Creating User Computations and User Variables

Version 10

March 3, 2025

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	1/25/2016	Initial publication
2	8/23/2018	Formatting changes
3	9/11/2019	Add information to *Section 2 User-Created Variables and Events* on page 2 that `compvar` is not required if a file has no computed expressions
4	3/25/2020	Retrofit to new template
5	8/5/2021	Added hyperlinks to existing cross-references for cyflex.com documents and added hyperlinked cross-references to usage help
6	5/16/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions
7	1/30/2024	Rebrand to TRP Laboratories
8	8/19/2024	Refreshed examples in Sections 3 through 7. Updated narrative in *Section 4 Specifying INTEGER Variable Specifications* on page 5 to state a variable can have an optional format field following the units field for formatting variable display. Refreshed example in *Section 11.2 Example* `gen_labels` *Specification File* on page 18
9	9/5/2024	Added Section *11.1 REAL_VARIABLE and INTEGER_VARIABLE Format* on page 16
10	3/3/2025	Removed `floating point: L Length` entry from *Section 11.1 REAL_VARIABLE and INTEGER_VARIABLE Format* on page 16

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.

Related Documents

Array Variables

Statistical Variables and Statistical Computations

Composition and Property Variables

CyFlex Documentation

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

Table of Contents

LIST OF TABLES

1 Overview

CyFlex® allows the user to create variables and to build computed expressions, the results of which are assigned to a variable. The method described in this document supports the creation of `REAL`, `LOGICAL`, `INTEGER`, and `STRING` variables.

Two applications are involved:

1. The `compvar` application continuously evaluates computed expressions and places the results in the target variable. Refer to *Section 10 The `compvar` Application* on page 15.
2. The `gen_labels` expression parses a specification file, creates the required variables and configures the `compvar` application to perform the computations. Refer to *Section 11 The `gen_labels` Application* on page 16.

Refer to the following for information on creating the listed variable types below:

- *Array Variables*
  - `REAL_ARRAY`
  - `LOGICAL_ARRAY`
  - `INTEGER_ARRAY`
  - `STRING_ARRAY`
- *Statistical Variables and Statistical Computations*
  - `STATISTICAL`
- *Composition and Property Variables*
  - `COMPOSITION`
  - `PROPERTY`

2 User-Created Variables and Events

REAL, LOGICAL, INTEGER, or STRING variables and associated computed expressions can be created by entries in a file which is then processed by the gen_labels application.

There may be multiple gen_labels specification files used in a system. Each copy of this type of specification file must have a name associated with it to identify which instance of the compvar application should perform the computed expressions defined in the file. This name is always at the top of the file and requires two lines:

```
@REG_NAME
     GL_turbo
```

Usually, there should be an instance of the compvar application running with this name (GL_turbo). There may be a case were the file contains no computed expressions, but simply creates a list of variables to be used by other applications. In that case, there is no need for a corresponding compvar.

The file where these variables are defined is separated into seven sections after the registered name specification. Each section is terminated by a line with a $ symbol in column 1.

1. Creation of Real Variables with optional computed expressions
2. Creation of Integer Variables with optional computed expressions
3. Creation of Logical Variables with optional computed expression
4. Creation of computed expression to be applied to existing variables
5. Creation of String Variables with optional computed expressions
6. Creation of computed expressions for display status (color/blinking) of existing variables
7. Creation of zero length events

Each specification for a variable consists of two lines of information.

1. The first line is the variable definition.
2. The second line is the optional computed expression, or a dash if there is no computed expression.

Creating events uses only one line and one field, the event name.

3 Specifying REAL Variable Specifications

Specify the fields listed in *Table 1 to create* **Real Variables.** Specified fields must be valid floating-point values.

Table 1: REAL Variable Fields

Field	Description	Example
Label	Alphanumeric string up to 79 characters	*my_variable*
Units	Any valid CyFlex units description	psi
Display Resolution	Number of decimal places	2
Initial Value	Optional value to be placed into the variable whenever this file is processed – use a dash to indicate that the value is not to be initialized. The initial value in the file must have the same units as specified in the units field.	100
Process Interval or Event Name	This specifies the rate at which the computed expression will be evaluated. If no computed expression is specified, then use a dash. This can be one of the alpha strings which designate a standard process interval (FAS, MED, SLO, WARP, USR1, USR2). Optionally, this may be specified as a numerical value in milliseconds or as an event name	SLO or 1000 or my_sync_event
History active flag	This must be either ON or OFF, indicating whether or not the variable is to be saved as part of compressed history.	ON
History tolerance	Value by which the variable must change to be stored in compressed history, but only if the History active flag is ON.	2.0
Transition event name	Event name that will be set if the value changes by an amount equal to the History tolerance.	*my_variable*-changed
Computed Expression – 2nd line of specification	Optional string representing a computed value	"other_var – 10[psi]"

