Introduction to state_mon

Version 6

February 13, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	8/8/2017	Initial publication
2	8/23/2018	Add SGS branding
3	4/2/2020	Retrofit to new template
4	12/2/2021	Added hypertext linked cross-reference to cyflex.com usage help for `state_mon` in *Section 1 Overview* on page 1
5	5/31/2022	Updated hypertext linked cross-reference to cyflex.com usage help for `state_mon` in *Section 1 Overview* on page 1
6	2/13/2024	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

List of Figures

LIST OF TABLES

1 Overview

`state_mon` is an advanced and inherently complex solution to very complex problems that are not easily dealt with using standard tools. In addition:

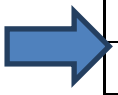
- `state_mon` is a service task that is launched and called from a `gp_test` using new `@STATE_MON_XXX` keywords.
- `state_mon` can be described as a means for implementing a Finite State Machine.

Refer to cyflex.com usage help for `state_mon` for supplemental information.

Consider an example shown in *Figure 1* where the state of the machine is defined by four logicals: A, B, C, and D, and the desired output is a combination of the logicals: E, F, G, H and I.

Figure 1: Finite State Machine Example

INPUTS			
A	B	C	D
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0



OUTPUTS				
E	F	G	H	I
1	0	1	1	1
1	0	1	0	1
1	1	1	1	1
1	0	1	0	1
1	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	0	0	0
0	0	1	0	1
1	0	0	0	0
1	1	1	0	0
0	1	1	1	1
1	0	1	1	0
0	0	1	0	1
0	1	1	1	0
0	1	1	1	0

Pertaining to the preceding figure:

- For N logical inputs, there are 2^N possible combinations leading to potentially different desired outputs. There are 16 possible combinations for this example.
- To implement the desired logic in `er_specs` or `gp_test`, an event must be created that would be set every time any one of the input logicals changes state. 16 event responses or `gp_test` modes would handle all the possible states. This is certainly doable, but it difficult to document and maintain.
- TRP Laboratories has addressed a requirement to manage a cart docking system that must examine the state of 16 logical inputs which translates to 65,536 potential combinations.
- `state_mon` handles the following conditions:
  - Inputs are not logicals
  - The state of the system is defined by stability, a timeout. or a limit violation,
  - Defining the transition to another state with the knowledge it is going to take time to reach that state

2 stbl Files

The `stbl` files format is similar to `vrbl` files:

```
@PROCESS_INTERVAL
  FAS
@FILE_FORMAT
  HORIZONTAL_LABELS
@STATE_VARIABLES
         crt_dock_pb:EQ_S  crt_undock_pb:EQ_S   ram_en_pb:EQ_S...
@STATE_VALUES_TABLE
emergency_inhibit      -                 -                   -    ...
dock_ready             released          released            -    ...
docking                pushed            released            pushed
undocking              released          pushed              pushed
docked                 released          released                 ...
undocked               released          released            -    ...
dock_limbo             released          released            -    ...
docking_pmp_inhibit      pushed            released              -  ...
docking_run_inhibit      pushed            released              -  ...
docking_f_grd_inhibit    pushed            released              -  ...
docking_shaft_inhibit    pushed            released              -  ...
docking_ram_inhibit      pushed            released      released   ..
docking_ls_inhibit       pushed            released            -    .
undocking_pmp_inhibit    released          pushed              -  ...
undocking_run_inhibit    released          pushed              -  ...
undocking_f_grd_inhibit  released          pushed              -  ...
undocking_clnt_inhibit   released          pushed              -  ...
undocking_shaft_inhibit  released          pushed              -  ...
undocking_ram_inhibit    released          pushed        released   ..
undocking_ls_inhibit     released          pushed              -  ...
docking_dual_button      pushed            pushed              -  ...
dock_ls_error            -                 -                   -  ...
default                  released          released      released  ...
```

- The `@PROCESS_INTERVAL` specifies how often the input variables are to be examined to verify the state of the system.
- Horizontal and vertical `@FILE_FORMAT`s are available as for `vrbl` files.
- The `@STATE_VARIABLES` is a list of the variables to be examined to determine the state of the system followed by a colon and an action code that will be described later.
- The `@STATE_VALUES_TABLE` starts with an index that is typically a string array element that describes the state of the system. There is an entry for each of the state variables that specifies the value of that variable in the given state. A dash, '-', is used to indicate that the value of that state variable for that particular state is not important.

3 state_mon Actions

Table 1 summarizes the available actions which can be associated with the state variable inputs in the `stbl` file.

