Getting Results with the PIDE Autotuner
# Contents

## Introducing the PIDE Autotuner
- Installation ................................................... 1
- Requirements .................................................. 1
- What’s New? .................................................... 2
  - New Data Type .............................................. 2
  - Enhanced PID (PIDE) Property Dialog Box ............. 2
  - PIDE Block Addition ...................................... 2
- Terms and Acronyms ........................................ 4

## Getting Started
- Getting Started .............................................. 5
  - The Autotune Tab .......................................... 5
  - The PIDE Autotune Dialog Box ......................... 8
  - Autotune Faults ........................................... 10

## Using the PIDE Autotuner
- Using the PIDE Autotuner ................................. 13
  - Configuring the Autotuner ............................... 13

## PIDE Technical Information
- PIDE Technical Information ............................... 17
  - PIDE User Tag Acquisition Flow ...................... 17

## PIDE Autotuner Tips and Tricks
- PIDE Autotuner Tips and Tricks ......................... 19
  - Using the PIDE Autotuner with Slow Temperature Loops .......... 19
INTRODUCING THE PIDE AUTOTUNER

The RSLogix 5000 PIDE Autotuner provides a simple, open-loop autotuner built into the PIDE instruction used in the Function Block Diagramming language. Because PIDE Auto-tuning is built into the controller, you can perform autotuning from PanelViews or any other operator interface devices, as well as RSLogix 5000. The PIDE block has been designed to use a second tag whose data structure is defined for the Autotuner. This allows the users to create tune tags only for the loops (PIDE blocks) they wish to auto-tune. These tune tags can also be accessed by any device, which will allow PanelViews, etc., to perform auto-tuning by setting and reading the appropriate values in the Autotune data structure in the controller. The user can choose to share these tune tags between different PIDE blocks to save memory if desired.

INSTALLATION

The PIDE Autotune feature is installed with the rest of the RSLogix 5000 software but must be activated with an activation disk, which is purchased separately. If Autotune is not activated, some of the Autotune-related fields and buttons on dialogs boxes will be visible but disabled.

In addition, if the PIDE Autotune feature is not activated:

- You can run the Autotuner from an operator interface for any PIDE blocks that have already been configured for autotuning, but you cannot configure additional PIDE blocks for autotuning.
- You cannot associate or change the association of an Autotune tag with a PIDE block
- You can delete an association between the PIDE block and a tune tag. You can also undo this operation.
- You can import and export projects with tags of type PIDE_AUTOTUNE and PIDE blocks with an associated tune tag.
- You can drag-and-drop or paste programs or routines that contain PIDE blocks with associated Autotune tags. A Cut/Copy/Paste and Drag/Drop operation is allowed for a PIDE block, but the associated Autotune tag will not be copied.

REQUIREMENTS

The requirements for using the Autotuner feature are:

- RSLogix 5000, release 10 or later
INTRODUCING THE PIDE AUTOTUNER

What’s New?

WHAT’S NEW?

There have been some changes and enhancements to the RSLogix 5000 software to support the Autotune function. The following sections explain in more detail.

NEW DATA TYPE

A data type, PIDE_AUTOTUNE, was added to support the Autotuning algorithm. Tags of this type are used to define the parameters for auto-tuning a PIDE block. This data type is available everywhere tags are created, whether the PIDE Autotune feature is activated or not. The Tag Editor and the Create Tag and Tag Properties dialog boxes provide access to the PIDE_AUTOTUNE data type.

Each PIDE block needs to reference a PIDE_AUTOTUNE tag in order to be tuned; for maximum flexibility, assign a different PIDE_AUTOTUNE tag to each PIDE block. This will allow you to tune multiple loops at the same time. If you wish to save memory you can assign the same PIDE_AUTOTUNE tag to multiple PIDE blocks. In this case, however, you will only be able to tune one loop at a time. Each tag of type PIDE_AUTOTUNE uses just over 1Kb (kilobyte) of memory.

