

CyFlex® Engine Controls

Version 9

October 28, 2024

Developed by Transportation Laboratories



Version History

Version	Date	Revision Description		
1	7/24/2017	Initial publication		
2	8/23/2018	Formatting revisions		
3	1/29/2019	Revise to reflect changes to the control programs		
4	3/30/2020	Retrofit to new template		
5 8/17/2021 Added Section 1.2 Summary of Engine page 1 with content of Table 1 that list usage help hyperlinked cross-reference Removed Appendix A which contained eng_ctrl_task and changed refer to Table 1 Renamed Appendix B to Appendix A a document cross-references according		Added Section 1.2 Summary of Engine Controls Tasks on page 1 with content of Table 1 that lists relevant cyflex.com usage help hyperlinked cross-references Removed Appendix A which contained usage help for eng_ctrl_task and changed references from Appendix A to Table 1 Renamed Appendix B to Appendix A and revised intra- document cross-references accordingly.		
		Renamed Appendix C to Appendix B and revised intra- document cross-references accordingly. Removed Appendix D which contained usages for set_eng_mode, sp, th, dy, other, dy_gains, dy_ramp, and reset_dyni_int. Included cyflex.com hyperlinked usage help cross-references to these commands in <i>Table 1</i>		
6	6 1/18/2022 Updated <i>Figure 1</i> on page 4 for varying Kp values Corrected the entry for the MD variable in <i>Table 4</i> on p Made miscellaneous corrections to Appendix <i>B. Exam</i> <i>Command Map</i> on page 38			
7	5/18/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions		
8	2/5/2024	Rebrand to TRP Laboratories		
9	10/28/2024	Remove mentions of deprecated ctrl_task command from Appendix A. Control Task Specification File on page 21		

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 <u>https://cyflex.com/</u>. View **Help & Docs** topics or use the **Search** facility to find topics of interest.



Table of Contents

1	OV	ERVIEW
	1.1	CONTROL SYSTEMS
	1.2	SUMMARY OF ENGINE CONTROLS TASKS
2	PID	CONTROLLERS
	2.1	PROPORTIONAL TERM
	2.2	INTEGRAL TERM
	2.3	DERIVATIVE TERM
3	GA	IN TUNING7
4	CYI	FLEX IMPLEMENTATION OF ENGINE PID CONTROLS8
	4.1	DEFINITIONS
	4.2	SELECTING THE DESIRED VALUE FOR THE CONTROL LOOP
	4.3	FINDING THE CONTROL LOOP ERROR
	4.4	DISPLAYING CONTROL LOOP VALUES
	4.5	CALCULATING THE PID CONTROLLER OUTPUT COMMAND
	4.5.	1 Integral Bounding
	4.5.	2 Simplified PID Controller Example
5	FEE	ED FORWARD CONTROL
	5.1	FEED FORWARD EXAMPLE
6	со	NTROLLER OUTPUT ADJUSTMENTS15
	6.1	Hysteresis
	6.2	OUTPUT BOUNDING
	6.3	OUTPUT COMMAND MAP
7	OP	EN/CLOSED LOOP OPERATION
8	MA	NUAL CONTROLS
9	AU	TO-TUNING A CONTROL LOOP
A	PPEN	DICES
	APPEN	NDIX A. CONTROL TASK SPECIFICATION FILE
	APPEN	NDIX B. EXAMPLE COMMAND MAP



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: PID CONTROLLER INFORMATION FLOW	. 9
FIGURE 5: PID CONTROLLER OUTPUT CALCULATION FLOW	11
FIGURE 6: SIMPLIFIED PID CONTROLLER EXAMPLE	12
FIGURE 7: FEED FORWARD FLOW	13
FIGURE 8: CONTROLLER OUTPUT ADJUSTMENTS FLOW	15
FIGURE 9: OPEN/CLOSED LOOP OUTPUT FLOW	17
FIGURE 10: MANUAL SET OF OPEN LOOP CONTROLLER OUTPUT FLOW	19

LIST OF TABLES

TABLE 1: CYFLEX ENGINE CONTROL PROGRAMS	1
TABLE 2: EFFECTS OF INDEPENDENTLY INCREASING A PARAMETER	7
TABLE 3: CONTROL LOOP VARIABLES	0
TABLE 4: CONTROLLER-BASED DISPLAY VARIABLES	7



1 Overview

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and has been universally accepted in industrial control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity which allows engineers and technicians to operate them in a simple, straightforward manner.

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 then summing those three components to compute the output.

