WHEN YOU NEED TO BE SURE

CyFlex® Limits Monitoring Applications

Version 6

May 17, 2022

Developed by SGS North America, Inc.

Version History

Version	Date	Revision Description
1	1/25/2019	Initial publication
2	8/23/2018	Format changes
3	9/17/2019	Add information about error counts
4	3/26/2020	Retrofit to new template
5	8/10/2021	Added hyperlinks to existing cross-references for cyflex.com documents and added hyperlinked cross-references to usage help
6	5/17/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions

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

The CyFlex® application `limit` monitors any CyFlex variable to compare the current value to an upper or lower limit and to either set a flag or initiate some type of action. When the variable value falls outside of a specified limit, the limit is said to be “violated”. This means it could be less than a lower limit or higher than an upper limit. There may be one or more specifications for any particular variable with different limits and/or different actions specified.

Most test systems have some kind of environmental characteristics that must be monitored continuously for safety reasons or to ensure that the system is operating in a useful manner. Use the `limit` application for this purpose and also to monitor the operating range of variables for other testing and data acquisition purposes.

More than one `limit` application may be running at any time, each with separate specifications. This can be used to separate the specifications for different systems in the test environment. The test object limits could be handled by one instance while the test cell or support systems could be handled by one or more other instances. Each instance of the `limit` application will have a unique registered name so that the specification files can be associated with a particular instance of the application.

The `limit` applications are typically launched in the `go.scp` startup script and run continuously, evaluating the specified limits. The specifications are defined in a file and processed by an application called `limit_specs`. The `limit_specs` application reads the file and sends the specifications to the `limit` application. The specification file will contain the registered name for the instance of `limit` that is assigned to evaluate the specifications.

1.1 Critical Operation and Error Counting

The `limit` application is typically launched as a *critical* application, that is, the `+c` command line option. For this case, the `limit`” application must be configured by the `limit_specs` application within 3 minutes of when it is first launched or it will kill the system watchdog. For this reason, the `go.scp` should not have a major separation between when `limit` is launched and when “`limit_specs`” is launched.

Refer to the following cyflex.com usage help documentation for related information:

- `limit`
- `limit_specs`

2 Limit Application

2.1 Launching the Limit Application

Launch the `limit` application in the system `go.scp` startup script as a critical process (`+c` option). Refer to cyflex.com usage help for `limit`.

Notes:

1. The `limit` application must be configured by the `limit_specs` application within 3 minutes of when it is first launched or it will kill the system watchdog. For this reason, the `go.scp` script should not have a major separation between when `limit` is launched and where `limit_specs` is launched.
2. The `limit` application will automatically attach to any process interval specified in the specification file. However, at least one process interval needs to be specified on the command line. This should be `SLO`.

2.2 Evaluation Process

The following lists the evaluation characteristics performed by the `limit` application:

- Evaluate limit specifications at a prescribed rate.
- Determine if a limit specification is “enabled” and ignore the limit if not enabled.
- Determine the limiting value if it is specified as a variable or computed expression.
- Compare the current variable value to upper or lower limit value.
- Evaluate how long an upper or lower limit has been exceeded and compare to the *period out* specification.
- Determine if the `age_limit` for update has been exceeded.
- Determine if the limit is in violation, that is, if the limit exceeds the limit value, is enabled, and exceeds `period_out` OR exceeds the `age_limit` .
- If in violation, compare the current value to the “extreme value” and capture a new “extreme value” and “extreme time” if the current value exceeds the former extreme value.

2.3 Limit Violation Actions

2.3.1 Violated Limits

The following describes actions for a limit that is being “violated”:

- Set the `violation_state` `LOGICAL` variable to `TRUE`.
- If the `latch_flag` variable is not already set, set it `TRUE` and capture the time.
- Compare the current value to the extreme value and capture a new extreme value if the deviation from the limit is larger.
- When first violated, set the event indicating the limit is in violation.
- Set the limit variable display status according to specifications.

