Virtual Zero and Span Reference

Version 4

February 15, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	6/8/2018	Initial publication
2	8/23/2018	Format with SGS brand
3	4/6/2020	Retrofit to new template
4	2/15/2024	Add hypertext-linked cross-references to cyfle.com usage help for `vzs` and `vzs_sensor` Rebrand to TRP Laboratories

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1 Overview* on page 1.

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

1 OVERVIEW 1
1.1 TERMINOLOGY 1
2 THEORY OF OPERATION 2
2.1 COMMON CAPABILITIES 2
2.1.1 Additional Common Behavior 2
2.2 SPECIFIC VZS CAPABILITIES 3
2.2.1 Additional vzs Behavior 3
2.3 SPECIFIC VZS_SENSOR CAPABILITIES 3
2.4 FUNCTIONS BEYOND VZS AND VZS_SENSOR 3
2.5 INTERACTION WITH OTHER CYFLEX COMPONENTS 4
3 STARTING THE APPLICATION 6
4 SPECIFICATION FILES 7
4.1 MULTI-RANGE VZS TASK SPECIFICATION FILE 7
4.2 SINGLE-RANGE VZS_SENSOR TASK SPECIFICATION FILE 14
5 EXAMPLE GP_TEST PROCEDURE 17

List of Figures

FIGURE 1: VZS INTERACTIONS WITH OTHER SYSTEM COMPONENTS 5

LIST OF TABLES

TABLE 1: TERMINOLOGY 1
TABLE 2: FUNCTIONS BEYOND VZS SCOPE 3

1 Overview

The virtual zero and span tasks supports the ability to perform bias and scaling adjustment on an input channel. There are two virtual zero and span tasks, each of which are addressed herein. These are:

1. `vzs`: the task that manages a multi-range gas analyzer, refer to cyflex.com usage help for `vzs`.
2. `vzs_sensor`: the task that manages a single range device such as a pressure transducer, refer to cyflex.com usage help for `vzs_sensor`.

These two tasks have many common attributes and vary from each other principally in the area of managing the multiple ranges vs. having only a single range.

1.1 Terminology

Table 1 describes terms used in this document.

Table 1: Terminology

Term	Definition
Virtual Span/Zero Adjustment	An operation performed on a device that is managed by the `vzs`/`vzs_sensor` task that determines an offset and/or scaling corrections that is applied prior to the application of a polynomial calibration.
Span	An external reference stimulus that is used with a zero, to determine offset and scaling corrections. A span stimulus is typically between 50% and 100% of the full range input of the device on the range being spanned.
Zero	An external reference stimulus that is used with a span, to determine a offset and scaling corrections. The zero-input reference is required to be exactly 0% of the range being spanned (i.e. a known small value cannot be accommodated).
VZS Range	The range a gas analyzer is working in according to the `vzs` task.
Device Range	The range a gas analyzer is working according to itself. **Note:** Device ranges can be mapped to VZS ranges provided that the mapping is of contiguous and increasing device ranges. Example: device range 2 through 4 can be mapped to VZS ranges 1 through 3.

2 Theory of Operation

2.1 Common Capabilities

Both `vzs/vzs_sensor` tasks have the following capabilities:

- Accept a virtual zero or span request

  This causes the task to issue a corresponding output stimulus request and to coordinate stabilization with an external service, typically a `gp_test` procedure.
- Accept an external stability complete or fail event

  Success causes the task to capture the virtual zero/span value and perform an adjustment if the other (span/zero) value is available.
- Accept a hardware zero/span request
  - This causes the task to change the active zero and span adjustments to 0 and 1 respectively so that the hardware zero and span operation is not effected by virtual zero and span corrections.
  - At completion of the h/w zero and span the raw values are used to compute a virtual zero and span correction.
- Store zero and span corrections to both shared memory and to the corresponding calibration table on disk.
- Indicate the present status of the virtual zero and span adjustment in a number of ways:
  - Whether the most recent adjustment is too old (based on a specified timeout period).
  - Whether the most recent adjustment exceeds a specified maximum adjustment threshold.

