TRP LABORATORIES

CyFlex® User Control Loops

Version 10

February 5, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	6/23/2016	Initial publication
2	8/23/2018	Added sentions: • 1.3. Summary of Control Tasks • 9. CyFlex Documents • 10. Revision History (replaced in Version 4 by this *Version History* section) Converted document to CMX format
3	1/29/2019	• Revised Section 9. CyFlex Documents to remove documents resource URL. • Add LV output option to specification file • Add `u_ctrl` to the command options • Correct flow diagram (target/reference) • Add reference to CyFlex.com
4	3/27/2020	Retrofit to new template
5	4/7/2020	Revise *Appendix B. Control Task Specification File* on page 21 to refresh with usage for `user_ctrl_specs` instead of `ctrl_specs`.
6	4/13/2021	Added specification usage for bumpless transfer on page 32 of *Appendix B. Control Task Specification File*.
7	8/11/2021	Revised *Table 1* on page 1 to add column with hyperlinked cross-references to cyflex.com usage help for listed control loop programs. Removed Appendix A which contained usage help for `u_ctrl`, u_`mode`, `u_target`, and `u_ramp` and changed Appendix A references to refer to *Table 1*. Renamed Appendix B to Appendix A. Revised cross-references formerly to Appendix B to Appendix A. Renamed Appendix C to Appendix B and removed usage for `user_ctrl_specs`. Included hyperlinked cross-reference to cyflex.com usage for `user_ctrl_specs` in *Table 1*. Revised cross-references formerly to Appendix C to Appendix B. Renamed Appendix D to Appendix C. Revised cross-references formerly to Appendix C to Appendix D.

Version	Date	Revision Description
8	1/18/2022	Updated *Figure 1* on page 4 for varying Kp values Corrected entries for `FA` and `MD` variables in *Table 4* on page 10 Corrected the explanation of error scaling in *Section 4 CyFlex Implementation of User PID Controls* on page 8 Added a description of the `@BUMPLESS` keyword in *Section 5.1 Transitions When Turning Feed Forward On/Off* on page 15 Corrected various spelling errors in *Appendix B. Control Task Specification File* on page 21 Corrected the command map table in *Appendix C. Example Command Map* on page 38
9	5/18/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions
10	2/5/2024	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.

Related Documents

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

1 OVERVIEW 1
1.1 Document Purpose 1
1.2 Control Systems 1
1.3 Summary of Control Tasks 1
2 PID CONTROLLERS 3
2.1 Proportional Term 3
2.2 Integral Term 4
2.3 Derivative Term 5
3 MANUAL TUNING 7
4 CYFLEX IMPLEMENTATION OF USER PID CONTROLS 8
4.1 Selecting the Desired Value for the Controller 8
4.2 Finding the Controller Error 9
4.3 Displaying Controller Values 9
4.4 Calculating the PID Controller Output Command 11
4.5 Integral Bounding 12
4.6 Simplified PID Controller Example 12
5 FEED FORWARD CONTROL 14
5.1 Transitions When Turning Feed Forward On/Off 15
5.2 Feed Forward Example 15
6 CONTROLLER OUTPUT ADJUSTMENTS 16
6.1 Hysteresis 16
6.2 Output Bounding 17
6.3 Output Command Map 17
7 OPEN OR CLOSED LOOP OPERATION 18
8 MANUAL CONTROLS 19
APPENDICES 20
Appendix A. Test Procedure Keywords 20
Appendix B. Control Task Specification File 21
Appendix C. Example Command Map 38

List of Figures

FIGURE 1: PLOT OF PV VS. TIME FOR THREE VALUES OF KP (KI AND KD HELD CONSTANT) 4
FIGURE 2: PLOT OF PV VS. TIME FOR THREE VALUES OF KI (KP AND KD HELD CONSTANT) 5
FIGURE 3: PLOT OF PV VS. TIME FOR THREE VALUES OF KD (KP AND KI HELD CONSTANT) 6
FIGURE 4: SET CONTROL LOOP TARGET FLOW 8
FIGURE 5: PID OUTPUT COMMAND CALCULATION FLOW 11
FIGURE 6: PID CONTROLLER EXAMPLE 12
FIGURE 7: FEED FORWARD PREDICTION CALCULATION FLOW 14
FIGURE 8: CONTROLLER OUTPUT ADJUSTMENTS FLOW 16
FIGURE 9: OPEN/CLOSED LOOP OUTPUT FLOW 18
FIGURE 10: MANUAL SET OF OPEN LOOP CONTROLLER OUTPUT FLOW 19

LIST OF TABLES

TABLE 1: CYFLEX CONTROL LOOP PROGRAMS 1
TABLE 2: EFFECTS OF INDEPENDENTLY INCREASING A PARAMETER 7
TABLE 3: CONTROLLER REAL VARIABLES CREATED BY A NEW USER CONTROL TASK 9
TABLE 4: CONTROLLER LOGICAL VARIABLES CREATED BY A NEW USER CONTROL TASK 10
TABLE 5: CONTROLLER STRING VARIABLES CREATED BY A NEW USER CONTROL TASK 10

1 Overview

1.1 Document Purpose

This document describes how to use the CyFlex user control loops software.

1.2 Control Systems

In a typical control system, the process variable is the system parameter that needs to be controlled, such as temperature (ºC), pressure (psi), or flow rate (liters/minute). A sensor is used to measure the process variable and provide feedback to the control system. The set point is the desired or command value for the process variable, such as 100 degrees Celsius in the case of a temperature control system.

The difference between the process variable and the set point at any given time is used by the control system algorithm to determine the desired actuator output to drive the system. For instance, if the measured temperature process variable is 100 ºC and the desired temperature set point is 120 ºC, then the actuator output specified by the control algorithm might be to drive a heater. Driving an actuator to turn on a heater warms the system, and results in an increase in the temperature process variable. This is called a closed loop control system because the process of reading sensors to provide constant feedback and calculating the desired actuator output is repeated continuously and at a fixed loop rate.

1.3 Summary of Control Tasks

Table 1 lists the control loop programs.

Table 1: CyFlex Control Loop Programs

Task	Purpose	Specification File	Cyflex.com Usage Help Reference
`user_ctrl_specs`	Specifies the user loop configuration via keywords	`user_control_spec_file`. See *Appendix B. Control Task Specification File..* Contains user loop specifications defined using test procedure keywords	Refer to `user_ctrl_specs`
`u_ctrl`	Sets the user control loop target, mode and/or ramps the target value	None	Refer to `u_ctrl`
`u_ramp`	Ramps the target value (set the user control loop ramping)	None	Refer `to` `u_ramp`
`u_target`	Sets the user control loop target.	None	Refer to `u_target`
`u_mode`	Chooses the user control loop mode – open or closed	None	Refer to `u_mode`

These files are located in the following directories on the test cell.

- Applications (tasks): `/cyflex/bin`
- Samples of specification files for the applications: *`/specs.def`* `(specs defaults)`

ⓝ***Note:***

User-configured spec files can reside in any directory if the full path is provided when the associated task is invoked. The default location is `/specs`.

2 PID Controllers

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and is universally accepted for industrial control. PID controllers perform in a wide range of operating conditions and offer functional simplicity.

The basic idea behind a PID controller is to read a sensor, then compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output.

The Proportional-Integral-Derivative control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller.

Defining ***u(t)*** as controller output, the final form of the PID algorithm is:

$$\mathrm{u}(t) = \mathrm{MV}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

where:

K_p:	Proportional gain, a tuning parameter
K_i:	Integral gain, a tuning parameter
K_d:	Derivative gain, a tuning parameter
e:	Error = Set Point – Feedback Variable
t:	Time or instantaneous time (the present)
τ:	Variable of integration; takes on values from time 0 to the present t

2.1 Proportional Term

The proportional term produces an output value that is proportional to the present error value. The proportional response can be adjusted by multiplying the error by a constant ***Kp***, called the proportional gain constant.

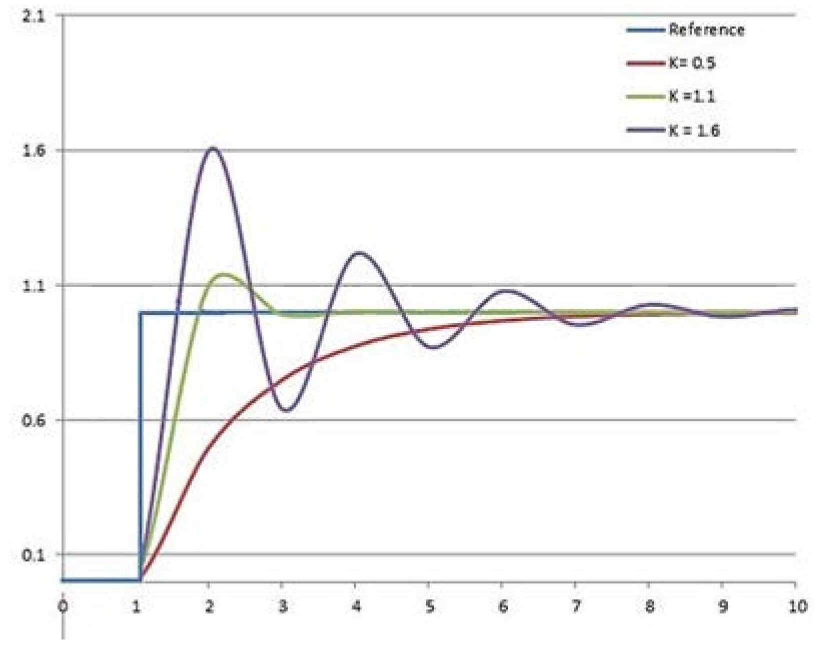
The proportional term is given by:

$$P_{\mathrm{out}} = K_p\, e(t)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances.

Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change in the early stages of a disturbance or change in set point.

Figure 1: Plot of PV vs. Time for Three Values of Kp (Ki and Kd held constant)



2.2 Integral Term

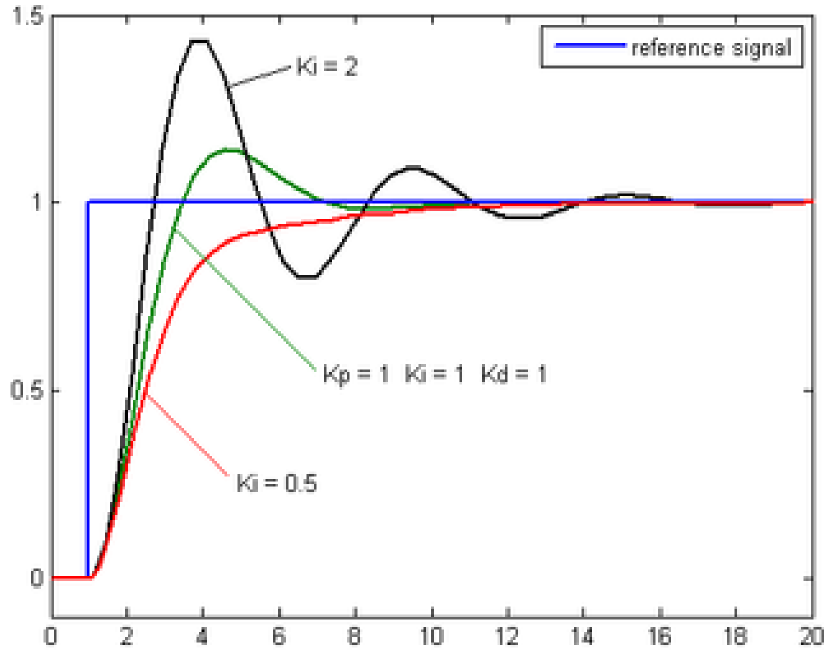
The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (***Ki***) and added to the controller output.

The integral term is given by:

$$I_{\text{out}} = K_i \int_0^t e(\tau)\, d\tau$$

The integral term accelerates the movement of the process towards the set point and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set point value.

Figure 2: Plot of PV vs. Time for Three Values of Ki (Kp and Kd held constant)



2.3 Derivative Term

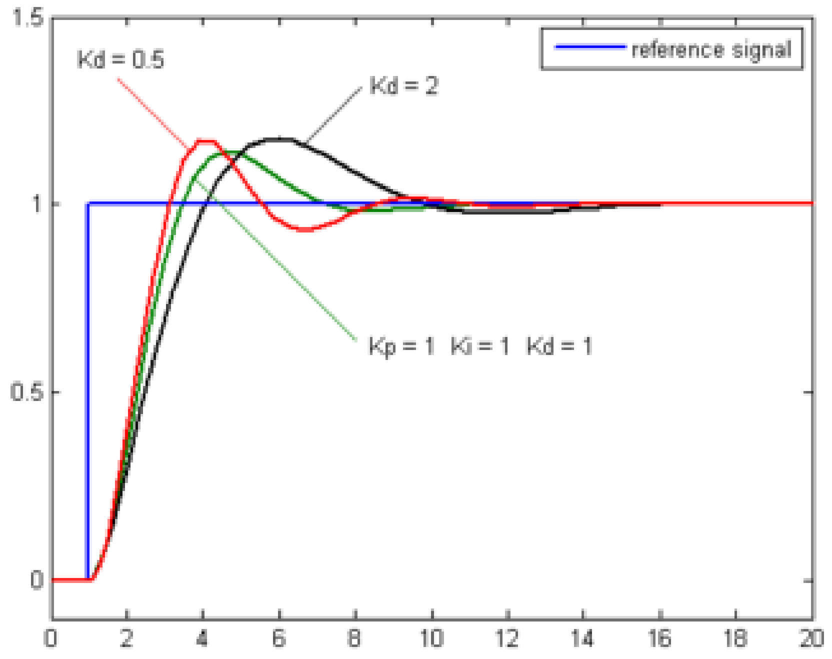
The derivative term is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain ***Kd***. The magnitude of the contribution of the derivative term to the overall control action is manipulated by the derivative gain, ***Kd***.

The derivative term is given by:

$$D_{\text{out}} = K_d \frac{d}{dt} e(t)$$

Derivative action, however, is seldom used in practice because of its inherent sensitivity to measurement noise. If this noise is severe enough, the derivative action will be erratic and actually degrade control performance. Large, sudden changes in the measured error (which typically occur when the set point is changed) cause a sudden, large control action stemming from the derivative term. This problem can be ameliorated to a degree if the measured error is passed through a linear low-pass filter.

Figure 3: Plot of PV vs. Time for Three Values of Kd (Kp and Ki held constant)



3 Manual Tuning

One simple tuning method that can be used with the controller on-line is to first set ***Ki*** and ***Kd*** values to zero. Then increase the ***Kp*** until the output of the loop begins to oscillate. Next, set the ***Kp*** to approximately half of that value. Then increase ***Ki*** until any offset is corrected in sufficient time for the process. Too much ***Ki*** will cause instability.

Finally, increase ***Kd***, if required, until the loop is acceptably quick to reach its reference after a load disturbance. Too much ***Kd*** will cause excessive response and overshoot.

Experiment by changing the setpoint back and forth in increasing step sizes until the entire operating range is covered.

In order to find the sign of the gains, recall the formula for the output as a function of the error. For the proportional term, this is:

$$P_{\mathrm{out}} = K_p\, e(t)$$

where:

The error is the set point minus the feedback value.

If an increase in the output is required when the set point is above the feedback value, then all of the gains are positive. If a decrease in the output is required when the set point is above the feedback, then all of the gains are negative.

A fast PID loop usually overshoots slightly to reach the set point more quickly; however, some systems cannot accept overshoot, in which case an over-damped closed-loop system is required, which will require a ***Kp*** setting significantly less than half that of the ***Kp*** setting that was causing oscillation.

Some systems being controlled are non-linear. That is, they have different input to output sensitivity at different operating conditions. Control tuning should generally be done at the most sensitive operating condition in order to avoid instabilities. The effects of non-linearity can be reduced or compensated for by using the error scaling multiplier, feed-forward, or command mapping features discussed elsewhere in this document.

Table 2: Effects of Independently Increasing a Parameter

Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_p small

4 CyFlex Implementation of User PID Controls

The CyFlex implementation of the general PID controller has a number of additional features added compared to the system described above. Some of these features provide additional flexibility while others allow switching from closed loop to other forms of control.

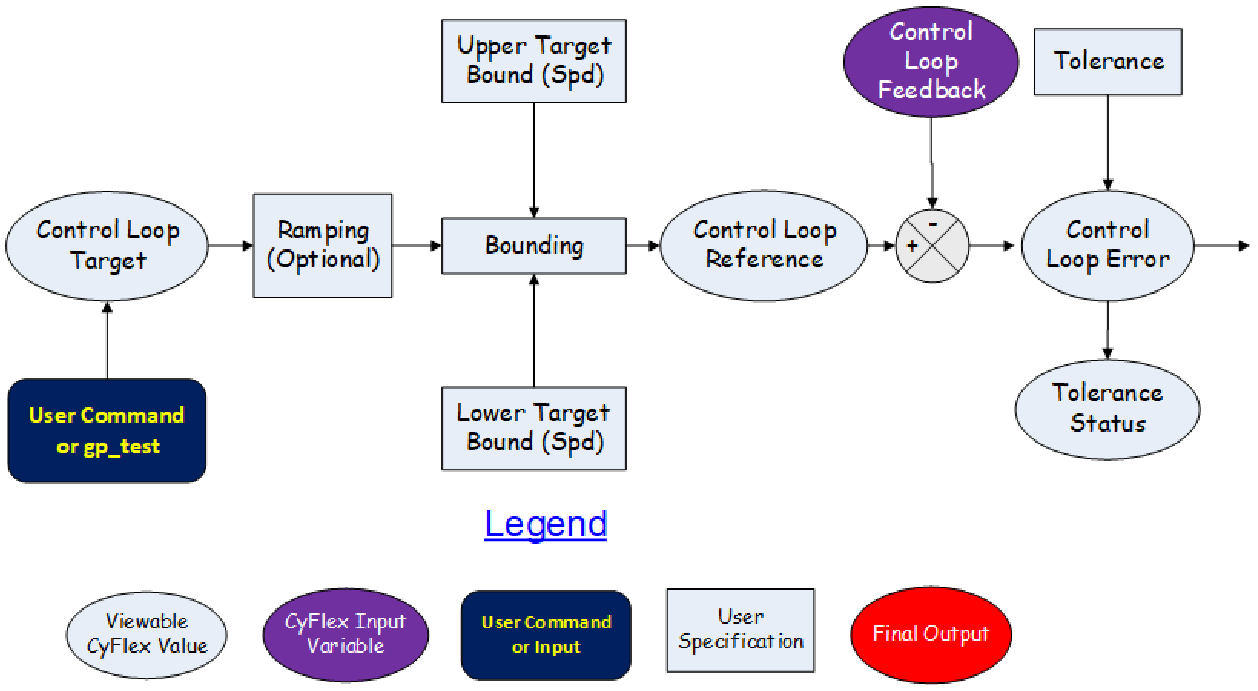
Refer to the flow diagrams below that include these features. The legend in the flow diagram immediately below applies also to the other diagrams.

