Programming Manual

Original Instructions

Rockwell Automation

iTRAK System

Firmware Version: 1.110

Allen-Bradley • Rockwell Software

Rockwell Automation

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

	**WARNING:** Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
	**ATTENTION:** Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.

	**SHOCK HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.
	**BURN HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.
	**ARC FLASH HAZARD:** Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Summary of Changes

This manual contains new and updated information.

Topic	Page
This revision is the first release of this programming manual.	All

The iTRAK® system is designed to provide an integrated Rockwell Automation motion solution. It combines the drive and motor/actuator into one component to simplify integration into your application.

The purpose of this manual is to provide you with the information needed to commission, tune, and program additional functions for an iTRAK system.

This manual is intended for qualified personnel. You must be able to program and operate an iTRAK system. In addition, you must have an understanding of the parameter settings and functions.

Additional Resources

These documents contain additional information concerning related products from Rockwell Automation.

Resource	Description
iTRAK System Technical Data, publication 2198T-TD001	Provides product specifications for the iTRAK system.
Kinetix Servo Drives Specifications Technical Data, publication KNX-TD003	Provides product specifications for the Kinetix® Integrated Motion over EtherNet/IP network, Integrated Motion over SERCOS interface, EtherNet/IP networking, and component servo drive families.
Kinetix Motion Accessories Specifications Technical Data, publication KNX-TD004	Provides product specifications for Bulletin 2090 motor and interface cables, low-profile connector kits, drive power components, and other servo drive accessory items.
iTRAK System User Manual, publication 2198T-UM001	Information on how to install, configure, start, and troubleshoot your iTRAK system.
Kinetix 5700 Servo Drives User Manual, publication 2198-UM002	Information on how to install, configure, start, and troubleshoot your Kinetix 5700 servo drive system.
iTRAK System, Firmware Revisions, publication 2198T-RN001	Provides information on updates to the iTRAK system firmware.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications website, http://www.rockwellautomation.com/global/certification/overview.page	Provides declarations of conformity, certificates, and other certification details.

You can view or download publications at http://www.rockwellautomation.com/global/literature-library/overview.page. To order paper copies of technical documentation, contact your local Allen-Bradley distributor or Rockwell Automation sales representative.

Notes:

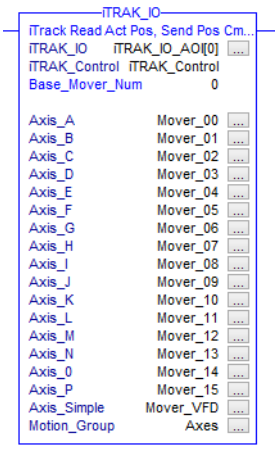
Getting Started

The iTRAK® system is configured and programmed with Studio 5000 Logix Designer®, version 21 and later. It is compatible with Allen-Bradley® controllers that support Integrated Motion on the EtherNet/IP network, with built-in Ethernet or with an Ethernet card that supports time synchronization. The controller implementation is designed to be similar to the programming experience of Kinetix® servo drives.

Each mover operates as one virtual axis within the system. This mapping lets you use Logix Designer application motion move commands, cam profiles, or stop commands, which normally control a rotary axis, to control the movers. The gateway tracks which mover is on which motor modules. It manages the communication between the controller and the motor modules, and directs motion commands to the appropriate location.

Communication over the EtherNet/IP network between the controller and the gateway is grouped into packets that contain the data for 16 movers. The number of active movers determines the number of packets that must be created and sent each Coarse Update Period. An Add-On Instruction that is called iTRAK_IO executes packet construction and processing. The iTRAK_IO instruction must be part of a program that executes in a Motion Event Task.

Figure 1 - iTRAK_IO Add-on Instruction



Each revision of iTRAK firmware is designed to work with specific functionality within the Logix Designer application project file. This programming manual revision is specific to firmware revision 1.110.

Before You Begin

ATTENTION: Before commissioning, verify that the emergency stop equipment works. Do not operate the machine if the safety circuit is not working.

Dangerous movements can occur immediately after power is applied or after an unspecified time of operation. The monitoring functions of the system are normally sufficient to avoid malfunction of the drives. Safety devices, for the protection of personnel from injury and property damage, cannot be relied upon until the integrated monitoring functions are made effective. Until integration is complete, you must assume that faulty drive movements can occur at any time.

ATTENTION: The iTRAK system can produce dangerous movements. There is a danger to life and risk of serious injury while integrating the system.

Faulty control of motors can cause dangerous movements. The following are common examples:

- Improper or wrong wiring or cable connection
- Operator Errors
- Wrong input of parameters
- Malfunction of sensors and encoders
- Defective components
- Software or firmware errors

Protection against Magnetic and Electromagnetic Fields during Operation

The motor modules, when in use, and permanent magnets pose a danger to persons with heart pacemakers, metal implants, and hearing aids.

ATTENTION: There is a risk of health hazard for persons with heart pacemakers, metal implants, and hearing aids while in proximity of magnetic and components that produce magnetic field.

- The movers have strong magnets.
- The track creates strong magnetic fields while energized during operation.
- Persons with heart pacemakers, metal implants, or hearing aids must not enter areas where components of the drive and control systems are mounted, commissioned, and operated.

Starter Projects

There are three iTRAK starter projects that are included with the system firmware in the Product Compatibility and Download Center. These starter projects are designed to ease your controller integration and provide sample logic to show how to implement the iTRAK system within your project.

Table 1 - Available Starter Projects

Project Name	Target Controller	Studio 5000® Version
iTRAK_CORE_Project_PCM.ACD	1756-L71	21.*xx*
iTRAK_CORE_Project_L7_IPS.ACD	1756-L71	28.11
iTRAK_CORE_Project_L8_IPS.ACD	1756-L85E	28.11

Figure 2 - iTRAK Core Project PCM

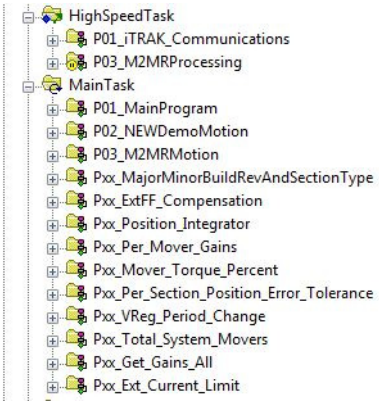
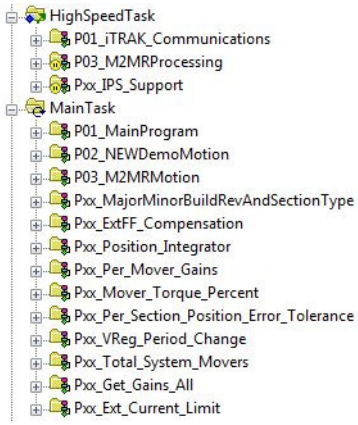
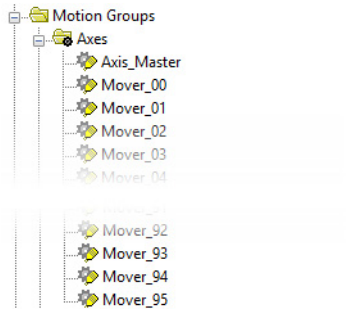


Figure 3 - iTRAK Core Project L7 IPS and L8 IPS



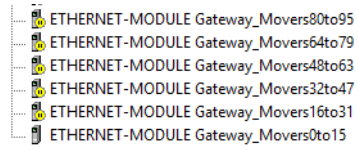
Each starter project includes a motion group that is configured with 96 mover axes.

Figure 4 - Motion Group



Each starter project includes multiple instances of the gateway in the I/O tree.

Figure 5 - Gateway in the I/O Tree



If you are using a starter project, you must set certain parameters that are based on your track configuration.

- Coarse Update Period must be adjusted.
- Unused mover axes must be moved to Ungrouped Axes.
- Position Unwind must be configured in active virtual axes.
- Gateway RPI must be set and additional connections must be uninhibited if needed.
- Set key track information that is used in the iTRAK_Control tag of the iTRAK_IO Add-On Instructions.

See Manual Configuration on page 11 for more detail on those operations.

Manual Configuration

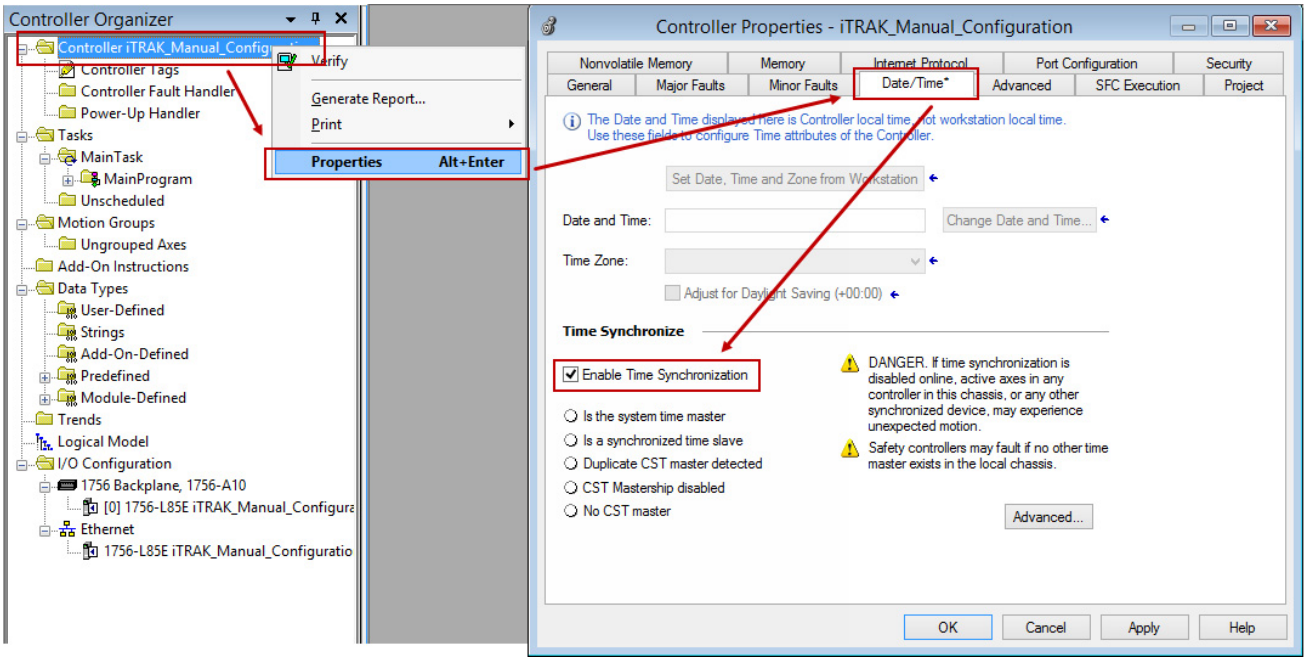
You can manually configure an existing project to recreate the functionality of the starter projects. The steps that are outlined in this section detail how to configure, but not how to add the Motion Group, Virtual Axes, or Generic Ethernet Connections.

IMPORTANT The files that you import follow the tag name scheme that is outlined in this document. If you use any other tag name scheme, you must make associations to those tags during the program import process. The details of that operation are not outlined here.

General

The iTRAK system communication requires time synchronization between the controller and the gateway.

1. From the Controller Organizer, double-click the controller or right-click and select Properties.
2. Click the Date/Time tab.
3. Check the box for Enable Time Synchronization.

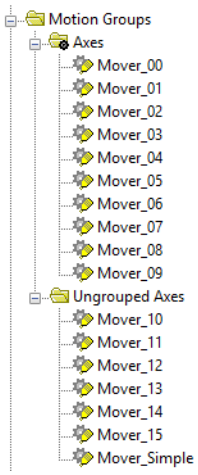


Motion Group and Virtual Axes

Virtual axes represent movers in the Logix Designer application. Axes must be created in multiples of 16 to complete the communication packet, even if fewer axes are used. Another virtual axis must also be created to represent a simple axis that can be left in the Ungrouped Axes folder.

In the example that is shown in Figure 6, only 10 movers are needed. The remaining six axes are created and left in the Ungrouped Axes folder in the Controller Organizer along with the simple axis.

Figure 6 - Example of When There Are Less Then 16 Axes



IMPORTANT The number of virtual iTRAK mover axes in the Motion Group must match the number of axes set in the iTRAK_Control.Data.ActiveMovers tag.

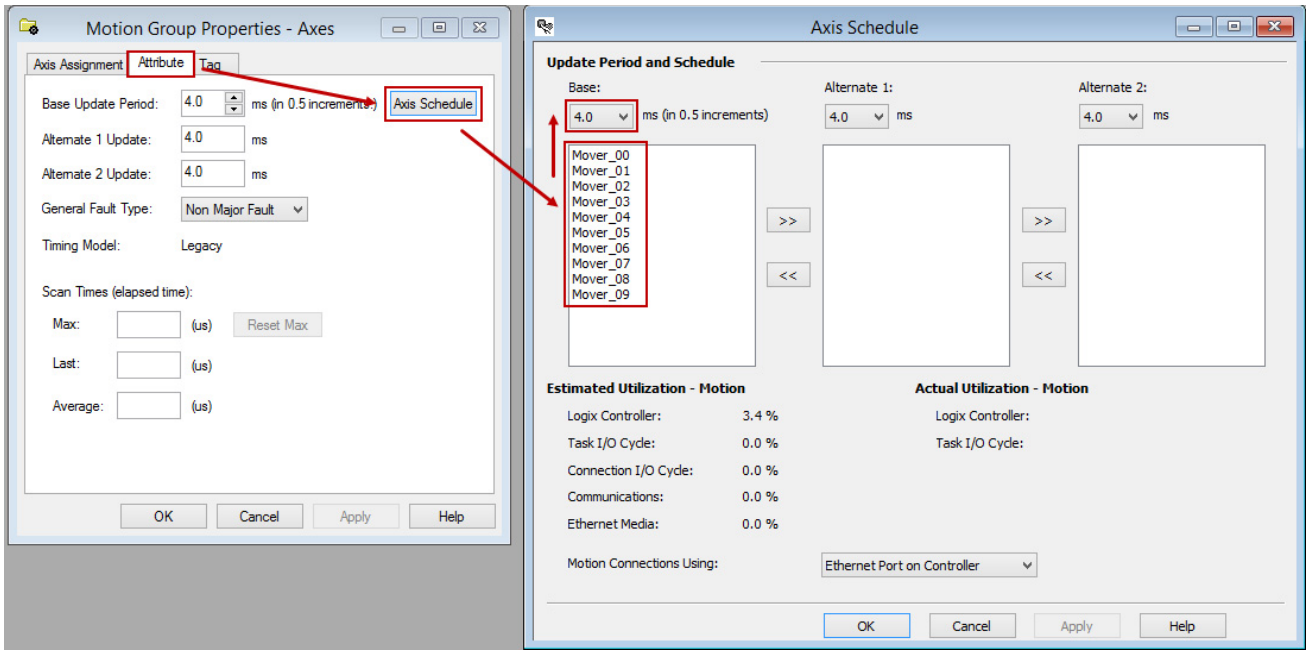
Balance the performance requirements of your system against the overhead time that is required and set the appropriate Coarse Update Period. Smaller Coarse Update Periods increase the performance of the motion application, but reduce the time for other tasks in the controller. The recommended Coarse Update Period is 2x the Requested Packet Interval (RPI), which is typically 1x or 2x the position loop update rate of the iTRAK system. You can determine the position loop update rate of your system by using the iTRAK System Technical Data publication, 2198T-TD001, or by using the iTRAK_Control.Cmd.RetrieveLoopPeriods controller tag. Higher Coarse Update Periods are possible, but servo performance can degrade as the Coarse Update Period increases. For more information, see the Integrated Architecture Builder® Knowledgebase article 1040301.

To set the Coarse Update Period, do the following.

1. From the Controller Organizer, double-click the motion group or right-click and select Properties.
2. Click the Attribute tab.

Your view can differ from the graphic that is shown depending on the version of Logix Designer application you are using.

3. Choose a Coarse Update Period or Base Update Period for the Motion Group.
4. If the Axis Schedule selection is visible, make sure that all mover axes are in the same schedule.



The virtual axes that are used by the movers must be configured. One track loop is one revolution of the virtual axis and the axis scale is mapped to millimeters. To configure the virtual axes that are used by the movers do the following.

1. From the Controller Organizer, double-click a virtual axis of a mover or right-click and select Properties.
2. Optionally, click the Units tab and set Position Units to millimeters or mm.
3. Click the Conversion tab.
4. Set the Positioning Mode to Rotary.

   This setting lets the axes unwind at the configured position.
5. Set the Conversion Constant to 1000.

   This value sets an effective command resolution of 1 µm.
6. Set the Position Unwind value according to your track length.

   This value must be equal to the circumferential track length in microns.
7. Click the Dynamics tab.

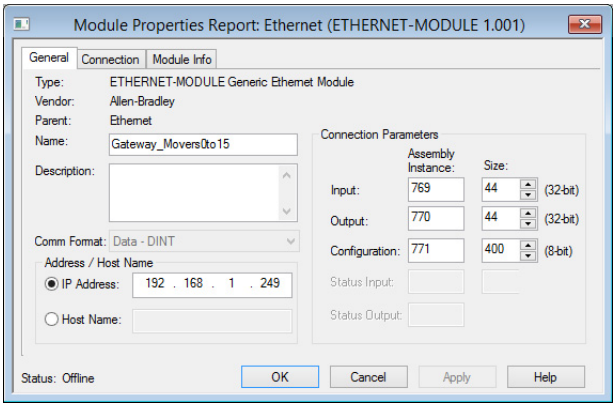
8. Set the Maximum Speed, Maximum Acceleration, Maximum Deceleration, Maximum Acceleration Jerk, and Maximum Deceleration Jerk.
9. Repeat these steps for all axes in the Motion Group.

Other axes can use the same Motion Group, including servos, variable frequency drives, converters, and other virtual axes.

Ethernet Communication

Each generic Ethernet module connection type can accommodate up to 16 movers. To configure the modules that are needed for your application, do the following.

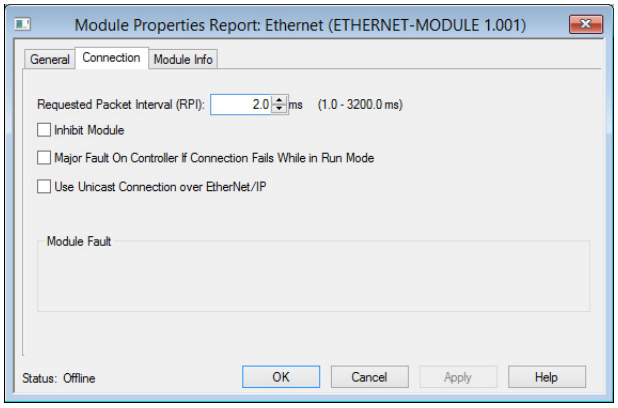
1. From the Controller Organizer, add a new ETHERNET-MODULE Generic Ethernet Module to the Ethernet network.
2. Click the General tab and set the Comm Format to Data - DINT.