2.1.1 Additional Common Behavior

1. There is a single instance of the `vzs*` task per input channel being managed.
2. The calibration table types required for use with both of the virtual zero and span tasks is `POLYNOMIAL_RANGE`.
3. The calibration table is typically applied to the input channel using an expression in gen_labels rather than by the analog input transfer layer. This expression has the form: `“@cal_table( raw_ai_chan, ‘<table_name>’ )”.`

   This is done for the following reasons:
   - The raw input from a digital gas analyzer is not an analog input channel.
   - The `vzs` task requires access to a raw input value.
4. The numeric processed applied by the `vzs*` tasks use the Numerical Recipes in C `zbrak` and `rtsafe` functions. 10 brackets and 1000 maximum iterations are used in the root finding process. Convergence has typically been observed within 3 to 5 iterations. The root convergence tolerance criteria used is $1*10^{-9}$ of full range.
5. The calibration polynomial specified for each range has to have a root within the specified full calibration zero and span range and is required to be monotonically increasing throughout the range. At least two coefficients are required (i.e. the polynomial is required to be at least first order).

2.2 Specific vzs Capabilities

The multi-range gas analyzer `vzs` task has the following capabilities in addition to the common set:

- Recognize when a range change has occurred on the multi-range device
  When a range change occurs, the `vzs` task activates the calibration constants and virtual zero and span corrections associated with that range (i.e. loading these values into shared memory from the calibration table file).
- Perform the operations above on a range-specific basis. Examples of this include:
  - Requesting the span stimulus specific to the active range when a span request is received.
  - Performing correction computations and validity using acquired and specified data that is specific to the active range.
  - Terminating an ongoing operation should a range change occur in the middle of it.
- Recognize and respond to a find best range request by cycling through enabled ranges until the range is found that maximizes the resolution of the present value and then indicating completion. Range changes will not occur automatically after completion (i.e. a one-shot operation and not a continuous auto-ranging mode).

2.2.1 Additional vzs Behavior

The `vzs` task can both command range changes and monitor range status. The commanded range may not match the range status for a number of reasons, including: analyzer in manual mode, analyzer does not support commanded range, analyzer does not have external range control, etc. The range status (not commanded range) is always used to load calibration data and to associate acquired zero and span data. A range status indication is required for `vzs` to operate properly.

2.3 Specific vzs_sensor Capabilities

The single range `vzs_sensor` task has the following capability in addition to the common set:

- Perform an update of bias and scale adjustment based on specifications of whether the virtual zero and/or virtual span are required prior to the adjustment being made.

Note:

In the `vzs` task both are always required prior to the adjustment being made.

2.4 Functions Beyond vzs and vzs_sensor

Table 2 lists functions beyond the scope of the `vzs*` tasks. These functions are typically required but are implemented elsewhere in the system.