1.1 Control Systems

In a typical control system, the process variable is the system parameter that needs to be controlled, such as speed (RPM) or torque (lb_ft). A sensor is used to measure the process variable and provide feedback to the control system. The set point is the desired target value for the process variable, such as 1800 RPM in the case of a speed control system.

At any given moment, the difference between the process variable and the target is used by the control system algorithm to determine the desired actuator output to drive the system. For instance, if the measured speed variable is 1700 RPM and the desired target speed is 1800 RPM, then the actuator output specified by the control algorithm might be decrease excitation on a dynamometer. Reducing excitation on the dynamometer causes the system to decrease load and results in an increase to the system speed. 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.

1.2 Summary of Engine Controls Tasks

Table 1 lists the engine controls programs.

Table 1 · C	vElex Engli	ne Control	Programs
Table 1. O	уг юл шиун		Trograms

Task	Purpose	Cyflex.com Usage Help Reference	
eng_ctrl_task	Provide PID control algorithms for engine throttle and dyno(s). Normally spawned by the by the eng_ctrl_specs task.	Refer to eng ctrl task and eng ctrl specs	
set_eng_mode	Set engine control mode	Refer to set eng mode	
sp	Set speed target	Refer to <u>sp</u>	
to	Set torque target	Refer to <u>to</u>	
th	Set throttle in open loop	Refer to <u>th</u>	
dy	Set dyno controller in open loop	Refer to <u>dy</u>	
other	Set target for 'other' control loop	Refer to other	



Task	Purpose	Cyflex.com Usage Help Reference	
dy_gains Set dyno gains		Refer to dy gains	
dy_ramp	Set dyno ram	Refer to dy ramp	
reset_dyno_int	Set dyno controller integral term to 0.0	Refer to reset dyno int	



2 PID Controllers

The PID 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:

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

where:

 $K_{p:}$ Proportional gain, a tuning parameter

 K_i : Integral gain, a tuning parameter

- K_d : Derivative gain, a tuning parameter
- *€*: 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 K_p , called the proportional gain constant.

The proportional term is given by:

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

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:

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

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 over shoot 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 K_d . The magnitude of the contribution of the derivative term to the overall control action is manipulated by the derivative gain, K_d .

The derivative term is given by:

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

Derivative action predicts system behavior and thus improves settling time and stability of the system. 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.

5









3 Gain Tuning

One tuning method that can be done with the controller on-line is to first set *Ki* and *Kd* values to zero. Increase the *Kp* until the output of the loop oscillates, then the *Kp* should be set to approximately half of that value. Next 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.

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:

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

where the error is the target 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 tuning 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.

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 <i>Kp</i> small

Table 2: Effects of Independently Increasing a Parameter

Refer to the following for related information:

- Section 4.5 Calculating the PID Controller Output Command on page 11
- Section 5 Feed Forward Control on page 13
- Section 6 Controller Output Adjustments on page 15



4 CyFlex Implementation of Engine PID Controls

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

Refer to *Figure 4* on page 9 for the information flow. The legend in the first section applies to all sections described.

4.1 Definitions

There are two distinct aspects to an engine PID control system: the control loop and the control device.

The control loop, such as engine speed, has elements of set point, feedback, error, and tolerance. The units of all of these elements are those of the feedback value. The name of the display variables for a control loop always begin with the user-specified loop name. For example, a control loop named Speed will create a variable called Speed_TR which will contain the set point/target value. The working units will be those of the variable – Speed.

An engine control device, or simply controller, will be one of two different systems: a throttle or a dynamometer. There will be a number of display variables created that are associated with a controller such as the output command, P,I & D terms, feed forward term, etc. Each of these variables will be created with the units of the controller output.

The gains for a control system come into play between the control loop and the control device. Ideally, the gains should have units of [output]/[feedback] but these units seldom exist in CyFlex. Therefore, all gain variables are assumed to have these units but for display purposes, the units are shown as [none].

Since there are multiple gains that may be used for an engine control system depending on the control mode, the ones that are actually in play are displayed as the loop_name_XG where X can be P, I, D, and F (feedforward). For example, Speed PG.

8



4.2 Selecting the Desired Value for the Control Loop



Beginning at the left most side of this diagram, set a target for the control loop. This can be done with the <sp>, <to>, or <other> commands or from within a general-purpose test procedure. Refer to *Table 1* on page 1 for cyflex.com usage help for these commands.

The closed loop target is *not* automatically set if either of two CyFlex features have been requested. First, for speed control, the user has the option of selecting maximum and minimum speed settings that keep the engine and dynamometer safe. These bounds are set in the engine parameter specification file along with many other engine and test cell values. The bounds may also represent the physical limits of the controlled process.