ENHANCED PID (PIDE) PROPERTY DIALOG BOX

The Enhanced PID Property dialog box has been updated with a new Autotune tab (see example on page 5). The Autotune tab will allow you to configure and acquire the PIDE_AUTOTUNE tag for the Autotune algorithm. A new Autotune dialog box handles starting the Autotune algorithm, monitoring outputs and loading the tuned gains. The other tabs in the PIDE Property dialog box have not changed.

Once the Autotune tag is associated and the code is downloaded to the controller, you would generally go Online and begin auto-tuning for the PIDE. But in some cases, you may want to configure all of your PIDE blocks for auto-tuning before you go back online. To accommodate this scenario, you are allowed to update the Process Type, PV Change Limit and CV Step Size while offline. If you do this to an Autotune tag that has been acquired by another PIDE, when you try to go back online you will be told that you must either download changes or upload what is in the controller.

For more information on the Autotune tab and auto-tuning operations, please see “Using the PIDE Autotuner” on page 13.

PIDE BLOCK ADDITION

The association of the Autotune tag will be available for editing at the bottom of the PIDE block. Select the tag using the same Tag Browser used throughout the FBD editor. If an Autotune tag has not been associated with this PIDE or if you request that this association be cleared, a question mark (“?”) will be displayed in this entry.

If the Autotune feature key is not present and no Autotune tag is associated with this PIDE then the tag name text (in this situation it is “?”) will appear grayed out and the inline editor is disabled. If an Autotune tag has been associated with this PIDE then the inline editor is fully operational, but the checking is done when the inline editor is closed. You are allowed to delete the association by simply deleting the tag name, but if you try to select another tag name, the action is canceled. A
message is displayed in the status bar informing you that you cannot do this operation because the
feature key is not present, and the previous tag name remains in this field.

**Import/Export Format Update**

A statement was added to the BLOCK import/export format. This optional statement contains the
tag name of the associated AUTOTUNE tag for a PIDE block and is only valid for PIDE blocks.

**Example:**

```plaintext
PIDE_BLOCK (ID := 0;
X := 180,
Y := 20,
Operand := PIDE_01,
VisiblePins := "PV, ..."
AutotuneTag := <AutotuneTag>
END_PIDE_BLOCK
```
**TERMS AND ACRONYMS**

**FBD**
Function Block Diagram

**Associate Tag**
The act of selecting an autotune tag to be used by a PIDE block, more than one PIDE can associate the same autotune tag.

**Acquire Tag**
The act of a PIDE block taking control of this autotune tag, once a tag is acquired no other PIDE block can use it for tuning.

**Release Tag**
The act of canceling an acquired tag, any PIDE block can release an autotune tag. This act takes precedence over all other autotune actions.
The user interface for the PIDE Autotuner consists mainly of the new Autotune tab that was added to the PIDE Properties dialog box. The Autotune functionality, a simple, open-loop Autotuner, is built right into the PIDE instruction.

The following sections describe the Autotune tab on the PIDE Properties dialog box and the PIDE Autotune dialog box.

**THE AUTOTUNE TAB**

The Autotune tab allows you to configure and acquire the PIDE_AUTOTUNE tag for the Autotune algorithm. The associated Autotune dialog box handles starting the Autotune algorithm, monitoring outputs and loading the tuned gains. The other tabs in the PIDE Property dialog box have not changed from the previous release of RSLogix 5000.