Table 2: Functions Beyond vzs Scope

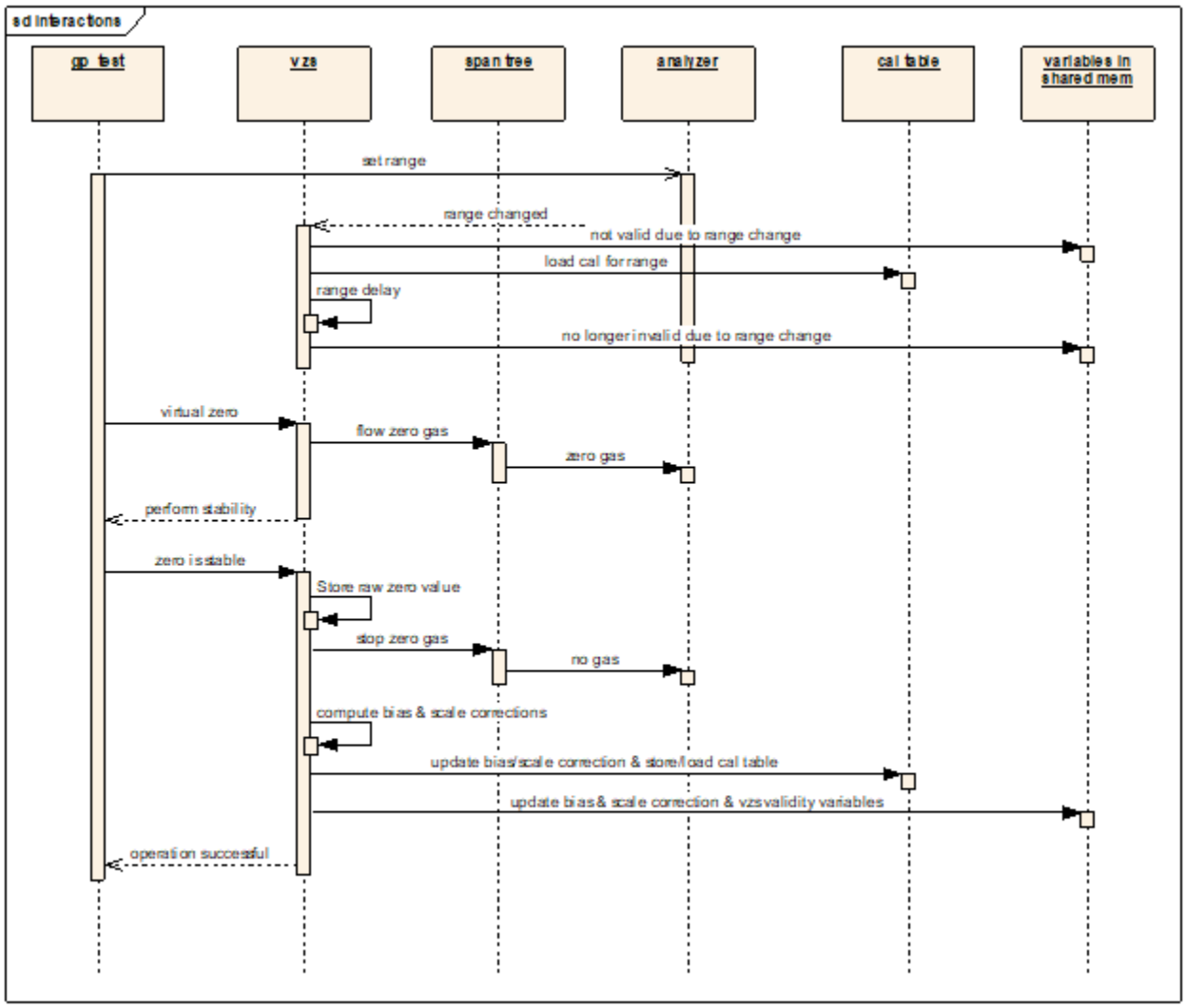
Function	Implementation Source
Conversion of raw input variable to final engineering units variable	`compvar`

Function	Implementation Source
Perform stability checks on zero and span operations	`gp_test`
Initiate range changes in response to test conditions	`gp_test` or user-initiated] **Note:** `vzs` does command range changes during a find vest range operation)
Determine when to perform zero and span	`gp_test` or user-initiated] **Note:** `vzs*` tasks do produce information that can help in this process
Manage anything having to do with multiple species from the same analyzer (e.g. `NOx` vs. `NO`)	N/A
Compute or evaluate the correctness of full calibration polynomial coefficients	Offline Excel process
Change span bottle concentrations; this is an important not to be overlooked	user/maintainer
Communication with an analyzer or sensor	IO sub-system or AK Interface Task

2.5 Interaction with Other CyFlex Components

Figure 1 on page 5 illustrates how the `vzs` interacts with other CyFlex system components during a virtual zero operation. This is provided to illustrate a typical interaction between the `vzs` task and other system components. A diagram of virtual span operation would be very similar. This diagram does not show non-normal steps, such as stability failure or adjustment limits exceeded. In these cases, a client op failed output event should result.

Figure 1: VZS Interactions With Other System Components



3 Starting the Application

The `vzs*` tasks are typically spawned in the startup process (i.e. in `go.scp`) as follows:

```
vzs [vzs_spec_file] &
```

For example:

```
vzs o2.VZSR3 &
```

The `vzs*` task will produce diagnostic output; therefore, the output of the task is typically redirected to either a file, a `log_rotate` process, or `/dev/null`.

For example:

```
Vzs o2.VZSR3 | log_rotate /data/errors/o2.VZSR3.%Y%m%d
2>/data/errors/o2.VZSR3.err &
```

The `log_rotate` will create a daily log file, keeping file size smaller and allowing `clean_up` to remove older logs. View the diagnostic output produced by the dump event by using the `tail` command as follows:

```
tail –c /data/errors/o2.VZSR3.20050815 (or whatever the date
is).
```

Use `CTRL-C` as needed to exit the `tail` session.

4 Specification Files

The `vzs*` tasks have fairly similar specification files. The `vzs_sensor` specification file is a subset of the `vzs` specification file and omits range specific specifications.