2.3.2 Limit Switched from Violated to Not Violated

The following describes options for handling of a limit that is switching from violated to not violated:

- Reset the violation state `LOGICAL` variable to FALSE.
- Reset the `period_out` counter.
- Set the event indicating that the limit is no longer in violation.

2.4 User-Configurable Responses

The actions described in *Section 2.3 Limit Violation Actions* on page 2 set flags, set events, and change display colors. The flags and display colors are indicators that a limit is in violation and are useful to an operator who is observing the display screen. However, the primary means of response to a violation is to configure the system to handle the violation event. This event is a signal that the limit is tripped (violated). The following are common methods of handling the event:

- Configure a specification for the `evnt_rsp` application to respond to the event. A potential response is to turn off relays, close values, set other events, log data, launch a script, etc. Refer to the following for related information:
  - *Event Response Utility*
  - Usage help for `evnt_rsp` on cyflex.com
- Configure a specification for the `compvar` application to respond to the event. A typical response is to compute something when the event is received. Refer to the following for related information:
  - *Creating User Computations and User Variables*
  - Usage help for `compvar` on cyflex.com
- Configure a specification for the Test Manager application to respond to the event and change the path of execution of a test procedure. Refer to the *Test Manager User Guide*.

Not all limit violations relate to serious situations that might require shutting down a test or taking safety-related actions. Consider the following:

- Use the violation flag of one limit specification as the enabling flag for another. This can let a limit specification operate only when the test fixture reaches a certain value of speed, pressure, temperature, etc.
- Use a violation event to kick off a data acquisition process by having the start event of the data acquisition process be the violation event.
- Launch a procedure of the Test Manager to advance to a new test mode when it receives the violation event.

3 Limit Specification Application

The `limit_specs` application processes limit specifications in a file. Refer to *Section 3.4 Limit Specifications File* on page 5 for details.

- The `limit_specs` application reads each specification and sends it to the appropriate instance of the `limit` application based on the registered name found in the file. Refer to cyflex.com usage help for `limit_specs` for related information.

3.1 Launching the Limit Specs Application

Launch the `limit_specs` application in the `go.scp` startup script. It. can also be run from the command line when the specifications are changed by the operator. When it has processed the specification file, it will report the number of limits found and the number of errors that were encountered, if any. Refer to *Section 3.3 Limit Specs Application Output* below for details.

3.2 Features

The `limit_specs` application features the following:

- Label of the variable to be evaluated (the `limit` variable")
- An optional `LOGICAL` *enable variable*" to enable or disable the evaluation process.
- Upper or lower limit type
- Rate of evaluation using standard process intervals: `FAS`, `MED`, `SLO`, `WARP`
- Limiting value in the form of constant, variable label, or computed expression
- A `LOGICAL` *flag variable* to hold the current state of "violation"
- A `LOGICAL` *latch variable* to hold the time when variable was first violated
- An optional event name to be set when the limit transitions from not-violated to violated
- An optional event name to be set when the limit comes out of the violated state
- An optional *period out* value for a time that the limit must be exceeded to be considered violated
- Color or blinking status of the variable display when in violation
- Age limit : an optional specification of a time during which the *limit variable* variable must be updated or the limit is considered to be violated regardless of the value.

3.3 Limit Specs Application Output

The following output will appear on the console where `limit_specs` is launched:

```
Limit_specs found 12 specifications in /specs/limit_specs.101
0 errors were encountered, leaving 12 active specs
```

The count of number of active specifications and number of errors will be written into 2 variables that are automatically created. The names of the variables will begin with the registered name of the limit task instance. The example below shows the names for the instance with the name of `Limit`.

```
LimitTotal     total number of specifications
LimitErrors    total number of errors found
```

3.4 Limit Specifications File

The limit specification format consists of a series of single line specifications containing the following fields:

