WHEN YOU NEED TO BE SURE

SGS

CyFlex® Knowledge Article

Emissions Bench Multiplexing for PAM/DARTS and EDA

Author: Daniel Oren

March 16, 2022

Background

Some gaseous emissions benches are capable of multiplexing between sample locations without the need to manually reconnect sample lines. When this happens, the names of the PAM keywords for emissions species must also change. It is important for downstream data processes to use the correct PAM keywords to identify the concentration measurements.

This document describes a CyFlex® configuration process that can be used to properly identify channels for both PAM/DARTS and Emissions Data Analysis (EDA). A specific example is used to illustrate the process, but many variations are possible.

PAM Plug-and-Play

CyFlex® plug-and-play has long been used as a method to instruct the data point task to include added channels related to portable measurement equipment when signaled to read PAM specifications. The intent is to make it possible to easily plug portable devices, such as lube oil soot carts and emissions benches, into a test cell and indicate their presence by setting a plug-and-play logical variable `ON`. A list of plug-and-play logical variables is specified in the `/cell/logi_specs` file and all are automatically created by the `sys_start` system start function.

The PAM `/specs/pnp/pam_pnp` plug-and-play file requires one or more lines with two tokens per line. The first token must be the name of a CyFlex® logical variable. For recognized plug and play devices, specify one of the existing variables in the `/cell/logi_specs` file.

The second token must be the complete path to a file having a PAM datapoint section which is to be read and added to the base specification read from the `/specs/pam_specs` file.

This capability can be extended to handle sample line multiplexing by creating logical variables that indicate which sample line is in use. The example below shows new sample line logical variables that must be created in a `gen_labels` file.

```
eng_enab        /specs/pnp/pam_specs.eng
at1_enab        /specs/pnp/pam_specs.at1
at2_enab        /specs/pnp/pam_specs.at2
exh_enab        /specs/pnp/pam_specs.exh
cvs_enab        /specs/pnp/pam_specs.cvs
```

PAM Specifications

The /specs/pnp/pam_specs files referenced in the PAM plug-and-play file have the same format as the $PAMDatapoint section of the base PAM specification file. The example below is the /specs/pam_specs.eng file referenced in the example above. The file must begin with $PAMDatapoint on the first line and end with $End.

The first token on each line will be the name of a CyFlex® channel used to receive the measured concentration data. In the example below, the data is captured in the MxConc array variable. The second token on each line can be either AVER or SNAP but for concentration measurements. Use AVER. The third token is the PAM keyword that will be used when sending data to PAM/DARTS and in the composite file that is used as input to EDA. Note that ENG indicates that the sample location is engine-out and it also corresponds to the .eng suffix on the file name.

```
$PAMDatapoint
  MxConc:HCO    AVER     ENG_CD_CO_MEA@1
  MxConc:CO2    AVER     ENG_CD_CO2_MEA
  MxConc:LCO    AVER     ENG_CD_CO_MEA@2
  MxConc:O2     AVER     ENG_CD_O2_MEA
  MxConc:THC    AVER     ENG_CW_HC_MEA
  MxConc:NOX    AVER     ENG_CD_NOX_MEA
$End
```

Consider a second example file: pam_specs.exh. This file is virtually identical except that the PAM keywords now begin with EXH to indicate that the sample location is exhaust pipe out. Similar files will exist for AT1 (first aftertreatment device outlet), AT2 (second aftertreatment device outlet) and CVS (constant volume sampling system).

```
$PAMDatapoint
  MxConc:HCO    AVER     EXH_CD_CO_MEA@1
  MxConc:CO2    AVER     EXH_CD_CO2_MEA
  MxConc:LCO    AVER     EXH_CD_CO_MEA@2
  MxConc:O2     AVER     EXH_CD_O2_MEA
  MxConc:THC    AVER     EXH_CW_HC_MEA
  MxConc:NOX    AVER     EXH_CD_NOX_MEA
$End
```

The concentration channels will be added to the list of channels used by datapoint for PAM/DARTS data acquisition and to the logger specification file logr_specs_10c.dat that is- used to gather transient emissions data. Exclude these channels from the base PAM specification file or from the @DATA_CHANNELS list in cvs_specs.dat.

Sample Line Logical Variables

The sample line logical variables used in the PAM plug-and-play file need to be tied in some way to the sample line selection mechanism. If the sample line multiplexing is built into the bench, the logic is usually obvious.

In the example below, the sample line location is indicated by a string variable, `MxSample_loc`, which is initialized by the operator. In this case, the valid strings to identify the sample location are assumed to be limited to ENG, AT1, AT2, CVS, and EXH.