In the following example, the `Courier font` is content in the spec file and the **bold Arial font** is used for descriptive comments that should not be included unless on a comment line.

4.1 Multi-Range vzs Task Specification File

```
# Name: /specs/o2.VZSR2
# Type: Gas Cart Virtual Zero/Span Task Spec File
# Purpose: To initialize the o2 vzs task

#*********************************************************************
#                     VZ&S Specification File
#*********************************************************************

[literals]
Version              = 2.01
Analyzer             = o2   - used to identify vzs instance
NumRngs            = 2           - number of virtual ranges –
                           less than or equal to number of real ranges
FirstRngNum        = 1           - first device range number
RequireMidRange    = 0           - mid-range check req’d for valid
output? (1/0)
NoGas               = 0     - GasRequest variable value to set when
idle
ZeroGas             = 11    - GasRequest variable value to set when
zeroing
SampleGas           = 12    - GasRequest variable value to set when
sampling
CalTable            = O2Cal - name of calibration table
                           – must be POLYNOMIAL_RANGE type
BestRngLimit      = 0.9           - upper limit used to determine if
present range
                           is adequate when a find best range
                      operation
                           occurs this upper limit is this value times
                      the
                           CaO2nc variable value for each range (e.g.
                      90%
                           of full cal concentration).

[events]                   - all are signal events (i.e. no data)
                              - Start of events that are inputs to vzs
Clear-Vzs-Event = eO2ClrVzsRq    - clears zero/span corrections to 0
and 1

Dump-Report-Event = eO2Dump - dumps diagnostic output to tasks stdout
```

Find-Best-Rng-Rqst-Event = eO2BstRngRq **- request to find best range**

Full-Span-Stability-Complete-Event = eO2VsStbDn **- indicates span is stable - typically evaluated and set by gp_test**

Full-Span-Stability-Failed-Event = eO2VsStbFl **- indicates span is unstable - typically evaluated and set by gp_test**

Gas-Off-Event = eO2GasOff **- requests that zero/span gases be turned off**
Hardware-Span-Rqst-Event = eO2HwSpRq **- indicates h/w span is active**
Hardware-Span-Comp-Event = eO2HwSpDn **- indicates h/w span is done**
Hardware-Zero-Rqst-Event = eO2HwZrRq **- indicates h/w zero is active**
Hardware-Zero-Comp-Event = eO2HwZrDn **- indicates h/w zero is done**
Mid-Rng-Span-Rqst-Event = eO2MrRq **- indicates mid-range check request**
Midrange-Stability-Complete-Event = eO2VmStbDn **- indicates mid-range span is stable - typically evaluated and set by gp_test**

Midrange-Stability-Failed-Event = eO2VmStbFl **- indicates mid-range span is unstable - typically evaluated and set by gp_test**

New-Rng-Detected-Event = eO2RngChgDet **- indicates that a range change has occurred – often a transition event on the range status integer variable**

New-Rng-Request-Event = eO2RngChgRq **- indicates that a range change is requested**

Periodic-Timer-Event = tmr-1000 **- periodic timer name – used for timeouts**

Virtual-Full-Span-Rqst-Event = eO2VsRq **- virtual span request (on present range)**