Example:

```
# REAL_VARIABLE specifications for GENERAL_USE
# label       units   format initial_value interval/event hst_flag  tolerance
# expression

# label       units   format initial_value interval/event hst_flag  tolerance
# ar_real     min     2      -             -              OFF       1.00
# -

# label      units  format   initial_value interval/event hst_flag  tolerance
# ar_real_2  sec    %10.4f   -             -              OFF       1.00
# -

$  Each section must end with a dollar sign
```

4 Specifying INTEGER Variable Specifications

Specify the fields listed in *Table 2* to create **Integer Variables**. The `INTEGER_TYPE` variable can have an optional format field following the `units` field for formatting variable display. Only valid options for integers are allowed. Refer to https://cplusplus.com/reference/cstdio/printf/ for supplemental information.

Table 2: INTEGER Variable Fields

Field	Description	Example
Label	Alphanumeric string up to 79 characters	*my_counter*
Units	Any valid CyFlex units description	gal
Initial Value	Optional value to be placed into the variable whenever this file is processed – use a dash to indicate that the value is not to be initialized. The initial value in the file must have the same units as specified in the `units` field.	–
Process Interval or Event Name	This specifies the rate at which the computed expression will be evaluated. If no computed expression is specified, then use a dash. This can be one of the alpha strings which designate a standard process interval (FAS, MED, SLO, WARP, USR1, USR2). Optionally, this may be specified as a numerical value in milliseconds or as an event name.	SLO or 1000 or *my_sync_event*
History active flag	This must be either “ON” or “OFF”, indicating whether or not the variable is to be saved as part of compressed history. The value will be stored as part of compressed history if the flag is ON and the value changes.	ON
Transition event name	Event name that will be set if the variable changes	*my_variable*-changed
Computed Expression – 2nd line of specification in the file	Optional string representing a computed value	“*my_counter* + 1[gal]”

Example:

```
# INTEGER_VARIABLE specifications for GENERAL_USE
# label       units   initial_value interval/event hst_flag  transition_evnt
# expression

# OR

# label  units format  initial_value interval/event hst_flag  transition_evnt
# expression


# label       units   initial_value interval/event hst_flag  transition_event
# MPR         none      1440            -             OFF
# -

# label units  format initial_value interval/event hst_flag  transition_event
# XYZ   none   %#x      1440          -             OFF
# -

$
```

5 Specifying LOGICAL Variable Specifications

Specify the fields listed in Table 3 to create **Logical Variables.**

Table 3: LOGICAL Variable Fields

Field	Description	Example
Label	Alphanumeric string up to 79 characters	*my_logi*
True transition event name	An event name which will be set when the value changes from FALSE to TRUE. The event will be created if it doesn't exist. This is an optional field. Use a dash to indicate no transition event.	*my_logi_on*
False transition event name	An event name which will be set when the value changes from TRUE to FALSE. The event will be created if it doesn't exist. This is an optional field.	*my_logi_off*
True description	An alphanumeric description of the true state for display	ON
False description	An alphanumeric description of the false state for display	OFF
Process Interval or Event Name	This specifies the rate at which the computed expression will be evaluated. If no computed expression is specified, then use a dash. This can be one of the alpha strings which designate a standard process interval (FAS, MED, SLO, WARP, USR1, USR2). Optionally, this may be specified as a numerical value in milliseconds or as an event name.	SLO or 1000 or *my_sync_event*
History active flag	This must be either ON or OFF, indicating whether or not the variable is to be saved as part of compressed history. The value will be stored as part of compressed history if the flag is ON and the value changes.	ON
Computed Expression – 2nd line	Optional string representing a computed value	" *my_counter* > 100[gal]"

Example:

```
# LOGICAL_VARIABLE specifications for GENERAL_USE
# label     true_event  false_event true_desc   false_desc int/event hst_flag
# expression


# label     true_event  false_event true_desc   false_desc int/event hst_flag
# ar_logi   -           -           true        false       -        OFF
# -

$
```

6 Specifying STRING Variable Specifications

Specify the fields listed in *Table 4* to create **String Variables**.