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

Use any of the above commands to automatically put the appropriate controller into closed loop mode if it was previously in open loop.

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

4.3 Finding the Control Loop Error

As mentioned in *Section 1* Overview on page 1, the control loop error is the difference between the control loop set point 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 = Control Loop Reference – Feedback Value

9



4.4 Displaying Control Loop Values

Specify a name for each control loop. Often times the name of the feedback variable is used, but this need not be the case. However, the name must coincide to a valid CyFlex variable that already exists when the engine control task is started because the program uses the units of this variable in its calculations.

From the Visio legend above, note that the control loop target, reference, and error values are all stored in viewable CyFlex variables. The variable names are the control loop name with a three-character suffix attached. *Table 3* shows all of the control loop variables that are created.

Control Loop Real Variables	Explanation			
LoopName_TR	Control loop target (set point when ramping is completed)			
LoopName_RF	Control loop reference (set point while ramping, with bounding)			
LoopName_ER	Control loop error			
LoopName_TO	Control loop tolerance			
LoopName_TP	Control loop reference plus the tolerance			
LoopName_TM	Control loop reference minus the tolerance			
LoopName_PG	Present control loop proportional gain which is a function of control mode.			
LoopName_IG	Present control loop integral gain which is a function of control mode.			
LoopName_DG	Present control loop derivative gain which is a function of control mode.			
LoopName_FV	Control loop feedback value which may be a function of control mode.			
LoopName_TL	Flag to indicate if the control loop is in-tolerance.			
CyFlex Event	Explanation			
LoopName_IN	CyFlex event that is set when a loop becomes in-tolerance			
LoopName_OT	CyFlex event that is set when a loop goes out of tolerance			

Table 3: Control Loop Variables



4.5 Calculating the PID Controller Output Command



The PID controller output is calculated as mentioned in *Section 1* Overview on page 1 with two important additions described below and in *Section 4.5.1 Integral Bounding* below .

First, specify a CyFlex variable that represents an error scaling multiplier. If specified, create a computed expression for the multiplier and update it 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, you may need small gains when the 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 similar to 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 may also be defined in the same file as a command map, as shown in *Appendix B. Example Command Map* on page 38. However, both methods should not be used for the same controller.

4.5.1 Integral Bounding

The second addition to the *Overview* is that the integral term can and will have separate bounding. If so desired, the integral bounds may be determined by a CyFlex variable or computed expression. In this way, the bounds can vary in any manner desired.

There are several benefits to including integral term bounding.

1. 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.



2. 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.

The first two conditions are called "integrator wind-up".

3. 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.

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

4.5.2 Simplified PID Controller Example

The example in *Figure 6* illustrates that the controller reference for speed has been set to 1850 rpm, the feedback value started at 1770 RPM and is now 1780 RPM. The PID gains are negative so that a decrease in command to the dynamometer raises the speed.



As an example:

```
Proportional Term = Error * Kp

Proportional Term = 70 rpm * (-0.6) % / rpm = -42.0 %

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

Previous Integral

Integral Term = ( 70 rpm * 1 sec) * (-0.3) % / (rpm - sec)

) - 24% = -45%

Derivative Term = ( ( Present Error - Previous Error ) /

Process Interval ) * Kd

Derivative Term = ( 70 rpm - 80 rpm ) / 1 sec * (-0.5) % -

Sec / rpm = 5.0 %

Total PID = ( -42 - 45 + 5 ) % = -82%
```



5 Feed Forward Control

There are often instances for accurately predicting 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.



For example, it is quite 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 preceding example, 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, specify a model using computed expressions, 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.

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

The above discussion of "bumpless" transfer applies when a controller is switched between open and closed loop.

If "bumpless" transfer is not desired, then two options are available.

- 1. First, feed forward 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. This gives the most flexibility.
- 2. Second, there is an option in the specification table that can set a controller for "bumpless" control or not. Using this option sets the behavior at all times.

5.1 Feed Forward Example

The amount of engine torque is closely tied to both speed and throttle position. Therefore, the engine can be mapped over the entire operating range to find the torque delivered for any combination. Depending on the time allowed for stabilization at any speed/load, the torque may be very accurate or just close.

In this case, the feed forward term will command the throttle close to the right place even before the torque can be measured. Proportional and integral terms can still be used to correct for nonlinearity or inaccuracy of our feed forward prediction and when the performance of the engine changes as the engine warms up.



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 as indicated in *Figure 8*.



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 a 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 B. Example Command Map* on page 38 for a throttle linearization. The map file also contains error scaling values if desired.