The logic to set the sample line logical variables for this example has been implemented with computed expressions as shown below. Logic has been added for a `MxSample_loc` of CVS so that option is available for potential future use.

```
#************************************************************************
# The following logical variables are used in the pam_pnp file to indicate
# which sample location is currently active so the correct pam_specs file
# will be read and the contents added to PAM/DARTS datapoint files and to
# the logger used to collect data during transient tests.  Only one of these
# logical variables will be true at any given time.
#************************************************************************
#label       true_event  false_event true_desc   false_desc int/event hst_flag
eng_enab     eng_enab_ev -           enabled     disabled   SLO       OFF
if ( @strcmp_lbl_lit(MxSample_loc,'ENG') ) then ON else OFF

#label       true_event  false_event true_desc   false_desc int/event hst_flag
at1_enab     at1_enab_ev -           enabled     disabled   SLO       OFF
if ( @strcmp_lbl_lit(MxSample_loc,'AT1') ) then ON else OFF

#label       true_event  false_event true_desc   false_desc int/event hst_flag
at2_enab     at2_enab_ev -           enabled     disabled   SLO       OFF
if ( @strcmp_lbl_lit(MxSample_loc,'AT2') ) then ON else OFF

#label       true_event  false_event true_desc   false_desc int/event hst_flag
exh_enab     exh_enab_ev -           enabled     disabled   SLO       OFF
if ( @strcmp_lbl_lit(MxSample_loc,'EXH') ) then ON else OFF

#label       true_event  false_event true_desc   false_desc int/event hst_flag
cvs_enab     cvs_enab_ev -           enabled     disabled   SLO       OFF
if ( @strcmp_lbl_lit(MxSample_loc,'CVS') ) then ON else OFF
```

Logical Variable Event Response

An event will be generated in the above example whenever one of the logical variables is set `ON`. This event will be processed by the event response task. In the example event response configuration shown below, these events are used to trigger another event (`PAM_specs_cfg`) that will cause a datapoint to reread the PAM specification file and the newly enabled file associated with the logical variable via the PAM plug-and-play file.

```
#*************************************************************************
#The xxx_enab_ev is a transition event triggered when MXSample_loc is changed.
# Datapoint needs to be told to reread pam_pnp and add the associated
# emissions variables to the list to be logged.
# The maximum number of input events for one entry is four, so two entries.
#*************************************************************************
@INPUT_EVENT
  eng_enab_ev
  exh_enab_ev

@OUTPUT_EVENT
  PAM_specs_cfg

@SCRIPT
  /specs/cmds/cvs_mk_logr_auto 1>/dev/null 2>/dev/null

@END

@INPUT_EVENT
  at1_enab_ev
  at2_enab_ev
  cvs_enab_ev

@OUTPUT_EVENT
  PAM_specs_cfg

@SCRIPT
  /specs/cmds/cvs_mk_logr_auto 1>/dev/null 2>/dev/null

@END
```

A `/specs/cmds/cvs_mk_logr_auto` script is also launched by event response to generate an updated version of the `logr_specs_10c.dat` logger specification file that is used to capture data during a transient cycle. This script assumes a current, valid test plan ID is contained in the `tpid` string variable. The process that builds the `logr_specs_10c.dat` file will add all the logical variables from the PAM plug-and-play file and all the CyFlex® concentration variables listed in all the PAM specification files referenced in the PAM plug-and-play file to the list of channels to be logged. This will occur only if the environment variable `CVS_MULTIPLEXING=` exists. This is important and will be mentioned again later.

The normal transient emissions process also generates the `logr_specs_10c.dat` file when the unmodified `cvs_mk_logr` script is run. Since the content does not change based on the sample line location, having it run again by event response is just a backup in case configuration files were changed out of sequence. It is a good practice to search the `logr_specs_10c.dat` file to verify that the logical variables and concentration channels have been included in the file.

Note that the processes triggered by event response will cause the concentration channels to be added to the list used by datapoint to capture data for PAM/DARTS and to the `logr_specs_10c.dat` file used to capture data during transient emissions tests. These channels should NOT be included in the base PAM specification file or in the list under `@DATA_CHANNELS` in the `cvs_specs.dat` file.

CVS_MULTIPLEXING= Environment Variable

The existence of the environment variable `CVS_MULTIPLEXING` is used as an indicator to the processes involved in building `logr_specs_10c.dat` and the composite file used as input to EDA, that multiplexing information will be captured in the logger data. This variable can be set in the `go` script or from the command line using `export CVS_MULTIPLEXING=`.