4.1 Selecting the Desired Value for the Controller

The process begins with setting a target or deferring to the `gp_test` program to set a target for the control loop. Refer to the left side of the diagram in *Figure 4* below This can be done with the `u_target` or `u_ctrl` commands or from within a general-purpose test procedure with the `@USER_LOOP` keyword.

See *Table 1* on page 1 for cyflex.com usage help references to user commands and to *Appendix A. Test Procedure Keyword* on page 20 for details about the test procedure keyword.

Figure 4: Set Control Loop Target Flow



The closed loop target is not automatically set if either one of three CyFlex features have been requested. First, is the user option of setting upper and lower bounds on the requested value. These bounds are set in the control task specification file (see *Appendix B. Control Task Specification File*) and are suggested to screen requests that may cause damage to the item being controlled. The bounds may also represent the physical limits of the controlled process. For example, bounds of 100-210 degrees Fahrenheit may be reasonable limits for control of a coolant temperature.

The second option that can be exercised on the target value is that the target can be ramped from the present value (with the `u_ramp` or `u_ctrl` commands), or from an arbitrary value in a test procedure, to the final value. The ramp rate can be set in a specification file, with the `u_ramp`, `u_ctrl` commands, or in a test procedure.

The third feature is that target will not be acted upon if the controller is in an open loop mode.

The ramped value can be displayed as the controller reference. Ramping provides a linear transition of the controller target from one value to the next at a specified rate.

4.2 Finding the Controller Error

The controller error is the difference between the controller reference and the actual feedback. It is important to note the sign of this difference because it will affect the sign of the gains that are used.

Error = Set Point – Actual

Or,

Error = Controller Reference – Feedback Value

4.3 Displaying Controller Values

Each control loop is given a user-defined name. Often times the name of the feedback variable is used, but this need not be the case.

From the legend in *Figure 4* on page 8, note that the controller target, reference, and error values are all stored in viewable CyFlex variables. The variable names are the controller name with a three-character suffix attached. *Table 3* shows all of the controller variables that are created by the new user control task.

Note:

Unless stated otherwise, any value may be changed in the following tables at any time.

Table 3: Controller Real Variables Created by a New User Control Task

Controller Real Variables	Descriptions
`BaseName_TR`	Controller target (set point when ramping is completed)
`BaseName_RF`	Controller reference (set point while ramping, with bounding)
`BaseName_ER`	Controller error – display only
`BaseName_PT`	Proportional correction term (scaled error multiplied by the proportional gain) – display only
`BaseName_IT`	Integral correction term (scaled error integrated over time and multiplied by the integral gain, with bounding) – display only
`BaseName_DT`	Derivative correction term (change in error times the derivative gain) – display only
`BaseName_FF`	Feed forward term (feed forward value times the feed forward gain) – display only
`BaseName_CM`	Final controller command (sum of PT, IT, DT, FF and bounding) – display only

Controller Real Variables	Descriptions
`BaseName_OL`	Controller open loop target. The actual open loop output may be bounded. If set, this value will also exist even when the controller is in closed loop.
`BaseName_PG`	Proportional gain value
`BaseName_IG`	Integral gain value
`BaseName_DG`	Derivative gain value
`BaseName_FG`	Feed forward gain value
`BaseName_LB`	Lower output bound value
`BaseName_UB`	Upper output bound value
`BaseName_IL`	Lower integral bound value
`BaseName_IU`	Upper integral bound value
`BaseName_LO`	Lower target bound value
`BaseName_HI`	Upper target bound value
`BaseName_TO`	Controller tolerance value
`BaseName_FV`	Controller feedback value (display only)

Table 4: Controller Logical Variables Created by a New User Control Task

Controller Logical Variables	Descriptions
`BaseName_TL`	`TRUE` or 1 when the controller is within the tolerance specified – display only
`BaseName_FA`	`TRUE` or 1 when feed forward has been set active
`BaseName_MD`	`TRUE` or 1 when the controller is in closed loop. The value will be `CLOSED_LOOP`. `FALSE` or 0 when in open loop. The value will be `OPEN_LOOP`.

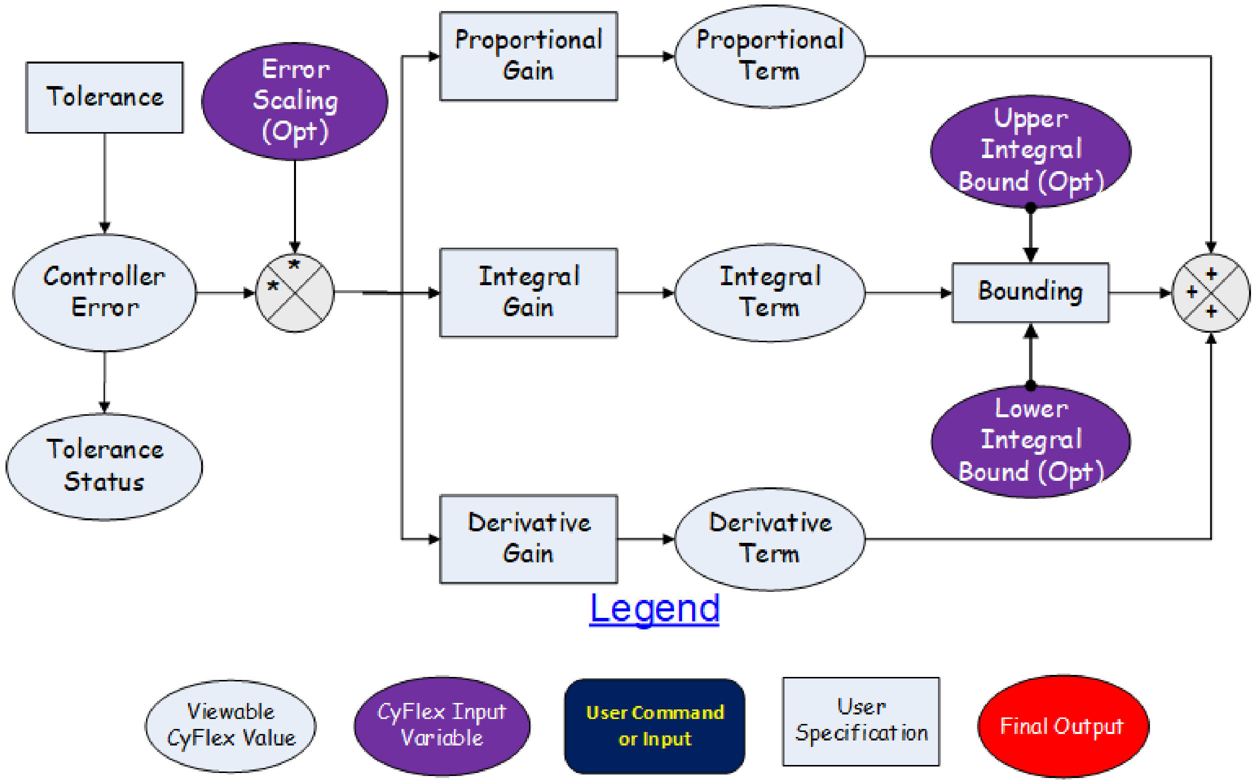
Table 5: Controller String Variables Created by a New User Control Task

Controller String Variables	Descriptions
`CtrlMode_X`	Where X is the controller number (zero based) `CLOSED_LOOP` or `OPEN_LOOP` – only for the old control task program
`Feedback_X`	Where X is the controller number (zero based) CyFlex label of feedback variable – only for the old control task program

4.4 Calculating the PID Controller Output Command

The PID controller output is calculated as described in *Section* 2 PID Controllers on page 3 with two important additions.