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 on to the output device.



7 Open/Closed Loop Operation

Choose whether to put the controller in open loop or closed loop. In closed loop the controller behaves as described in *Section 1.1 Control Systems* on page 1 with the options that have been included in the specification file.

Open loop has 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.



When a controller mode is changed from open loop to closed loop or vice versa, the controller BaseName MD variable is set TRUE (1) for closed loop and FALSE (0) for open loop.

The <sp>, <to>, <dy>, and <other> commands are used to set the controller targets in open and closed loop. Refer to *Table 1* on page 1 for cyflex.com usage help for these commands.

Table 4 lists all of the controller-based display variables. CyFlex supports three separate controllers: up to two dynamometers and one throttle.

Controller Real Variables	Explanation		
BaseName_PT	Proportional correction term (scaled error multiplied by the proportional gain)		
BaseName_IT	Integral correction term (scaled error integrated over time and multiplied by the integral gain, with bounding)		
BaseName_DT	Derivative correction term (change in error times the derivative gain)		

Table 4: Controller-based Display Variables



Controller Real Variables	Explanation		
BaseName_FF	Feed forward term (feed forward value times the feed forward gain)		
BaseName_CM	Final controller command (sum of PT, IT, DT, FF and bounding)		
BaseName_OL	Controller open loop command. The actual open loop output may be bounded. If set, this value will also exist even when the controller is in closed loop.		
BaseName_LB	Controller lower bound.		
BaseName_UB	Controller upper bound.		
BaseName_IL	Controller integral lower bound.		
BaseName_IU	Controller integral upper bound.		
BaseName_FG	Controller feed forward gain		
Controller Logical Variables	Explanation		
BaseName_FA	TRUE or 1 when feed forward has been set active		
BaseName_MD	TRUE or 1 when the controller is in closed loop. FALSE or 0 when in open loop.		



8 Manual Controls

There is a feature to set the open loop output of a controller with a potentiometer or other CyFlex real variable. This is called the manual or local setting. The computer tracks the voltage of this potentiometer as read in the input so that when the controller is switched into remote/auto, the output will be "bumpless" or continuous.

Specify a CyFlex real variable that will contain the manual setting and a logical variable that enables/disables local control.

Some applications use this feature because human intervention to guide the control loop through the startup process is helpful, with control being turned over to the computer when the user is satisfied with the condition of the system.







9 Auto-Tuning a Control Loop

Gain tuning of a process can be difficult to master, especially if done by someone that does so infrequently. A poorly tuned system can lead to either slow response or significant errors, both of which can be costly. Therefore, CyFlex provides a means of automatically tuning a control loop which is a modified form of the classic Ziegler-Nichols tuning method.

The Ziegler–Nichols tuning method, developed by John G. Ziegler and Nathaniel B. Nichols. It is performed by setting the *I* (integral) and *D* (derivative) gains to zero. The "P" (proportional) gain is then increased from zero until it reaches the **ultimate gain**, at which point the output of the control has stable and consistent oscillations. Then the ultimate gain and oscillation period are used to set the P, I, and D gains.

The CyFlex method assumes that the user has set the initial gains and a target value and is asking the computer to help improve the settings. The algorithm takes the following steps:

- 150 executions of the controller are used to establish the standard deviation of the present system error. The actual time to do this is dependent on the specified loop update rate.
- Open loop commands that are both above and below the present value are calculated as percentages of the existing value. Note that the best place to perform this tuning is with the output near the mid-point of its range.
- The open loop output is alternately set to the upper and lower values.
- Each open loop value is maintained until the feedback value exceeds the present error by 25%.
- The measured error and oscillation period are used in a similar manner as Ziegler/Nichols in order to arrive at new gains.
- The new gains are then displayed to the user.

A certain amount of noise in the feedback value is to be expected. Therefore, this process should be attempted several times and the gain values averaged before updating the values in the specification file.

Initiate auto-tuning with either of the following methods:

- Type autotune *loop* name on the command line.
- Use the control display GUI and select the auto-tuning button on the command tab.

If the system does not seem to establish regular, predictable oscillations, the auto-tuning process may give up and produce an alert. Also, the auto-tuning process may be aborted by typing abort_autotuning *loop_name* or by selecting the abort auto-tuning button on the control display GUI.