3. Enter the Name, Connection Parameters, and IP Address from this table.

Name	Movers Assigned	Assembly Instance (Size)			IP Address
		Input	Output	Configuration	
Gateway_Movers0to15	0…15	769 (44)	770 (44)	771 (400)	192.168.1.249 or user-defined
Gateway_Movers16to31	16…31	779 (44)	780 (44)	781 (0)	192.168.1.248 or (user-defined last octet minus 1)
Gateway_Movers32to47	32…47	789 (44)	790 (44)	791 (0)	192.168.1.247 or (user-defined last octet minus 2)
Gateway_Movers48to63	48…63	799 (44)	800 (44)	801 (0)	192.168.1.246 or (user-defined last octet minus 3)
Gateway_Movers64to79	64…79	809 (44)	810 (44)	811 (0)	192.168.1.245 or (user-defined last octet minus 4)
Gateway_Movers80to95	80…96	819 (44)	820 (44)	821 (0)	192.168.1.244 or (user-defined last octet minus 5)

4. Click the Connection tab and set the RPI to one or two times the position loop update rate of the iTRAK system.



5. Clear the Inhibit Module checkbox.

   If the module is needed, based on the number of movers in your system, clear the Inhibit Module checkbox.

6. Clear the Use Unicast Connection over EtherNet/IP checkbox.

7. Repeat steps 1...6 until all six Ethernet modules are configured or the number of Ethernet modules can accommodate all movers on the system.

8. Set Gateway_Movers0to15:C.Data[0] to the appropriate value for your controller using this table.

Controller	Value
CompactLogix 5380 CompactLogix 5480 ControlLogix 5580	16#02
CompactLogix 5370 ControlLogix 5570	16#00

Gateway_Movers0to15:C	{...}	{...}		AB:ETHERNET_MODULE:C:0
Gateway_Movers0to15:C.Data	{...}	{...}	Hex	SINT[400]
Gateway_Movers0to15:C.Data[0]	16#02		Hex	SINT
Gateway_Movers0to15:C.Data[1]	16#00		Hex	SINT

Import Program P01_iTRAK_ Communications

IMPORTANT To use the iTRAK system, you must use the program P01_iTRAK_Communications in an Event Task triggered by the Motion Group Execution. This program and its execution provide timely processing and transmission of data. Placement in any other task does not produce stable motion control.

This program contains the logic and source-protected Add-On Instructions that are required for the controller and gateway. The program performs the

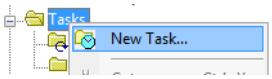
abstraction between the virtual movers and the drive electronics. The program P01_iTRAK_Communications contains two routines.

Table 2 - P01_iTRAK_Communications Routines

Routine	Description
R00_iTRAK_IO	The following steps are executed in this routine. 1. Synchronous Copy File from the gateway input assembly to the tag used for iTRAK_Control in the iTRAK_IO Add-On Instruction. 2. Execution of the iTRAK_IO Add-On Instructions. 3. Synchronous Copy File from the tag that is used for iTRAK_Control in the iTRAK_IO Add-On Instruction to the gateway output assembly. 4. Immediate output of the gateway output assembly.
R01_Headway	Collision checks are primarily performed in the gateway however it can be helpful when you commission the system to execute those checks in parallel in the controller. This routine compares the command position of adjacent movers and reports faults if there are overlaps.

To create an Event Task triggered from the Motion Group Execution, do the following.

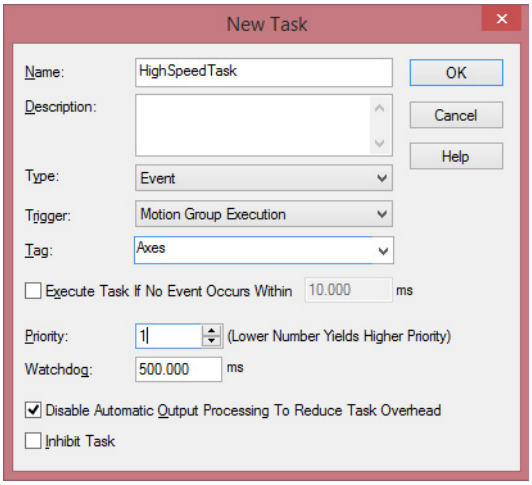
1. In the Controller Organizer, right-click on Tasks and select New Task...



2. Choose a name for the task, set the Type to Event, Trigger to Motion Group Execution, and Tag to your Motion Group.

   We recommend that you set the Priority to 1.

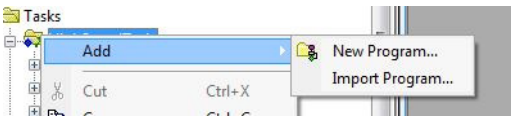
3. Click OK.



To import the program P01_iTRAK_Communications into your project, do the following.

1. From the Controller Organizer, right-click on the folder associated the Event Task that the Motion Group execution triggers.
2. Choose Import Program.

The Import Program can be under the Add menu.

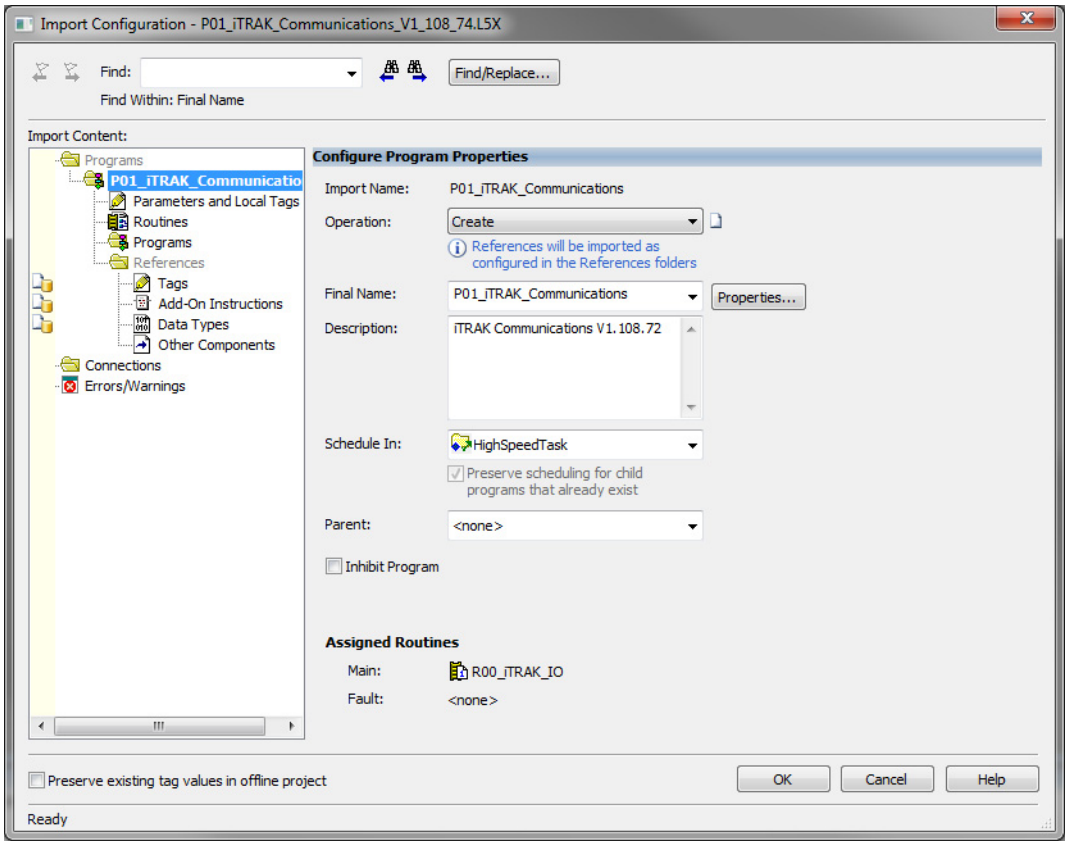


Logix Designer application version 28 is shown. Other versions can have another path.

3. Select the P01_iTRAK_Communications file, click Import.

   The file name P01_iTRAK_Communications_*Vmajor_minor_build*.L5X represents the version of the program.

   The Import Configuration dialog box appears.



4. Select each of the folders in the Import Content tree and set the parameters that are listed in this table.

   Follow the guidelines that are shown for an Existing iTRAK Project or a New iTRAK Project. Do not change parameters other than the ones listed in this table.

Folder	Parameter	Existing iTRAK Project	New iTRAK Project
P01_iTRAK_Communications Configure Program Parameters	Operation	Overwrite	Create
	Schedule In	HighSpeedTask	HighSpeedTask
Tags	Axes	Use existing	Create
	Axis_Master	Use existing	Use existing if available, if not create one.
	Gateway_Movers*X*to*Y*:I	Use existing	Use existing
	Gateway_Movers*X*to*Y*:O	Use existing	Use existing
	iTRAK_Control	Overwrite	Create
	Mover_*XX*	Use existing	Use existing if available, if not create one.
	Mover_VFD	Use existing	Create
Add-On Instructions	iTRAK_IO	Overwrite	Create
Data Types	All	Overwrite	Create

5. To complete the import process, click OK

Import Program P01_MainProgram

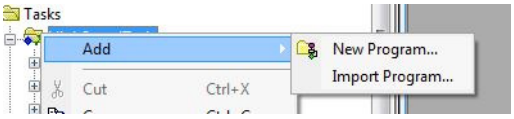
IMPORTANT To use the iTRAK system, you must use the P01_MainProgram program. The P01_MainProgram program contains routines and logic that support general operations of the track including the ability to configure, enable, disable, and retrieve status information.

The program P01_MainProgram does not need to be in a high-priority or scheduled task. In the Starter Projects, it is executed in a Continuous Task with a 500 ms Watchdog.

To import the program P01_MainProgram into your project, do the following.

1. In the Controller Organizer, right-click on the task you want to place the program into.
2. Choose Import Program.

   The Import Program can be under the Add menu.

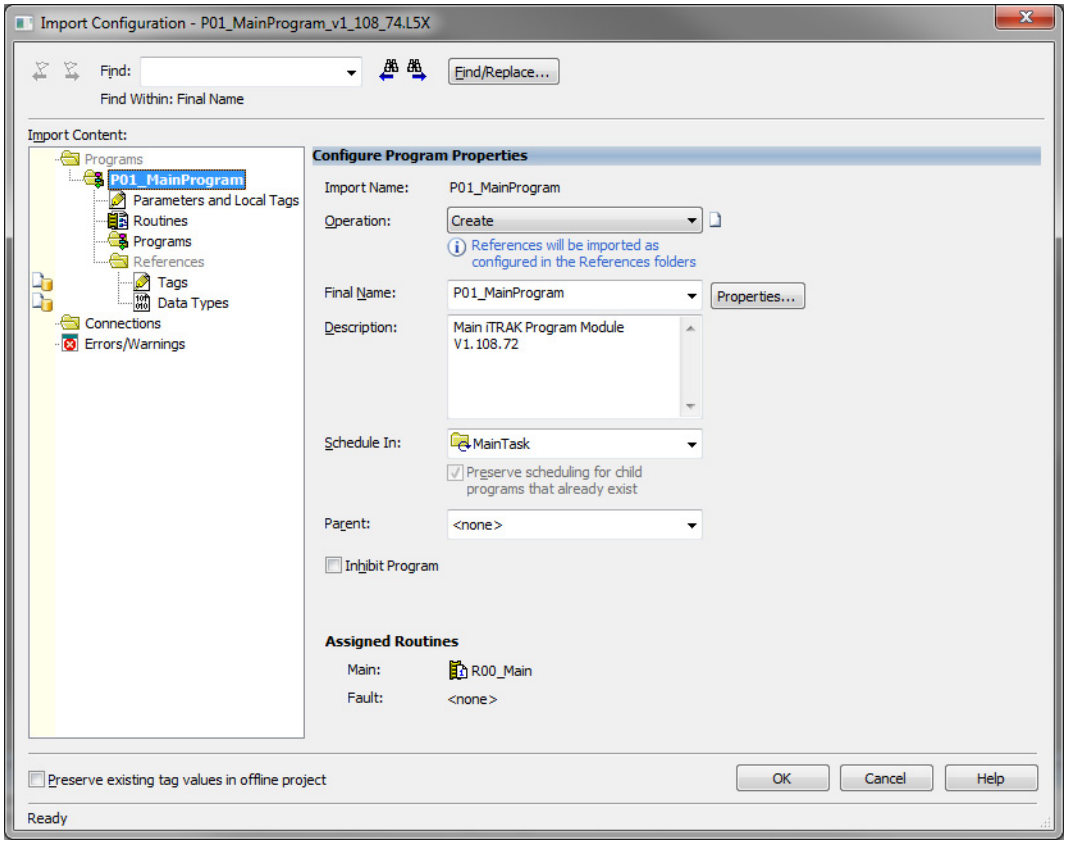


   Logix Designer application version 28 is shown. Other versions can have another path.

3. Select the P01_MainProgram file, click Import.

   The file name P01_MainProgram_*Vmajor_minor_build*.L5X represents the version of the program.

   The Import Configuration window is displayed.



4. Select each of the folders in the Import Content tree and set the parameters that are shown in this table.

   Follow the guidelines that are shown for an Existing iTRAK Project or a New iTRAK Project. Do not change parameters other than the ones listed in the table.

Folder	Parameter	Existing iTRAK Project	New iTRAK Project
P01_MainProgram Configure Program Parameters	Operation	Overwrite	Create
	Schedule In	MainTask	MainTask
Tags	All	Review for errors, overwrite if possible	Use default selection
Data Types	All	Review for errors, overwrite if possible	Use default selection

5. To complete the import process, click OK.

Update an Existing Project File to Use Firmware Revision 1.110

If your existing project is used with firmware revision 1.097 or earlier, contact ICTSupport@ra.rockwell.com for assistance in updating the project.

For version 1.098 and later, follow the steps in the previous sections for adding the programs P01_iTRAK_Communciations and P01_MainProgram to your project file.

IMPORTANT If you have modified the User-Defined Data Types used by the iTRAK system, the modifications do not carry forward when you update. If the modifications must be maintained, consider moving them into separate data structures.

If you have changed the naming conventions used such as Axis_00 instead of Mover_00, you must correlate the imported tags to the existing tags in your system.

Configure the iTRAK System

Whether you use the Starter Projects or configure your own project file, some parameters must be set before motion is commanded.

The configurable settings of the iTRAK system are set in the Data member of the iTRAK_Control tag that is specified in the iTRAK_IO Add-On Instruction. The full list of members in that data structure is specified in Data Types - Appendix A on page 61.

These items must be set before operation.

- Track Configuration settings
  - Track Length
  - Axis Unwind
  - Active Mover Count
  - Number of Zones
  - Zone Start Position [0]
  - Zone Begin Report Offset [0]
  - Zone Begin Command Offset [0]
  - Zone Begin Hold Offset [0]
  - Zone End Report Offset [0]
  - Zone End Command Offset [0]
  - Zone End Hold Offset [0]
  - Zone Length [0]
- Motion Control settings
  - Renumber Movers at Start
  - Motion Polarity
  - Reverse Mover Numbering
  - Position Offset
  - Mover Numbering Offset

- Tuning settings
  - Headway Tolerance
  - Standstill Window for All Motor Modules
  - Standstill Window for Curve Motor Modules
  - Position Error Tolerance
  - Velocity Proportional Bandwidth for All Motor Modules
  - Velocity Integrator Bandwidth for All Motor Modules
  - Position Proportional Bandwidth for All Motor Modules
  - Position Derivative Bandwidth for All Motor Modules
  - Acceleration Feed Forward for All Motor Modules

Detailed information for the values to use for these parameters can be found See UDT_iTRAK_Data on page 66.

Notes:

Motion Commands

The iTRAK® system is designed to feel familiar if you've used Kinetix® servos before, however there are some procedural differences. Use this chapter to learn how to enable the track, command motion, disable the track, and clear faults.

Enable and Start the Track

The best practice is to use the sample code that was delivered with your iTRAK system as an example or start point for developing your code. It shows the sequence of operations clearly and simplifies the bits.

If you are using the sample code, latch the iTRAK_Control.Cmd.iTRAKStart bit. After a few moments, the iTRAK system is enabled and the iTRAK_Control.Status.ReadyForMotion bit goes high, which indicates that motion instructions can be commanded.

If you are developing your own sequence, do the following.

1. Write gains to the iTRAK system.
2. Renumber movers if needed.
3. Latch iTRAK_Control.Cmd.ServoOn.
4. If motion is required for renumbering, perform motion, disable the iTRAK system, and then repeat step 1...step 3.

Command Motion

The iTRAK system is ready for motion commands when the system is not faulted, the gateway is running, the DC bus contactor is closed, and the track is enabled. If you are using the sample code in your program, the program sets iTRAK_Control.Status.ReadyForMotion tag when these conditions are met.

There are three ways to execute motion commands on the iTRAK movers.

- The movers can be commanded using Logix Designer application motion instructions, such as MAM and MAPC. The movers are virtual axes, so not all functions affect them; MSO and MSF are examples of instructions that do not apply to movers.

  TIP If your goal is to jog all axes around the track, the preferred instruction is the MAG instruction. The MAG instruction electronically gears the axes to one master axis. Then use the MAJ on the master axis.

- You can use Rockwell Automation supplied Add-On Instructions (AOIs). These instructions combine standard instructions with extra logic to make sure that movers queue rather than collide, and synchronize to other servo axes without exceeding acceleration or velocity limits. These AOIs are available from your local Rockwell Automation iTRAK application consultant.
- You can develop your own AOIs to accomplish your application-specific tasks.

IMPORTANT If you use Logix Designer application motion instructions and write your own AOIs, it is your responsibility to make sure that command positions do not overlap. The gateway issues a headway fault if command positions overlap.

Stop and Disable the Track

If you are using the sample code, latch the iTRAK_Control.Cmd.iTRAKStop to bring the system to a coordinated stop. This command executes a Motion Group Stop command and then disables the iTRAK system.

TIP The use of the iTRAKStop bit stops all axes in the Motion Group with the same settings. If non-mover axes are in the Motion Group, the use of iTRAKStop bit can cause undesirable effects.

If you are developing your own sequence, you must stop the iTRAK system in a coordinated manner. If a master axis is driving the machine, bring it to a coordinated stop. Once all axes are stopped, latch the iTRAK_Control.Cmd.ServoOff bit.

Clear Faults

If you are using the sample code in your program, the program indicates that a fault has occurred when it sets iTRAK_Control.Status.Faulted bit.

If you are developing your own core code, a fault has occurred if any of the following conditions are true:

- The iTRAK_Control.Status.GatewayFaultCode does not equal 0
- The iTRAK_Control.Status.SectionFaultCode does not equal 0
- The iTRAK_Control.Status.FaultMessageLine1 is not a NullString

On rare occasions, the iTRAK_Control.Status.RenumberMoversBeforeClearingFault tag can be latched. If it is latched, toggle the iTRAK_Control.Cmd.RenumberMovers bit, and wait until the bit is reset.

Make sure that the fault conditions have been cleared, then toggle the iTRAK_Control.Cmd.Fault_Reset bit high to reset faults on the system. Some faults require the gateway to be reset, which would use the iTRAK_Control.Cmd.Reset_Gateway bit instead.

Notes:

Core Functions

The core functions in this chapter can be essential to your operation. Review these functions to see if the features are useful in your application.

Changing IP Address of the Gateway

The iTRAK® system uses a set of six contiguous IP addresses that vary only in the final (fourth) octet, for communication between the controller and the system. Each of these IP addresses is responsible for communicating the information for a group of 16 movers. The IP addresses are assigned to the mover groups in an order that descends in last octet in the IP address. Even if you do not use all six mover groups and I/O modules in your program, six contiguous IP addresses must be available. The IP addresses must not conflict on your network for the iTRAK system to function properly.