Table 1: State Variable Inputs Actions

Action	Description
`LO`	lower limit
`LO_W, LO_C, LO_S`	lower limit warning, critical, state change
`UP`	upper limit
`UP_W, UP_C, UP_S`	upper limit warning, critical, state change
`NE`	not equal
`NE_W, NE_C, NE_S`	not equal warning, critical, state change
`EQ`	equal
`EQ_W, EQ_C, EQ_S`	equal warning, critical, state change
`DV`	deviation
`DV_W, DV_C, DV_S`	deviation warning, critical, state change
`SD`	standard deviation
`SD_W, SD_C, SD_S`	standard deviation warning, critical, state change
`CV`	coefficient of variation
`CV_W, CV_C, CV_S`	coefficient of variation warning, critical, state change
`TD`	time delay
`TD_W, TD_C, TD_S`	time delay warning, critical, state change

4 gp_test @STATE_MON_XXX Keywords

Three `state_mon` related `gp_test` keywords are used to specify the necessary inputs to the process. The `@STATE_MON_SPEC_FILES` keyword tells the process to read a line specified by an index from one or more `stbl` files.

Depending on the state of the inputs specified in the file(s) and the requested `action_code`, the process could be deemed to have failed or succeeded and the appropriate exit path will be taken based on the `@STATE_MON_ACTIONS` keyword.

In the event that some subset of the inputs indicates a state change, an abort or an emergency situation, or if a timeout occurs, other exit paths may be taken based on the `@STATE_MON_EXCEPTIONS` keyword.

```
@STATE_MON_ACTIONS
# success path    failure path    read_mode    action_code
  10              15              READ         'VERIFY'

@STATE_MON_SPEC_FILES
# spec file pathname                file index label
  /specs/gp/stbls/stbl_state.dock   dock_string?dock_index

@STATE_MON_EXCEPTIONS
# time_out timeout_path state_change_path  warning_path  critical_path
  5[sec]    90            95                 50            60
```

5 state_mon Action Codes

Three action codes are available:

1. IMMEDIATE
2. VERIFY
3. MONITOR

These describe the action that `state_mon` is expected to execute when interpreting the `stbl` file contents. *Table 2* describes their behaviors.

Table 2: state_mon Action Code Behaviors

action_code	Behavior
IMMEDIATE	Parameters will be evaluated once. Those with actions ending in _C, _W or _S will cause immediate termination to the associated path if they evaluate as FALSE. Parameters having actions without _C, _W or _S extensions will cause termination to the failure path if any evaluate as FALSE. The success path will be taken if all evaluate as TRUE. The order of precedence is state change path, critical path, warning path then success/failure path. Parameters with some stability actions do not make sense in the context of a single evaluation - they will be ignored. Deviations will be evaluated on the basis of a single sample even if a longer sample period is specified.
VERIFY	Parameters will be evaluated continuously at the rate specified in the stbl file. Those with actions ending in _C, _W or _S will cause immediate termination to the associated path if they evaluate as FALSE. Parameters having actions without _C, _W or _S extensions are not necessarily expected to evaluate as TRUE when this mode is first entered. They will be evaluated continuously and the success path will be taken only when all evaluate as TRUE. The mode will only terminate due to a timeout, state change or if a warning or critical fault occurs. The order of precedence is state change path, critical path, warning path then success/failure path.
MONITOR	Parameters will be evaluated continuously at the rate specified in the stbl file. Those with actions ending in _C, _W or _S will cause immediate termination to the associated path if they evaluate as FALSE. If any of the parameters having actions without _C, _W or _S extensions evaluate as FALSE, the mode will exit to the failure path. The mode will only terminate due to a timeout, state change or if a warning or critical fault occurs. The order of precedence is state change path, critical path, warning path then success/failure path.

6 String Arrays as Indices

Strings that describe the state of the system are used as indices into both the `stbl` and `vrbl` files. Use string arrays to increment the index of the array and search through the state table until finding the state that matches the current inputs.

```
@REG_NAME
  AR_dock

#
**************************************************************************
# Array to use as an index into the stbl and vrbl files that control
the
# test cell docking system.
#
**************************************************************************
#array_label          source        units   format
 dock_string          STRING_ARRAY  none    -

#dimensions enums
23

dock_string_single:0=emergency_inhibit
dock_string_single:1=dock_ready
dock_string_single:2=docking
dock_string_single:3=undocking
dock_string_single:4=docked
dock_string_single:5=undocked
dock_string_single:6=dock_limbo
dock_string_single:7=docking_pmp_inhibit
dock_string_single:8=docking_run_inhibit
dock_string_single:9=docking_f_grd_inhibit
dock_string_single:10=docking_shaft_inhibit
dock_string_single:11=docking_ram_inhibit
dock_string_single:12=docking_ls_inhibit
dock_string_single:13=undocking_pmp_inhibit
dock_string_single:14=undocking_run_inhibit
dock_string_single:15=undocking_f_grd_inhibit
dock_string_single:16=undocking_clnt_inhibit
dock_string_single:17=undocking_shaft_inhibit
dock_string_single:18=undocking_ram_inhibit
dock_string_single:19=undocking_ls_inhibit
dock_string_single:20=docking_dual_button
dock_string_single:21=dock_ls_error
dock_string_single:22=default
```