Table 4: STRING Variable Fields

Field	Description	Example
Label	Alphanumeric string up to 79 characters	*my_description*
Initial value	Optional string to be placed into the variable whenever this file is processed – use a dash to indicate that the value is not to be initialized.	'something happening'
Process Interval or Event Name	This specifies the rate at which the computed expression will be evaluated. If no computed expression is specified, then use a dash. This can be one of the alpha strings which designate a standard process interval (FAS, MED, SLO, WARP, USR1, USR2). Optionally, this may be specified as a numerical value in milliseconds or as an event name.	SLO or 1000 or *my_sync_event* or -
History active flag	This must be either ON or OFF, indicating whether or not the variable is to be saved as part of compressed history. The value will be stored as part of compressed history if the flag is ON and the value changes.	ON
Transition event name	Event name that will be set if the string content changes	new_desc
Computed Expression – 2nd line of specification	Optional string representing a computed value	reason_string

Example:

```
# label          initial_value      event/interval   hst_flag
# expression

# label          initial_value      event/interval   hst_flag
# ar_string      '-'                -                OFF
# -

$ end of strings
```

7 Existing Variables

7.1 Specifying Computed Values for Existing Variables

Specify the fields listed in *Table 5* to create **computed values for existing variables**.

Table 5: Computed Values for Existing Variables Fields

Field	Description	Example
Label	Alphanumeric string up to 79 characters. This variable will not be created by this `gen_labels`. It must have been created elsewhere.	`some_variable`

Process Interval or Event Name	This specifies the rate at which the computed expression will be evaluated. If no computed expression is specified, then use a dash. This can be one of the alpha strings which designate a standard process interval (`FAS`, `MED`, `SLO`, `WARP`, `USR1`, `USR2`). Optionally, this may be specified as a numerical value in milliseconds or as an event name.	`SLO`
		`or 1000`
		`or *my_sync_event*`
		`or -`

Computed Expression – 2nd line of specification	Optional string representing a computed value	`" p1 / p2 "`

Example:

```
# These variables must be defined elsewhere and exist before
# processing this gen_labels file

# label            interval/event
# expression

# label            interval/event
# ac_air_ot_tW_TR  SLO
# if( imt_map_enab ) then ( imt_calc ) else ( ac_air_ot_tW_TR )

$
```

7.2 Specifying Computed Display Status for Existing Variables

Specify the fields listed in *Table 6* to create **computed display status for existing variables**.

Table 6: Computed Display Status for Existing Variables Fields

Field	Description	Example
Label	Alphanumeric string up to 79 characters. This variable will not be created by this `gen_labels`. It must have been created elsewhere.	`some_variable`
Process Interval or Event Name	This specifies the rate at which the computed expression will be evaluated. If no computed expression is specified, then use a dash. This can be one of the alpha strings which designate a standard process interval (`FAS`, `MED`, `SLO`, `WARP`, `USR1`, `USR2`). Optionally, this may be specified as a numerical value in milliseconds or as an event name.	`SLO` or `1000` or *`my_sync_event`* or `-`
Computed Expression – 2nd line of specification	Optional string representing a computed display status	`" if (RPM < idle_spd) then RED else GREEN "`

Example:

```
label          interval/event
# expression
#
#*********************************************************************************

#  The result of the expression must be one of the following:
#
#     NORMAL             use normal display color
#     BLINK              blink the normal display color
#     COLOR              use the 'exception' color
#     COLOR_BLINK        blink the exception color
#
#     BLUE
#     GREEN
#     CYAN
#     RED
#     MAGENTA
#     YELLOW
#     WHITE
#     BLINK_BLUE
#     BLINK_GREEN
#     BLINK_CYAN
#     BLINK_RED
```

```
#     BLINK_MAGENTA
#     BLINK_YELLOW
#     BLINK_WHITE
#*************************************************************************

#
# label           interval/event
# expression
#
# label           interval/event
# AVL_TESTPATH    TC305ParameterFile.xpa
# if (ChgLock_Status > 0[none]) then GREEN else CYAN

$ end of computations of display status for existing variables -
```

8 Creating Events

Specify the field listed in Table 7 to create **events**.

Table 7: Create Events Field

Field	Description	Example
Event name	Alphanumeric string up to 31 characters.	`an_event_is_born`

9 Testing Computed Expressions

Use the `get_comp` application to evaluate an expression from the command line. Complicated expressions can be problematic to write with the correct syntax. The `get_comp` application executes validity test of an expression.

Command syntax:

```
get_comp    " <expression> "   [u=units]
```

Where:

- `expression` is the computed expression to be tested. [Required]
- `u=units`: optional output units

The output value is printed to the stdout (console) device and defaults to the base SI units of the dimension used in the expression.