EXAMPLE

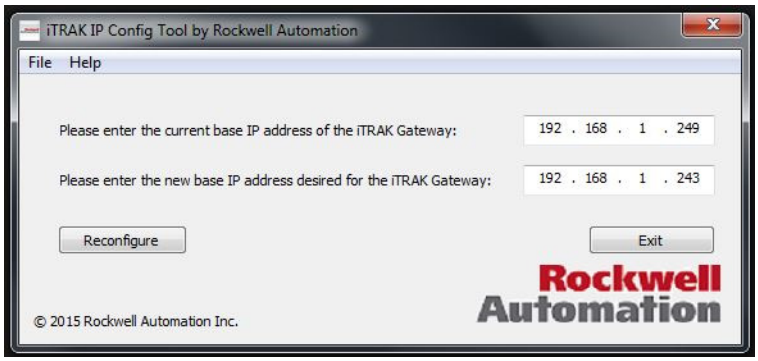
1. Movers 0...15 are assigned to the highest IP address.
2. Movers 16...31 are assigned to the IP address that is one lower than the previous IP address.
3. Continued until all 96 movers are assigned.

By default, the iTRAK system uses the six IP addresses that start with 192.168.1.249, and continue downward to 192.168.1.244. If you would like to change the default set of IP addresses, install and use the available iTRAK IP Configuration and Logs Retriever Tools from Knowledgebase article 898817

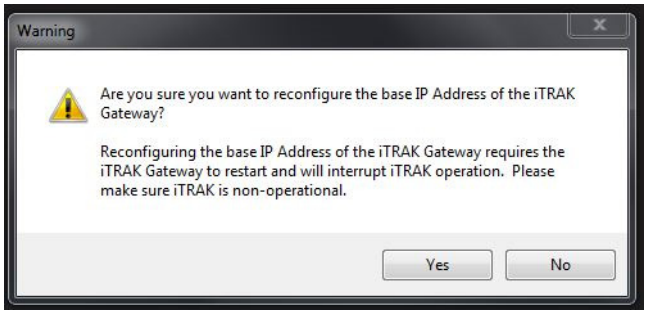
After installing the iTRAK IP Config Tool on your PC, complete these steps to change the IP address set. This example changes the IP addresses from the default set of 192.168.1.249...192.168.1.244 to the new set of 192.168.1.243...192.168.1.238.

1. Verify that there are no IP address conflicts with the current IP addresses or the future IP addresses.
2. Connect your iTRAK system to the network.
3. Connect your PC to the same network as the iTRAK system.
4. To run the iTRAK IP Config Tool, click the desktop icon.

5. Enter the current highest IP address of the iTRAK system, the highest IP address in the new IP address set, and click the Reconfigure button. In the IP address boxes in the window, the cursor will move automatically to the next number field after three digits have been typed.



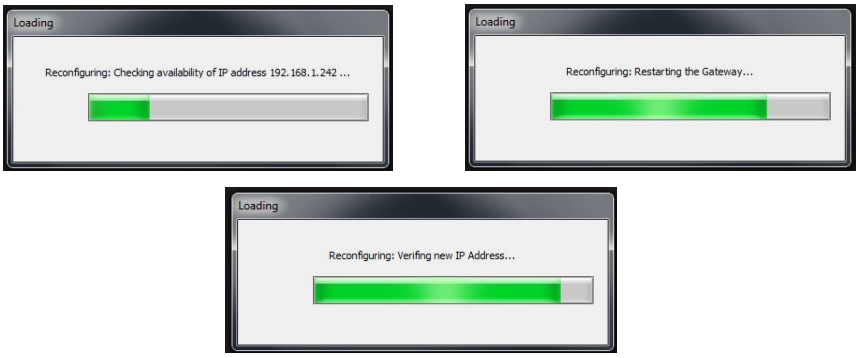
6. The iTRAK IP Config Tool confirms that you want to change the IP address set.



If, after checking your settings, you still want to make the change, click Yes. Otherwise, click No.

The iTRAK IP Config Tool verifies the availability of the six IP addresses in the new set.

7. Restart the iTRAK system, and verify that the new IP addresses are set.

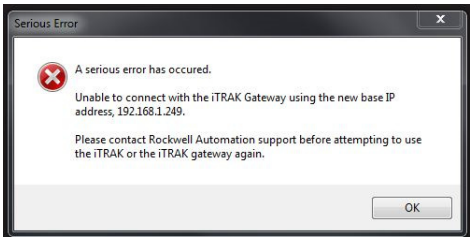


8. On the iTRAK IP Config Tool, click Exit.

   The iTRAK system uses the new IP address set for communications.

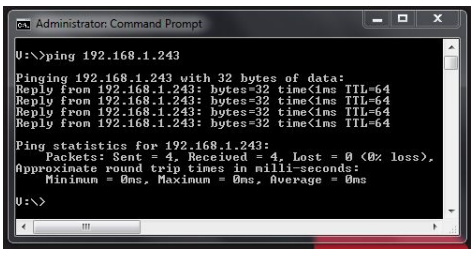
9. Set all I/O modules in your Logix Designer application to their new IP addresses.

IMPORTANT If the iTRAK system takes longer to restart than the iTRAK IP Config Tool expects, the tool generates a false error message. The message indicates that the tool was unable to connect with the iTRAK system and an incorrect base IP address is displayed.



If you receive an error message, do the following.

1. In the dialog box, click OK.
2. Wait for the iTRAK system to finish rebooting.
3. To confirm that the new IP addresses are set properly, query the first IP address in a Command dialog box.



Firmware Revision and Motor Module Type Information

The sample code provides the means of retrieving the Major Revision, Minor Revision, Build Number, and Motor Module Type number for each of the motor modules on the iTRAK system, and the Major Revision, Minor Revision, and Build Number of the gateway. The sample code sends explicit messages between the controller and the gateway.

The program Pxx_MajorMinorBuildRevAndSectionType, found in all sample ACD files, contains the necessary code to obtain this information. If you need this information, we recommended that you import this program, without changing it, into your Logix Designer application. It does not need to be in a high frequency or high priority task.

To retrieve the version information, latch the iTRAK_Version.GetRevBuildNumAndType controller tag. When the application automatically unlatches the tag, the version and type information members are populated. The members are listed here.

Table 3 - iTRAK Version Members

iTRAK_Version Member	Range for [*n*]	Description
iTRAK_Version.GatewayMajorRev	–	Major Revision number of the gateway firmware on the iTRAK system.
iTRAK_Version.GatewayMinorRev	–	Minor Revision number of the gateway firmware on the iTRAK system.
iTRAK_Version.GatewayBuildNum	–	Build number of the gateway firmware on the iTRAK system.
iTRAK_Version.SectionMajorRev[*n*]	0…63	Major Revision number of the firmware on motor module *n* of the iTRAK system.
iTRAK_Version.SectionMinorRev[*n*]		Minor Revision number of the firmware on motor module *n* of the iTRAK system.
iTRAK_Version.SectionBuildNum[*n*]		Build number of the firmware on motor module *n* of the iTRAK system.
iTRAK_Version.SectionFirmwareType[*n*]		Type number of the firmware on motor module *n* of the iTRAK system.

Valid type number and their motor modules are listed here.

Table 4 - iTRAK Motor Module Firmware Type Number

iTRAK Motor Module Firmware Type Number		Description
Force Optimized, Use Only for PCM	**Current Optimized, Use Only for iTRAK Power Supply**	
5	505	Straight motor module, 100 mm magnet length
6	506	Straight motor module, 50 mm magnet length
13	513	Straight motor module,150 mm magnet length
14	4514	Constant-radius curve motor module,150 mm magnet length
25	4525	Reserved for straight motor module,100 mm magnet length
107	4607	Constant-radius curve motor module, 100 mm magnet length
108	4608	Constant-radius curve motor module, 50 mm magnet length
112	4612	Reserved for curve motor module, 100 mm magnet length
113	4613	Reserved for curve motor module, 150 mm magnet length

Table 4 - iTRAK Motor Module Firmware Type Number (Continued)

iTRAK Motor Module Firmware Type Number		Description
Force Optimized, Use Only for PCM	**Current Optimized, Use Only for iTRAK Power Supply**	
122	4622	Reserved for curve motor module, 100 mm magnet length
212	4712	Reserved for curve motor module 100 mm magnet length
213	4713	Reserved for curve motor module 150 mm magnet length
222	4722	Reserved for curve motor module 100 mm magnet length

Set Zero Position

The iTRAK System User Manual, publication 2198T-UM001, has detailed information on the wiring of motor modules. The physical wiring of the motor modules to the gateway determines the initial zero position of the iTRAK system. It can be redefined virtually by using configuration settings in the iTRAK_Control.Data tags.

The zero position of the system is the most clockwise position of motor module zero. It can be shifted to align to the motion commands and processes being executed around the track. When examining the position sensing surface of the motor modules, the zero position is always shifted counterclockwise for positive values and clockwise for negative values. The motion polarity adjustment does not impact zero position adjustment. The virtual zero position is not used during renumbering; only the hardwired zero position is used during renumbering.

Table 5 - Define Set Zero Position

Set Zero Position	Tag	Range	Procedure
Virtually adjust the Zero Position of the track.	iTRAK_Control.Data.PositionOffset	– Track Length…Track Length	1. Disable the track 2. Put the controller either off-line or in Program mode. 3. Set the iTRAK_Control.Data.PositionOffset to the adjustment required to change the zero position of the track relative to the hard-wired zero position. Maintain the value through a First Scan condition.

Homing and Renumbering

Each mover is identified in Logix Designer application with a unique virtual axis name and those axes are correlated to the physical track with the iTRAK_IO Add-On Instruction. Some applications do not need to maintain correlation between physical movers and virtual axes through power cycles. Some applications do require the unique identification of each mover or identification of paired movers, such as for left and right tooling.

Motion Polarity

The positive direction of motion of the iTRAK system is initially defined as counter-clockwise when viewing the indicator lights on the top of the motor modules. It can be redefined by using configuration settings in the iTRAK_Control.Data tags

Table 6 - Set Value for Motion Polarity.

Set Positive Direction	Tag	Value	Procedure
Counter-clockwise	iTRAK_Control.Data.MotionPolarity	0	1. Disable the track. 2. Put the controller either off-line or in Program mode. 3. Set iTRAK_Control.Data.MotionPolarity to 0. Maintain the value through a First Scan condition.
Clockwise		1	1. Disable the track 2. Put the controller either off-line or in Program mode. 3. Set iTRAK_Control.Data.MotionPolarity to 1. Maintain the value through a First Scan condition.

Numbering Direction

During renumbering, movers are numbered such that the mover that is at the highest position on the track becomes the first mover (Mover 0), the mover at the second highest position on the track becomes the second mover, and so on. This process is called Forward Numbering.

Optionally, you can set the mover number order such that the mover at the lowest position on the track becomes the first mover, the mover at the second lowest position on the track becomes the second mover, and so on. This process is called Reverse Numbering.

Table 7 - Set Value for Numbering Direction

Numbering Order	Tag	Value	Procedure
Forward Numbering	iTRAK_Control.Data.ReverseMoverNumbering	0	1. Disable the track 2. Put the controller either off-line or in Program mode. 3. Set iTRAK_Control.Data.ReverseMoverNumbering to 0. Maintain the value through a First Scan condition.
Reverse Numbering		1	1. Disable the track 2. Put the controller either off-line or in Program mode. Set iTRAK_Control.Data.ReverseMoverNumbering to 1. Maintain the value through a First Scan condition.

Numbering Offset

If you know which physical mover is in the position that makes it Mover 0 during renumbering, you can apply an offset so the correct mover number is assigned. There are two tags that can be used.

Table 8 - Set Value for Numbering Offset

Desired Action	Tag	Value	Procedure
Clockwise adjustment set during power-up	iTRAK_Control.Data. MoverNumberingOffset	0…(number of movers-1)	1. Disable the track 2. Put the controller either off-line or in Program mode. 3. Set the iTRAK_Control.Data.MoverNumberingOffset to the number of movers between the zero position and the desired Mover 0 the opposite direction of numbering. Maintain the value through a First Scan condition.
Counterclockwise Adjustment Set During Each Renumbering	iTRAK_Control.Data.RenumberingMoverStartNumber		1. Disable the track 2. To shift in the same direction of numbering, set the iTRAK_Control.Data.MoverNumberingOffset to the number of movers Maintain the value through the renumbering process.

Applying Renumbering

The four tags that are mentioned in this section can be combined in many ways. Review the examples to determine how to use them in your application.

Renumbering Process Description

Renumbering is the process that the gateway uses to assign individual movers to a memory location. Individual mover data is stored in an array and the mover number defines the array element. These are important points about renumbering:

- All movers are identical, but must be uniquely identified.
- Similar to homing, but the process does not inherently guarantee that the movers have the same number.
- It can take up to two Coarse Update Periods to complete.
- Gateway must be running, but the track cannot be enabled.

If any of the following conditions occur, renumbering is required:

- Initial Commissioning.
- RenumberMoverBeforeClearingFault status bit is high.
  - Fault 11 with subcode other than 7.
  - Fault 12.
- Movers were manually moved more than 300 mm (11.8 in.) when power is off.

If the following conditions occur, renumbering occurs automatically:

- Movers were manually moved 300 mm (12 in.) when power is off.
- Power was removed from the system while the track was enabled.
- Power was removed from the system within 10 seconds of a disable command being sent to the gateway.

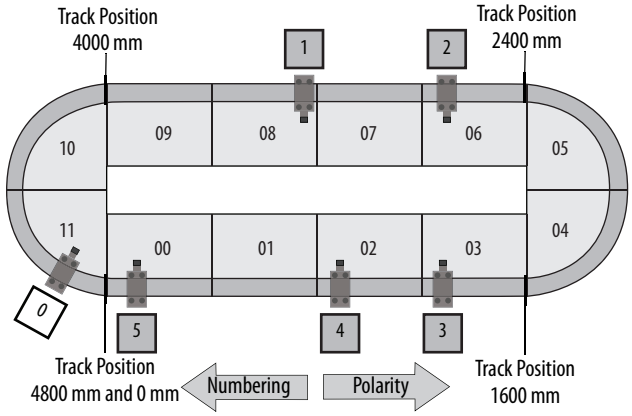
Renumbering Examples

This section contains several examples of how to configure the renumbering of your iTRAK system.

In each example, if MoverNumberingOffset or RenumberingMoverStartNumber are not applied, different shading is shown on the mover that would be identified if as Mover 0.

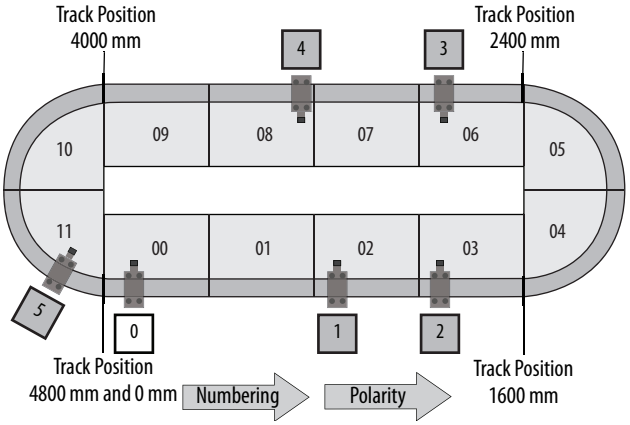
Example 1 - Default Configuration

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



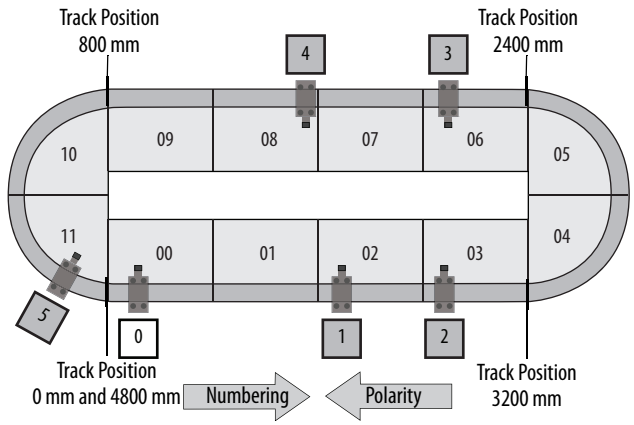
Example 2 - Reverse Numbering

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



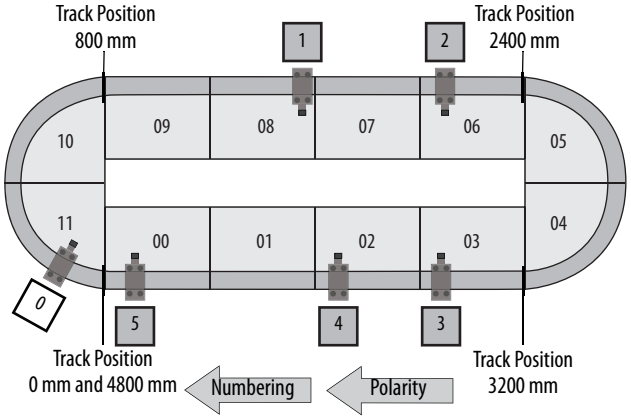
Example 3 - Polarity

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



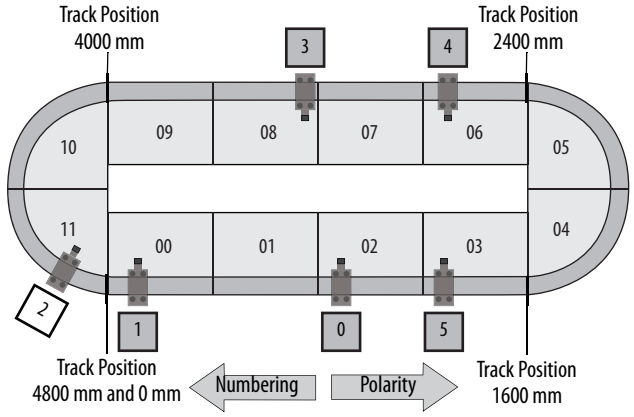
Example 4 - Polarity and Reverse Numbering

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



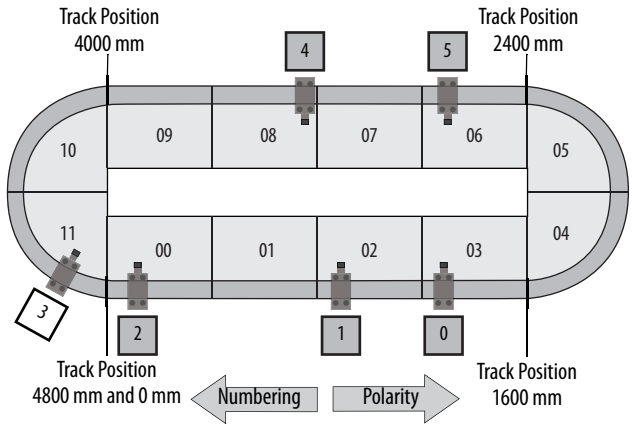
Example 5 – Start Number

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	2



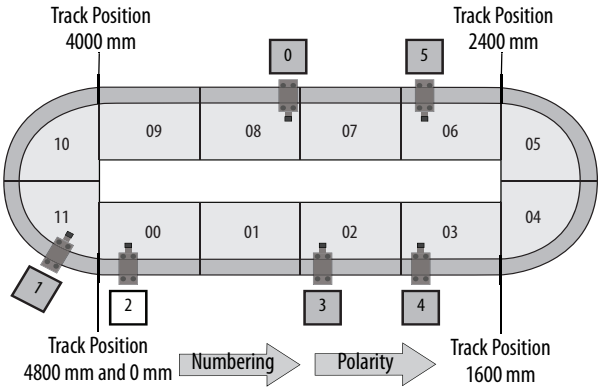
Example 5a – Start Number with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	3



