

WHEN YOU NEED TO BE SURE

SGS

CyFlex® Knowledge Article

Converting from Old to New Control Programs

Author: David Ruthmansdorfer

April 21, 2021

1 Introduction

A script has been created for CyFlex release 6.3.24 to simplify the process of going from the old control programs to the new control programs. The script uses entries in your `ctrl_specs.xx` file to populate an `eng_ctrl_specs.xx` file. See the example below:

```
[tcl@mynode1 specs]$ ctrl_spec_converter /specs/ctrl_specs.1
Your new eng_ctrl_specs file can be found in
/tmp/new_eng_ctrl_specs.1
```

After review and update, move the newly formatted file to the `/specs` directory, which is the default search path for `eng_ctrl_specs` application, or a `/cell/sub_systems` subdirectory if preferred.

Enter the following for full details on usage and compatibility of the script:

```
$ use ctrl_spec_converter
```

2 Converting the Old Control Program to the New Control Program

Execute the following steps:

1. Verify there are no errors with the existing control spec file. With CyFlex running, execute the following:

```
clear_error
ctrl_specs
errs
```

2. Determine whether there are any errors generated related to `ctrl_task` and remedy them.
3. Run the `ctrl_spec_converter` application with the old `ctrl_specs.xx` file. For example:

```
ctrl_spec_converter /specs/ctrl_specs.1
```

4. Move the new file to its final directory. For example:

```
$ mv /tmp/new_eng_ctrl_specs.1 /specs/eng_ctrl_specs.1
```

5. Back up the old user control spec file:

```
$ cp /specs/ctrl_specs.1_user /specs/ctrl_specs.1_user.old_ctrl_task
```

6. Edit the user control spec file in one of the two following ways. The advantage of method (b) is that user loops can be grouped in more meaningful ways.

a. **Quickest way:**

Add the `critical` column to all the `@USER_LOOP_CTRLER` keywords.

The old format follows:

```
@USER_LOOP_CTRLER
# label      command      interval      open-loop
# root      output units      ramp rate
AirVolFlow  %              MED           2.0
```

The new format has the critical entry:

```
@USER_LOOP_CTRLER
# label      command      interval      open-loop      critical
# root      output units      ramp rate
AirVolFlow  %              MED           2.0           N
```

b. **Best way:**

To maximize the benefit of the new control programs, break each of the control loops into a separate file for each loop or group the control loops by category (`user_ctrl_specs.cell_ventilation` for example).

For each control loop, copy and paste the section from the `@USER_LOOP_CTRLER` keyword to the following `$END` into a new file. A good naming convention for individual loops is to name the file `user_ctrl_specs.loop_name` so it is easy to identify.

Note:

Automatic reading of plug and play (PNP) files is no longer supported. Each former PNP specification must be directly launched.

When complete, add the `critical` column mentioned above in step 6.a and review the comments for each keyword to see if new capabilities can be used to advantage.

7. Edit the `go` script:

- Remove all instances of `ctrl_task`.
- Remove `comp_ctrl`.
- Remove `ctrl_specs`. Ensure this command is not running in multiple places in the `go` script.
- Substitute the following in place of running `ctrl_specs`:

```
eng_ctrl_specs /specs/eng_ctrl_specs.xx
# full path is not needed if file is located in specs
directory
user_ctrl_specs /specs/user_ctrl_specs.clnt_ot_t
user_ctrl_specs /specs/user_ctrl_specs.FMS
user_ctrl_specs /specs/user_ctrl_specs.exp_tnk_p
```

Note:

If all of the user control loops are in one file, you can replace all of the `user_ctrl_specs` instances with the following individual line:
`user_ctrl_specs /specs/ctrl_specs.xx_user`

3 New Features

3.1 Bumpless Transfer

```
# -----
#
#
#   Bumpless Transfer
#   (OPTIONAL ENTRY)
#
# The engine control task can operate in one of two ways when feed forward
# control is made either active or inactive and when a controller is
# switched from open to closed loop.
#
# The first way (Y or YES) is called bumpless transfer. In that mode, the
# output will NOT suddenly change when feed forward is turned on/off
# or the control mode is changed. Instead, the integral term will be
# adjusted by the feed forward term and the output will remain
# constant (bumpless).
#
# In the second way (N or NO), the output is instantly changed by the feed
# forward term when a switch is made. This is how the old control
# task (ctrl_task) has worked since some time in 2005 and has been found to
# be necessary for transient emissions testing.
#
# If this keyword is not used, then transfer will NOT be bumpless so that
# behaviour of the new eng_ctrl_task is the same as the older ctrl_task
#
# NOTE: Always use all 3 entries even if a second dyno is not being used
#
@BUMPLESS_TRANSFER
# First Dyno      Second Dyno      Throttle
#                (if dual dyno)
#   YES           YES           NO
```

3.2 Change Display Precision of Dyno and Throttle Control Loop Variables

If a higher or lower precision is required for the dyno or throttle controller variables, use the keywords below to change them.

```

# -----
#
#   Specify some attributes of the created controller variables.
#   (OPTIONAL ENTRIES)
#
# @DYNO_LOOP_VRBL_OPTS_CM
#   <display_precision>      <history_active>      <history_tolerance>
#
#   where:
#       display  - The number of decimal places to be displayed by
#       precision the display task when the variable
#                   is displayed. Default is 1
#
#       history  - A flag indicating if the variable should be included
#       active   in the history log. Valid entries for this field
#                   are Y, YES, T or TRUE. All other entries are
#                   set to FALSE.
#
#   NOTE: The history save flag is now obsolete since ALL variables are
#           saved in history files. Entries should still be made
#           until the eng_ctrl_specs program can be modified.
#
#       history  - The amount the variable must change for it to be
#       tolerance written to the history log. Default is 5.0.
#                   A value of 0 will NOT be accepted.
#
#   NOTE: There is a similar specification for each of the created
#   variables.
#           The only thing that is different is the suffix of the keyword.
#   The
#           additional keywords for created variable options are:
#
# @DYNO_LOOP_VRBL_OPTS_PT
# @DYNO_LOOP_VRBL_OPTS_IT
# @DYNO_LOOP_VRBL_OPTS_DT
# @DYNO_LOOP_VRBL_OPTS_FF
# @DYNO_LOOP_VRBL_OPTS_OL
# @THROTTLE_LOOP_VRBL_OPTS_CM
# @THROTTLE_LOOP_VRBL_OPTS_PT
# @THROTTLE_LOOP_VRBL_OPTS_IT
# @THROTTLE_LOOP_VRBL_OPTS_DT
# @THROTTLE_LOOP_VRBL_OPTS_FF
# @THROTTLE_LOOP_VRBL_OPTS_OL
# -----

```

3.3 Functional Changes

Characteristics of the old control program:

- Long keywords (TORQUE, SPEED) have delays of 50ms in them which can cause issues on very fast transients.
- The delays can be avoided by using the short keywords (TRQ, SPD).
 - There is an additional difference with the short keywords in that the control mode (open/closed) is not changed the way that you think it might.
 - The long keywords set a target **and** put the appropriate controller in closed loop.
 - The short keywords only set a target.

Characteristics of the new control program:

- The delays have been removed with the long keywords and long keywords are usable for fast transients.
- The long keywords set both the target and open/closed mode.
- The short keywords only set the targets.

4 Additional Information

Refer to the following documents for supplemental information:

- [User Control Loops](#)
- [Engine Controls](#)