7 Trace File Format

The following is an example of `state_mon` messages:

```
gp_state_mon.dock,15  | Read stateball file in IMMEDIATE mode
 T/D-16:43:28 09/06/13

smCFG to 1732 action=IMMEDIATE read_once=0 nfiles=1


state_mon - start of IMMEDIATE mode - 09/06/13 16:43:28.45

         time:(spec_file_num)    vrbl_name:state (crit, warn)  vrbl_value,
spec_value
                                          action  pass  pass
  time=  0.00:(0)              crt_dock_pb:SE_C (1,1) vrb   =    released,
sp.val=    released
  time=  0.00:(0)            crt_undock_pb:SE_C (1,1) vrb   =    released,
sp.val=    released
  time=  0.00:(0)               ram_en_pb:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)                   estop:SE_C (1,1) vrb   =      normal,
sp.val=      normal
  time=  0.00:(0)                 run_sns:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)              ds_grd_ls_F:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)                 ds_up_ls:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)              crt_dock_ls:SE_C (1,1) vrb   =  not_docked,
sp.val=  not_docked
  time=  0.00:(0)            dock_clnt_lsl:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)            crt_undock_ls:SE_C (0,1) vrb   =ram_not_extd,
sp.val=    ram_extd
  time=  0.00:(0)          dock_hyd_pmp_en:SE_C (1,1)  Don't Care Spec

 T/D-16:43:28 09/06/13
   state_mon -  SE_C or SE_W failed, took state_mon critical failure path

gp_state_mon.dock,25  | Critical path-NO MATCH from IMMEDIATE action
 T/D-16:43:28 09/06/13
   immediate -

gp_state_mon.dock,15  | Read stateball file in IMMEDIATE mode
 T/D-16:43:28 09/06/13

smCFG to 1732 action=IMMEDIATE read_once=0 nfiles=1


state_mon - start of IMMEDIATE mode - 09/06/13 16:43:28.45

         time:(spec_file_num)    vrbl_name:state (crit, warn)  vrbl_value,
spec_value
                                          action  pass  pass
  time=  0.00:(0)              crt_dock_pb:SE_C (1,1) vrb   =    released,
sp.val=    released
  time=  0.00:(0)            crt_undock_pb:SE_C (1,1) vrb   =    released,
sp.val=    released
  time=  0.00:(0)               ram_en_pb:SE_C (1,1)  Don't Care Spec
```

```
  time=  0.00:(0)                    estop:SE_C (1,1) vrb   =      normal,
sp.val=      normal
  time=  0.00:(0)                  run_sns:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)               ds_grd_ls_F:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)                  ds_up_ls:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)               crt_dock_ls:SE_C (1,1) vrb   =  not_docked,
sp.val=  not_docked
  time=  0.00:(0)             dock_clnt_lsl:SE_C (1,1)  Don't Care Spec
  time=  0.00:(0)             crt_undock_ls:SE_C (1,1) vrb   =ram_not_extd,
sp.val=ram_not_extd
  time=  0.00:(0)           dock_hyd_pmp_en:SE_C (1,1)  Don't Care Spec

 T/D-16:43:28 09/06/13
   state_mon -  state_mon replied - OK

gp_state_mon.dock,40  | State match found.  Now read the furball file using
the
 T/D-16:43:28 09/06/13
   AUXILIARY_TASK -  auRPL6a4-140
```

8 Additional Reference

8.1 Strategies to Minimize the Number of States

- Make use of the “Don’t Care” spec to collapse many states into one. For example, if there is an emergency, in many cases it does not matter what buttons are being pushed. The system must be put into a safe state.
- Create one logical that consolidates multiple similar inputs. For example, there are four coolant level sensors on the HedgeHog cart docking system that have been consolidated into one.

8.2 Hardware Logic Duplication For Clarity

There is often hardware logic designed into various circuits that will inhibit operator inputs. For example, the estop will often disable the input from a button that the operator would use to request the cell systems to turn on. It is a good practice to duplicate the same logic in the state tables so the operator will be made aware if there is another hardware system that is inhibiting the operation he wishes to perform.

8.3 Testing For Complete Coverage

- Once the state table(s) is/are generated, test them to verify that all possible input combinations are recognized.
- `vrbl` files are created that contain every possible combination of each of the inputs. This can be done relatively easily in a spreadsheet.
- Where necessary, the inputs are `set_inactive` and each line of the `vrbl` file is read in individually. The resulting state is written to a file for later analysis. Every possible set of inputs should result in a defined state. If there are gaps, they need to be understood and addressed regardless how remote the possibility of an occurrence appears to be.
- `gp_tests` have been developed that provide an example framework for these tests.