- `limit_variable` is the label of a `REAL`, `INTEGER`, `REAL_ARRAY`, `INTEGER_ARRAY`, `STATISTICAL`, `COMPOSITION`, or `PROPERTY` variable
- `limit_value` may be expressed as a constant, variable label or computed expression. The units of the limit value must have the same units dimension as the limit variable but may have different units within that dimension. For example, if the limit variable has units of pressure, then the limit value must have units of pressure, but either may be [psi], [in_hg], [kpa], etc.
- `upper/lower` flag indicates Upper or Lower limit – must be entered as either a `U` or `L` single character
- `Process_interval` is the rate at which the limit will be evaluated. Specify `SLO`, `MED`, `FAS`, or `WARP`.
- `Display_type` when violated is an optional field which can be filled with a dash if not required. Specify one of the following:
  - `BLINK` to blink the default color
  - `INVERSE` to reverse background/ foreground colors
  - `BLUE`
  - `GREEN`
  - `CYAN`
  - `RED`
  - `MAGENTA`
  - `YELLOW`
  - `WHITE`
  - `BLACK`
  - `BLINK_BLUE`
  - `BLINK_GREEN`
  - `BLINK_CYAN`
  - `BLINK_RED`
  - `BLINK_MAGENTA`
  - `BLINK_YELLOW`
  - `BLINK_WHITE`
  - `BLINK_BLACK`
- `enable` a `LOGICAL` variable to enable or disable the limit. This is an optional field which can be filled with a dash (`-`) if the limit is to be always active.
- `violation_event` is the event name that will be set when transitioning from not-violated to violated. The event will be created as a zero-length event if it does not exist. This is an optional field which can be filled with a dash if not required. There are 3 events that the system always creates and have common usage.
  - `flag_limit`: no define usage
  - `abort_limit`: generally used to do an orderly shutdown of the test
  - `emergency`: generally used to do an emergency shut down of the test
- `normal_event` is the event name that will be set when transitioning from violated to not-violated. This event will be created if it does not exist. This is an optional field that can be filled with a dash if not required.
- `violation_flag` is a `LOGICAL` variable which will be set `TRUE` when this limit is

being violated. This is an optional field that can be filled with a dash if not required.

- `latch_flag` is a `LOGICAL` variable that will be set `TRUE` when the limit is first violated and will remain `TRUE` until reset by a user command. This is an optional field that can be filled with a dash if not required.
- `period_out` is a time interval in which the variable must be continuously exceeding the limit before the limit is considered to be violated. This is an optional field that can be filled with a dash if not required.
- `age_limit` is a time interval during which the limit variable must be updated else it is considered to be violated. This is an optional field that can be filled with a dash, `NO_AGE_LIMIT`, or left blank if not required.

All of the preceding fields are read by `limit_specs` as if they are on a single line, however they usually do not format easily on an 80-character line so it is normal usage to separate them into 4 lines with each of the first 3 terminated by a back-slash character which indicates line-continuation to the parser. A typical example is shown below.

```
#variable limit_value \
#         upper/lower    interval       display \
#         enable         violation_event     normal_event \
#         violation_flag latch_flag          period_out
     age_limit
cmp_in_t  86[deg_f] \
          U         SLO            BLINK_RED \
          Enab_lwr_lmt   flag_limit          -    \
          A_cmp_in_t     AL_cmp_in_t         0[sec]
     60[sec]
```

4 Additional Applications

4.1 Limit Report Application

Information about limits that have been violated is stored in a shared memory data structure called a limit latch. Previously violated limits information gets stored in this structure and can be dumped out into a readable form by the `limit_rpt` application. The output is directed to that standard output, which is usually to the console where the command is launched. Refer to cyflex.com usage help for `limit_rpt` for related information.

Some applications will automatically launch `limit_rpt` and redirect the output to be appended to a file named `/data/errors/limit_rpt.YYMMDD` where `YYMMDD` represents year, month, and day. This is normally done when the latch structures are about to be cleared and we wish to preserve a record of the information stored in the latches. The applications which do this are:

- `limit_specs`; refer to cyflex.com usage help for `limit_specs`
- `reset_ltch`; see below.
- `gp_test` whenever a new test is started by the `nt` command; refer to cyflex.com usage help for `gp_test`

4.2 Reset Latch Application

The `reset_ltch` application generates a limit report and then clears all of the information stored in the limit latches. Refer to cyflex.com usage help for `reset_ltch`.