The `esvd_star5.b09` task that is involved in building the composite file will look at the initial state of the sample line logical variables in the logger data to decide which PAM specification file should be referenced when deciding which PAM keywords should be associated with the concentration channels when the data is written to the composite file.

Emissions Bench File Configuration

Information from an `em_bench` file (emissions bench) is used when building the `*5` section of the composite file which is used as input to `em_data_analysis`. The `em_bench` file holds information related to the configuration of the emissions bench along with data gathered during pre- and post-test verifications. This file uses PAM keywords so that the information will align with the names assigned to the logger channels that are the concentration measurements.

The `em_bench` file is usually created using a flash report or by a general-purpose test procedure. Many of the records have a format like that shown below, an `@KEYWORD` followed by multiple lines of data associated with individual species measurements. In this example, the keyword shows that the values are the span verification measurements made prior to the test.

```
@CONT_CHAN_PRETEST_SPAN
  EXH_CD_CO2_MEA        156210.87  PPM
  EXH_CD_CO_MEA            504.06  PPM
  EXH_CD_O2_MEA         213231.40  PPM
  EXH_CW_HC_MEA            302.96  PPM
  EXH_CW_CH4_MEA           296.95  PPM
  EXH_CD_NOX_MEA           603.49  PPM
```

There is another format that can be used in the em_bench file that does not rely on PAM keywords for most of the records. Using the alternate format, the channel names might look like this:

```
@CONT_CHAN_PRETEST_SPAN
  MxConc:CO2            156210.87  PPM
  MxConc:CO                504.06  PPM
  MxConc:O2             213231.40  PPM
  MxConc:THC               302.96  PPM
  MxConc:CH4               296.95  PPM
  MxConc:NOX               603.49  PPM
```

The association between channel name and PAM keyword is then defined by an `@SPECIES_TAG_TO_KEYWORD_MAPPING` entry that does not appear in the `*5` section of the composite file. It is instead used to instruct the process that creates the content for the `*5` section so it can alter the other entries to use the proper PAM keyword when writing the entries.

```
@SPECIES_TAG_TO_KEYWORD_MAPPING
  MxConc:CO2              EXH_CD_CO2_MEA
  MxConc:CO               EXH_CD_CO_MEA
  MxConc:O2               EXH_CD_O2_MEA
  MxConc:THC              EXH_CW_HC_MEA
  MxConc:CH4              EXH_CW_METHANE_MEA
  MxConc:NOX              EXH_CD_NOX_MEA
```

This process is known to work independently from the PAM plug-and-play process. The combination has never been tested as this is written. There is little motivation to do that testing since if the process that writes the emissions bench file knows enough to create this entry, it also knows enough to write the other keyword entries using the current PAM keywords in the first place. This topic is mentioned in the hope that labs using the old method will convert to using PAM keywords.

MEXA Bench Processes

An attempt is underway to standardize the CyFlex® processes for communicating with Horiba emissions benches using the MEXA LAN protocol and message protocol. Part of this process involves the use of variable files to capture and store data related to bench configurations and operations such as pre- and post-test zero and span. This data is often stored in a variable file that has the `MxSample_loc` used as an index. The example below is a shortened version of the file that loads a PAM keyword into the array variables that have the keywords for a given species.

```
VERTICAL_LABELS
#*************************************************************************
# The MxPAM_keyword array is used in *5 section keywords as the primary
# identifier.  There needs to be an entry for every species for which data
# will be written to the composite file even if the data is not going to be
# used for the primary emissions calculations.  All entries should be valid
# PAM keywords.  The keywords will be initialized based on the specified
# MxSample_loc.
#*************************************************************************

                                ENG                    EXH
   MxPAM_keyword:HCO      none   'ENG_CD_CO_MEA'       'EXH_CD_CO_
   MxPAM_keyword:CO2      none   'ENG_CD_CO2_MEA'      'EXH_CD_CO2_MEA'
   MxPAM_keyword:LCO      none   'ENG_CD_CO_MEA'       'EXH_CD_CO_MEA'
   MxPAM_keyword:O2       none   'ENG_CD_O2_MEA'       'EXH_CD_O2_MEA'
   MxPAM_keyword:THC      none   'ENG_CW_HC_MEA'       'EXH_CW_HC_MEA'
   MxPAM_keyword:CH4      none   'ENG_CW_METHANE_MEA' 'EXH_CW_METHANE_MEA'
   MxPAM_keyword:NOX      none   'ENG_CD_NOX_MEA'      'EXH_CD_NOX_MEA'
   MxPAM_keyword:EGRCO2   none   'INT_MNF_CD_CO2_MEA' 'INT_MNF_CD_CO2_MEA'
```

This and similar files are read as part of the bench initialization process.