Appendices

Appendix A. Control Task Specification File

```
#
#
     Default Engine Control Mode
#
         (REQUIRED ENTRY)
#
#
#
    Modes are defined as:
#
#
        1
             Dyno control dyno torque; throttle control speed
             Dyno control net torque; throttle control speed
#
        2
            Dyno control other channel; throttle control speed
#
        3
#
        4
            Dyno control speed; throttle control gross torque
           Dyno control speed; throttle control net torque
Dyno control speed; throttle control other channel
#
        5
#
        6
#
@ENG CONTROL MODE
  5
      # Dyno control speed; throttle control net torque
 _____
#
#
#
#
     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).
#
```



@BUMPLESS TRANSFER # First Dyno Second Dyno Throttle # (if dual dyno) YES YES NO # # # There are 3 different CONTROL LOOPS that may be specified. These # are: # # 1 - Speed # 2 - Torque 3 - Other # # # # The engine control task can control two of the three values listed # depending on the control mode chosen above. At least two of them MUST BE specified. The "other" feedback variable must be a CyFlex # REAL variable that already exists. # # # Speed and torque loop names are used for their labels ONLY and for # units checking. The eng ctrl task will NOT use the value of speed # and torque entered. Instead, it will use the proper values based on the engine control mode - ctl_spd, net torque, gross torque, # # dyno torque, etc. Refer to perf labels for definitions. # # # Speed and Torque are existing variables that contain the # units desired for display variables. # # "Other" variable such as average exhaust temperature, # boost pressure, etc. (Optional if both Speed and Torque # are specified but required if not) # # The eng ctrl specs task will truncate these labels to 28 # characters and append the following extensions, RF, TR, # ER, etc. for the control variable reference, target, etc. # terms. # # All control display values are listed below. Loopname # for these variables is taken from the @SPEED LOOP NAME, # @TORQUE LOOP NAME, and @OTHER LOOP NAME labels. # # (Reference of feedback variable) loopname RF loopname_TR (Target of feedback variable)
loopname_ER (Error of feedback variable)
loopname_TO (Tolerance variable/value) # # # loopname_PG (Proportional gain) loopname_IG (Integral gain) # #



feedback DG (Derivative gain) # loopname FV (Feedback values used for the # control display) # loopname TL (Error is in tolerance LOGICAL VARIABLE) # # loopname IN (Name of ERROR-IS-IN-TOLERANCE # event) # (Name of ERROR-IS-OUT-OFloopname OT # TOLERANCE event) # Units that will be associated with the $_{\rm FV}$ variable. # units # To work properly, these units should be the same as the # label entered and both should be the same used in the # perf labels file for (ctl spd, Dyno torq, Net torq, and # Gross torq). # # The closed loop ramp rate (feedback units/sec) for the ramp rate # control variable. This may be a value, label, or 79 # character expression. If a label or expression are # used, they must have units of the feedback variable. # # loop tolerance This is the error (+-) under which the control loop # is said to be in tolerance. # # IMPORTANT NOTE: Many commands, as well as gp test, will overwrite # the ramp rate so it is likely that the value in the # specification file will only be used initially. @SPEED LOOP NAME #label(28) units CL-ramp rate loop tolerance Speed rpm 25 5 @TORQUE LOOP NAME #label(28) units CL-ramp rate loop tolerance Torque lb ft 25 10 @OTHER LOOP NAME #label(28) units CL-ramp rate loop tolerance AveExhTemp deg f 5 20 # # # There are 3 different CONTROLLERS that may be specified. These # are: # # 1 - DYNO # 2 - DYNO 2 (optional - for dual dyno control) # 3 - THROTTLE



#

_____ # # Dyno Dual Loop Control - for water brake dynos with inlet and # outlet restrictions that need to be switched as a function of an # independent variable. This option may also be used when two separate dynos are used in series - such as when a small # # motoring dyno is in series with a water brake or eddy current # dyno. # # If this keyword is used, the first dyno (DYNO) will be used # below the threshold and DYNO2 will be used above the threshold. # # If this keyword is NOT used, then DYNO will be used all of the # time. # # (OPTIONAL ENTRY) # # # Active this specifies whether the dyno switch and # second dyno controller are active: # 0, N, or NO = inactive # 1, Y, or YES = active # # is the value of the switch variable below Threshold value # which the dyno will switch to the second dyno # controller. It should be of the form # 123.4[units]. # # Threshold hysteresis is the amount of hysteresis that is applied to # eliminate switching between controllers due to # operating near the threshold crossing. It # should be of the form 12.3[units]. # # Switch variable is the label of a REAL variable that will be # used to determine when to switch between # controllers. This variable must already exist. # # # Example: Threshold = 100[lb ft]# Hysteresis = 10[lb ft] # Switch Variable = Dyno Torq # # If torque is 50 lb-ft, the first dyno # controller is used, and continues to be used # until torque exceeds 110 lb-ft (100 threshold # + 10 hysteresis). At this point the second # dyno controller is used, and is continued to



