFSG
Fluke 8840A	DMM
Fluke 8842A	DMM
Fluke 9640A	LFSG LVLG HFSG SWPG
Fluke PM 6690	LFCTR
HP 3325A	FGEN
HP 3325B	FGEN
HP 3335A	LVLG
HP 3336A	LVLG
HP 3336B	LVLG
HP 3336C	LVLG
HP 5334A	LFCTR
HP 5350A	UWCTR
HP 5350B	UWCTR
HP 5350M	UWCTR
HP 5351A	UWCTR

Flexible Standard Instruments in MET/CAL 7.2	
HP 5351B	UWCTR
HP 5351M	UWCTR
HP 5352A	UWCTR
HP 5352B	UWCTR
HP 5352M	UWCTR
HP 8340A	SWPG
HP 8340B	SWPG
HP 8341A	SWPG
HP 8341B	SWPG
HP 8642A	HFSG UWSG
HP 8642B	HFSG UWSG
HP 8643A	HFSG UWSG
HP 8644A	HFSG UWSG
HP 8644B	HFSG UWSG
HP 8645A	HFSG UWSG
HP 8647A	HFSG UWSG
HP 8648A	HFSG UWSG
HP 8648B	HFSG UWSG
HP 8648C	HFSG UWSG
HP 8648D	HFSG UWSG
HP 8656A	HFSG UWSG
HP 8656B	HFSG UWSG
HP 8657A	HFSG UWSG
HP 8657B	HFSG UWSG
HP 8662A	HFSG UWSG
HP 8663A	HFSG UWSG
HP 8664A<	HFSG UWSG
HP 8665A	HFSG UWSG
HP 8665B	HFSG UWSG
HP 8671A	HFSG UWSG
HP 8671B	HFSG UWSG
HP 8672A	HFSG UWSG
HP 8672S	HFSG UWSG
HP 8673B	HFSG UWSG
HP 8673D	HFSG UWSG
HP 8673C	HFSG UWSG

Flexible Standard Instruments in MET/CAL 7.2	
HP 8673D	HFSG UWSG
HP 8673E	HFSG UWSG
HP 8673G	HFSG UWSG
HP 8673H	HFSG UWSG
HP 33120A	FGEN
HP 34401A	DMM
HP 83620A	HFSG UWSG SWPG
HP 83620B	HFSG UWSG SWPG
HP 83622A	HFSG UWSG SWPG
HP 83622B	HFSG UWSG SWPG
HP 83623A	HFSG UWSG SWPG
HP 83623B	HFSG UWSG SWPG
HP 83624A	HFSG UWSG SWPG
HP 83624B	HFSG UWSG SWPG
HP 83630A	HFSG UWSG SWPG
HP 83630B	HFSG UWSG SWPG
HP 83640A	HFSG UWSG SWPG
HP 83640B	HFSG UWSG SWPG
HP 83642A	HFSG UWSG SWPG
HP 83650A	HFSG UWSG SWPG
HP 83650B	HFSG UWSG SWPG
HP 83711A	HFSG UWSG
HP 83711B	HFSG UWSG
HP 83712A	HFSG UWSG
HP 83712B	HFSG UWSG
HP 83731A	HFSG UWSG

Flexible Standard Instruments in MET/CAL 7.2	
HP 83731B	HFSG UWSG
HP 83732A	HFSG UWSG
HP 83732B	HFSG UWSG
HP 83751A	HFSG UWSG
HP 83751B	HFSG UWSG
HP 83752A	HFSG UWSG
HP 83752B	HFSG UWSG
Marconi 2023	HFSG
Marconi 2024	HFSG
Philips PM 6680	LFCTR
Philips PM 6681	LFCTR
Philips PM 6685	LFCTR
Rohde & Schwarz SMY02	HFSG
Rohde & Schwarz SMY03	HFSG
Rohde & Schwarz SMY43	HFSG
Tabor 8550	FGEN
Tabor 8551	FGEN
Wavetek 39A	FGEN
Wavetek 80	FGEN
Wavetek 81	FGEN
Wavetek 195	FGEN
Wavetek 395	FGEN
Wavetek 900	LFCTR
Wavetek 901	LFCTR
Wavetek 905	LFCTR