Figure 5: PID Output Command Calculation Flow



First, a CyFlex variable can be specified that represents an error scaling multiplier. It is generally a computed expression that updates at a rate that is suitable to the control scheme. Scaling the error has the same effect as changing all of the PI&D gains as a function of something else in the system.

For example, there may be a need for small gains when a particular speed value is low to account for additional transport delays while having normal gains when the speed is high. To accomplish this, a computed expression for the gain scale may be something like the following:

```
GainScale = MinimumValue + PresentSpeed / RatedSpeed
```

When the `PresentSpeed` is equal to `RatedSpeed`, the `GainScale` would equal 1.0 + the minimum value. As `PresentSpeed` reduces, the `GainScale` would reduce in a linear fashion.

A gain scaling table in the same file may also be defined as a command map, as shown in Appendix *C. Example Command Map on page 38*. However, both methods should not be used for the same controller.

4.5 Integral Bounding

The second addition to the Overview is that the integral term can, and will, have separate bounding. With the old control task, setting a bounding value is not needed but, instead, labels of CyFlex variables must be specified in which the integral upper and lower bounds are stored. With the new user control task, the bounds can be values, labels for CyFlex variables, or computed expressions. In this way, the bounds can vary in any manner desired by the user.

There are several benefits to including integral term bounding. First, if the system is very slow to respond in some areas of operation, the integrated error might reach an unreasonable value.

This can make recovery times longer than desired. Second, if the system is unable to reach the point of zero error in steady state, the integrated error would continue to climb and approach the output bounds, which may not be desirable. These two conditions are called “integrator wind-up”.

Third, where feed forward (discussed below) is used as the primary means of control, it may be desirable to add only limited integral authority to account for system modeling errors. Refer to *Section 5 Feed Forward Control* on page 14.

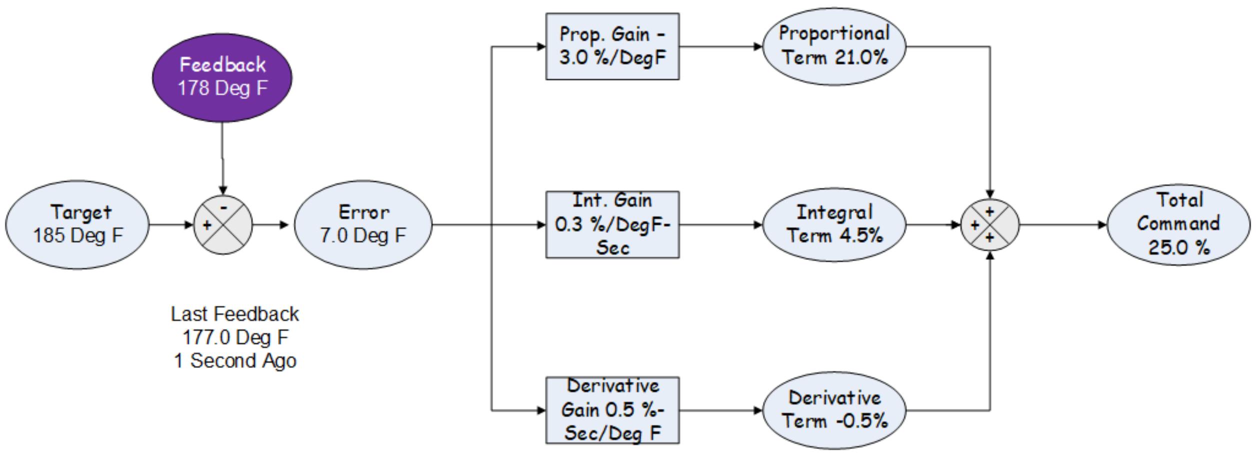
If more restrictive integral bounds are not specified, then the integral term is automatically limited to the output bound minus the feed forward term.

4.6 Simplified PID Controller Example

In the example in *Figure 6*, the controller target for a heater is set to 185 Deg F, the feedback value started at 177 Deg F and is now 178 Deg F. The PID gains are positive so that an increase in command to the heater raises the temperature.

Had the example control loop been for a cooling system, then less cooling would have been required to reach the set point and the gains would have to be negative.

Figure 6: PID Controller Example



In the preceding example:

Proportional Term = `Error * `K_p

Proportional Term = `7.0 Deg F * 3.0 %/ Deg F = 21.0 %`

Integral Term = `( Error * Process Interval + Previous Integral ) * Ki`

Integral Term = `( 7 Deg F * 1 sec + 8 Deg F – sec ) * 0.3 % / (Deg F – sec) = 4.5%`

Derivative Term = `( ( Present Error – Previous Error ) / Process Interval ) * Kd`

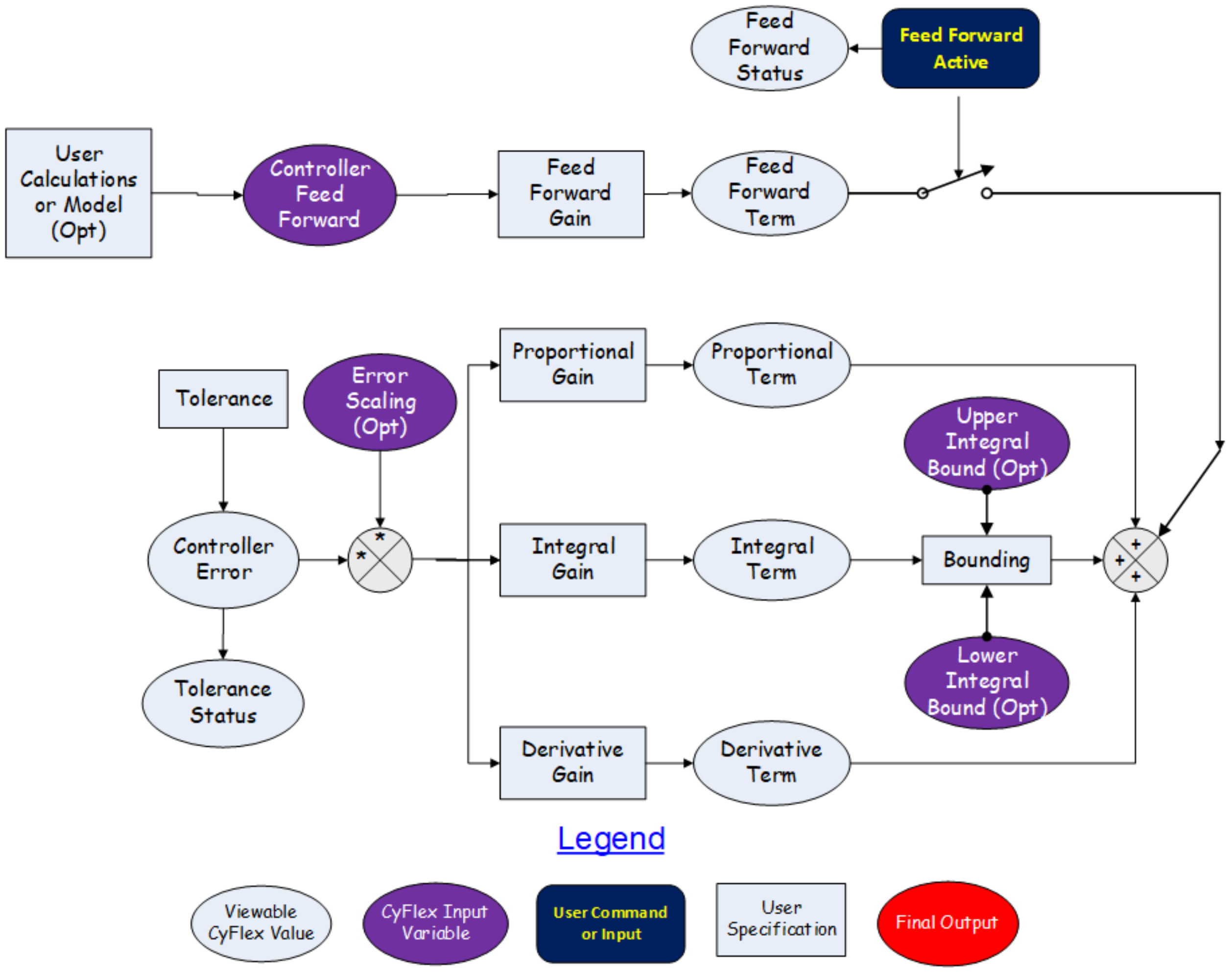
Derivative Term = `( 7 Deg F – 8 Deg F ) / 1 sec * 0.5 % - Sec / Deg F = -0.5 %`

Total PID = `( 21.0 + 4.5 - 0.5 ) % = 25%`

5 Feed Forward Control

There are often instances to accurately predict the behavior of a system relative to changes in the controlled system. In those cases, it may be better to use the output of the predicted behavior as the predominant control mechanism and add small amounts of PID or PI output to bring the system to the target value.

