# Purpose

This document will provide details on the system interaction when aFactoryTalk client application requests data from a ControlLogix controller. Configuration techniques will be examined which can impact system performance. Tools will be introduced which can be used to evaluate system performance, monitor critical resources, and troubleshoot issues for the system.

# Overview of components

There are several key components involved when a FactoryTalk client application requests data from a ControlLogix controller. The components include the Ethernet Network, FactoryTalk Application (which include the FactoryTalk Directory Server, FactoryTalk Application Server, and FactoryTalk Application Client), Rockwell Automation Data Server (RSLinx Enterprise or RSLinx Classic), and the ControlLogix controller. Figure 1 shows a common layout with a network FactoryTalk application.



Figure 1 – Overview of Components

## Ethernet Network

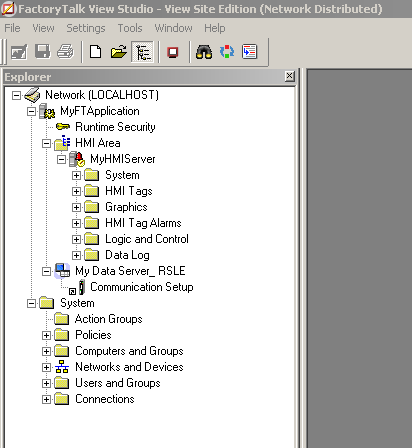
Achieving optimal ControlLogix performance with a FactoryTalk application client relies on having a strong and stable network (as does ***any*** network solution). Ethernet communications rely on a network that does not have noise, interrupted connectivity, excessive collisions, or broadcast storms. A network is only as good as the selected hardware**.**

In addition, if the networking services are configured incorrectly, this may inhibit or adversely affect communication performance. It’s important to remember each network is unique to each company. Most are configured by their IT departments, based on their policies.

If there are issues with a customer’s network configuration, Rockwell Automation has a team, Rockwell Automation Network and Security Services, which can Audit, Analyze, Plan, Design, and implement network solutions. This is a contracted solution which is available for customers.

## Factorytalk Application

The FactoryTalk Application can reside on the same PC or be separated depending on the application type. Local FactoryTalk applications will typically have all parts of the system reside on the same PC. (This may include the Rockwell Automation Data Server). Network FactoryTalk applications can separate their functions and have them run on separate PCs; These include the FactoryTalk Directory Server, FactoryTalk Application Server, and FactoryTalk Application Client(s). See Figure 2 for an example of a FactoryTalk View SE application. Let’s look at the functions of these components.



Rockwell Automation Data Server: RSLinx Enterprise

Factortalk View SE HMI Server

Network FactoryTalk Directory hosted on Local PC

Factortalk Application name:

MyFTApplication

Figure 2 – Example of a Network FactoryTalk Application with FactoryTalk View SE HMI Application Server.

The FactoryTalk Directory Server (FTD) exists whether the FactoryTalk Application type is local or network. The FactoryTalk Directory (FTD) is the centerpiece of the FactoryTalk Services Platform, providing a central lookup service for all servers participating in an application. The FTD server determines which computer is hosting the server(s) when the FactoryTalk application client requests data from a controller. The role of the FTD in the Rockwell environment is analogous to the role of a Domain Controller in the Microsoft Windows environment.

There are many FactoryTalk application servers such as FactoryTalk View SE, Historian SE, FactoryTalk AssetCentre, e.t.c., which among other things provide configuration of tags that will be requested from the contorllogix controller. One of the roles of the Factorytalk application server are to configure which tags in the controller that the FactoryTalk application client will request, the rate that tags will be requested, and the way they will be accessed.

The FactoryTalk application client performs the requests for tag values configured by the FactoryTalk Application server. The FactoryTalk Application Client requests the tag values from the Rockwell Automation data server and once received, provides the information to the FactoryTalk application server.