Virtual-Zero-Rqst-Event = eO2VzRq **- virtual zero request (on present range)**

Zero-Stability-Complete-Event = eO2VzStbDn **- indicates zero is stable - typically evaluated and set by gp_test**

```
Zero-Stability-Failed-Event = eO2VzStbFl - indicates zero is unstable
                                - typically evaluated and set by
                                gp_test

Span-Gas-On-Event = eO2SpOn     - indicates request to flow span gas
                                for present range without performing
                                an adjustment – used for diagnostics

Zero-Gas-On-Event = eO2ZrOn     - indicates request to flow zero gas
                                without performing an adjustment –
                                used for diagnostics


                            - Start of events that are outputs from vzs
Client-Op-Success-Event = eO2VzsOk  - indicates that last request is
complete

Client-Op-Failed-Event = eO2VzsFl    - indicates that last request
failed

Start-Full-Stability-Event = eO2StrVsStb - indicates that span
stability
                                should start, typically handled by
                                gp_test

Start-Mid-Rng-Stability-Event = eO2StrVmStb - indicates that mid-range
span
                                stability should start, typically
                                handled by gp_test

Start-Zero-Stability-Event = eO2StrVzStb - indicates that zero
stability
                                should start, typically handled by
                                gp_test


[vars/input/integer]
# INTEGER variable name for range request
# a value of -1 indicates a find best range request
# other values are specified in table below.

# a range change sensing (request) event is required to trigger
processing
# for this action.
RngRequest          = o2_rng_rq

# INTEGER variable name for actual range sense
PresentRng          = o2_rng_s

[vars/input/real]
```

```
# label of analyzer raw output
# its value is an input to VZ manager, units are typically mV
AnalyzerRaw          = o2_araw   - this is not calculated by vzs!

# label of analyzer calibrated output - pre validation
# its value is an input to VZ manager, units are typically PPM
AnalyzerConc         = o2_ppm         - this is not calculated by vzs!

# REAL variable name for range change delay [units must be time]
# This is usually on the order of 5 seconds
RngChangeDelay     = rng_dly          - after a range change vzs task
waits this
                                long before using analyzer values


# REAL variable name for span or zero gas timeout [units must be time]
# This is usually on the order of 5 minutes
ZeroGasTimeout       = zr_sp_flw_tmout – vzs turns GasRequest variable
to off
                                value after zero gas on for this time
                                to save gas

SpanGasTimeout       = zr_sp_flw_tmout  – vzs turns GasRequest
variable to off
                                value after span gas on for this time
                                to save gas



# REAL variable name for span or zero gas age limit [units must be
time]
# This is usually on the order of 8 hours
VzsAgeLimit          = zr_sp_age_ul – age limit goes invalid after
this time

[vars/output/integer]
# INTEGER variable for requesting a specific valve configuration for
this
# analyzer
GasRequest           = o2_sv_sel – typically mapped to index variable
of
                                furball file to set DOs


# INTEGER variable name for range command to analyzer
CommandedRng      = o2_rng_cmd   – typically mapped to either AK
command or
                                DOs to command range, depending on
                                analyzer type
```

```
# INTEGER variable name for state of VZS
VzsState            = O2VzsState – diagnostic output



[vars/output/string]
# STRING variable showing state description of VZS task
VzsStateDesc        = O2VzsStateDesc – diagnostic output

VzsOperatorMsg = O2UserMsg       – Operator prompt – typically on a
display

[vars/output/logical]