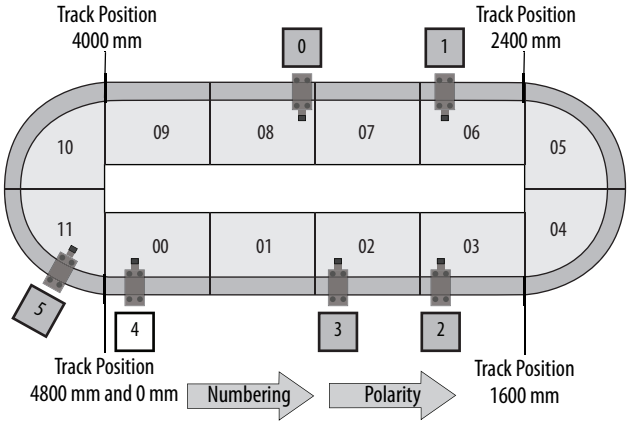
Example 6 - Reverse Numbering and Start Number

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	2



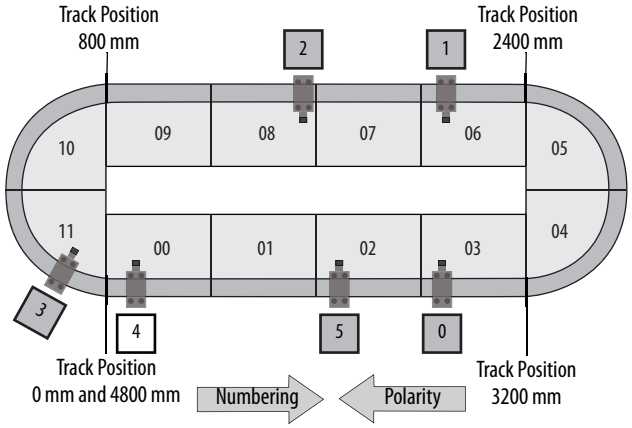
Example 6a - Reverse Numbering and Start Number with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	4



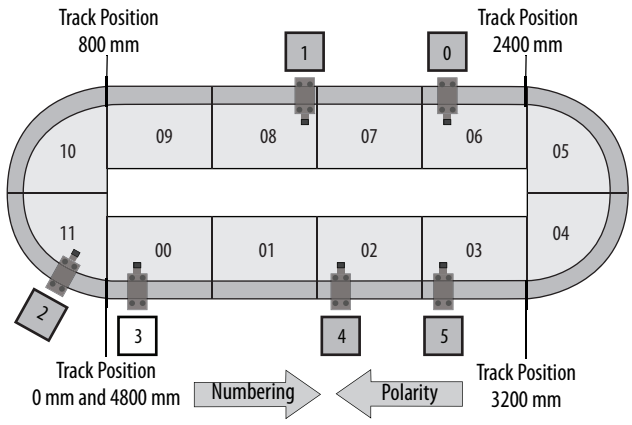
Example 7– Polarity and Start Number

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	4



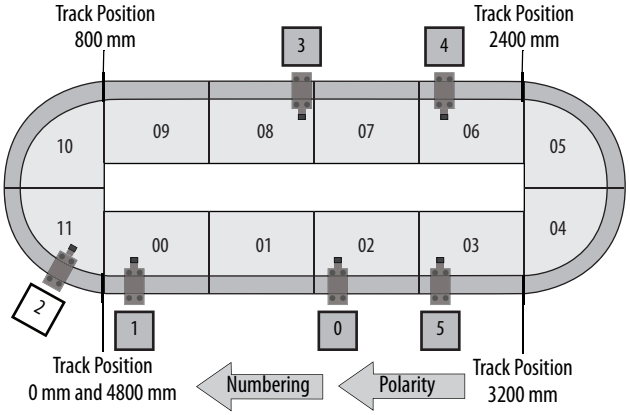
Example 7a – Polarity and Start Number with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	3



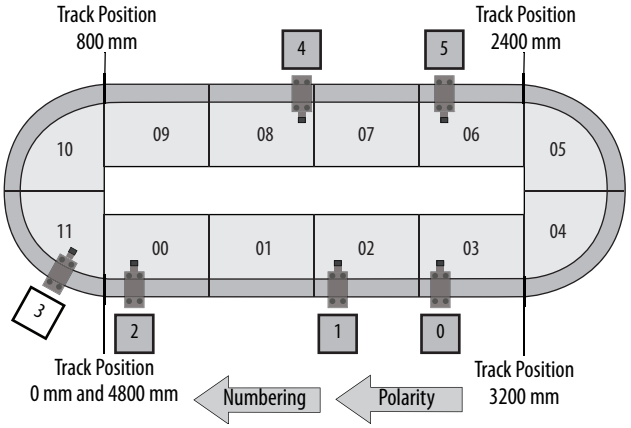
Example 8 – Polarity, Reverse Numbering, and Start Number

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	2



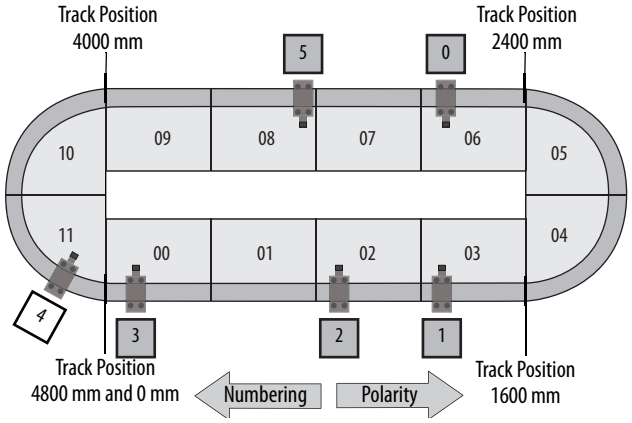
Example 8a – Polarity, Reverse Numbering, and Start Number with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	0
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	3



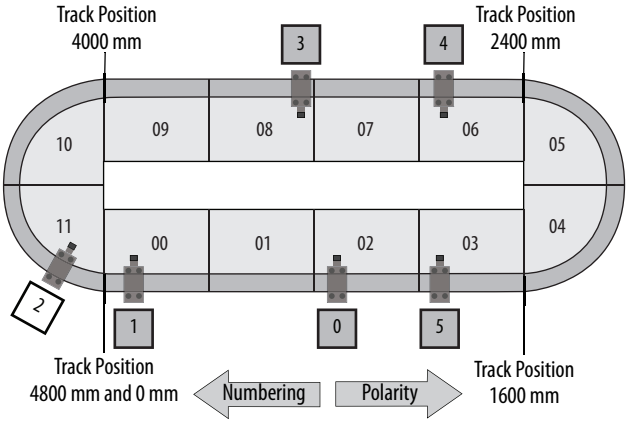
Example 9 – Offset

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	2
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



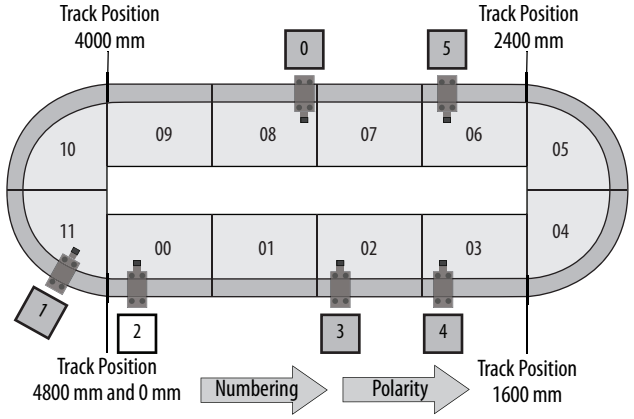
Example 9a – Offset with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	4
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



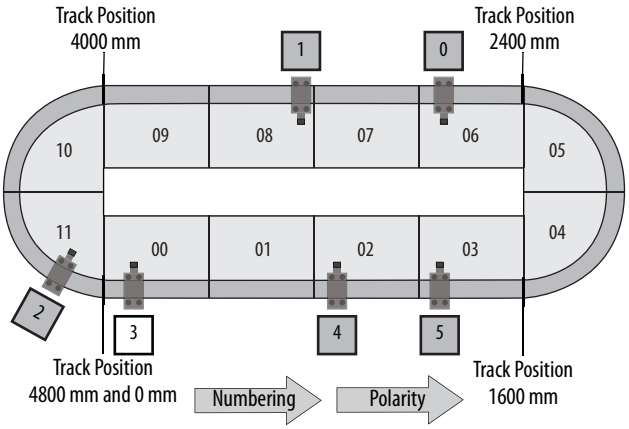
Example 10 – Reverse Numbering and Offset

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	2
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



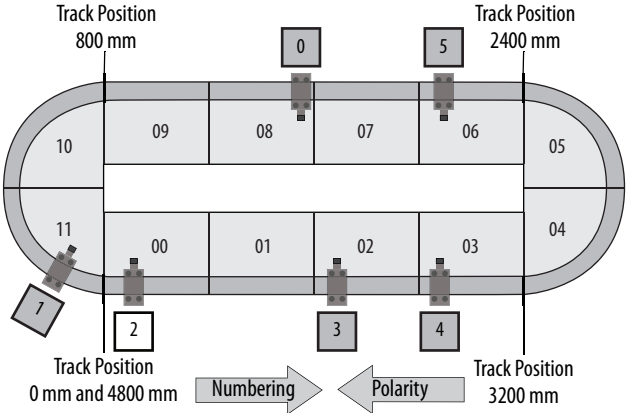
Example 10a – Reverse Numbering and Offset with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	3
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



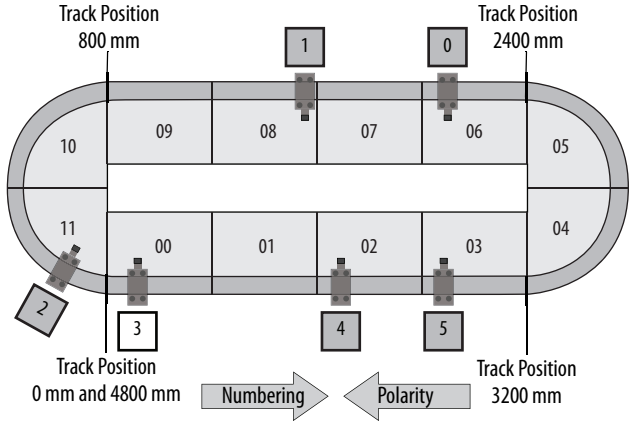
Example 11 – Polarity and Offset

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	2
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



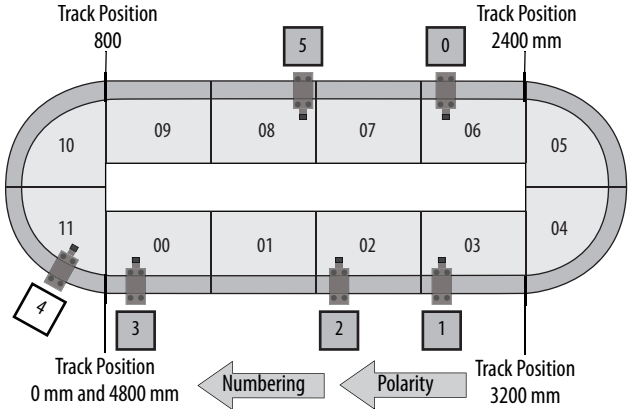
Example 11a – Polarity and Offset with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	3
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



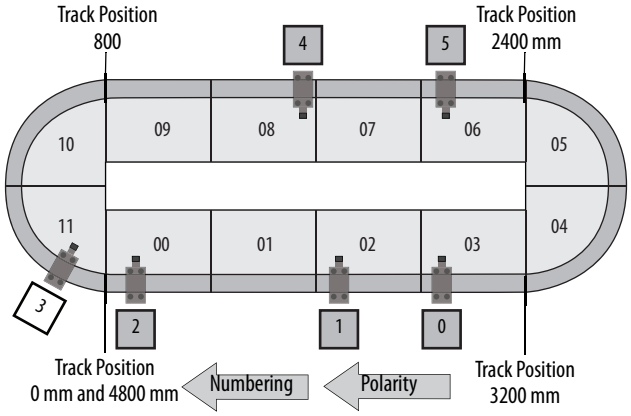
Example 12 - Polarity, Reverse Numbering, and Offset

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	2
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



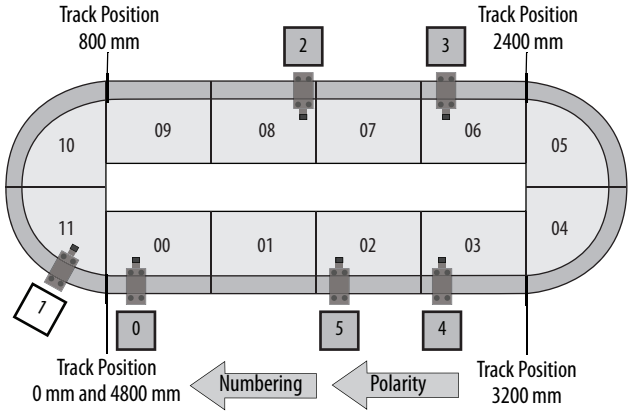
Example 12a - Polarity, Reverse Numbering, and Offset with Different Values

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	3
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	0



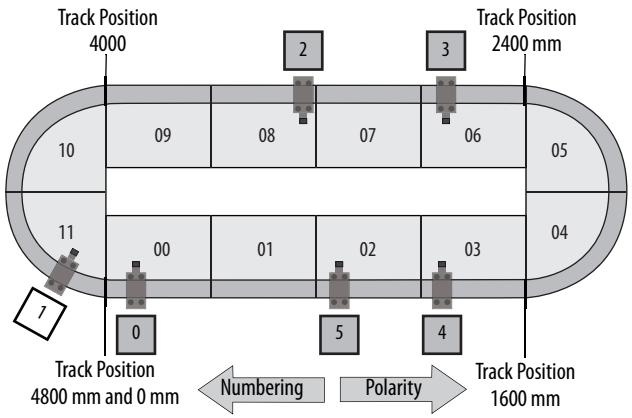
Example 13 - Reverse Numbering, Offset, and Start Number

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	1
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	1
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	3
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	4



Example 14 - Offset and Starting Number

Steps	iTRAK_Control.Data.	Range	Value
1. Identify Track 0 Position	–	–	–
2. Identify Forward Direction	MotionPolarity	0 = CCW 1 = CW	0
3. Identify Numbering Direction	ReverseMoverNumbering	0 = high to low 1 = low to high	0
4. Identify Numbering Offset (Increment against numbering direction.)	MoverNumberingOffset	0…(number of movers - 1)	3
5. Identify Numbering Starting Number (Increment along numbering direction.)	RenumberingMoverStartNumber	0…(number of movers - 1)	4



Trending

The iTRAK system provides the Actual Position, Position Error, and up to one other value for trending in Logix Designer application. The other value can be selected from the following list.

- Commanded current from the velocity loop (mA) per mover
- Voltage of the common rail (mV) per motor module
- Voltage of the high rail (mV) per motor module
- Actual velocity (mm/s) per mover
- Velocity error (mm/s) per mover
- Negative output current (mA) per motor module
- Positive output current (mA) per motor module
- Difference between gateway and sensor reported mover position (µm) per mover
- Commanded current of coil number 11 (mA) per motor module
- Internal temperature (°C x 512) per motor module
- Mover position scaling coefficient at the start of the motor module per motor module.
- Mover position scaling coefficient at motor module end per motor module

The preferred mechanism for selecting information to trend is by using the Util_Trace_RD_*xxxxx* controller tags that are found in P01_MainProgram-> R05_ParameterExchange. Selections can be made or changed at any time.

The data that is returned is synchronized within the track (mover to mover). However, a delay of up to three Coarse Update Periods can be seen when trended on the same plot with Actual Position data from Kinetix® servo drives.

Headway Checking in the Controller

The allowable space between movers is called headway. If the headway between movers gets below zero, a collision between movers can occur. You can configure the length of your tooling, or any additional space you want around your movers to make sure no collisions occur, in the iTRAK_Control.Data.HeadwayTolerance tag. Headway checking is performed in two places: the gateway by default, and optionally in the controller.

Headway checking in the gateway can only be disabled by setting iTRAK_Control.Data.HeadwayTolerance to zero.

Headway checking in the controller must be explicitly enabled with the iTRAK_Control.Cmd.DoLogixHeadwayPrecheck tag. It can be useful when troubleshooting motion profiles before the track is physically assembled or debugging code while commissioning. It executes in program P01_iTRAK_Communications in the routine R01_Headway. The program puts a burden on the controller in a high-speed task, so it must be inhibited when not troubleshooting.

iTRAK Power Supply

The Kinetix 5700 iTRAK power supply requires logic in the controller to configure the iTRAK system and monitor the power supply. There is sample logic included with each of the Logix Designer application starter projects in the Pxx_IPS_Support program in the HighSpeedTask task. The Logix Designer application can also be downloaded from Knowledgebase article 778917. The application performs the following necessary functions:

- Sends the power supply type, either Power Control Module or iTRAK power supply to the gateway.
- Sends DFE.DCBusUpStatus to the gateway. The bus up status is used by the gateway to control when the iTRAK power supply produces a DC bus.
- Monitors DFE.DCBusUnload in the sample application to perform a controlled stop in the event of a fault.

The Logix Designer application also provides examples for disabling the iTRAK power supply via the gateway. The axis associated to the Kinetix 5700 DC-power supply must be named DFE or those tags need to be updated to match the name given to the Kinetix 5700 DC-power supply.

To use the Kinetix 5700 iTRAK power supply with your system, you need to configure the correct commutation table. Each magnet plate length has its own commutation table and there are separate tables for use with the iTRAK power supply and the Power Control Module.

To use the appropriate tables for a Kinetix 5700 iTRAK power supply, do the following.

1. Set iTRAK_Control.Data.FlashSectionsCommutationType to 0 for optimal current tables.
2. Flash the modules using the appropriate iTRAK_Control.Cmd.FlashSections*xx*mm command.
3. Verify that the correct commutation tables were used.

   Read the Section Type from the modules.

Power Control Module

If you are using the Power Control Module with your system, you need to configure the correct commutation table. Each magnet plate length has its own commutation table and there are separate tables for use with the iTRAK power supply and the Power Control Module.

To use the appropriate tables for a Power Control Module, do the following.

1. Set iTRAK_Control.Data.FlashSectionsCommutationType to 1 for optimal Kf tables.
2. Flash the modules using the appropriate iTRAK_Control.Cmd.FlashSections*xx*mm command.
3. Verify that the correct commutation tables were used.

   Read the Section Type from the modules.