The following table describes the fields and buttons on the Autotune tab of the PIDE Properties dialog box.
### Feature Description

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag: Name</td>
<td>Displays the name of the Autotune tag associated with this PIDE block. If no tag is associated, it displays “?”</td>
</tr>
<tr>
<td>Tag Status</td>
<td>Displays one of three possible states:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Available</strong>: The Autotune tag has not been acquired by any other PIDE block. This is also indicated in the database when Autotune.AtuneAcquired is <strong>FALSE</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Acquired</strong>: The Autotune tag has been acquired by this PIDE block and can be used for autotuning. This is also indicated in the database when Autotune.AtuneAcquired is <strong>TRUE</strong> and PIDE.AtuneReady is <strong>TRUE</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>Acquired by Another PIDE</strong>: The Autotune tag has been acquired by another PIDE block and must be released before it can be acquired by this PIDE block and used for autotuning. This is also indicated in the database when Autotune.AtuneAcquired is <strong>TRUE</strong> and PIDE.AtuneReady is <strong>FALSE</strong></td>
</tr>
<tr>
<td>Acquire Tag button</td>
<td>Sets PIDE.AtuneAcquire to <strong>TRUE</strong> — acquires the Autotune tag for this PIDE block. When the tag is already acquired by this PIDE, the button is disabled. This is also indicated in the database when PIDE.AtuneReady and Autotune.AtuneAcquired are <strong>TRUE</strong>. If the tag name is “?” the button is disabled. If the Autotune tag has been acquired by another PIDE this button is also disabled.</td>
</tr>
<tr>
<td>Release Tag button</td>
<td>Sets PIDE.AtuneUnacquire to <strong>TRUE</strong> — releases a tag that has been acquired by any PIDE block. This button is enabled if any PIDE block has acquired this tag. It will release the tag from this and any other PIDE block that acquired it. If the Autotune tag was acquired by another PIDE block, clicking this button displays a message indicating that this action will abort an existing Autotune already in progress.</td>
</tr>
</tbody>
</table>
**Getting Results with the PIDE Autotuner**

**Feature** | **Description**  
--- | ---  
Process Type | Sets the value of the Autotune parameter ProcessType. Available types (integrating or self-regulating) are: Temperature, Pressure, Flow, Level, Position, Velocity, Integrating, Non-integrating, or Unknown.  
A process is called an *integrating process* if a step change in loop output causes the PV to linearly increase until it reaches a process limit. A level loop is an example of an integrating process. If level is controlled by opening and closing an inlet valve, and if the inlet valve is opened, the level will rise until the tank is filled.  
A process is called a *non-integrating process* if a step change in loop output causes the PV to change in response and level out at a new value. A flow loop is an example of a non-integrating process.  

![Integrating Process](image)

![Non-integrating Process](image)

PV Change Limit | Sets the value of the Autotune parameter PVTuneLimit. If the CV step change causes the PV to change beyond the PV Change Limit, the Autotune will abort. You can use this feature to make sure that an autotune does not cause your PV to increase or decrease beyond the limit.  
CV Step Size | Sets the amount that the CV will change when autotuning is started. When autotuning is complete, the CV will return to its original value. You can enter either a positive (step up) or negative (step down) value.  
Execution State | Ready, In Progress, Complete, Aborted  
Proportional | Displays the value of the PIDE parameter, PGain.  
Integral | Displays the value of the PIDE parameter, IGain.  
Derivative | Displays the value of the PIDE parameter, DGain  
**Autotune** button | Opens the Autotune dialog box. It is enabled only if the PIDE.AtuneReady = TRUE, Autotune.AtuneAcquired = TRUE, and PIDE.AtuneDataInv = FALSE
THE PIDE AUTOTUNE DIALOG BOX

Once you have configured an Autotune tag, click the Autotune button to open the PIDE Autotune dialog box. The PIDE Autotune dialog box allows you to Autotune and load the gains for the acquired PIDE. If data from a previous Autotune is still valid then you can immediately select and load the gains. When you click the Start button the Autotune algorithm will begin to run. At any point during this operation you can abort the Autotune process by clicking the Abort button. If the Autotune algorithm fails, the reason for the failure displays in the Autotune status box. When the Autotune algorithm has completed successfully you can select which gains to load into the PIDE block. The selection process is accomplished via the radio buttons next to the different Response speeds. You can manually update any of the gains by selecting Current and typing new values in the boxes. The boxes are only enabled when Current is selected. You can now load these gains into the PIDE block by clicking the Set Gains in PIDE button.