be used until torque drops below 90 lb-ft (100 threshold - 10 hysteresis). # # # When the first dyno is in use, the output of # the second dyno will be set to the lower # bound. Similarly, when the second dyno is # control, the output of the first dyno will be # set to its upper bound. # # Therefore, care must be taken select bounds # that accurately reflect the requirements at # the switch point. Also, hysteresis should be # kept to a reliable setting. # _____ # -#@DUAL DYNO CONTROL # Active Threshold_value Threshold_hysteresis Switch_variable
1 100[lb_ft] 10[lb_ft] Torque # _____ # # Specify the label and computation rate for the controller # (REQUIRED ENTRY) # # @DYNO CTRLER # <ctrler name> <output units> <interval> <open loop ramp rate> # # where: # ctrler name - The root of the variables created for the # controller. # The labels created are: # ctrler name CM (Command output) ctrler_name_CM(Command Output)ctrler_name_OL(Open loop target)ctrler_name_PT(Proportional term)ctrler_name_IT(Integral term)ctrler_name_DT(Derivative term)ctrler_name_FF(Feed forward term)ctrler_name_FA(Feed forward active) # # # # # # # # ctrler_name_FG ctrler_name_LB (Feedforward gain) # (Lower bound) ctrler_name_UB # (Upper bound) # # ctrler name IL (Integral lower bound) # ctrler_name_IU (Integral upper bound ctrler_name_LO (Lower output limit) (Integral upper bound) # # ctrler name UO (Upper output limit)



##########	<pre>Specify the output channel for the controller (REQUIRED ENTRY) @DYNO_OUTPUT_CHAN One of 4 output types may be specified, AO, CO, DO, or RV</pre>				
щ ФD,	YNO_CTRLER # label # root ou Dyno	tput units %	interv FAS	open-1 ral ramp 1 5	Loop cate
# # # # # .		the specifi initially. This keywor keywords!!!	cation fil d must app	e will only be ear before oth	e used ner DYNO
# # # #	ramp rate value, CyFlex label, or "expression" I IMPORTANT NOTES: Many commands, as well as gp_test, will ove the ramp rate so it is likely that the valu				
# # #	open loop	- The open	loop ramp	rate value. 7	This can be a
" # #		- Dyno and	throttle c	controls should	d be executed at
# # # #	interval	- The calcu Valid ent	lation int	erval for the LO, MED, FAS,	control loop. WARP, USR1, and
# # # #	command output units	- the units NOTE: The to	s assigned e units of _CM, _PT,	to created var the controller _FT and other	riables r are assigned variables.
# # # #		NOTE: Since root, chara	e 3 charact the root acters.	ers are append should have a	ded to the label maximum of 28
# # # #		ctrler_ ctrler_	_name_MD _name_OC	(Mode of cont VARIABLE) (Open/Closed VARIABLE)	crol - LOGICAL STRING
			MD		1



# # #	channel	 is the analog output channel number that will receive the Dyno_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_constan	<pre>t- a recursive filter factor between 0 and 0.99</pre>
" # #	CO <channel> <zero_< th=""><th>value> <span_value> <filter_constant></filter_constant></span_value></th></zero_<></channel>	value> <span_value> <filter_constant></filter_constant></span_value>
# # #	where: channel	- is the counter output channel number that will receive the Dyno_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_constan	t- a recursive filter factor between 0 and 0.99
# # #	RV <target_label></target_label>	<bias> <gain> <filter></filter></gain></bias>
# # #	where: target_label	- is the real variable where the Dyno_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
# # #	DO <do label="" pwm=""></do>	<base_period> [enable_label]</base_period>
# #	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.
 # # # # #	<pre>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,eg. 30[sec].</pre>
# # # #	enable_label - The label of a LOGICAL_VARAIBLE that is used to disable/enable the PWM output.
# #	LV <lv label="" pwm=""> <base_period> [enable_label]</base_period></lv>
# #	where:
# # # #	LV_PWM_label - is the label of the LOGICAL variable that will provide the PWM output. This label must already exist.
+ + + + + + + + + + + + + + + + + + +	<pre>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,eg. 30[sec].</pre>
# # # #	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.
@DY	YNO OUTPUT CHAN