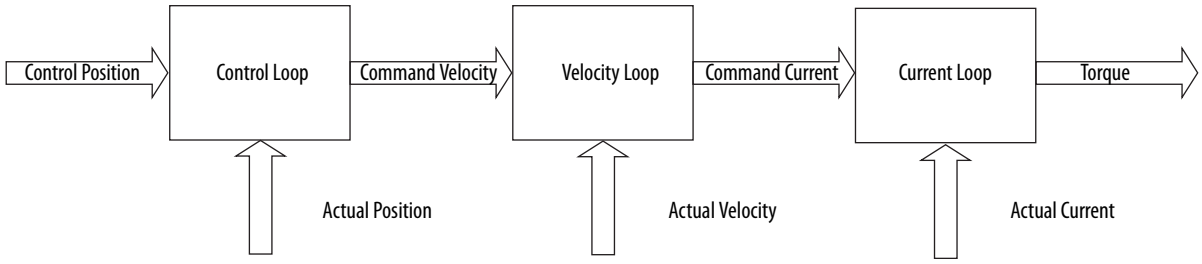
Systems that were commissioned using previous firmware revisions maintain the existing commutation table through a firmware flash to revision 1.110. The commutation tables are only updated when the FlashSections*xx*mm command is used; however, the operational firmware of the motor modules are updated when the gateway is updated.

Tuning

This chapter describes how to tune the iTRAK® system for optimizing the performance of your system.

Control Loop Diagram

The control loop consists of a three-stage cascaded loop.



The position loop gains and velocity loop gains can be adjusted to optimize them for your system. The current loop gains are not user-accessible.

Download Gains to the iTRAK System

The iTRAK supports three separate sets of gains. Motor module gains for curves or for the entire track can be set individually. Tracks typically have separate gains for curved and straight motor modules, however it is uncommon to have individual motor modules gains. Use the guidance of a Rockwell Automation specialist to implement individual tuning of motor modules.

Table 9 - Gain Tags

Gain Type	iTRAK_Control.Cmd Tag	Action/Result	Associated Gain Tags from iTRAK_Control.Data
Section	SetGainsSection	When used with Data.SetGainsSectionNumber, it writes a set of gains to an individual motor module.	GainsSectionPositionBandwidth GainsSectionPositionIntegratorBandwidth (1) GainsSectionPositionDerivative GainsSectionVelocityBandwidth GainsSectionVelocityIntegratorBandwidth
Curves	SetGainsCurves	When used, it writes gains to all curved motor modules, including any individual curved motor modules that were written to previously.	CurveStandstillWindow GainsCurvePositionBandwidth GainsCurvePositionIntegratorBandwidth (1) GainsCurvePositionDerivative GainsCurveVelocityBandwidth GainsCurveVelocityIntegratorBandwidth iTRAK_CurvesVelocity_FF (2)
All	SetGainsAll	When used, it writes gains to all motor modules, including any individual straight motor module or curved motor module that were written to previously.	StandstillWindow PositionerrorTolerance HeadwayTolerance GainsAllPositionBandwidth GainsAllPositionIntegratorBandwidth (1) GainsAllPositionDerivative GainsAllVelocityBandwidth GainsAllVelocityIntegratorBandwidth iTRAK_AllVelocity_FF (2) GainsAllAccelerationFeedForward GainsAllGravityMagnitudeKg GainsAllGravityZeroLocation

(1) These tags are not part of iTRAK_Control UDT; they are from the controller scoped tag iTRAK_Position_Integrator

(2) These tags are controller scoped tags.

IMPORTANT Change gains by using the SetGainsAll followed by SetGainsCurves. If motor module gains are used, SetGainsSection must be used last in the sequence. Failure to follow the order can result in incorrect gains being applied.

System Tuning Procedure

This section defines a procedure for how to modify tuning gains to achieve desired performance. It is written generically so that once executed on all straight motor modules, it can be iterated for all curve motor modules, or individual motor modules.

Overview

There are five sets of procedures that are executed in series to tune the straight motor modules in your system. Iterate these procedures to tune the curved motor modules after the straight motor modules have been tuned. The procedures are detailed in the following sections.

- Determine Goals and Set Initial Gains
- Develop Application Code and Create Trends
- Tune Velocity Loop
- Tune Position Loop
- Apply Acceleration Feedforward

In addition, there are advanced tuning functionality options such as gravity compensation, position integrator hold, and the use of dead-zones. These functions are detailed after general tuning; however, it can save time to review those features and implement them before starting the tuning procedure.

Determine Goals and Set Initial Gains

Your application has specific requirements that are critical to the quality of the operation. Know your requirements before starting the tuning process. You can save time in the tuning process if you clear goals. Examples include position error while moving, velocity error while tracking, and stop repeatability.

TIP It can be helpful to calibrate each mover to a mechanical datum and enter a corrective offset in the iTRAK_Control.Data.MoverOffset tag for each mover in the system. The use of corrective offset compensates for mechanical tolerance stack-up in the mover and can improve the performance of the system.

The default values for system tuning are set in the Logix Designer application sample. The values are appropriate for simple commissioning. When you are ready to start tuning the system, set the gains in the table. This table also includes conventions for abbreviating the name of each gain when it is used in the flow charts.

Table 10 - Gain Tag Defaults and Suggested Values

Tag	Abbreviated Name	Default Value	Suggested Initial Value
iTRAK_Control.Data.GainsAllPositionBandwidth	K_{pp}	75	20
iTRAK_Position_Integrator.GainsAllPositionIntegratorBandwidth	K_{pi}	0	3
iTRAK_Control.Data.GainsAllPositionDerivative	K_{pd}	2	3
iTRAK_Control.Data.GainsAllVelocityBandwidth	K_{vp}	10	5
iTRAK_Control.Data.GainsAllVelocityIntegratorBandwidth	K_{vi}	20	0
iTRAK_Control.Data.GainsAllAccelerationFeedforward	K_{aff}	50	0

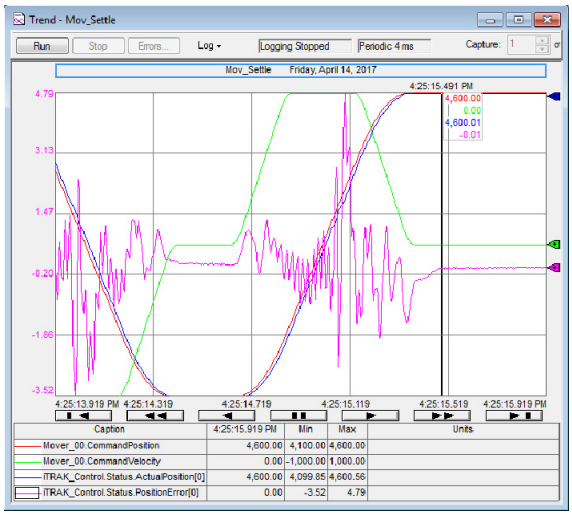
Develop Logix Designer Application and Create Trends

When you are tuning the system, it is helpful to have Logix Designer application to make the mover perform critical moves and a method to monitor the success of the move. The iTRAK system is not included in Motion System Tuning Application Techniques, publication MOTION-AT005, but the appendices contain useful examples for the types of motion commands, application logic, and trends.

Provide a simple bidirectional move in your Logix Designer application that can be repeated easily. Have the application approximate what the mover does in the actual application and exercise the system to test the requirements that are critical to the quality of the operation. The application code can include MAM, MAPC, and MATC instructions. MAJ instructions are not recommended because they continue to run after they are started, which can lead to collisions without careful planning of the code.

Trends are useful to monitor mover information. A typical trend includes the Command Position, Command Velocity, Actual Position, Position Error, and Current or Torque. If you choose to monitor any variables beyond Actual Position and Position Error, you must enable the trends for those variables.

Figure 7 - Typical Trend



TIP After each change in tuning values, use Cmd.SetGainsAll to apply gains to the motor modules

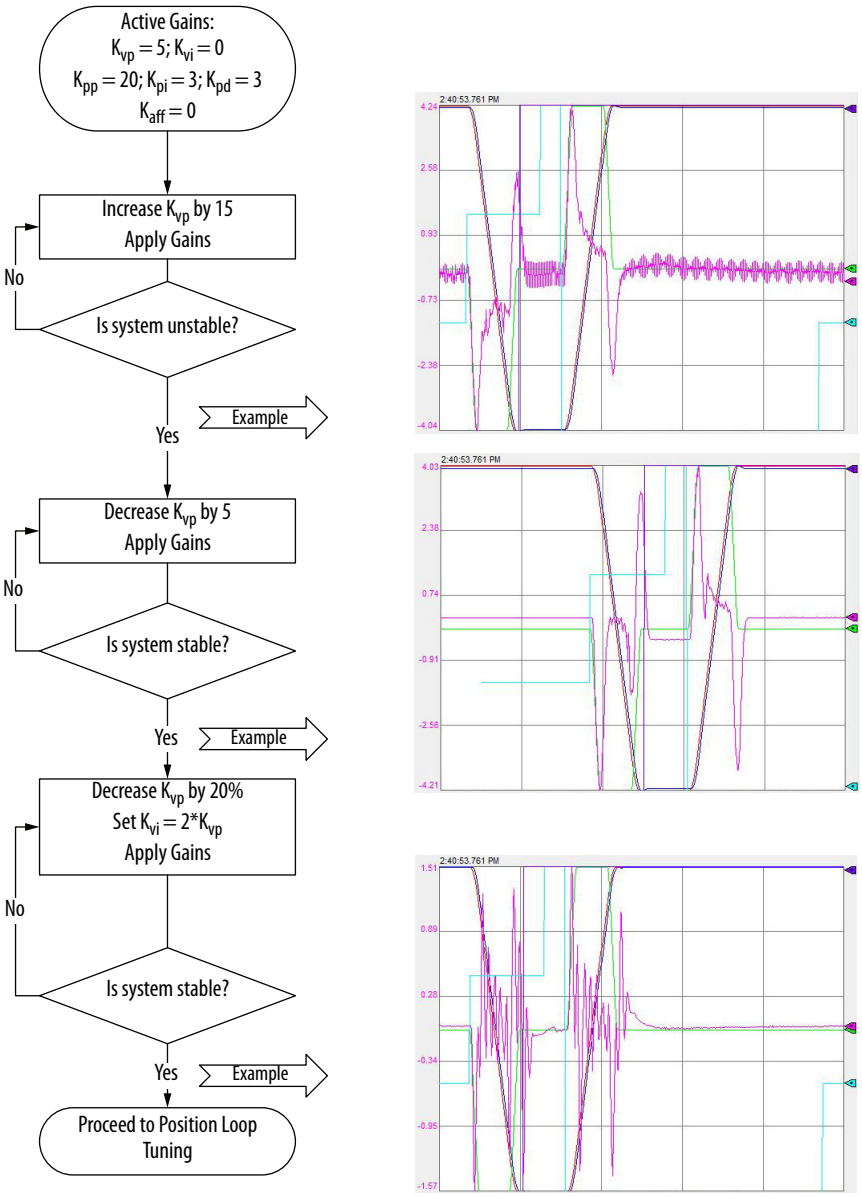
IMPORTANT After tuning the straight motor modules, verify that the tuning works on the curve motor modules. If instability exists on the curves, repeat the process with the curve gains.

Tune Velocity Loop

The velocity loop is the first loop to tune. Do the following or use the flowchart in Figure 8.

1. Increase K_{vp} by 15; repeat until the system becomes unstable.
2. Decrease K_{vp} by 5; repeat until system becomes stable.
3. Decrease K_{vp} by 20% and set $K_{vi} = 2^* K_{vp}$; repeat until system is stable.

Figure 8 - Velocity Loop Tuning



Tune the Position Loop

After the velocity loop is tuned, tune the position loop. Do the following or use the flowchart in Figure 9 and Figure 10.

1. Increase K_{pp} by 30; repeat until system becomes unstable.
2. Decrease K_{pp} by 10; repeat until system becomes stable.
3. Decrease K_{pp} by 20%.
4. Increase K_{pd} by 1; repeat until the time to settle decreases, system becomes unstable, or $K_{pd} = 5$
5. Decrease K_{pd} by 1; repeat until the time to settle is decreased or constant, or $K_{pd} = 3$
6. Increase K_{pi} by 1; repeat until the time to settle increases or $K_{pi} = 5$
7. Decrease K_{pi} by 1.

Figure 9 - Tune the Position Loop

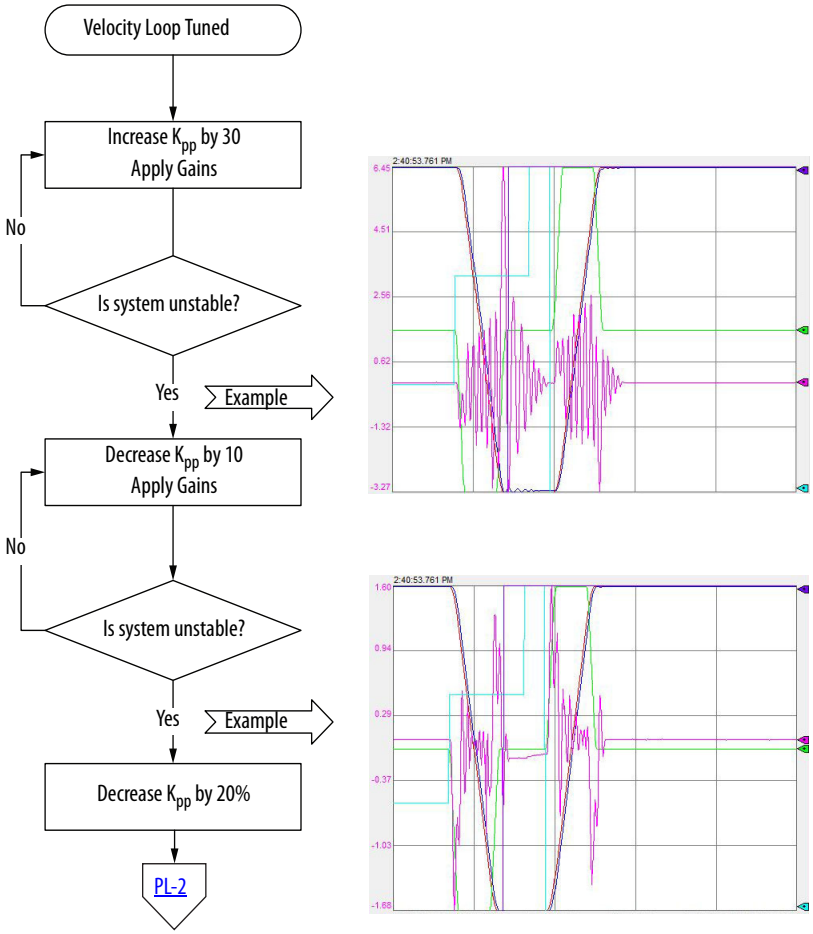
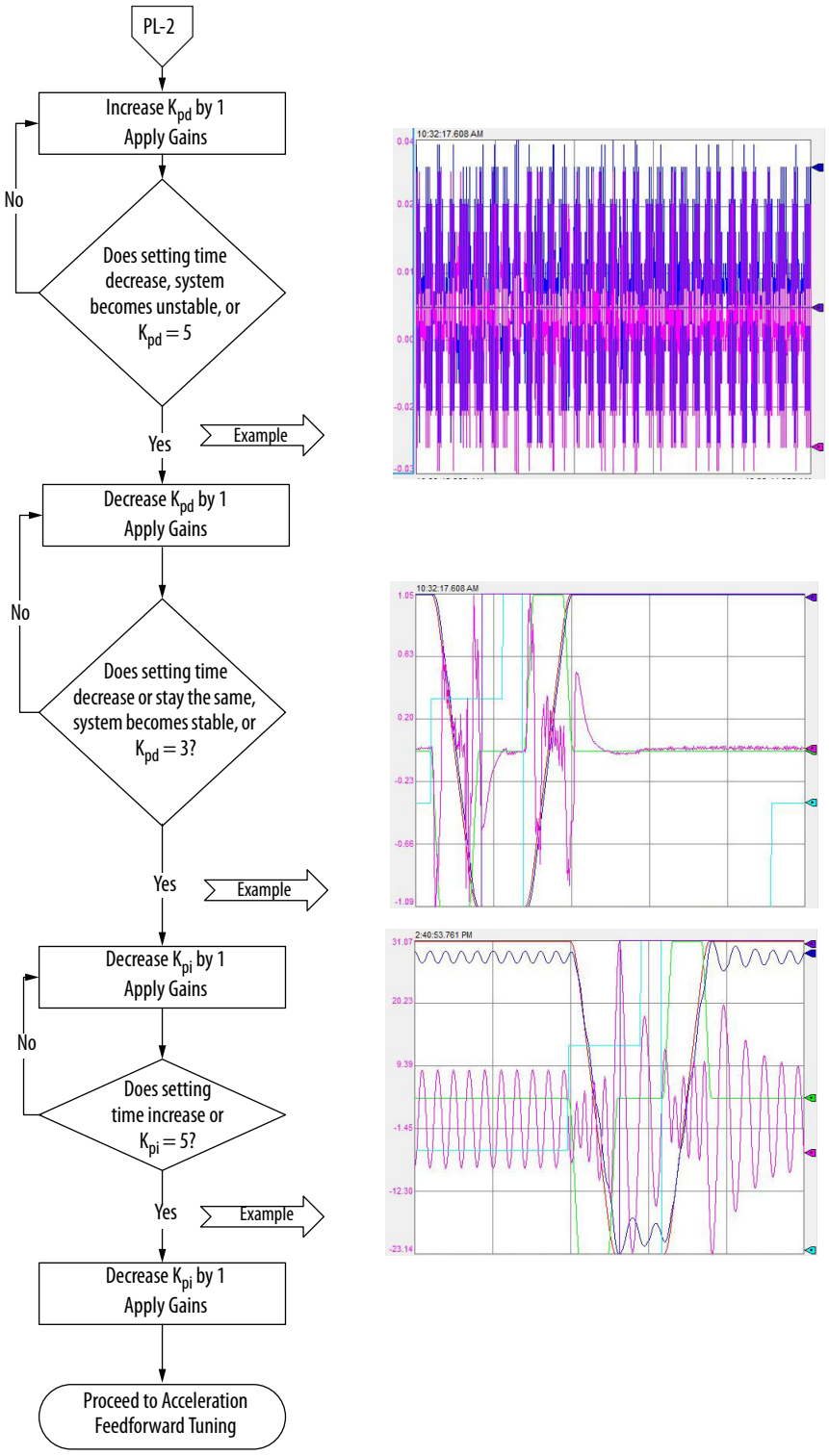


Figure 10 - Tune the Position Loop (Continued)

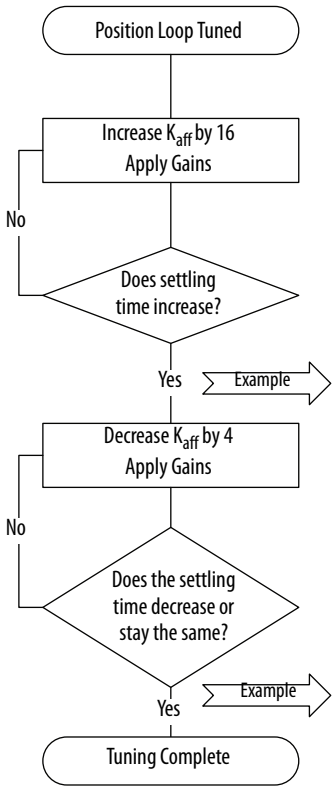


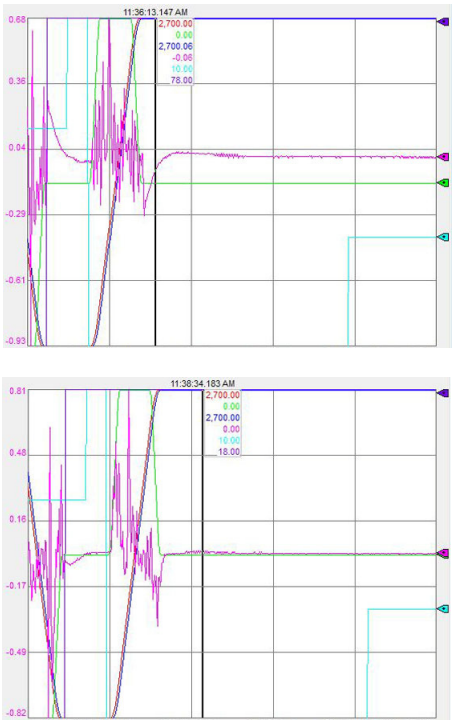
Apply Acceleration Feedforward

After the velocity loop and position loop are tuned, tune the acceleration feedforward. To tune the acceleration feedforward, do the following or use the flowchart in Figure 1.