The following table describes the fields and buttons on the PIDE Autotune dialog box.
### Feature Description

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start button</strong></td>
<td>Allows you to start Autotune. It is enabled when the PIDE.AtuneReady parameter is TRUE and is disabled when the PIDE.AtuneOn parameter is TRUE. In order to start autotuning, the PIDE loop must be in Manual mode (either Operator Manual or Program Manual).</td>
</tr>
<tr>
<td><strong>Abort button</strong></td>
<td>Requests Autotune abort. It is enabled when the PIDE.AtuneOn parameter is TRUE. Sets PIDE.AtuneAbort to TRUE.</td>
</tr>
<tr>
<td><strong>Set Gains in PIDE button</strong></td>
<td>Sets the PIDE current gains with the selected tuned gains. The tuned gains loaded are based on the currently selected gains. This button is enabled when the PIDE.AtuneDone parameter is TRUE. Sets PIDE.AtuneUseGains to TRUE.</td>
</tr>
<tr>
<td><strong>Autotune Gains radio buttons</strong></td>
<td>Determine which set of gains will be downloaded to the controller when Set Gains in PIDE is clicked. If Slow, Medium or Fast Response is selected then the Autotune parameter ResponseSpeed will be updated to reflect the selection. If Current is selected then the Current Integral, Derivative and Proportional Gains boxes are enabled. You can then adjust these values prior to clicking Set Gains in PIDE. The default value is the value contained in the ResponseSpeed in the PIDE.</td>
</tr>
</tbody>
</table>

**Slow Response** -
- **Proportional**
  - Displays the value of the Autotune parameter, PGainTunedSlow.
- **Integral**
  - Displays the value of the Autotune parameter, IGainTunedSlow.
- **Derivative**
  - Displays the value of the Autotune parameter, DGainTunedSlow.

**Medium Response** -
- **Proportional**
  - Displays the value of the Autotune parameter, PGainTunedMed.
- **Integral**
  - Displays the value of the Autotune parameter, IGainTunedMed.
- **Derivative**
  - Displays the value of the Autotune parameter, DGainTunedMed.

**Fast Response** -
- **Proportional**
  - Displays the value of the Autotune parameter, PGainTunedFast.
- **Integral**
  - Displays the value of the Autotune parameter, IGainTunedFast.
- **Derivative**
  - Displays the value of the Autotune parameter, DGainTunedFast.
**GETTING STARTED**

*The PIDE Autotune Dialog Box*

---

**Feature** | **Description**
--- | ---
Current - Proportional | Displays the value of the PIDE parameter, PGain. When enabled these values can be changed by the user.
Current - Integral | Displays the value of the PIDE parameter, IGain. When enabled these values can be changed by the user.
Current - Derivative | Displays the value of the PIDE parameter, DGain. When enabled these values can be changed by the user.
Time Constant, Deadtime and Gain | Used to estimate the validity of the gains found by the Autotuner. If the process model represented by the Time Constant, Deadtime, and Gain is similar to how you know your process actually runs, then the resulting gains should give good results.
Execution State | Displays the execution state: Ready, In Progress, Complete, Aborted
Autotune Status | Displays OK or Detailed Fault status information (see Autotune Faults table). The faults will be displayed one after another with a comma as a separator. If there are more than two lines of text, there will be a vertical scrollbar.