AO <channel> <zero_value> <span_value> <filter_constant>



AO	1	0	100	0.0			
 Sp	ecify the (OPTIONA	command out L ENTRY)	put bounds f	or the controller			
@DYNO <1	_OUTPUT_B	DUNDS d> <uppe< td=""><td>er_bound></td><td></td><td></td></uppe<>	er_bound>				
wh	ere: lower bo	und - This e for th "expre	entry may be ne output ter ession". The	a value of the lower m, a label, or an default lower bound	bound is 0.		
		A lowe bound	er bound must is specified	be provided when an	upper		
	upper bo	und - This e for th "expre	- This entry may be a value of the upper bound for the output term, a label, or an "expression". The default upper bound is 100.				
		An upp bound	per bound mus is specified	t be provided when a	n lower		
DYNO_OUT # <lowe 0</lowe 	PUT_BOUND r_bound>	S <upper_k 100[%]</upper_k 	oound>				
Specify the integral term bounds for the controller (OPTIONAL ENTRY)							
f # @DYNO_INTEGRAL_BOUND_LABELS # <lower_bound> <upper_bound></upper_bound></lower_bound>							
wh	where: lower integral bound - This entry may be the value of the lower bound for the integral term, a label, or an "expression" A lower bound must be provided when an upper bound is specified.						
	upper in	tegral bound	d - This entr upper bound label, or an An upper bou lower bound	y may be the value o for the integral term "expression" nd must be provided is specified.	f the m, a when ar		



```
#
#
  NOTE: If the @DYNO INTEGRAL BOUND LABELS is not entered, integral
#
       bounds will default to 0 and 100. This may not be
#
       appropriate if negative commands are required.
#
#
#
  _____
@DYNO INTEGRAL BOUND LABELS
  #<lower bound label>
                             <upper bound label>
                              100[8]
    dyno int LB
 _____
#
#
#
     Specify the hysteresis for the controller
#
      (OPTIONAL ENTRY)
#
#
 @DYNO HYSTERSIS
#
    <value>
#
#
    where:
#
      value - is the hysteresis value of the controlled device.
#
             If not specified a value of 0.0 will be assumed.
#
#
 _____
@DYNO HYSTERSIS
  #<value>
    0.5
 _____
#
#
#
     Specify the command map pathname for the controller
#
       (OPTIONAL ENTRY)
#
#
 @DYNO 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.
#
#
    _____
#@DYNO COMMAND MAP FILE
#
 <pathname>
```

```
# /specs/throttle_map
```



#

```
#
#
      Specify the error scaling variable for the controller
#
         (OPTIONAL ENTRY)
#
#
   @DYNO ERROR SCALING LABEL
#
      <scale error label>
#
#
      where:
#
         scale error - The label whose value will be used to scale
#
             label
                      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. This is
#
                      a convenient way to make the gains a function
#
                      of another variable(s). The units of this
#
                      variable must be [none].
#
#
        _____
@DYNO ERROR SCALING LABEL
  #<scale error label>
    gain scale var
#
 _____
#
#
      Specify the feed forward variable for the controller
#
         (OPTIONAL ENTRY)
#
#
   @DYNO FEED FORWARD
#
      <label>
              <active> <gain>
#
#
      where:
#
                 - A label of an CyFlex variable is entered. Its
         label
#
                   value is multiplied by the gain. The variable
#
                   must be defined elsewhere and already exist in
#
                   the system.
#
#
                 - A flag indicating if the feed forward term is
         active
#
                   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.
#
#
                 - The feed forward gain that will be used when feed
         gain
#
                   forward is active. This can be a constant,
#
                   label, or expression.
```



NOTE: If this keyword is not entered, then no feed # forward term will be computed. # _____ # @DYNO FEED FORWARD # <label> <active> <gain> dyno ff NO dyno ff gain # _____ # # Specify the local command label for the controller # (OPTIONAL ENTRY) # @DYNO LOCAL COMMAND LABEL # # <local label> # # where: local label - is the name of the CyFlex REAL 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. # # _____ #@DYNO LOCAL COMMAND LABEL # <local label> # dyno local setting _____ # # # Specify the remote sense LOGICAL VARIABLE for the controller # (OPTIONAL ENTRY) # @DYNO 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. # # _____



#@DYNO REMOTE SENSE LABEL # <remote label> # dyno local remote # _____ # # Specify the initial PID gains for the controller. These values # will be reflected in the variables <ctrler name PG, IG, and DG> and may be changed by the user at any time after initial # # configuration. # An option (+s) exists with the dy_gains and th_gains command to # modify this file by storing the gains that are presently in use. # # Enter # <use dy gains> for more information. # # (REQUIRED ENTRY) # # @DYNO XXXX GAINS <proportional> <integral> <derivative> # # # where: # proportional - The proportional gain value # # integral - The integral gain value # # derivative - The derivative gain value # # ______ @DYNO SPEED GAINS # proportional integral derivative -0.600000 -0.300000 0.000000 @DYNO DYNO TORQUE GAINS # proportional integral derivative 0.11 0.011 1.0 @DYNO NET TORQUE GAINS # proportional integral derivative 1.20 0.12 0.012 @DYNO OTHER GAINS # proportional integral derivative 1.30 0.13 0.013 # ______ #