Examples:

```
     get_comp "100[psi]"
output> 689475.7 This result will be in base SI units for pressure, which is pascals

     get_comp "100[psi]" u=in_hg
output> 204.177[in_hg]
```

Notice that a side effect of the normal way computed expressions are handled in CyFlex is that one can use the `get_comp` command to do units conversion as in the next example.

```
     get_comp "if Engine_Run then 5[psi] else 10[psi]" u=psi
output> 5[psi]

     get_comp "@cal_table( 5[mv], 'cmp_in_p' )"
output>(the y-value in the /cell/tables/cmp_in_p.tbl file interpolated for x=5[mv])
```

Refer to usage help for `get comp` for related information.

10 The compvar Application

The compvar application is a memory-resident application that processes the computed expressions that were defined in a gen_labels specification file. Multiple instances of compvar may run simultaneously to operate on expressions at different rates, priorities, or to handle different sets of associated data. Usually, expressions that are to be computed at high rates will be configured to run at higher priorities.

Each instance of compvar must have a unique registered name. The registered name of an instance of compvar is used to associate it to a particular gen_labels specification file in which that name appears as the registered name.

For a particular instance of compvar and a particular process interval, the expressions are always evaluated in the same sequence as they appear in the gen_labels specification file.

Command syntax:

```
compvar   name priority process_intervals [+c]
```

Where:

- name is the registered name.
- priority is a specified number between 11 and 19. Higher number is higher priority.
- process_intervals: specify FAS, MED, or SLO. SLO is required.
- +c is an optional argument which indicates that this is a “critical” application. A failure of the application will cause a system watchdog shutdown.

Example:

```
compvar GL_turbo 18 SLO +c &
```

Launches an instance of compvar to handle the computed expressions created in the gen_labels file which contains the GL_turbo registered name.

Note:

The compvar application will attach to all the process intervals that are specified in the corresponding gen_labels file. Only the SLO rate needs to be specified on the compvar command line

Refer to compvar usage help for related information.

11 The gen_labels Application

The gen_labels application is a specification file processor. It reads a particular file, creates variables and events, configures the computed expressions for a compvar application and then exits. It can be executed from the command line after modifications have been done to a specification file and is also normally installed in the go.scp startup script.

When launching compvar and gen_labels, the order is not important. All of the configuration information is stored in memory.

- If compvar runs first, it will wait for a configuration message from the gen_labels application before processing the specifications and performing the computations.
- If gen_labels runs first, it stores the configuration information in memory and then when compvar starts up, it finds the information and begins processing.

If the information in a gen_labels' file does not include computed expressions and is used instead to simply create and initialize variables and events, there is no need to run a corresponding compvar application running since it would be doing nothing if there are no computed expressions to process.

Command syntax:

gen_labels [*specification file name*]

Where:

- *specification file name* is the name of the file to process. The default is /specs/gen_labels.*NNN* (where NNN is the cell name – from /cell/*cell_name*)

Example:

```
gen_labels /specs/gen_labels.GL_turbo
```

Refer to gen labels usage help for related information.

11.1 REAL_VARIABLE and INTEGER_VARIABLE Format

REAL_VARIABLE and INTEGER_VARIABLEs can take a format string in place of resolution. For INTEGER_VARIABLEs this is an optional field. Refer to the following spec file excerpt for syntax details. *Section 11.2* on page 18 contains a complete example spec file.

```
The format matching should be: %[flags][width][.precision][length]specifier

The '%' and specifier are required.

Flags: 0, #, +, -

Width: number only, NOT *

Precision: number only, NOT .*

Length:
    integers: h, l, j, z, t
```

```
Specifiers:
    floating point: e, E, f, F, g, G, a, A
    integers: d, i, u, o, x, X

Replace REAL Variable example section with:

# REAL_VARIABLE specifications for GENERAL_USE
# label       units   format initial_value interval/event hst_flag  tolerance
# expression

# label       units   format initial_value interval/event hst_flag  tolerance
# ar_real      min     2         -             -            OFF       1.00
# -

# label      units  format   initial_value interval/event hst_flag  tolerance
# ar_real_2    sec   %10.4f       -             -           OFF       1.00
# -

$  Each section must end with a dollar sign

Replace INTEGER Variable example section with this:

# INTEGER_VARIABLE specifications for GENERAL_USE
# label        units   initial_value interval/event hst_flag  transition_evnt
# expression

# OR

# label  units format  initial_value interval/event hst_flag  transition_evnt
# expression


# label       units   initial_value interval/event hst_flag  transition_event
# MPR          none    1440             -            OFF
# -

# label units  format initial_value interval/event hst_flag  transition_event
# XYZ   none    %#x     0x7ffc          -            OFF
# -
```