1. Increase K_{aff} by 16; repeat until the time to settle increases.
2. Decrease K_{aff} by 4; repeat until the time to settle decreases or is constant

Figure 11 - Acceleration Feedforward





Position Integrator Hold

When you are using the position integrator (K_{pi}), it is beneficial to implement an Integrator Hold. When Position Integrator Hold is enabled, it pauses the windup of the integrator until the command velocity is zero, enables higher proportional gains during the dynamic portions of the command profile, and then only executes the integrator to close final position error.

To enable the Integrator Hold, use the sample program Pxx_Position_Integrator.

Command Dead-zones

You can implement a dead-zone around the command position to reduce oscillation when the mover is close enough to the command position. Implement dead-zones separately for straights and curves by using the StandstillWindow and CurveStandstillWindow respectively. When the mover is within the specified standstill window, there is no servo action on the mover.

Figure 12 - Standstill Window with No Servo Action

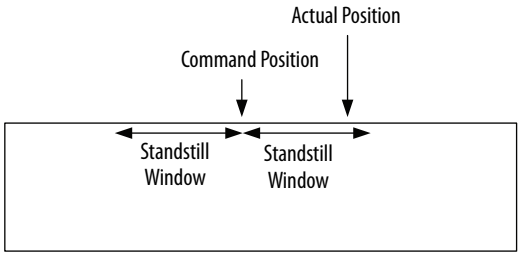
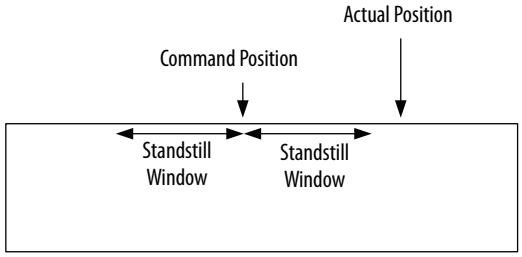


Figure 13 - Standstill Window with Servo Action



Do not use on applications with external forces such as gravity or clamping because it causes the mover to oscillate continuously in and out of the dead-zone. You can disable the dead-zone by entering a value of zero into the appropriate tag.

Gravity Compensation

Gravity is not a concern for horizontal iTRAK systems, as gravity is exerting the same downward force perpendicular to the direction of motion continuously, no matter where the movers are on the track. However, with vertical and stand-up systems, this condition is not the case. Whenever a mover is traveling upwards on the track, the force of gravity is impeding the motion of the mover and requires the track to exert more force to maintain the motion profile. Conversely, the track requires less force on a mover that is traveling downwards on the track. As with horizontal tracks, horizontal moves on a vertical or stand-up track require no force adjustment.

Movers experience a change in the effect of gravitational forces when they traverse curves on non-horizontal tracks. Each time the mover goes through a curve, the impact of gravity will change. As the loads on movers get heavier, the effect of the gravitational forces becomes greater, to the point where the standard servo control loops are not be able to provide adequate performance without additional compensation. An automatic Gravity Feedforward Compensation feature was added the iTRAK system.

To make the appropriate adjustment, the internally calculated feedforward value is then added into the velocity regulator commanded current output.

Gravity feedforward compensation feature is described as follows.

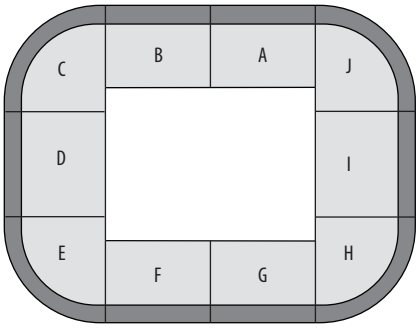
The gravity feedforward compensation code needs three inputs:

- The mass of the mover and its tooling determines the magnitude of the compensation.
- The mover magnet length, which is automatically pulled from the motor module type, is used to compensate for gravitational change on the curves.
- The location of the hardwired Section Zero, relative to the track orientation, determines which curves are transitions from flat to vertical surfaces.

To use the gravity feedforward compensation feature of the iTRAK system, two controller tags must be set, to set the tags do the following.

1. TRAK_Control.Data.GainsAllGravityMagnitudeKg is the total mass of the mover and its tools in kilograms. The maximum value is 100 kilograms. When the value is set to zero, gravity compensation is disabled.
2. Set the iTRAK_Control.Data.GainsAllGravityZeroLocation tag to indicate the location of the hard-wired actual zero position regarding the track orientation and gravity. Determine the location by viewing your track from the position sensor side. Set the tag to one of the four values that are illustrated here.

Figure 14 - Gravity Compensation Motor Modules



Location of Hardwired Zero	Tag Value	Motor Modules in Diagram
Top	1	A, B, C
Left	2	D, E
Bottom	3	F, G, H
Right	4	I, J

Notes:

Data Types

The iTRAK® system requires specific information from the controller LOGIX 5000™. That information is organized into User-defined Data Types (UDTs), which are included in the Starter Projects and the AOI definitions.

UDT_iTRAK_Control

The iTRAK_IO Add-On Instruction has an In-Out Reference parameter that is called iTRAK_Control, which has the data type UDT_iTRAK_Control. UDT_iTRAK_Control has four members as shown here.

Table 11 - UDT_iTRAK_Control Members

Tag Name	Data Type
Cmd	UDT_iTRAK_Cmd
Status	UDT_iTRAK_Status
Data	UDT_iTRAK_Data
Msg	UDT_iTRAK_Msg

UDT_iTRAK_Cmd

The UDT_iTRAK_Cmd data type contains the command variables for the iTRAK system. As of firmware revision 1.110, the UDT_iTRAK_Cmd has the following members.

Table 12 - UDT_iTRAK_Cmd Data Types

Tag Name	Full Name and Description	Values		Read-write	Data Type
iTRAKStart	iTRAK Start On a rising edge, this tag starts the iTRAK initialization sequence in the application code. It is used in the iTRAK sample project to trigger the Start Sequence in P01_MainProgram, R03_Start. It is not used in the iTRAK_IO instruction.	Default: Options:	0 0 = No action 1 = Initiate Start sequence	RW	BOOL
iTRAKStop	iTRAK Stop On a rising edge, this tag starts the iTRAK shutdown sequence in the application code. It is used in the iTRAK sample project to trigger the Stop Sequence in P01_MainProgram, R04_Stop. It is not used in the iTRAK_IO instruction.	Default: Options:	0 0 = No action 1 = Initiate Stop sequence	RW	BOOL
ServoOn	Servo On On a rising edge, if the I/O conditions are correct, this tag enables the DC bus to the motor modules via the USB I/O module.	Default: Options:	0 0 = No action 1 = Enable DC bus	RW	BOOL

Table 12 - UDT_iTRAK_Cmd Data Types (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
ServoOff	Servo Off On a rising edge, this tag disables the DC bus to the motor modules via the USB I/O module.	Default: Options:	0 0 = No action 1 = Disable DC bus	RW	BOOL
ReportPositionOutsideZone	Report Position Outside Zone This tag is no longer used and will be removed in a future revision.	Default: Options:	0 N/A	RW	BOOL
FeedbackUpdateOff	Feedback Update Off When held at the high level, this tag prevents the reporting of feedback from the track. This option can reduce the time that the gateway takes to process, but makes troubleshooting more difficult.	Default: Options:	0 0 = Feedback reported 1 = Feedback not reported	RW	BOOL
FaultReset	Fault Reset On a rising edge, if the fault cause is no longer present, this tag attempts to clear the fault in the gateway.	Default: Options:	0 0 = No action 1 = Reset faults	RW	BOOL
ResetGateway	Reset Gateway On a rising edge, this tag forces a restart of the gateway firmware program, without completely restarting the operating system of the gateway.	Default: Options:	0 0 = No action 1 = Initiate renumbering	RW	BOOL
RenumberMovers	Renumber Movers On a rising edge, this tag renumbers the movers from the first motor module. See Motion Polarity, Homing and Renumbering, and Set Zero Position on page 31 for more information.	Default: Options:	0 0 = No action 1 = Initiate renumbering	RW	BOOL
FlashSections50mm	Update Motor Module with 50 mm Firmware On a rising edge, this tag updates the commutation tables that are associated with 50 mm long magnets via application code. The update is independent of coil size. This tag stays high until the flashing is complete. The tag returns to the low value when the motor module is ready to use. Do not cycle power or restart the gateway during this time. The gateway can automatically restart during this process, monitor the tag to verify the completion of this process. is complete.	Default: Options:	0 0 = No action 1 = Write 50 mm commutation	RW	BOOL
FlashSections100mm	Update Track Motor Modules with 100 mm Firmware On a rising edge, this tag updates the commutation tables that are associated with 100 mm long magnets via application code and is independent of coil size. This tag stays high until the flashing is complete. The tag returns to the low value when the motor module is ready to use. Do not cycle power or restart the gateway during this time. The gateway can automatically restart during this process, monitor the tag to verify the completion of this process. is complete.	Default: Options:	0 0 = No action 1 = Write 100 mm commutation	RW	BOOL
FlashSections150mm	Update Track Motor Modules with 150 mm Firmware On a rising edge, this tag updates the commutation tables that are associated with 150 mm long magnets via application code and is independent of coil size. This tag stays high until the flashing is complete. The tag returns to the low value when the motor module is ready to use. Do not cycle power or restart the gateway during this time. The gateway can automatically restart during this process, monitor the tag to verify the completion of this process. is complete.	Default: Options:	0 0 = No action 1 = Write 150 mm commutation	RW	BOOL
SetGainsAll	Set Gains for All Motor Modules On a rising edge, this tag initiates the sequence in application code to write the gain values to all motor modules, including curves, from the Data tag. These gains become active as soon as the gateway receives them.	Default: Options:	0 0 = No action 1 = Initiate gain write	RW	BOOL

Table 12 - UDT_iTRAK_Cmd Data Types (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
SetGainsCurves	Set Gains for Curve Motor Modules On a rising edge, this tag initiates the sequence in application code to write the curve motor module gain values from the Data tag. These gains become active as soon as the gateway receives them. If your curve motor module gains differ from your straight motor module gains, this action must be performed following any Set Gains for All Motor Modules command. The execution of SetGainsCurves restores the curve motor module values that were overwritten by the Set Gains for All Motor Modules command.	Default: Options:	0 0 = No action 1 = Initiate gain write	RW	BOOL
SetGainsSection	Set Gains for a Single Motor Module On a rising edge, this tag initiates the sequence in application code to write the motor module gain values from the Data tag for the motor module indicated by the SetGainsSectionNumber tag. These gains become active as soon as the gateway receives them. To correct any overwrite that can have resulted from a preceding Set Gains for All Motor Modules or Set Gains for Curve Motor Module command, the Set Gains for a Single Motor Module must be issued after those commands.	Default: Options:	0 0 = No action 1 = Initiate gain write	RW	BOOL
GetGainsAll	Get Gains for All Motor Modules On a rising edge, this tag initiates the sequence in application code to read the actual gain values being used in each motor module.	Default: Options:	0 0 = No action 1 = Initiate gain read	RW	BOOL
TraceSelect	Select Trace Data On a rising edge, this tag initiates the sequence in application code to write the trace configuration to the gateway.	Default: Options:	0 0 = No action 1 = Initiate trace write	RW	BOOL
RunMoverRegistration	Run Mover Registration This tag is reserved for future use.	Default: Options:	0 N/A	RW	BOOL
SetDataSection	Write Data to Motor Modules On a rising edge, this tag initiates the sequence in application code to write certain advanced tuning parameters	Default: Options:	0 0 = No action 1 = Initiate data write	RW	BOOL
DoLogixHeadwayPrecheck	Execute Headway Check-in Logix While held high, this tag enables headway checking in the controller and headway checking in the gateway. This tag increases Logix Designer application execution time. It is used in the iTRAK sample project P01_iTRAK_Communications, R01_Headway. It is not used in the iTRAK_IO instruction.	Default: Options:	0 0 = No action 1 = Additional headway check in controller	RW	BOOL
ShutdownGateway	Shutdown Gateway This tag is reserved for use by Rockwell Automation® engineers.	Default: Options:	0 N/A	RW	BOOL
RetrieveLoopPeriods	Retrieve Loop Rates When this tag is latched in the Parameter Exchange routine, a parameter message id that is configured and sent to retrieve the periods of the various regulators on the iTRAK system. The tag is unlatched after the message is configured but before the values are retrieved. In the sample code, this tag is also latched every time the controller tag iTRAK_Control.Data.GatewayRunning has a rising edge transition.	Default: Options:	0 0 = No action 1 = Retrieve the regulator periods	RW	BOOL

UDT_iTRAK_Status

The UDT_iTRAK_Status data type contains all status information for the iTRAK system. As of revision 1.110, the UDT_iTRAK_Status has the following members.

Table 13 - UDT_iTRAK_Status

Tag Name	Full Name and Description	Values		Read-write	Data Type
GatewayRunning	Gateway Running This tag indicates if the gateway program is processing time-stamped data and is time-synchronized to the controller.	Default: Options:	0 0 = The gateway is not running 1 = The gateway is running	RO	BOOL
DCBusContactor	DC Bus Contactor This tag indicates that high voltage is available for the motor modules through the power supply as detected by the USB I/O module.	Default: Options:	0 0 = High voltage is not available 1 = High voltage is available	RO	BOOL
ReadyForMotion	Ready For Motion This tag indicates that the system is enabled with servo control ready for motion control commands. This tag is set in the Logix Designer application and neither the gateway or iTRAK_IO AOI modifies it directly.	Default: Options:	0 0 = Not ready for motion commands 1 = Ready for motion commands	RO	BOOL
Faulted	Faulted This tag indicates if the system has a fault present.	Default: Options:	0 0 = Not Faulted 1 = Faulted	RO	BOOL
RenumberMoversBeforeClearingFault	Renumber Movers Before Clearing Fault Certain faults require that a renumbering operation is performed on the track before the fault can be cleared. This tag indicates renumbering is required.	Default: Options:	0 0 = Clear faults without renumbering 1 = Renumber before the fault is cleared	RO	BOOL
GatewayFaultCode	Gateway Fault Code This tag contains the fault code from the gateway.	Default: Range	0 See iTRAK System User Manual, publication 2198T-UM001	RO	INT
SectionFaultCode	Motor Module Fault Code This tag contains the fault code from the motor module.	Default: Range:	0 See iTRAK System User Manual, publication 2198T-UM001	RO	INT
SectionNumberFaulted	Motor Module Number Faulted This tag contains the number of the motor module that is reporting the fault in Motor Module Fault Code.	Default: Range:	0 0…(Motor Module Count - 1)	RO	INT
SectionDeviceFaulted	Motor Module Device Faulted This tag indicates if the motor module fault is power-related, which can impact multiple motor modules or is position-related, which is isolated to one motor module.	Default: Options:	0 0 = Position related 1 = Power related	RO	INT
SectionFaultData	Motor Module Fault Data This tag contains extended fault code information specific to each fault code that can help diagnose and troubleshoot faults.	Default: Range:	0 See iTRAK System User Manual, publication 2198T-UM001	RO	INT
FaultMessageLine1	Fault Message Line 1 This tag contains the first line of text that would be displayed on the HMI for the specific fault code.	Default: Range:	"" See iTRAK System User Manual, publication 2198T-UM001	RO	STRING
FaultMessageLine2	Fault Message Line 2 This tag contains the second line of text that would be displayed on the HMI for the specific fault code.	Default: Range:	"" See iTRAK System User Manual, publication 2198T-UM001	RO	STRING
CoarseUpdatedPeriod	Coarse Update Period This tag indicates the Coarse Update Period being used by the motion system.	Default: Range:	12000 0…32000000	RO	DINT

Table 13 - UDT_iTRAK_Status (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
CSTTimeActualPositionRecorded	CST Time Actual Position Recorded Time stamp of last recorded feedback communication for each mover group.	Default: Range:	0 $0 \ldots 2^{31}$	RO	DINT[6]
ActualPosition	Actual Position Each element of this array tag indicates the actual position of the corresponding mover in the system. This information is not reassociated to the mover virtual axis. The actual position in the mover virtual axis is most likely not correct.	Units: Default: Range:	mm 0 0…Track Length	RO	REAL[96]
PositionError	Position Error Each element of this array tag indicates the position error of the corresponding mover in the system. Values outside of the minimum or maximum values are capped before communication from the gateway. This information is not reassociated to the mover virtual axis. The actual position in the mover virtual axis is most likely not correct.	Units: Default: Range:	mm 0 -32.768…+32.766	RO	REAL[96]
TraceData	Trace Data This tag returns the requested trend data for each of the movers on the system. The values and their interpretation are dependent on the type of trend data requested.	Default: Range:	0 $-2^{31} \ldots +2^{31}$	RO	DINT[96]
PositionLoopPeriod	Position Loop Rate The tag contains the position regulator period in microseconds.	Units:	µs	RO	INT
VelocityLoopPeriod	Velocity Loop Rate The tag contains the velocity regulator period in microseconds.	Units:	µs	RO	INT
CurrentLoopPeriod	Current Loop Rate The tag contains the current regulator period in microseconds.	Units:	µs	RO	INT

UDT_iTRAK_Data

The UDT_iTRAK_Data data type contains all configurable variables for the iTRAK system. As of revision 1.110, the UDT_iTRAK_Data has the following members.