Figure 7: Feed Forward Prediction Calculation Flow



For example, it is easy to map an engine over the entire speed/load range to know what the effect of any given throttle position is relative to engine torque. A reverse map of speed and desired torque can be generated that would give a throttle position that would be a very good first approximation of actual engine torque. This throttle position could become a feed forward term in the above equation.

In the example above, the PID portion of the controller would only be used to generate a command that would account for errors in the model output.

In the CyFlex controller, a model using computed expressions can be specified, 2 or 3-D tables, or any combination of these. It is important that the result of the model is in the same units as the specified command output.

5.1 Transitions When Turning Feed Forward On/Off

Feed forward can be turned on or off at any time but be aware of what will happen at those transitions. Because there is a goal to make transitions “bumpless”, when feed forward is first turned off, the integral term is adjusted so that:

$$NewIntegralTerm = OldIntegralTerm + OldFeedForwardTerm$$

Note that the *NewIntegralTerm* is subject to the optional integral bounds when this adjustment is made.

In this way, there is no step in the output when the feed forward is turned off. Instead, the integral term is allowed to change until the system comes into balance.

Similarly, when the feed forward in turned on:

$$NewIntegralTerm = OldIntegralTerm - NewFeedForwardTerm$$

If “bumpless” transfer is not desired, then two options are available:

1. First, the feed forward control can be enabled at all times and the model output set to zero when not desired. In this way, the integral term will not be adjusted, and the user has the flexibility to decide how each transition is to happen.
2. Second, the `@BUMPLESS` keyword can be used in the specification file and the value set to `NO`. This method will *always* cause a “bumpy” transfer. If the `@BUMPLESS` keyword is not used, the transitions will be “bumpless”.

5.2 Feed Forward Example

The amount of engine cooling required is closely proportional to the power that the engine is producing. Therefore, engine power could be used as feed forward in the engine cooling loop. Further, it is known that the cooling valve needs to reach 80% open to cool a 400 hp engine when the cooling water is 80 degrees.

Feed Forward term = Feed Forward Variable * Feed Forward Gain

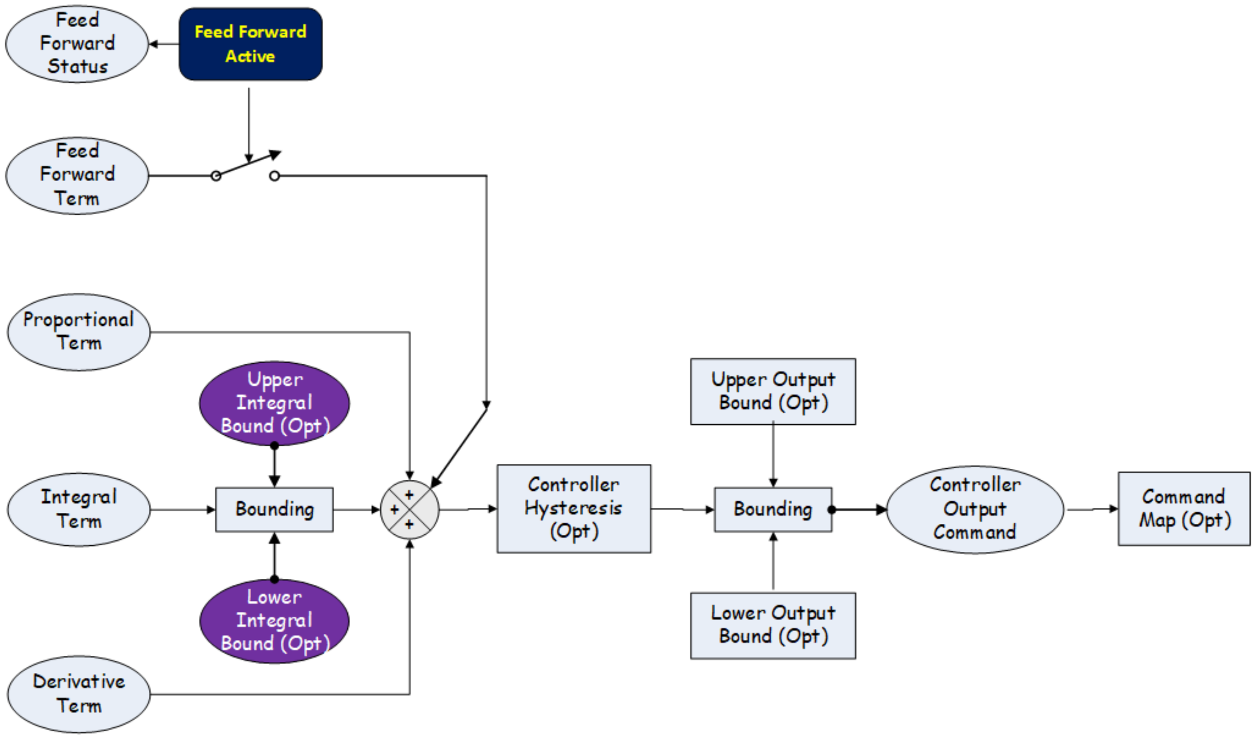
Feed Forward Term = 400 hp * 0.2% command / hp = 80%

In this case, the feed forward term will position the cooling valve close to the right place even before the coolant has a chance to flow to the heat exchanger. Proportional and integral terms can still be used to correct for non-linearity or inaccuracy of the feed forward prediction and when the performance of the heat exchanger changes over time.

6 Controller Output Adjustments

Once the PID + Feed Forward calculations are made, options for hysteresis, output bounding, and command mapping are allowed before the command is turned into a “real world” output.

Figure 8: Controller Output Adjustments Flow



6.1 Hysteresis

The first of these options for modifying the output is the addition of hysteresis compensation to the command value. In CyFlex, the hysteresis term can be thought of as the value that would be required to overcome the dead band in a worn linkage. For example, if a linkage is worn to the equivalent of 1% of its motion, when the direction of travel is reversed, the actuator output must change by 1% before the linkage would begin to move the actuated system in the reverse direction. By setting the hysteresis value to 1% in the specification file, the system will automatically add or subtract 1% to/from the output when the direction of travel changes. If the direction of the output continues to be the same, then no additional hysteresis correction is made.

In some actuators the hysteresis changes as a function of position. In these cases, choose a value that represents either the condition of least hysteresis, or the value at the nominal operating condition, if that is fairly constant.

6.2 Output Bounding

The output of the control algorithm is normally bounded so that 0% command represents the minimum position of the actuator and 100% represents maximum output. However, there is no reason to limit the bounds to these values.

For example, if the output of the controller is directed to an electrical current output device that is capable of 0-20 ma, but the device being driven wants a 4-20 ma signal, then it is reasonable to set the lower bound at 20%.

6.3 Output Command Map

The last way to modify the controller command is to specify a mapping function. This functionality is normally used where the system is non-linear or where linear gains do not provide sufficient speed, stability, or accuracy.

An example command map file is shown *in Appendix C. Example Command Map* on page 38 for a throttle linearization. The map file also contains error scaling values.

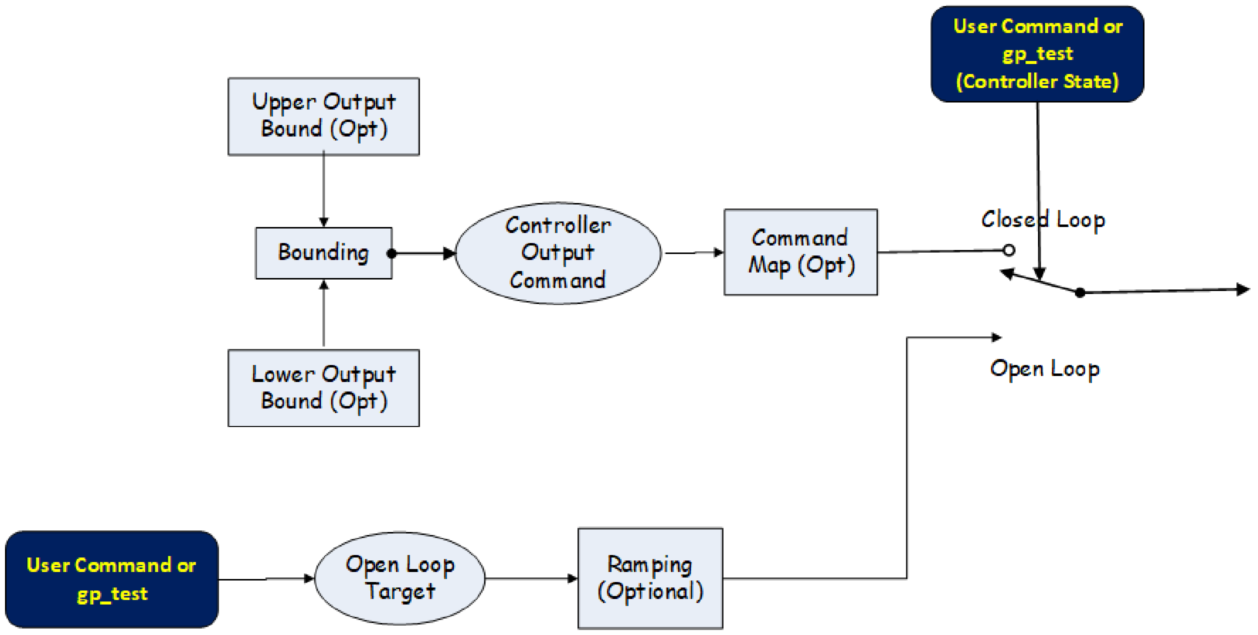
Another means to accomplish this same functionality is to send the output of the controller to a real variable rather than to an actual output device. In this case, the real variable can be manipulated in any chosen way before sending it to the output device.