11.2 Example `gen_labels` Specification File

```
#*********************************************************************
#
#        Version number (required)
#
#*********************************************************************

# SPEC_FILE_VERSION
  VERSION_2  (do not remove this line)

#*********************************************************************
#
#    Registered name - optional if using only one instance of compvar,
#                      otherwise, it is required and must match the
#                      registered name argument when launching compvar
#
#*********************************************************************

@REG_NAME
 GL_PM2


#*********************************************************************
#
#        G e n e r a l   U s e   S p e c i f i c a t i o n s
#
#*********************************************************************
#
#  This file contains the labels, units, and format information for
#  the #  GENERAL_USAGE rdata, ldata, idata, and sdata arrays.
#  The rdata is an array of REAL_VARIABLES, the ldata
#  is an array of INTEGER_VARIABLEs, the ldata is an array
#  of LOGICAL_VARIABLES, and the sdata is an array of
#  STRING_VARIABLEs.
#
#  The file is arranged into blocks, each terminated by a required
#  line #  with a single dollar sign ($):
#
#     block 1: Real Variables
#     block 2: Integer Variables
#     block 3: Logical Variables
#     block 4: Computations on existing Variables
#     block 5: String Variables
#     block 6: Computation of the display status (color)of an existing
#              variable
#     block 7: creating zero length (signal) events
#
```

```
#*****************************************************************

# REAL_VARIABLE specifications for GENERAL_USE
# label  units format initial_value interval/event hst_flag  tolerance
# expression

# label  units format initial_value interval/event hst_flag  tolerance
# ar_real min     2      -             -            OFF       1.00
# -

# label     units  format initial_value interval/event hst_flag  tolerance
# ar_real_2 sec    %10.4f  -             -            OFF       1.00
# -

$  Each section must end with a dollar sign

#*****************************************************************

# INTEGER_VARIABLE specifications for GENERAL_USE
# label  units  initial_value interval/event hst_flag  transition_evnt
# expression

# OR

# label units format initial_value interval/event hst_flag transition_evnt
# expression


# label units  initial_value interval/event hst_flag  transition_event
# MPR   none   1440          -              OFF
# -

# label units  format  initial_value interval/event hst_flag transition_event
# XYZ   none   %#x     1440          -              OFF
# -

$

#*****************************************************************

# LOGICAL_VARIABLE specifications for GENERAL_USE
# label true_event false_event true_desc  false_desc int/event hst_flag
# expression


# label   true_event  false_event true_desc false_desc int/event hst_flag
# ar_logi -           -           true      false       -         OFF
# -

$
```

```
#*********************************************************************

# PRE-EXISTING COMPUTED_VARIABLE specifications for GENERAL_USE

# These variables must be defined elsewhere and exist before
# processing this gen_labels file

# label            interval/event
# expression

# label            interval/event
# ac_air_ot_tW_TR  SLO
# if( imt_map_enab ) then ( imt_calc ) else ( ac_air_ot_tW_TR )

$

#*********************************************************************

# STRING_VARIABLE creations and definitions
#
# label          initial_value      event/interval   hst_flag
# expression

# label          initial_value      event/interval   hst_flag
# ar_string      '-'                -                OFF
# -

$ end of strings

#*********************************************************************

# Section for computing DISPLAY STATUS of pre-existing variables.
#
# label         interval/event
# expression
#
#*********************************************************************

#  The result of the expression must be one of the following:
#
#     NORMAL            use normal display color
#     BLINK             blink the normal display color
#     COLOR             use the 'exception' color
#     COLOR_BLINK       blink the exception color
#
#     BLUE
#     GREEN
#     CYAN
#     RED
#     MAGENTA
```

```
#     YELLOW
#     WHITE
#     BLINK_BLUE
#     BLINK_GREEN
#     BLINK_CYAN
#     BLINK_RED
#     BLINK_MAGENTA
#     BLINK_YELLOW
#     BLINK_WHITE
#
#*********************************************************************

#
# label           interval/event
# expression
#
# label           interval/event
# AVL_TESTPATH    TC305ParameterFile.xpa
# if (ChgLock_Status > 0[none]) then GREEN else CYAN

$ end of computations of display status for existing variables -

#*********************************************************************

# Section for defining signal events
#
# event names
#
#*********************************************************************


# event names
# start_event
# stop_event


$ end of events
```