Table 14 - UDT_iTRAK_Data

Tag Name	Full Name and Description	Values		Read-write	Data Type
TrackLength	Track Length Total length of track, in microns. Changes to this parameter can only be made when the track is disabled.	Units: Default: Range:	µm 4800000 0…2560000000	RW	DINT
AxisUnwind	Axis Unwind The value that is used in the Position Unwind field in the Virtual Axis Properties. This value is typically the same as the Track Length, except in Multi-path tracks. Changes to this parameter can only be made when the track is disabled.	Units: Default: Range:	µm 4800000 0…2560000000	RW	DINT
ActiveMovers	Active Mover Count Total number of movers currently active on the track. Changes to this parameter can only be made when the track is disabled.	Default: Range:	10 0…96	RW	DINT
MotionPolarity	Motion Polarity Defines the direction of positive motion, when viewing the top of the track, where the plastic feedback board covers are located. See Motion Polarity, Homing and Renumbering, and Set Zero Position on page 31 for detailed use. Changes to this parameter can only be made when the iTRAK system is disabled and only becomes active after a first-scan pass of the controller.	Default: Options:	0 0 = Counterclockwise Forward 1 = Clockwise Forward	RW	BOOL
ReverseMoverNumbering	Reverse Mover Numbering Defines the direction of mover renumbering. When disabled, mover 0 is assigned to the mover with the most positive position, and mover N is the lowest positive position. When enabled, mover 0 is assigned to the lowest positive position, and mover N is the most positive position. Changes to this parameter can only be made when the iTRAK system is disabled and only become active after a first-scan pass of the controller.	Default: Options:	0 0 = Mover 0 Most Positive 1 = Mover 0 Least Positive	RW	BOOL
PositionOffset	Position Offset Use this value to move the zero position of the track logically. The new zero position is at a relative difference from the hardwired zero position, equal to the value entered, in the direction specified by Motion Polarity. Changes to this parameter can only be made when the iTRAK system is disabled and becomes active after a first-scan pass of the controller. Changes to this parameter do not change where renumbering begins, which is dependent on the setting of the Mover Numbering Offset tag relative to the system wiring.	Units: Default: Range:	µm 0 0…TrackLength	RW	DINT
MoverNumberingOffset	Mover Numbering Offset Use this value to offset mover numbering. The value represents the relative adjustment of the movers from the hard-wired zero position of the track, according to the MotionPolarity and ReverseMoverNumbering fields. When the mover number (ActiveMovers-1) is assigned, numbers roll over to 0 and continue until all movers have been numbered. Changes to this parameter can only be made when the iTRAK system is disabled and become active after a first-scan pass of the controller	Default: Range:	0 0…ActiveMovers	RW	DINT
NumberOfZones	Number of Zones Straight and closed tracks must use a value of 1 for this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Default: Range:	1 1…8	RW	SINT

Table 14 - UDT_iTRAK_Data (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
ZoneStartPosition	Zone Start Position Straight and closed tracks must set all the data elements to a value of 0 for this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Units: Default: Range:	µm 0 0…TrackLength	RW	DINT[8]
ZoneBeginReportOffset	Zone Begin Report Offset Straight and closed tracks must set all the data elements to a value of 0 for this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Units: Default: Range:	µm 0 0…TrackLength	RW	DINT[8]
ZoneBeginCommandOffset	Zone Begin Command Offset Straight and closed tracks must set all the data elements to a value of 0 for this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Units: Default: Range:	µm 0 0…TrackLength	RW	DINT[8]
ZoneBeginHoldOffset	Zone Begin Hold Offset Straight and closed tracks must set all the data elements to a value of 0 for this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Units: Default: Range:	µm 0 0…TrackLength	RW	DINT[8]
ZoneEndReportOffset	Zone End Report Offset Straight and closed tracks must set data element [0] to the Track Length, and use a value of 0 for all other data elements of this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Units: Default: Range:	µm 4800000 0…TrackLength	RW	DINT[8]
ZoneEndCommandOffset	Zone End Command Offset Straight and closed tracks must set data element [0] to the Track Length, and use a value of 0 for all other data elements of this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Units: Default: Range:	µm 4800000 0…TrackLength	RW	DINT[8]
ZoneEndHoldOffset	Zone End Hold Offset Straight and closed tracks must set data element [0] to the Track Length, and use a value of 0 for all other data elements of this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation to commission.	Units: Default: Range:	µm 4800000 0…TrackLength	RW	DINT[8]
ZoneLength	Zone Length Straight and closed tracks must set data element [0] to the Track Length, and use a value of 0 for all other data elements of this parameter. Multi-path tracks can have other values that require assistance from Rockwell Automation commission.	Units: Default: Range:	µm 4800000 0…TrackLength		DINT[8]
ParameterStatus	Parameter Status Data in parameter exchange structure is ready for download to the gateway. When the iTRAK_IO AOI parses this tag, any information in the parameter messages starts to be sent to the gateway asynchronously. The iTRAK_IO AOI sets this value low when the data has been copied.	Default: Options:	0 0 = No action 1 = Initiate parameter exchange to gateway	RW	SINT
ParameterCode	Parameter Code Indicates what type of data is stored in Parameter Data for download to the gateway. See Appendix B on page 79 for more information.	Default: Range:	0 See Appendix B	RW	INT
ParameterSize	Parameter Size Indicates how much data is stored in Parameter Data for download to the gateway. See Appendix B on page 79 for more information.	Default: Range:	0 See Appendix B	RW	SINT
ParameterData	Parameter Data Stores parameter data before download to the gateway. See Appendix B on page 79 for more information.	Default: Range:	0 See Appendix B	RW	INT[12]

Table 14 - UDT_iTRAK_Data (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
ParameterStatusReturn	Parameter Status Return Data in parameter messages was uploaded from the gateway. The iTRAK_IO AOI sets this value high when the data has been copied. Set this value low to let more data to be uploaded.	Default: Options:	0 0 = Ready to read data 1 = Data was uploaded from gateway	RW	BOOL
ParameterCodeReturn	Parameter Code Return Indicates what type of data is stored in Parameter Data Return after upload from the gateway. See Appendix B on page 79 for more information.	Default: Range:	0 See Appendix B	RW	INT
ParameterSizeReturn	Parameter Size Return Indicates how much data is stored in Parameter Data after upload from the gateway. See Appendix B on page 79 for more information.	Default: Range:	0 See Appendix B	RW	SINT
ParameterDataReturn	Parameter Data Return Stores parameter data after upload from the gateway. See Appendix B on page 79 for more information.	Default: Range:	0 See Appendix B	RW	INT[12]
MoverOffset	Mover Offset Measured offset between position magnet and mover magnet. This value is set individually for each mover in the array so it requires absolute identification of the movers. This tag can be used to increase the mover-to-mover stop repeatability and accuracy by calibrating each position magnet relative to the mover magnet.	Units: Default: Range:	µm 0 $-2^{31}\ldots+2^{31}$	RW	DINT[96]
RenumberMoversAtStart	Renumber Movers at Start While set high, this tag triggers renumbering during each iTRAK Start request. It is used in the iTRAK sample project to trigger specific Parameter Interface in P01_MainProgram, R03_Start. It is not used in the iTRAK_IO instruction.	Default: Options:	0 0 = Do not renumber movers after each iTRAKStart rising edge 1 = Renumber movers after each iTRAKStart rising edge	RW	BOOL
SetGainsSectionNumber	Set Gains Motor Modules Number This tag is used to determine which motor module gets the specific per-motor module gain set to be communicated in the sample logic. It is not used inside of the iTRAK_IO instruction. Values outside of the valid range can cause a major fault on the controller. See Tuning on page 49 for more information.	Default: 0 Range:	0 0…63	RW	DINT
HeadwayTolerance	Headway Tolerance This tag indicates the minimum spacing that must be maintained between the command positions of adjacent movers. If the difference in command position between movers is below this value, a headway fault is generated. This value is used in the Logix Designer application headway checking and in the gateway headway checking.	Units: Default: Range:	mm 40 $0\ldots2^{31}$	RW	DINT
StandstillWindow	Standstill Window for All Motor Modules This tag specifies the dead-zone around the command position while at zero command velocity. It must not be used on areas of the track that have external forces that are applied. This tag applies to all motor modules on the track. Values outside of the range disable the feature.	Units: Default: Range:	µm 25 1…19999	RW	DINT
CurveStandstillWindow	Standstill Window for Curve Motor Modules This tag specifies the dead-zone around the command position while at zero command velocity for curve motor modules. It must not be used on areas of the track that have external forces that are applied. This tag applies to all curve motor modules on the track. Values outside of the range disable the feature.	Units: Default: Range:	µm 19999 1…19999	RW	DINT

Table 14 - UDT_iTRAK_Data (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
PositionErrorTolerance	Position Error Tolerance This tag indicates the maximum allowable amount of position error for any single mover on any motor module of the track before a Position Window fault is generated. This value is applied to all motor modules of the track when the Set Gains All command is used.	Units: Default: Range:	µm 200 0…Track Length	RW	DINT
GainsAllVelocityBandwidth	Velocity Proportional Bandwidth for All Motor Modules This tag is the proportional gain for the velocity loop for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information.	Default: Range:	10 $0…2^{15}$	RW	DINT
GainsAllVelocityIntegratorBandwidth	Velocity Integrator Bandwidth for All Motor Modules This tag is the integral gain for the velocity loop for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information.	Default: Range:	20 0…127	RW	DINT
GainsAllPositionBandwidth	Position Proportional Bandwidth for All Motor Modules This tag is the proportional gain for the position loop for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information.	Default: Range:	75 $0…2^{15}$	RW	DINT
GainsAllPositionDerivative	Position Derivative Bandwidth for All Motor Modules This tag is the derivative gain for the position loop for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information.	Default: Range:	2 0 …127	RW	DINT
GainsAllAccelerationFeedForward	Acceleration Feed Forward for All Motor Modules This tag is the acceleration feed forward for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information.	Default: Range:	50 0…127	RW	DINT
GainsAllGravityMagnitudeKg	Gravity Magnitude for All Motor Modules This tag is the gravity compensation for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information.	Units: Default: Range:	Kg 0.0 0…327.67	RW	REAL
GainsAllGravityZeroLocation	Gravity Zero Location This tag sets the zero position of the gravity compensation. This value is applied to all motor modules of the track when the Set Gain All command is used. See Tuning on page 49 for more information.	Default: Range:	0 See Tuning	RW	DINT
GainsCurvesVelocityBandwidth	Velocity Proportional Bandwidth for Curve Motor Modules This tag is the proportional gain for the velocity loop for curve motor modules. This value is applied to curve motor modules of the track when the Set Gains Curves command is used. See Tuning on page 49 for more information.	Default: Range:	4 $0…2^{15}$	RW	DINT
GainsCurvesVelocityIntegratorBandwidth	Velocity Integral Bandwidth for Curve Motor Modules This tag is the integral gain for the velocity loop for curve motor modules. This value is applied to curve motor modules of the track when the Set Gains Curves command is used. See Tuning on page 49 for more information	Default: Range:	8 0…127	RW	DINT
GainsCurvesPositionBandwidth	Position Proportional Bandwidth for Curve Motor Modules This tag is the proportional gain for the position loop for curve motor modules. This value is applied to curve motor modules of the track when the Set Gains Curves command is used. See Tuning on page 49 for more information.	Default: Range:	50 $0…2^{15}$	RW	DINT

Table 14 - UDT_iTRAK_Data (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
GainsCurvesPositionDerivative	Position Derivative Bandwidth for Curve Motor Modules This tag is the derivative gain for the position loop for curve motor modules. This value is applied to curve motor modules of the track when the Set Gains Curves command is used. See Tuning on page 49 for more information.	Default: Range:	3 0…127	RW	INT[64]
GainsSectionVelocityBandwidth	Velocity Proportional Bandwidth for Specific Motor Modules This tag is the proportional gain for the velocity loop for specific motor modules. This value is applied to the specified motor module of the track indicated by Set Gains Motor Module Number value when the Set Gains Motor Module command is used. See Tuning on page 49 for more information.	Default: Range:	0 $0…2^{15}$	RW	INT[64]
GainsSectionVelocityIntegratorBandwidth	Velocity Integral Bandwidth for Specific Motor Modules This tag is the integral gain for the velocity loop for specific motor modules. This value is applied to the specified motor module of the track indicated by Set Gains Motor Module Number value when the Set Gains Motor Module command is used. See Tuning on page 49 for more information.	Default: Range:	0 0…127	RW	SINT[64]
GainsSectionPositionBandwidth	Position Proportional Bandwidth for Specific Motor Modules This tag is the proportional gain for the position loop for specific motor modules. This value is applied to the specified motor module of the track indicated by Set Gains Motor Module Number value when the Set Gains Motor Module command is used. See Tuning on page 49 for more information.	Default: Range:	0 $0…2^{15}$	RW	INT[64]
GainsSectionPositionDerivative	Position Derivative Bandwidth for Specific Motor Modules This tag is the derivative gain for the position loop for specific motor modules. This value is applied to the specified motor module of the track indicated by Set Gains Motor Module Number value when the Set Gains Motor Module command is used. See Tuning on page 49 for more information.	Default: Range:	0 $0…2^{15}$	RW	INT[64]
GatewayInput	Gateway Input This tag is reserved for internal use, do not write to it.	–	–	RO	DINT[264]
GatewayOutput	Gateway Output This tag is reserved for internal use, do not write to it.	–	–	RO	DINT[264]
SetDataSectionNumber	Set Data Motor Modules Number This tag is used in the sample logic to determine which motor module gets the specific per-motor module advanced tuning information to be communicated. See Tuning on page 49 for more information.	Default: Range:	0 0…63	RW	DINT
TAFFScalingCoefficient	Tangential Acceleration Feedforward Scaling Coefficient This tag is used to modify the Tangential Acceleration Feedforward coefficient that is applied to Tangential Acceleration information associated with variable radius curved motor modules. See Tuning on page 49 for more information.	Units: Default: Range:	% 100 80…120	RW	DINT
IPSMode	iTRAK Power Supply Mode This tag indicates whether the track is being powered by the Kinetix® 5700 iTRAK Power Supply.	Default: Options:	0 0 = iTRAK Power Supply is not used 1 = iTRAK Power Supply is in use	RW	INT
VregPeriod	Velocity Regulator Period This tag is reserved for internal use, do not write to it.	–	–	RO	INT
SectionCount	Motor Module Count This tag is reserved for internal use, do not write to it.	–	–	RO	INT

Table 14 - UDT_iTRAK_Data (Continued)

Tag Name	Full Name and Description	Values		Read-write	Data Type
RenumberingMoverStartNumber	Renumber Mover Start Number When performing a renumbering operation, this value is used as the starting mover number in place of 0. During the renumbering process, when the mover number reaches the number of active movers on the system, it rolls over to 0 and continues from there. See Homing and Renumbering on page 32.	Default: Range:	0 0…Active Movers-1	RW	DINT
InternalIndex	Internal Index This tag is reserved for internal use, do not write to it.	–	–	RO	DINT
UnsolicitedGatewayParam	Unsolicited Gateway Parameter This tag is reserved for internal use, do not write to it.	–	–	RO	INT
ParameterFeedbackStatus	Parameter Feedback Status This tag is reserved for internal use, do not write to it.	–	–	RO	BOOL
GainsAllVelocityFeedForward	Velocity Feed Forward for All Motor Modules This tag is the velocity feedforward gain for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information. The recommended value is 100 even though the default value is 0.	Default: Range:	0 0…100	RW	INT
GainsCurvesVelocityFeedForward	Velocity Feed Forward for Curve Motor Modules This tag is the velocity feedforward gain for curved motor modules. This value is applied to curved motor modules of the track when the Set Gains Curves command is used. See Tuning on page 49 for more information. The recommended value is 100.	Default: Range:	0 0…100	RW	INT
FlashSectionsCommutationType	Set the Type of Motor Module Commutation Tables Sets the Commutation Table Motor Module type to be used with the iTRAK_Control.Cmd.FlashSectionsxxmm command. For Optimal current, set this value to 0, which is the default. For Optimal Kf, which is the commutation table present in Revision1.108 and earlier firmware, set this value to 1.	Default: Range:	0 0 = Select the optimal current commutation tables. 1 = Select the optimal K_f commutation tables.	RO	INT

UDT_iTRAK_Msg_Control

This user-defined type is instantiated in the sample code in the iTRAK_MSG controller tag. This tag is used with the Parameter Interface to make sure that user requests are sent to the iTRAK system one at a time. It reports the status of a completed request.

Table 15 - UDT_iTRAK_Msg_Control

Tag Name	Full Name and Description	Values		Read-write	Data Type
Interlock	Interlock This tag is latched immediately before a parameter interface message being sent, and remains latched until the parameter request is successfully completed, or a fault occurs.	Default: 0 Options:	0 0 = New parameter requests can be initiated 1 = Parameter request is in process	RO	BOOL
DN	Done This tag indicates that the most recently sent parameter request that was completed successfully.	Default: Options:	0 0 = Not completed 1 = Successful completion	RO	BOOL
ER	Error This tag indicates that the most recently sent parameter request was not successful, either due to a timeout, or some other error.	Default: Options:	0 0 = No error 1 = Most recent parameter request failed	RO	BOOL
ONS	One Shot This tag is a one shot that is used internally by the message control code.	–	–	RO	BOOL
Timer	Timer This tag is a timer that is used internally by the message control code.	–	–	RO	TIMER

Feature Based Data Types

Some of the iTRAK features are implemented using sample logic. Those features often have specific data types that are associated with the sample code.

UDT_iTRAK_Version

This user-defined type is instantiated in the sample code in the iTRAK_Version controller tag. It is associated with the Pxx_MajorMinorBuildRevAndSectionType program sample code, which is used to retrieve the firmware revision and motor module type information from the iTRAK system.

Table 16 - UDT_iTRAK_Version

Tag Name	Full Name and Description	Values		Read-write	Data Type
GetRevBuildNumAndType	Get Revision, Build Number, and Type When latched, this tag generates a message request to obtain the gateway firmware for: Major revision Minor revision Build number and motor modules: Major revision Minor revision Build number Firmware motor module type	Default: Options:	0 0 = No action 1 = Initiate message	RW	BOOL
GatewayMajorRev	Gateway Major Revision This tag will store the Major Revision number of the gateway after the message has completed.	Default: Typical Value:	0 1	RO	INT
GatewayMinorRev	Gateway Minor Revision This tag will store the Minor Revision number of the gateway after the message has completed.	Default: Typical Value:	0 110	RO	INT
GatewayBuildNum	Gateway Build Number This tag will store the Build Number of the gateway after the message has completed.	Default: Typical Value:	0 *xxx*	RO	INT
SectionMajorRev	Motor Module Major Revision This tag will store the Major Revision number of each motor module in the array after the message has completed.	Default: Typical Value:	0 1	RO	INT[64]
SectionMinorRev	Motor Module Minor Revision This tag will store the Minor Revision number of each motor module in the array after the message has completed.	Default: Typical Value:	0 110	RO	INT[64]
SectionBuildNum	Motor Module Build Number This tag will store the Build Number of each motor module in the array after the message has completed.	Default: Typical Value:	0 *xxx*	RO	INT[64]
SectionFirmwareType	Motor Module Firmware Type This tag will store the Firmware Type of each motor module in the array after the message has completed.	Default: Range:	0 See Firmware Revision and Motor Module Type Information on page 30	RO	INT[64]

UDT_iTRAK_FW_Version

This user-defined type is instantiated in the sample code in the iTRAK_Min_Compatible_FW_Version controller tag. It is set to indicate the minimum compatible firmware revision with which the programs, AOI, and tags in the sample code can be used. The Major Revision, Minor Revision, and Build Number must match the firmware that is running on your iTRAK system.

Table 17 - UDT_iTRAK_FW_Version

Tag Name	Full Name and Description	Values		Read-write	Data Type
MajorRev	Major Revision This tag contains the minimum required Major Revision number of the firmware that is required for this Logix Designer application to work correctly.	Default:	0	RO	INT
MinorRev	Minor Revision This tag contains the minimum required Minor Revision number of the firmware that is required for this Logix Designer application to work correctly.	Default:	0	RO	INT
BuildNum	Build Number This tag contains the minimum required Build number of the firmware that is required for this Logix Designer application to work correctly.	Default:	0	RO	INT

UDT_iTRAK_Position_Integrator

This user-defined type is instantiated in the iTRAK_Position_Integrator controller tag and lets you set the position integrator gain and the integrator hold enable feature.