7 Open or Closed Loop Operation

Choose whether to put the controller in open loop or closed loop. In closed loop, the controller behaves as described above with the options that have been included in the specification file.

Open loop enables more direct control over the output. With a test procedure or user command, the output can be set directly or ramped linearly from one position to another. *Figure 9* illustrates this concept.

Figure 9: Open/Closed Loop Output Flow



The following occurs when a controller mode is changed from open loop to closed loop or vice versa:

1. The `CtrlMode_X` string variable (X is the zero-based controller number) is set to "`CLOSED_LOOP`" or "`OPEN_LOOP`" for loops that are processed by the old control task only. The new user control task references all loops by name rather than number.
2. The controller `BaseName_MD` variable is set TRUE (1) for closed loop and FALSE (0) for open loop.

Use the `u_mode`, `u_target`, and `u_ctrl` commands to set the controller in open and closed loop. Refer to *Table 1* on page 1 for cyflex.com usage help references to user commands.

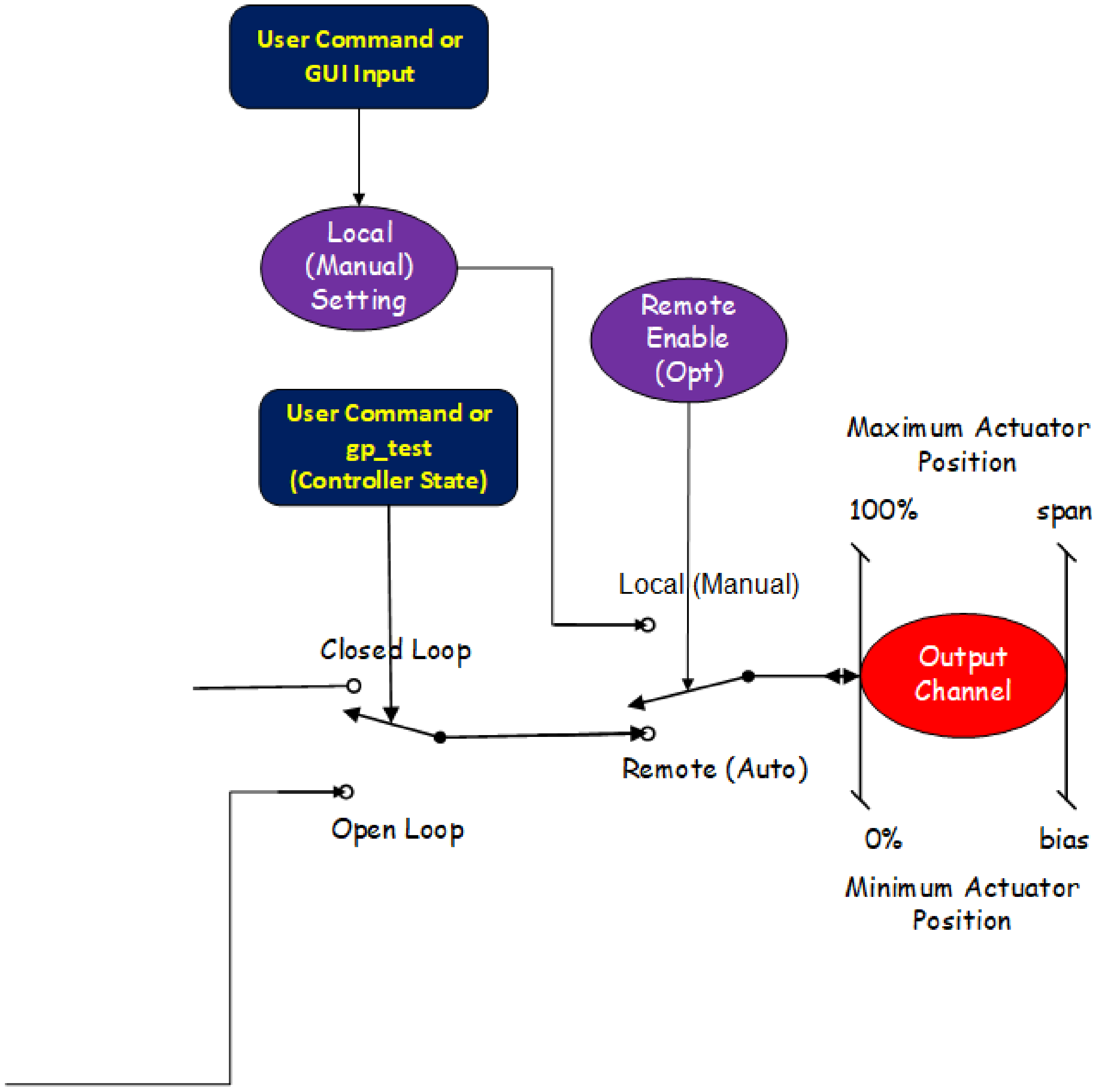
8 Manual Controls

Setting the open loop output of a controller with a potentiometer is called the manual or local setting. The computer tracks the voltage of this potentiometer so that, when the controller is switched into remote/auto, the output will be “bumpless” or continuous.

A CyFlex real variable can be specified that will contain the manual setting and a logical variable that enables/disables local control.

Some applications use this feature because the control loop can be user-guided through the start-up process with control being turned over to the computer when the user is satisfied with the condition of the system.

Figure 10: Manual Set of Open Loop Controller Output Flow


User Command or GUI Input
Local (Manual) Setting
Remote Enable (Opt)
User Command or gp_test (Controller State)
Maximum Actuator Position
100%
span
Local (Manual)
Closed Loop
Output Channel
Remote (Auto)
Open Loop
0%
bias
Minimum Actuator Position

Appendices

Appendix A. Test Procedure Keywords

@USER_LOOP

This is the test procedure keyword in the specification file used to change the target for a control loop already established in the control specification file.

Note:

User controls are any closed loop controls other than engine/dynamometer control loops. Typically, they are used for controlling the temperature or pressure of fuel, coolant, air, etc.

If the open loop start and end targets are different, and the optional ramp rate is not specified, the target is ramped linearly based on the mode timeout value.

The target fields and ramp rate may be expressed as either constants, variable labels, or computed expressions. If the target is a constant, the units must be appended as shown in the examples below.

The ramp rate has an implied "units per second" rate.

```
@USER_LOOP
#open/closed    variable       start_target   [end_target]    [ramp_rate]
CLOSED_LOOP     port_in_p      150[deg_F]
CLOSED_LOOP     fuel_temp      100[deg_F]     104[deg_F]      0.1 [dt_F]
OPEN_LOOP       air_inlet_t    100[%]
CLOSED_LOOP     the_temp       100[deg_F]     My_target_t
CLOSED_LOOP     the_pressure   0[in_H2O]      "my_target_p + 5.0[in_H2O]"
```

The `end_target` value is optional.