---

**AUTOTUNE FAULTS**

The following table defines the possible faults generated by Autotune:

<table>
<thead>
<tr>
<th>Autotune Faults</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtuneFault</td>
<td>Autotune has generated any of the following faults.</td>
</tr>
<tr>
<td>PVOutOfLimit</td>
<td>Either PV or the deadtime-step ahead prediction of PV exceeds PVTuneLimit during Autotuning. When True, Autotuning is aborted.</td>
</tr>
<tr>
<td>ModeInv</td>
<td>The PIDE mode was not Manual at start of Autotuning or the PIDE mode was changed from Manual during Autotuning. When True, Autotuning is not started or is aborted.</td>
</tr>
<tr>
<td>CVWindupFault</td>
<td>WindupHIn or WindupLIn is True at start of Autotuning or during Autotuning. When True, Autotuning is not started or is aborted.</td>
</tr>
<tr>
<td>StepSizeZero</td>
<td>StepSizeUsed=0 at start of Autotuning. When True, Autotuning is not started.</td>
</tr>
<tr>
<td>CVLimitsFault</td>
<td>CVLimitsInv and CVManLimiting are True at start of Autotuning or during Autotuning. When True, Autotuning is not started or is aborted.</td>
</tr>
</tbody>
</table>
### Autotune Faults

<table>
<thead>
<tr>
<th>Fault</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVInitFault</td>
<td>CVInitializing is True at start of Autotuning or during Autotuning. When True, Autotuning is not started or is aborted.</td>
</tr>
<tr>
<td>EUspanChanged</td>
<td>CVEUSpan or PVEUSpan change during Autotuning. When True, Autotuning is aborted.</td>
</tr>
<tr>
<td>CVChanged</td>
<td>CVOper is changed when in Operator control or CVProg is changed when in Program control or CV becomes high/low or ROC limited during Autotuning. When True, Autotuning is aborted.</td>
</tr>
<tr>
<td>AtuneTimedOut</td>
<td>Elapsed time is greater than TestLength since step test is started. When True, Autotuning is aborted.</td>
</tr>
<tr>
<td>PVNotSettled</td>
<td>The PV is changing too much to Autotune. When True, Autotuning is aborted. Wait until PV is more stable before autotuning.</td>
</tr>
<tr>
<td>TagReleased</td>
<td>The acquired tag was released by another PIDE block</td>
</tr>
<tr>
<td>TagInvalid</td>
<td>The acquired tag is not of the correct data type</td>
</tr>
</tbody>
</table>
GETTING STARTED

The PIDE Autotune Dialog Box
Beginning with Release 10 of ControlLogix, Autotune is configured from RSLogix 5000; the control algorithm resides in the Logix controller firmware. It provides a simple, open-loop Autotuner built right into the PIDE instruction.

**Configuring the Autotuner**

1. Create the Autotune Program Tag for the loop by right-clicking on Program Tags and selecting New Tag.

2. Name the tag and select PIDE_AUTOTUNE as the Data Type. Click OK.
3. Assign the Autotune tag to the loop (see example below).

4. View Block Properties for the PIDE tag by clicking the browse button (it’s the little blue button with three dots on it; see the example shown in step 3). The PIDE Properties dialog box opens.

5. Click the Acquire Tag button if the Autotune tag is not yet acquired. If another PIDE loop has acquired the Autotune tag, you can click Release Tag to force the other loop to release the Autotune tag and allow this loop to acquire it.

**NOTE:** If the other loop is in the middle of an autotune, the autotune will be aborted.
6. Select the type of process.

7. Configure the Autotune parameters: **PV Change Limit** and **CV Step Size**.
   If the PV reaches the PV Change Limit while an autotune is in progress, the autotune will abort. For autotuning to complete successfully, set the PV Change Limit far enough above or below the current PV so that the CV Step Size will not cause the PV to go beyond the PV Change Limit (depending on how the process will respond to the CV Step).

   CV Step Size is the amount that you want the output to step when autotuning is started. When the autotuning completes, the output is returned to its previous value. CV Step Size can be positive (step up) or negative (step down).

8. Click the **Autotune** button.

9. Place the loop in Manual mode and wait for the PV to settle to a steady state value.

10. Click **Start** and the Autotuner will start collecting data and will step the CV by the amount specified in the CV Step Size box.
11. Once the Autotuner has collected enough data, it will reset the CV to the original value and display the calculated values.