# LOGICAL variable indicating VZS valid for present range
VzsValid            = O2VzsVld   - can/should be used in gen_labels
                                   expression to force final value to a
                              known
                                   bad value (e.g. -999999)

# LOGICAL variable indicating mid range check valid for present range
VzsMidRngValid    = O2VzsMrVld   - can/should be used in gen_labels
                                   expression to force final value to a
                              known
                                   bad value (e.g. -999999)

# LOGICAL variable indicating that full span data is available for
# present range
VzsFullSpanDataAvail= O2VsAvail

# LOGICAL variable indicating that zero data is available for present
range
VzsZeroDataAvail    = O2VzAvail

# LOGICAL variable indicating that zero/span age limit is OK for
present range
VzsAgeLimitValid    = O2VzsAgeLmtVld - can/should be used in
gen_labels
                                   expression to force final value to a
                              known
                                   bad value (e.g. -999999)


# LOGICAL variable indicating that range delay time has elapsed for
present range
VzsRngDelayValid  = O2VzsRngDlyVld - can/should be used in gen_labels
                                   expression to force final value to a
                              known
                                   bad value (e.g. -999999)
```

```
# Per range specifications
[r1/in]
Enable          = o2R1en        - only enabled ranges can be requested
                                  Or used in find best range operation

CaO2nc          = O2R1CaO2nc    - span gas concentration during full
cal

SpanConc        = O2R1CyO2nc    - present full span bottle
concentration

MidConc         = O2R1CyO2nc    - present mid span bottle
concentration

NoAdjTol        = VzsNoAdjPctUl - don't adjust if adjustment less than
                                  this much – helps avoid non-value
                                  adding discontinuities in data

MaxAdjLimit     = VzsMaxAdjPctUl    - don't adjust if adjustment
exceeds
                                  this much – Inhibit variable for range
                                  set if exceeded

MidRngLimit     = VzsMrRngPctUl     - tolerance used to check mid-
range span
                                  results - Inhibit variable for range
                                  set if exceeded

MidRngGas       = 1             - value to set to GasRequest variable
for
                                  Mid-range check on this range

SpanGas         = 2             - value to set to GasRequest variable
for
                                  Span operation on this range


[r1/out]
Inhibit         = o2_r1_vzs_inh     - present validity for Range 1
ZeroMvMsd       = O2R1Vz        - raw (mv) value for last virtual zero
SpanMvMsd       = O2R1Vs        - raw (mv) value for last virtual span
BiasMv          = O2R1VzsBias   - last computed bias correction (in
mv)
ScaleMvPerMv    = O2R1VzsSlp        - last computed gain correction
(unitless)
MidMvMsd        = O2R1MrMvMsd   - raw (mv) value for last mid-range
check
MidMvTheor      = O2R1MrMvTh        - expected raw value for mid-
range check
```

```
ZAge              = O2R1VzAge     - time since last successful virtual
zero
SAge              = O2R1VsAge     - time since last successful virtual
span

                                  - the following variables are same as
                                  above for additional ranges – see
                                  comments above
[r2/in]
Enable            = o2R2en
CaO2nc           = O2R2CaO2nc
SpanConc          = O2R2CyO2nc
MidConc           = O2R2CyO2nc
NoAdjTol          = VzsNoAdjPctUl
MaxAdjLimit       = VzsMaxAdjPctUl
MidRngLimit       = VzsMrRngPctUl
MidRngGas         = 2
SpanGas           = 3

[r2/out]
Inhibit           = o2_r2_vzs_inh
ZeroMvMsd         = O2R2Vz
SpanMvMsd         = O2R2Vs
BiasMv            = O2R2VzsBias
ScaleMvPerMv      = O2R2VzsSlp
MidMvMsd          = O2R2MrMvMsd
MidMvTheor        = O2R2MrMvTh
ZAge              = O2R2VzAge
SAge              = O2R2VsAge
```