Appendix B. Control Task Specification File

```
#   Notes:
#
#   1 - A new way to specify user control loops is now available in CyFlex.
#       Each user loop may be specified in a unique file and the specifications
#       are entered via keywords.  If a particular specification is not needed,
#       then its associated keyword should not be entered. When a keyword
#       is not entered the controller will assume default values.  A minimum
#       of 4 keywords and the associated specification must be entered for each
#       controller.
#
#   2 - A separate instance of the user_ctrl_task will be launched for each
#       control loop.  If the specifications are reprocessed with a loop
#       removed, the instance of the user_ctrl_task will remain running in its
#       original configuration.  If that loop is no longer desired, the
#       proper instance of user_ctrl_task must be slayed or the command
#       remove_loop <loop_name> may be run.
#
#   3 - Each user loop may be specified in a unique file or they may be
#       grouped in any convenient way.
#
#   4 - The PNP functionality is no longer supported.
#
#   The following describes the available controller keywords, the
#   specifications that must follow each keyword, whether or not the
#   keyword is required, and any associated default values.  Following
#   the description is an example specification for a user control loop.
#
#----------------------------------------------------------------------------
#
#     Specify the label and computation rate for the controller
#        (REQUIRED ENTRY)
#
#  @USER_LOOP_CTRLER
#     <loop_name>   <command>     <interval>  <open_loop>    <critical>
#                    <output>                 <ramp_rate>
#                    <units>
#      loop_name       pct          MED        10             +c
#
#     where:
#        loop_name - This will become root of the variables created for the
#                    controller.  The labels created are:
#                     loop_name_CM    (ctrler command output)
#                     loop_name_LB    (ctrler lower output bound)
#                     loop_name_UB    (ctrler upper output bound)
#                     loop_name_OL    (ctrler open loop target)
#                     loop_name_TR    (target of the control loop)
#                     loop_name_LO    (lower target bound)
#                     loop_name_HI    (upper target bound)
#                     loop_name_RF    (reference of the control loop)
#                     loop_name_FV    (loop feedback value)
#                     loop_name_PT    (ctrler proportional term)
#                     loop_name_IT    (ctrler integral term)
#                     loop_name_IL    (lower bound on integral term)
#                     loop_name_IU    (upper bound on integral term)
#                     loop_name_DT    (ctrler derivative term)
#                     loop_name_FF    (ctrler feed forward term)
#                     loop_name_FA    (ctrler feed forward active)
#                     loop_name_ER    (error of the control loop)
#                     loop_name_TO    (control loop tolerance)
#                     loop_name_PG    (proportional gain)
#                     loop_name_IG    (integral gain)
#                     loop_name_DG    (derivative gain)
#                     loop_name_FG    (feed forward gain)
#                     loop_name_MD    (controller open/closed mode)
#                     loop_name_TL    (error is in tolerance LOGICAL VARIABLE)
#                     loop_name_IN    (name of ERROR IS IN TOLERANCE event)
#                     loop_name_OT    (name of ERROR IS OUT OF TOLERANCE event)
#                     loop_name_FB    (name of the feedback variable)
#                     loop_name_OC    (string to indicate open/closed mode)
#                     loop_name_TP    (target plus tolerance - for plots)
#                     loop_name_TM    (target minus tolerance - for plots)
#                     loop_name_RR    (closed loop ramp rate)
#                     loop_name_OR    (open loop ramp rate)
#
#        NOTE: The loop name does not have to be the same as the name of the
#              feedback variable
#
#        command   - the units assigned to created variables
#        output      NOTE: The units of the feedback variable are assigned
#        units             to _RF, _TR, & _ER variables.
#
#        interval  - The calculation interval for the control loop. Valid
#                    entries are SLO, MED, FAS, USR1, USR2 and WARP.
#
#        open loop - The open loop ramp rate.  This can be a value, CyFlex
#        ramp rate   label, or "computed expression" - 79-character max length.
#                    Units must be the same as the command output.
#
#        NOTE: If the value is zero or if the expression evaluates to
#              zero, ramping will be turned off and the reference value
#              will jump immediately to the loop target.
#
#        critical  - Should this loop run as critical for the watchdog
#
#                    Use +c, +C, y, or Y for critical,
#
#                    Use n, N, or - for non critical
#
#                    IMPORTANT - In order to change the critical nature of a
#                                loop the appropriate instance of the
#                                user_ctrl_task must be slayed and then
#                                the specifications must be reprocessed
#
#---------------------------------------------------------------------------
#
#     Specify the feedback variable for the controller
#        (REQUIRED ENTRY)
#
#  @CTRLER_FEEDBACK_VRBL
#     <label>     <closed_loop_ramp_rate>    <error_term_tolerance>
#
#     where:
#        label       - The label of the parameter being controlled by the
#                      loop.
#
#        closed loop - The closed loop ramp rate.  This can be a value, CyFlex
#        ramp rate     label, or "computed expression" - 79-character max length.
#                      Units must be the same as the feedback variable.
#
#        NOTE: If the value is zero or if the expression evaluates to
#              zero, ramping will be turned off and the reference value
#              will jump immediately to the loop target.
#
#        error term  - The maximum value of the error term for the control
#        tolerance     to be within tolerance. When the error is less than
#                      value the label 'loop_name_TL' will be TRUE, otherwise
#                      it will be FALSE.
#
#----------------------------------------------------------------------------
#
#     Specify the initial PID gains for the controller
#
#     The values will be displayed in _PG, _IG, and _DG and can
#     be changed by the user at any time.
#
#     It is up to the user to update this file with new values
#     as the values are not written to this file if they are changed
#
#        (REQUIRED ENTRY)
#
#  @CTRLER_GAINS
#     <proportional>    <integral>     <derivative>
```

```
#
#     where:
#        proportional - The proportional gain value
#
#        integral     - The integral gain value
#
#        derivative   - The derivative gain value
#
#---------------------------------------------------------------------------
#
#     Specify the output channel for the controller
#        (REQUIRED ENTRY)
#
#  @CTRLER_OUTPUT_CHAN
#
#     One of 4 output types may be specified, AO, CO, DO, RV, or LV
#
#
#        AO  <channel> <zero_value> <span_value> <filter_constant>
#
#           where:
#              channel        - is the analog output channel number that
#                                will receive the _CM value.
#
#              zero_value     - the command output value associated with
#                                0% of the range of the AO device.
#
#              span_value     - the command output value associated with
#                                100% of the range of the AO device.
#
#              filter_constant- a recursive filter factor between 0 and 0.99
#
#        CO  <channel> <zero_value> <span_value> <filter_constant>
#
#           where:
#              channel        - is the counter output channel number that
#                                will receive the _CM value.
#
#               zero_value     - the command output value associated with
#                                0% of the range of the CO device.
#
#               span_value     - the command output value associated with
#                                100% of the range of the CO device.
#
#               filter_constant- a recursive filter factor between 0 and 0.99
#
#         RV  <target_label>  <bias>  <gain>   <filter>
#
#            where:
#               target_label   - is the real variable where the
#                                _CM value will be placed
#
#               bias           - an offset applied to the value before
#                                units conversion
#
#               gain           - a multiplier applied to the value
#                                before units conversion
#
#               filter         - a recursive filter factor between 0 and 0.99
#
#             IMPORTANT: If the RV destination is chosen, the var_transfer task
#                        must also be running at a rate at least as fast as the
#                        loop timer
#
#         DO  <DO PWM label>  <base_period> [enable_label]
#
#            where:
#
#                DO_PWM_label - is the label of the DO bit that will
#                               provide the PWM output. This label
#                               must already exist and be contained in
#                               the do_specs file.
```

```
#
#               base_period  - A label whose value will be used as the
#                              base period of the PWM output. A numeric
#                              value may be specified. For this case, the
#                              value must also contain the time units,
#                              e.g. 30[sec].
#
#               enable_label - The label of a LOGICAL_VARAIBLE that
#                              is used to disable/enable the PWM
#                              output.
#
#        LV  <LV_PWM_label>  <base_period> [enable_label]
#
#           where:
#
#              LV_PWM_label - is the label of a LOGICAL variable that will
#                             provide the PWM output. This label
#                             must already exist.
#
#              base_period  - A label whose value will be used as the
#                             base period of the PWM output. A numeric
#                             value may be specified. For this case, the
#                             value must also contain the time units,
#                             e.g. 30[sec].
#
#              enable_label - The label of a LOGICAL_VARAIBLE that
#                             is used to disable/enable the PWM
#                             output.
#
#  NOTE ON FILTERS:
#
#     A recursive filter factor of 0 means that no filtering of the output
#     is applied.  A factor of 0.5 means that 50% of the previous value
#     will be added to 50% of the most recent value.  This causes the
#     output to be less noisy and change more slowly.
#
```

```
#     As the filter factor approaches 1.0 the percentage of the most recent
#     value approaches 0% and the responsiveness of the controller becomes
#     very slow.  A value of 1.0 should never be used.
#
#
#---------------------------------------------------------------------------
#
#     Specify some attributes of the created controller variables.
#         (OPTIONAL ENTRIES)
#
#  @USER_LOOP_VRBL_OPTS_TR
#     <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 YES or NO. Default is NO.
#
#        NOTE: The history flag is now obsolete since ALL variables are
#              now saved in history files.  Entries should still be made
#              until the user_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
#
#     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:
#
#  @USER_LOOP_VRBL_OPTS_RF
#  @USER_LOOP_VRBL_OPTS_ER
#  @USER_LOOP_VRBL_OPTS_CM
#  @USER_LOOP_VRBL_OPTS_PT
#  @USER_LOOP_VRBL_OPTS_IT
#  @USER_LOOP_VRBL_OPTS_DT
#  @USER_LOOP_VRBL_OPTS_FF
#  @USER_LOOP_VRBL_OPTS_OL
#
#----------------------------------------------------------------------------