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 # # - A flag indicating if the variable should be history # included in the history log. Valid entries for active # this field are Y, YES, T or TRUE. All other # entries are set to FALSE. # # 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 # # _____ @DYNO LOOP VRBL OPTS CM #<display_precision> <history active> <history tolerance> 1 YES #@DYNO LOOP VRBL OPTS PT # #<display precision> <history active> <history_tolerance> # Ν 1 #@DYNO LOOP VRBL OPTS IT # #<display_precision> <history_active> <history tolerance> # Ν #@DYNO LOOP VRBL OPTS DT # #<display_precision> <history active> <history tolerance> # 1 3 Ν #@DYNO LOOP VRBL OPTS FF # #<display_precision> <history_active> <history_tolerance> # 1 3 Ν #@DYNO LOOP VRBL OPTS OL #<display precision> <history active> <history tolerance>



3 1 Ν # # _____ # # # The above keywords and explanations apply to the THROTTLE and DYNO2 # controllers. # # _____ # # NOTE: This keyword must appear before other THROTTLE keywords!!!!! # @THROTTLE CTRLER open-loop ramp rate # label # root output units interval 90 5 Throttle FAS @THROTTLE OUTPUT CHAN # AO <channel> <zero value> <filter constant> 0.0 0 100 AO 2 @THROTTLE OUTPUT BOUNDS #<lower_bound> <upper_bound> 0 100 @THROTTLE INTEGRAL BOUND LABELS #<lower bound label> <upper bound label> throt int LB 100 @THROTTLE HYSTERSIS #<value> 0 @THROTTLE ERROR SCALING LABEL #<scale error label> gain scale var @THROTTLE FEED FORWARD # <label> <active> <gain> throttle ff YES throt_ff_gain @THROTTLE SPEED GAINS



proportional integral derivative 0.02 0.02 0.00 @THROTTLE GROSS TORQUE GAINS # proportional integral derivative 0.02 0.02 0.00 @THROTTLE NET TORQUE GAINS # proportional integral derivative
0.060000 0.200000 0.000000 @THROTTLE OTHER GAINS # proportional integral derivative
 0.3 0.23 0.003 # ______ # # The above explanations apply to the second dyno controller as well # # Use these keywords ONLY if the dual dyno option is used! # # _____ # # NOTE: This keyword must appear before other DYNO2 keywords!!!!! # @DYNO2 CTRLER # label open-loop # root output units interval ramp rate Dyno2 00 FAS 5 @DYNO2 OUTPUT CHAN # AO <channel> <zero value> <filter constant> AO 3 0 100 0.0 @DYNO2 OUTPUT BOUNDS #<lower bound> <upper bound> 0 100 @DYNO2 INTEGRAL BOUND LABELS #<lower bound label> <upper bound label> -50 100



@DYNO2_HYSTE: # <value> 0</value>	RSIS							
<pre>#@DYNO2_ERROR_SCALING_LABEL # #<scale_error_label> # gain_scale_var</scale_error_label></pre>								
#@DYNO2_FEED_FORWARD # # <label> # dyno2_ff</label>			<active> YES</active>		<gain> dyno2_ff_gain</gain>			
<pre>@DYNO2_SPEED_GAINS # proportional integral derivative -0.600000 -0.300000 0.000000</pre>								
@DYNO2_DYNO_ # proport 1.0	TORQUE_G ional	AINS integr 0.11	al	derivat 0.011	zive			
@DYNO2_NET_T # proport 1.20	ORQUE_GA ional	AINS integr 0.12	al	derivat 0.012	zive			
@DYNO2_OTHER # proport 1.30	_GAINS ional	integr 0.13	al	derivat 0.013	cive 3			
YEND								



Appendix B. Example 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 0.00000E00 1.000000E-02 1.000000E-02 1.000000E-02 1.200000E01 1.226994E00 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 8.00000E01 8.866608E01 8.692828E01 1.000000E02 1.000000E02 1.000000E02 1.000000E02 # Number of coordinates 3 # Z Value Scaling Factor 7.000000E02 4.100629E-01 1.300000E03 1.00000E00 2.100000E03 6.301887E-01