12. Select type of process response that is required by your control Loop:
   - **Slow** - Slow response, no overshoot
   - **Medium** - Faster response with typically minimal overshoot
   - **Fast** - Fastest response with potential for more overshoot

![Process Response Types](image)

13. Click **Set Gains in PIDE** and set your loop to Automatic mode and you are done. You have tuned your loop in record time!
The following diagram shows the steps in the Autotune flow:

1. Begin
2. Is tag acquired by this PIDE?
   - Yes: Update the Process Type, PV, and CV parameters
   - No: Is tag acquired by another PIDE?
     - Yes: Request release of tag
     - No: Is Autotune ready?
       - Yes: PIDE in Manual mode?
         - Yes: Enter Manual mode in PIDE
         - No: Start Autotune
       - No: Exit
3. Is tag released?
   - Yes: Request to acquire tag
Getting Results with the PIDE Autotuner
This chapter provides tips and tricks for using the PIDE Autotuner. Additional tips and tricks will be added from time to time, so be sure to check the Rockwell Extranet at http://www.software.rockwell.com/extranet/ for updated documentation every now and then.

Using the PIDE Autotuner with Slow Temperature Loops

You may find the following items useful if you are using the Autotuner and have slow temperature loops.

- Make sure that the update time for your temperature loop is appropriate for your process. Most temperature loops only need to execute once every second or two since the process response is so slow. If you execute the PIDE instruction much faster than needed, the Autotuner may not be able to store a sufficient amount of data samples to calculate an accurate set of gains. If you have slow temperature loops, it is recommended that the PIDE instructions be executed in a slow periodic task.

- Temperature loops often require large derivative gains to obtain the best control. Because the derivative term of a loop works on change in PV or change in error, loops with large derivative gains are very sensitive to noise on the PV signal. If the PV for your loop is noisy, you should use filtering on the analog input module or use a filter function block to filter the PV signal before it is wired into the PIDE instruction. This will stop the noise signal from causing the CV output to bounce in response to the noisy PV.

- If the temperature loop takes a long time to rise to a new steady state value after a CV step change, the PIDE Autotuner will try to tune the loop as an integrating loop in order to provide a faster result, even though you chose “Temperature” as the process type. This will sometimes give less than optimal results. If this is the case, try setting the process type to “Non-Integrating” and run the Autotuner again. This will force the Autotuner to wait for the PV to begin to settle into a new steady state value and will sometimes give better results for this type of loop.

- If you are controlling a temperature loop with a pulsed output, and are typically using the Split-Range Time-Proportioning (SRTP) block, make sure that the SRTP block or similar output pulsing logic executes in a much faster task. For these types of loops, you should generally execute the PIDE instructions in a slow (one or two second) periodic task, but the SRTP instructions should execute every 10-50 milliseconds in a faster periodic task. This will allow the pulsed output to have the resolution necessary to effectively control the heating or cooling elements. You can use a controller-scoped tag to pass the CV output from the PIDE instruction to the input of the SRTP instruction.
Using the PIDE Autotuner with Slow Temperature Loops
INDEX

A
Acquire Tag 4
activation disk 1
Associate Tag 4

D
data types
PIDE_AUTOTUNE 2

E
Enhanced PID Property (modal) dialog box 2

F
FBD 4

I
Import/Export format update 3
installation 1

P
PIDE Autotuner 1
activation disk 1
installing 1
tips and tricks 19
using 13
using with slow temperature loops 19
PIDE block 2
sample block glyph 17
PIDE Property dialog box 2
PIDE user tag acquisition flow diagram 17
PIDE_AUTOTUNE 2

R
Release Tag 4
requirements 1
RSLogix 5000
what’s new in 2

S
slow temperature loops 19

T
technical information 17
terms and acronyms 4
tips and tricks 19

U
using the PIDE Autotuner 13