#
#     Specify an attribute of the created controller LOGICAL variables.
#         ( !!!!!!!! THIS OPTION IS NOW OBSOLETE - ALL VARIABLES ARE NOW
#                    SAVED AUTOMATICALLY !!!!!!!!! )
#
#  @USER_LOOP_VRBL_OPTS_TL
#     <history_active>
#
#     where:
#        history  - A flag indicating if the LOGICAL_VARIABLE should be included
#        active     in the history log. Valid entries for this field
#                   are YES or NO. Default is NO.
#
#     NOTE:  There is a similar specification for each of the created LOGICAL
#            variables. The only thing that is different is the suffix of the
#            keyword. The additional keyword for created LOGICAL variable
#            options are:
#
#  @USER_LOOP_VRBL_OPTS_FA
#
#----------------------------------------------------------------------------
#
#     Specify the command output bounds for the controller
#        (OPTIONAL ENTRY)
#
#  @CTRLER_OUTPUT_BOUNDS
#     <lower_bound>     <upper_bound>
#
#     where:
```

```
#        lower bound - This entry may be a value, label of a variable, or a
#                      computed expression with a maximum length of 79
#                      characters. The default lower bound is 0.
#
#        upper bound - This entry may be a value, label of a variable, or a
#                      computed expression with a maximum length of 79
#                      characters. The default upper bound is 100.
#
#
#----------------------------------------------------------------------------
#
#     Specify the integral term bounds for the controller
#        (OPTIONAL ENTRY)
#
#  @CTRLER_INTEGRAL_BOUND_LABELS
#     <lower_bound>      <upper_bound>
#
#     where:
#        lower bound - This entry may be a value, label of a variable, or a
#                      computed expression with a maximum length of 79
#                      characters. The default lower bound is -100.
#
#        upper bound - This entry may be a value, label of a variable, or a
#                      computed expression with a maximum length of 79
#                      characters. The default upper bound is 100.
#
#----------------------------------------------------------------------------
#
#     Specify the hysteresis for the controller
#        (OPTIONAL ENTRY)
#
#  @CTRLER_HYSTERSIS
#     <value>
#
#     where:
#        value - is the hysteresis value for the controlled device. If not
#                specified a value of 0.0 will be assumed.
#
#----------------------------------------------------------------------------
#
#     Specify the command map pathname for the controller
#        (OPTIONAL ENTRY)
#
#  @CTRLER_COMMAND_MAP_FILE
#     <pathname>
#
#     where:
#        pathname - is the full path name of the file to use as a command
#                   map for the output command.  If this key word is not
#                   entered, then no command map will be applied to the
#                   output command.
#
#----------------------------------------------------------------------------
#
#     Specify the error scaling variable for the controller
#        (OPTIONAL ENTRY)
#
#  @CTRLER_ERROR_SCALING_LABEL
#     <scale_error_label>
#
#     where:
#        scale_error - The label of the variable whose value will be used to
#          label       scale the computed error term.  The computed error is
#                      multiplied by this value before the output term is
#                      computed.  If this keyword is not entered no scaling
#                      will be performed.  The units on this variable must
#                      be [none]
#
#----------------------------------------------------------------------------
#
#     Specify the feed forward variable for the controller
#        (OPTIONAL ENTRY)
```

```
#
#  @CTRLER_FEED_FORWARD
#     <label>     <active>    <gain>
#
#     where:
#        label    - A label of a CyFlex variable is entered. The user must
#                   perform any computations on the value outside the user
#                   control task.
#
#        active   - A flag indicating if the feed forward term is initially
#                   active or inactive. Valid entries for this field are
#                   YES, Y, T, TRUE.  Any other value will be taken as
#                   NO or FALSE.  The value can be changed on the fly
#                   by setting the XXXXXX_FA variable TRUE or FALSE.
#
#        gain     - The feed forward gain that will be used when feed forward
#                   is active.  This can be a number, label, or expression
#                   that is up to 79 characters in length
#
#            NOTE: If this keyword is not entered, then no feed forward
#                  term will be computed.
#-------------------------------------------------------------------------------------

#     Specify whether bumpless transfer is desired when feed forward is turned ON or OFF.
#     If this keyword is not used, the loop will have bumpless transfer.
#
#        (OPTIONAL ENTRY)

#  @BUMPLESS_TRANSFER
#     <YES or NO>
#
#     where:
#        YES      - Changing from feed forward ON/OFF will result in
#                   a smooth transition in the output by adjusting the
#                   integral term - assuming the change does not cause the
#                   integral term to reach a user specified limit
```

```
#
#        NO       - Changing from feed forward ON/OFF will result in
#                   an abrupt change in the output and will not affect
#                   the integral term unless a user specified output
#                   limit is reached.
```

```
#----------------------------------------------------------------------------
#
#     Specify the local command label for the controller
#        (OPTIONAL ENTRY)
#
#  @CTRLER_LOCAL_COMMAND_LABEL
#     <local_label>
#
#     where:
#        local_label - is the name of the CyFlex variable that contains
#                      the local command output when the control loop
#                      is not under computer control. If this keyword
#                      is not entered then no local command will be applied.
#
#----------------------------------------------------------------------------
#
#     Specify the remote sense LOGICAL VARIABLE for the controller
#        (OPTIONAL ENTRY)
#
#  @CTRLER_REMOTE_SENSE_LABEL
#     <remote_label>
#
#     where:
#        remote_label - is the label of the CyFlex LOGICAL_VARIABLE that
#                       will be used to determine if remote (computer)
#                       control is to be used or local control. If this
#                       keyword is not entered, then only remote (computer)
#                       control will be used.
#
```

```
#-----------------------------------------------------------------------------
#
#     Specify the upper and lower bounds of the target for the feed back
#     variable   (OPTIONAL ENTRY)
#
#  @USER_LOOP_TARGET_BOUNDS
#     <lower bound>   <upper bound>
#
#     where:
#        <lower bound> - 1. The value of the lower bound that should be
#                           placed on the target of the controller,
#                           e.g. 34
#
#                        2. The value of the lower bound, with units specified,
#                           that should be placed on the target of the
#                           controller. Note that the units type must be the
#                           same as the output, e.g. 34[PSI]
#
#                        3. A CyFlex variable name. The specified variable
#                           must already exist at the time that user_ctrl_specs
#                           is run.  It will not be created. The value of
#                           this variable will then be used as the lower
#                           bound of the target,
#                           e.g. Oil_rfl_tar_low_bnd
#
#                        4. A double quoted expression. The expression will be
#                           evaluated and the result will be used as the lower
#                           bound of the target.  The length of the expression
#                           is limited to 79 characters,
#                           e.g. "Oil_Rf_tar_low_bnd + barometer"
#
#         <upper_bound>  - Same as lower bound.
#
#     The bounds are independent and any of the above options can be specified
#     for each bound.
#
```

```
#---------------------------------------------------------------------------
#
#     Specify the initial mode and set point for the controller
#     (OPTIONAL ENTRY)
#
#  @INITIAL_CTRLER_STATE
#     <controller mode>           <setpoint>      <always/only on go>
#
#      where:
#
#         <controller mode> - OPEN_LOOP   - The controller will be placed
#                                           in the open loop mode when
#                                           user_ctrl_specs is run.
#
#                             CLOSED_LOOP - The controller will be placed
#                                           in the closed loop mode when
#                                           user_ctrl_specs is run.
#
#         <setpoint>        - the target value of the specified <mode>.
#                             This entry may be a constant, label, or
#                             computed expression.
#
#         <always/only on go> - A value of GO will cause the initial state
#                               to ONLY be set when the the user_ctrl_task
#                               is launched.  Any other value will cause
#                               the loop to be initialized every time
#
#
```

```
#---------------------------------------------------------------------------
#
#     Indicate the end of the specifications for a user loop controller
#
#  $END Note: Additional user loops may be specified following the $END.
#
#-------------------------Example User Loop Specification ------------------

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

@CTRLER_FEEDBACK_VRBL
   # label     closed loop    error term
   #            ramp rate      tolerance
   AirVolFlow     50.0           10.0

@CTRLER_GAINS
   # proportional      integral      derivative
     0.01              0.001         0.0

@CTRLER_OUTPUT_CHAN
   #output type      channel   bias       span       filter constant
      AO               1       0.0        100.           0.0

@INITIAL_CTRLER_STATE
   # initial mode    initial state     ALWAYS/only on GO
     OPEN_LOOP       50[%]             GO

$END
```

Appendix C. Example Command Map

```
####################### Command Map ##################################

# Z Variable Definition

RPM

# Number of z_values

2

# Number of commands

9

#  Z Values

   9.000000E02   1.800000E03

#   Normal Cmds         Delinearized Cmds

   0.000000E00   1.000000E-02    1.000000E-02  1.000000E-02
   1.226994E00   1.200000E01     8.895646E-01  6.102686E00
   9.221557E00   1.764406E01     6.620429E00   4.580000E01
   6.131737E01   5.442304E01     4.396506E01   6.560000E01
   7.433962E01   6.361658E01     5.330000E01   7.148061E01
   9.075472E01   7.520543E01     5.730000E01   7.889337E01
   9.652695E01   7.928055E01     8.395945E01   8.150000E01
   9.754601E01   8.000000E01     8.866608E01   8.692828E01
   1.000000E02   1.000000E02     1.000000E02   1.000000E02

####################### End of Command Map ##########################


####################### Gain Scale Table ############################

# Number of coordinates

3

#         Z Value         Scaling Factor

          7.000000E02     4.100629E-01
          1.300000E03     1.000000E00
          2.100000E03     6.301887E-01

####################### End of Gain Scale Table #####################
```