Table 18 - UDT_iTRAK_Position_Integrator

Tag Name	Full Name and Description	Values		Read-write	Data Type
GainsAllPositionIntegratorBandwidth	Position Integrator Bandwidth for All Motor Modules This tag is the integral gain for the position loop for all motor modules. This value is applied to all motor modules of the track when the Set Gains All command is used. See Tuning on page 49 for more information.	Default: Range:	0 0…127	RW	DINT
GainsCurvesPositionIntegratorBandwidth	Position Integrator Bandwidth for Curved Motor Modules This tag is the integral gain for the position loop for curved motor modules. This value is applied to curved motor modules of the track when the Set Gains Curves command is used. See Tuning on page 49 for more information.	Default: Range:	0 0…127	RW	DINT
GainsSectionPositionIntegratorBandwidth	Position Integrator Bandwidth for Specific Motor Modules This tag is the integral gain for the position loop for specific motor modules. This value is applied to the specified motor module of the track indicated by Set Gains Motor Module Number value when the Set Gains Motor Module command is used. See Tuning on page 49 for information.	Default: Range:	0 0…127	RW	DINT[64]
MinCompatibleFWVersion	Minimum Compatible Firmware Version The minimum compatible firmware version for this implementation.	–	–	RW	UDT_iTRAK_FW_Version
FeatureRevision	Feature Revision The internal feature revision number for this implementation.	–	–	RO	INT
SetIntegratorHold	Set Integrator Hold When this tag is latched in the Integrator Hold routine, a parameter message is configured and sent to enable or disable the integrator hold feature on the iTRAK system. The tag will be unlatched after the message is configured. In the sample code, this tag is not automatically set.	Default: Options:	0 0 = No action 1 = Configure parameter request and initiate message	RW	BOOL
IntegratorHoldEnable	Integrator Hold Enable This tag is used in the Integrator Hold routine to enable or disable the position integrator hold, which prevents position integrator windup unless the command velocity is zero.	Default: Options:	0 0 = Disable Integrator Hold 1 = Enable Integrator Hold	RW	DINT

Additional Controller Tags in the Sample Code

This section shows you the additional controller tags in the sample code that are not part of the iTRAK user-defined types.

Before revision 1.108, the Util_Trace... tags listed in Table 19 used to be program local tags inside the P01_Main_Program of the Main Task in the sample code. These tags have been changed to controller tags for ease-of-use. The P05_Parameter_Exchange routine in P01_Main_program contains code to make sure that only one of these Util_Trace... tags can be set at a time.

Table 19 - Additional Controller Tags in the Sample Code

Tag Name	Full Name and Description	Values		Read-write	Data Type
Util_Trace_RD_Axis_FErr_and_Touque_mamps	Trace Following Error and Torque This tag triggers configuration of the trending data for the following. • Position Error by Mover • Command Current by Mover See Trending on page 46 for more information.	Default: Options:	0 0 = No action 1 = Initiate the requested trend operation	RW	BOOL
Util_Trace_RD_Axis_FErr_and_Touque_mamps	Trace Following Error and Velocity Loop This tag triggers configuration of the trending data for the following. • Position Error by Mover • Velocity Loop Update Rate See Trending on page 46 for more information.	Default: Options:	0 0 = No action 1 = Initiate the requested trend operation	RW	BOOL
Util_Trace_RD_Axis_FollowErr_and_Vel_mms	Trace Following Error and Velocity Loop This tag triggers configuration of the trending data for the following. • Position Error by mover • Actual Velocity by mover See Trending on page 46 for more information.	Default: Options:	0 0 = No action 1 = Initiate the requested trend operation	RW	BOOL
Util_Trace_RD_Sec_Common_Voltage_mv	Trace Motor Module Common Voltage This tag triggers configuration of the trending data for the Mid-bus Voltage by motor module See Trending on page 46 for more information.	Default: Options:	0 0 = No action 1 = Initiate the requested trend operation	RW	BOOL
Util_Trace_RD_Sec_High_Voltage_mv	Trace Motor Module High Voltage This tag triggers configuration of the trending data for the High-bus Voltage by motor module. See Trending on page 46 for more information.	Default: Options:	0 0 = No action 1 = Initiate the requested trend operation	RW	BOOL
Util_Trace_RD_Sec_Neg_Current_mamps	Trace Motor Module Positive Current For this tag, the abbreviated name is contrary to the function of this tag. This tag triggers configuration of the trending data for the Positive Current by motor module. See Trending on page 46 for more information.	Default: Options:	0 0 = No action 1 = Initiate the requested trend operation	RW	BOOL
Util_Trace_RD_Sec_Pos_Current_mamps	Trace Motor Module Negative Current For this tag, the abbreviated name is contrary to the function of this tag. This tag triggers configuration of the trending data for the Negative Current by motor module. See Trending on page 46 for more information.	Default: Options:	0 0 = No action 1 = Initiate the requested trend operation	RW	BOOL

Parameter Interface

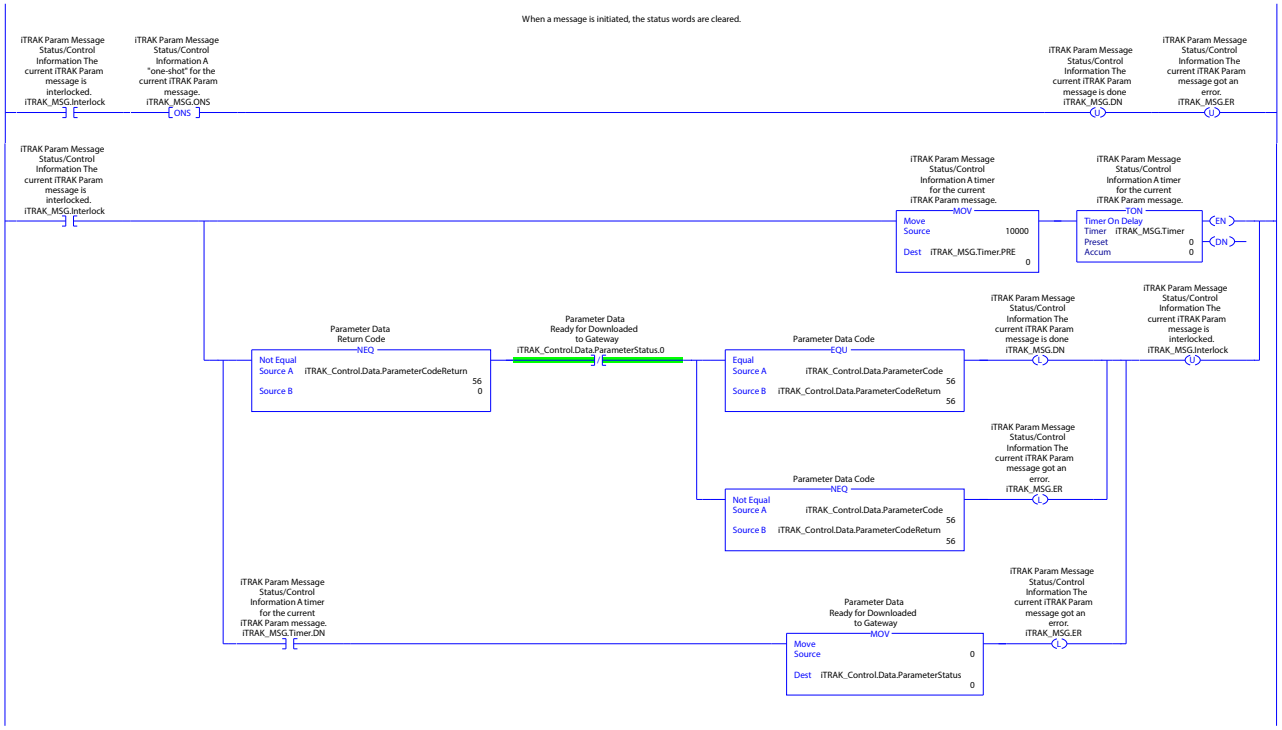
P05_ParameterExchange Routine

The iTRAK® system has a service communication channel that occurs asynchronously. You can use this mechanism to load parameters and retrieve information from the track. It is recommended that you incorporate the entire P05_ParameterExchange routine in your project. This routine includes the preferred code for using this feature, including the message interlock, which is essential to preventing message collisions.

The message interlock code is documented using the this table.

Rung	Code
31	XIC iTRAK_MSG.Interlock ONS iTRAK_MSG.ONS OTU iTRAK_MSG.DN OTU iTRAK_MSG.ER
32	XIC iTRAK_MSG.Interlock BST MOV 10000 iTRAK_MSG.Timer.PRE TON iTRAK_MSG.Timer ? ? NXB BST NEQ iTRAK_Control.Data.ParameterCodeReturn 0 XIO iTRAK_Control.Data.ParameterStatus.0 BST EQU iTRAK_Control.Data.ParameterCode iTRAK_Control.Data.ParameterCodeReturn OTL iTRAK_MSG.DN NXB NEQ iTRAK_Control.Data.ParameterCode iTRAK_Control.Data.ParameterCodeReturn OTL iTRAK_MSG.ER BND NXB XIC iTRAK_MSG.Timer.DN MOV 0 iTRAK_Control.Data.ParameterStatus OTL iTRAK_MSG.ER BND OTU iTRAK_MSG.Interlock BND

The message interlock rungs are shown here.



To use the Parameter Exchange routine, do the following.

1. To set data, do the following.
   a. Apply desired parameter code.
   b. Apply appropriate parameter size, specific to each parameter code.
   c. Fill parameter data by using the correct format for that parameter code.
2. To initiate data transfer, do the following.
   a. Wait for the message interlock to be low.
   b. Clear the data in parameter code return.
   c. Set parameter status to 1 to initiate transfer.
   d. Set message interlock high.
3. Monitor message interlock for completion, then do the following.
   a. Confirm that parameter code return and parameter status return are correct.
   b. If needed, copy data from parameter data return.

The parameters that you can access have tags and sample code included in the Full Featured Sample Projects. If you choose not to use the sample code, use the pre-defined interfaces for data formatting and sequencing that are included in the Full Featured Sample Projects to learn how to implement the code in your application,

Table 20 - Parameter Code Description

Feature	Description	Parameter Codes	Sample Code Location
Adjust velocity loop rate	The velocity loop rate is determined by the gateway. Adjusting it down can lead to task overlaps within the motor module and faults. Adjusting it up can lead to significantly diminished performance. Adjust only under the guidance of Rockwell Automation support.	11	Pxx_VReg_Period_Change
Using the Kinetix® 5700 iTRAK Power Supply	The default power supply can be adjusted to utilize the Kinetix 5700 iTRAK Power Supply with this code.	14 and 15	Pxx_IPS_Support
Position Integrator Hold	The position integrator can be configured to turn off while the command position is changing. This can improve response and stability in highly dynamic moves while maintaining repeatability at the end of the move.	20	Pxx_Position_Integrator
Maximum Mover Torque	The maximum amount of commanded torque or current that can be applied to a single mover is ±8000 mA. This adjustment lets you to change this value to anywhere in the range of 0%…100% of normal, or 0…±8000 mA. You can only change the maximum torque percentage when the track is disabled.	22	Pxx_Mover_Torque_Percent
Trace or Trending	Actual Position and Position Errors are returned by default to the controller. This parameter exchange feature lets you to select one additional data value to be returned regularly.	44 and 144	P01_MainProgram → R05_Parameter Exchange → Rungs 22…33
Flash *xx* mm Firmware for Optimal K_f	The three different magnet plate lengths require different commutation tables. Use this feature to send the appropriate commutation table to the motor module. Commutation tables can only be updated while the track is disabled. Power cycling of the motor module is required to make the new tables active.	50, 100, and 150	P01_MainProgram → R05_Parameter Exchange Rungs 10…19
Flash xx mm Firmware for Optimal Current	The three different magnet plate lengths require different commutation tables. Use this feature to send the appropriate commutation table to the motor module. Commutation tables can only be updated while the track is disabled. Power cycling of the motor module is required to make the new tables active.	51, 101, and 151	P01_MainProgram → R05_Parameter Exchange Rungs 10…19

Table 20 - Parameter Code Description (Continued)

Feature	Description	Parameter Codes	Sample Code Location
Position Error Tolerance Per Section	By default, the iTRAK system has a single position error tolerance for the entire track. Your process can require that position error tolerances in certain motor modules be different. For example, a high-speed motor module without accuracy requirements could utilize a higher tolerance to prevent nuisance trips. This parameter lets you to modify specific motor modules with different position error tolerances.	54	Pxx_Per_Section_Position _Error_Tolerance
Tuning	The tuning gains for the iTRAK system can be set for all motor modules, all curves, or individual motor modules. These tuning gains are transmitted to the gateway with the parameter exchange function. See Tuning on page 49 for more information.	55, 57, and 155	P01_MainProgram → R05_Parameter Exchange → Rungs 1…7
External Force Feedforward	It can be useful to apply an external force feedforward to account for known disturbances in the system such as cam-actuated mechanics. This features lets you assign an external force trim by position at a resolution of 1 mm.	66	Pxx_ExtFF_Compensation
Renumber Movers	Movers are only uniquely identified while power is maintained on the system. If your application requires identification through power cycles, additional identifiers, such as a photo reflective flag on Mover #1 and application code to detect it, can be used with this feature to reset mover numbering. See Homing and Renumbering on page 32.	67	P01_MainProgram → R05_Parameter Exchange Rungs 20…21

Notes:

Add-on Instructions

Export these AOIs from the sample code file and import into you Logix Designer application file.

iT_GSSN Add-on Instruction

Use the iT_GSSN add-on instruction to retrieve motor module serial numbers.

About the AOI

The iT_GSSN AOI retrieves the serial numbers of all motor modules on the iTRAK® system and stores them in an output array.

Figure 15 - Logix Designer Application Example

Table 21 - Parameters

Parameter	Description
iT_GSSN	The control tag for the AOI.
iTRAK_Control	The instance of UDT_iTRAK_Control that you want to use.
Section_Serial_Number	This tag is an INT array of 64 elements that receives the section serial number from the gateway.
<iT_GSSN >.EN	The AOI is enabled.
<iT_GSSN >.PC	The AOI completed successfully.
<iT_GSSN >.ER	The AOI encountered an error.
<iT_GSSN >.ERR	The error code that is associated with the error. This parameter is not visible, but is available.
<iT_GSSN >.EXERR	Additional error information, if applicable This parameter is not visible, but is available.

Table 22 - Error Codes

ERR Value	Description
0	No error code.
-4	The internal parameter code value that the gateway returned did not match the parameter code that was sent.
-5	The currently loaded revision of firmware does not support this function.

There is no extended error information for the iT_GSSN AOI.

Motor module serial number are received in groups of eight from the gateway. Only the actual number of motor modules on the track are assigned the serial numbers and put into the output tag. The rest of the information from the gateway, which is invalid, is ignored.

iT_SPO Add-on Instruction

Use this AOI to set the position offset of a mover.

About the AOI

The iT_SPO AOI sets the position offset field in the iTRAK_Control.Data.PositionOffset tag according to the actual position of a passed-in mover number. The instruction accounts for the motion polarity of the track, and takes the average of 100 readings of the iTRAK_Control.Status.ActualPosition [MoverNumber] value to improve the accuracy of the result.

The iT_SPO AOI lets you position a mover on the track where you want the zero point to be, disable the track, and then execute the instruction to readjust the zero point. After execution, ActualPosition tag of the mover is zero, ± a few microns. Mover numbers remain as they were before execution.

A Logix Designer application example that calls the iT_SPO AOI is given here. The iTRAK_Position_Offset tag is only shown here as a sample. You can use whatever tags you want.

IMPORTANT Use this AOI, like the Position Offset values for movers, only when absolute identification of movers is being maintained.

Figure 16 - Logix Designer Application Example

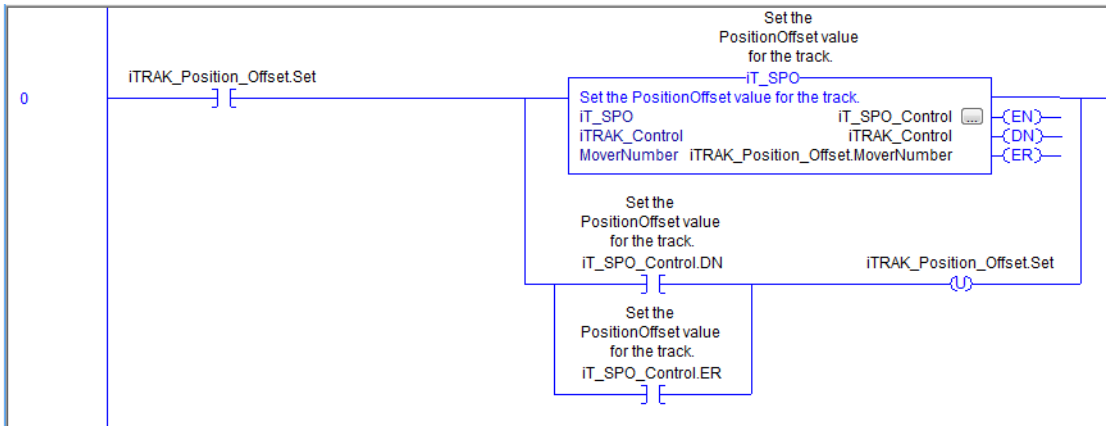


Table 23 - Parameter

Parameter	Description
iT_SPO	The control tag for the AOI.
iTRAK_Control	The instance of UDT_iTRAK_Control that you want to use.
MoverNumber	The specific mover number on the track that you want to set as the zero position.
<iT_SPO>.EN	The AOI is enabled.
<iT_SPO>.DN	The AOI completed successfully.
<iT_SPO>.ER	The AOI encountered an error.
<iT_SPO>.ERR	The error code that is associated with the error. This parameter is not visible, but is available.
<iT_SPO>.EXERR	Additional error information, if applicable. This parameter is not visible, but is available.

Table 24 - Error Codes

ERR Value	Description
0	No error code.
-1	The track must not be enabled when this instruction is executed.
-2	An input parameter value is greater than its upper limit.
-3	An input parameter is less than its lower limit.

Table 25 - Extended Error Codes

EXERR Value	Description
0	No extended information.
-1	ERR applies to the Mover Number parameter.

Notes:

E

F

G

H

R

S

T

U

V

W

Z

Notes:

Rockwell Automation Support

Use the following resources to access support information.

Technical Support Center	Knowledgebase Articles, How-to Videos, FAQs, Chat, User Forums, and Product Notification Updates.	https://rockwellautomation.custhelp.com/
Local Technical Support Phone Numbers	Locate the phone number for your country.	http://www.rockwellautomation.com/global/support/get-support-now.page
Direct Dial Codes	Find the Direct Dial Code for your product. Use the code to route your call directly to a technical support engineer.	http://www.rockwellautomation.com/global/support/direct-dial.page
Literature Library	Installation Instructions, Manuals, Brochures, and Technical Data.	http://www.rockwellautomation.com/global/literature-library/overview.page
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	http://www.rockwellautomation.com/global/support/pcdc.page

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete the How Are We Doing? form at http://literature.rockwellautomation.com/idc/groups/literature/documents/du/ra-du002_-en-e.pdf.

Rockwell Automation maintains current product environmental information on its website at http://www.rockwellautomation.com/rockwellautomation/about-us/sustainability-ethics/product-environmental-compliance.page.

Allen-Bradley, Integrated Architecture Builder, iTRAK, Kinetix, LOGIX 5000, Rockwell Automation, Rockwell Software, Studio 5000, and Studio 5000 Logix Designer are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 2198T-PM001B-EN-P - September 2017
Supersedes Publication 2198T-PM001A-EN-P - April 2017

Copyright © 2017 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.