4.2 Single-Range vzs_sensor Task Specification File

Note:

The notes below only detail the differences between `vzs_sensor` and `vzs`. Refer to the preceding spec file for notes on elements from this spec file for which there are no comments.

```
# Name: /specs/opac.VZ
# Template: /specs/opac.VZ
# Purpose: To initialize the opac vzs_sensor task

#*****************************************************************
#                    VZ&S Sensor Specification File
#*****************************************************************

[literals]
Version             = 2.01
Sensor              = opac
CalTable            = opac

[events]
Clear-Vzs-Event = eOpacClrVzsRq
Dump-Report-Event = eOpacDump
Full-Span-Stability-Complete-Event = eOpacVsStbDn
Full-Span-Stability-Failed-Event = eOpacVsStbFl
Stimulus-Off-Event = eOpacStimOf
```
- different keyword tag from above Gas off event
```
Hardware-Span-Comp-Event = eOpacHwSpDn
Hardware-Span-Rqst-Event = eOpacHwSpRq
Hardware-Zero-Comp-Event = eOpacHwZrDn
Hardware-Zero-Rqst-Event = eOpacHwZrRq
Periodic-Timer-Event = tmr-1000
Virtual-Full-Span-Rqst-Event = eOpacVsRq
Virtual-Zero-Rqst-Event = eOpacVzRq
Zero-Stability-Complete-Event = eOpacVzStbDn
Zero-Stability-Failed-Event = eOpacVzStbFl
Span-On-Event = eOpacSpOn
Zero-On-Event = eOpacZrOn

Client-Op-Success-Event = eOpacVzsOk
Client-Op-Failed-Event = eOpacVzsFl
Start-Full-Stability-Event = eOpacStrVsStb
Start-Zero-Stability-Event = eOpacStrVzStb

[vars/input/integer]
# zero mode
# 1 = required, 2=use latest actual or cal
ZeroMode        = opac_zr_md
```
- function not included above – allows you to specify whether a zero is required or not prior to adjustment

```
# span mode
# 1 = required, 2=use latest actual or cal
SpanMode        = opac_sp_md
```
- function not included above – allows you to specify whether a apsn is required or not prior to adjustment

```
[vars/input/real]
# label for AI raw input
# its value is an input to VZ manager, units are typically mV
AIRaw          = opac_mv

# label of calibrated output
# its value is an input to VZ manager, units are typically PPM
AIOut           = opac

# REAL variable name for span or zero gas timeout [units must be time]
# This is usually on the order of 5 minutes
ZeroTimeout       = zr_sp_tmout
SpanTimeout       = zr_sp_tmout

# REAL variable name for span or zero gas age limit [units must be
time]
# This is usually on the order of 8 hours
VzsAgeLimit          = zr_sp_age_ul

# REAL variable name for value of span during full cal from which
# polynomial coeff were derived
CalSpanVal           = OpacSpCal
```
- different keyword than vzs

```
# REAL variable name for value of span during virtual span operation
VirtSpanVal         = OpacSpVal
```
- different keyword than vzs

```
# REAL variable name for variable that indicates lower limit to
adjustment
# i.e. no adust (but success) if adjust of less than this is
calculated
NoAdjTol         = VzsNoAdjPctUl

# REAL variable name for variable that indicates upper limit to
adjustment
# i.e. no adust (but failure) if adjust of more than this is
calculated
# in this case the bias and gain are calculated and available for
display
# but are not updated in the cal table in memory or on disk
MaxAdjLimit      = VzsMaxAdjPctUl

[vars/output/real]
# Set to cal value if not to be measured
ZeroMvMsdOrCal       = OpacVz
```

```
# Set to cal value if not to be measured
SpanMvMsdOrCal       = OpacVs
# Resulting bias - also stored in cal table
BiasMv               = OpacVzsBias
# Resulting gain - also stored in cal table
ScaleMvPerMv         = OpacVzsSlp

# How old (in hours) the virtual zero/span is - counter starts at
10000
VzAge                 = OpacVzAge
VsAge                 = OpacVsAge

[vars/output/integer]
# INTEGER variable name for state of VZS
VzsState            = OpacVzsState

[vars/output/string]
# STRING variable showing state description of VZS task
VzsStateDesc        = OpacVzsStateDesc
VzsOperatorMsg = OpacOperatorMsg

[vars/output/logical]
# LOGICAL variable indicating VZS valid for present range
VzsValid            = OpacVzsVld

# LOGICAL variable indicating that full span data is available for
# present range
VzsFullSpanDataAvail= OpacVsAvail

# LOGICAL variable indicating that zero data is available for present
range
VzsZeroDataAvail    = OpacVzAvail

# LOGICAL variable indicating that zero/span age limit is OK for
# present range
VzsAgeLimitValid    = OpacVzsAgeLmtVld

# LOGICAL variable indicating that a zero stimulus should be applied
ActivateZeroStimulus    = opac_zr_stim- different keyword than vzs


# LOGICAL variable indicating that a span stimulus should be applied
ActivateSpanStimulus    = opac_sp_stim      - different keyword than vzs

##########################################################################
```

5 Example gp_test Procedure

The following gp_test procedure illustrates how a client `gp_test` procedure might interact with a `vzs*` task to perform a span operation:

```
# Emissions Cart (Bench) - Low Concentration CO Span QC Check Process
–
# 2 Range Analyzer

#start_mode ( mode where the test begins )
1

@INSTANCE
   GP_O2

#-------------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   1                 0.1[sec]          2
   # description
   Starting QC Span Checks

# Placeholder initial mode

@PROCEDURE
/specs/gp/analyzer/gp_set_o2_range

#-------------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   2                 0.1[sec]          3
   # description
   Requesting VZS to perform virtual span

@SET_EVENTS
   # start_type    event_name
   AT_START        eO2VsRq

#-------------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   3                 0[sec]            4
   # description
   Waiting for stability request from VZS

# Wait for request for stability
@TERMINATION_EVENTS
   #event_name                              termination_path
   eO2StrVsStb                             4
```

```
#-----------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   4                 0.1[sec]          5
   # description
   Respond that stability is satisfied

# Note - could add @STABILITY for more stringent criteria to this mode

@SET_EVENTS
   # start_type    event_name
   AT_END          eO2VsStbDn

#-----------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   5                 120[sec]            702
   # description
   Waiting for completion of virtual span step

@TERMINATION_EVENTS
   #event_name          termination_path
   eO2VzsOk  800
   eO2VzsFl  701

#-----------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   701               5[sec]          900
   # description
   Virtual span failed

#-----------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   800               0.1[sec]          RETURN
   # description
   Successfully completed span QC check process

#-----------------------------------------------------------------------
@MODE
   #mode_number      max_timeout       default_next_mode
   900               0.1[sec]          RETURN
   # description
   Failed to complete span QC process
```