## Rockwell Automation Data Server

The Rockwell Automation Data Server, which is added to the Application, is the component which services the request for tag values from the FactoryTalk client application, optimizes the request with the controller(s), and returns the values to the client. One of its primary functions is to synchronize the tags that are being requested into tag group (sometimes called optimized packets) with the controller, which will increase performance with the FactoryTalk client application .

## ControlLogix Controller

The Controllogix controller contains all the tags and their values that are being requested and passed to the FactoryTalk application. Based on the bandwidth available, the controller creates packets in alignment with the data server, so the FactoryTalk Client Application can receive tags values at the specified update rate.

# Component Details

Although multiple components have been mentioned, The 2 most important components which affect performance are the ContolLogix Controller and the Rockwell Automation Data Server.

Before we look at what happens when a Client requests a tag value, let’s look at some more details the controller and data server.

## ControlLogix Controller

First, let’s get a basic understanding of the internal structure of a ControlLogix Controller.

### Anatomy of a Logix Controller

The ControlLogix controller contains 2 CPUs and 2 areas of memory. **Figure 3** shows the Anatomy of a ContolLogix Controller.

In the Controllogix controller, memory is separated into two distinct areas. One for Logic and Data and the other for I/O:

* Logic and Data contains memory for the program logic, tag data, and Rockwell Automation data server tag groups (commonly known as Trend Objects)
* I/O contains memory for I/O data, force tables, message buffers, and produced/consumed tags

The ContorlLogix controller uses two 32Bit CPUs which interact to perform control. They are the Logix and Backplane CPU:

* Logix CPU executes application code and performs message processing (Unscheduled or Class 3 communication).
* Backplane CPU communicates with I/O and sends and receives data from backplane (Scheduled or Class 1 communication)
  + Because this CPU is operating independently from the Logix CPU, all I/O information is sent/received asynchronous to program execution.

Because the Logix controllers have 32 bit CPUs, the most efficient operation will occur with 32-bit data structures. (PLC-5s and SLCs were 16 bit.)

In code execution, data type conversion will occur if 32-bit data types are not used. This can increase scan times and memory usage, especially if there is a high amount of conversions occurring.

This is one of the first places to look for optimizing code execution.



Figure 3 - Anatomy of a ControlLogix Controller

## Rockwell Automation Data Servers

Rockwell Automation provides 2 primary data servers:

* RSLinx Enterprise
* RSLinx Classic

RSLinx Enterprise is the preferred data server for all controllers when communicating with FactoryTalk applications. Although it was designed with the ControlLogix controller in mind, it can also be used to communicate with legacy controllers (such as the PLC-5s and SLCs).

RSLinx Classic is the legacy data server and should only be used when features are not available in RSLinx Enterprise. The following features are currently not supported in RSLinx Enterprise:

* Unsolicited messages (available in RSLinx Enterprise 5.70)
* Complex paths are needed (Ethernet to DH+, DH-485)
* Alias Topic Support
* DDE

## System Operation

What happens when a FactoryTalk® Application client request tags from a ControlLogix Controller??

Here is the sequence of events that occur when a FactoryTalk® Application Client requests tag values from a ControlLogix controller:

* The FactoryTalk Client Application requests tags from the controller.
* The Rockwell Automation data server arranges the requested tags into tag groups (sometimes referred to as ControlLogix Optimized Packets)
* The Rockwell Automation Data Server establishes CIP connections to the controller.
  + One connection for Shared Services.
  + Minimum 1 connection for reads (maximum of four depending on the number of tags groups)
  + One connection for writes (created the first time a client writes a tag value to the controller.)
* The ControlLogix Controller creates tag groups (sometimes referred to as trend objects) that correspond to the tag groups created by the data provider.
* Once the handshaking between the data provider and the controller has been performed, communication between the ControlLogix controller and the FactoryTalk Application Client is established, and tag values are updated.

The way tags groups are created is the reason the values update faster than 3rd party data servers that communicate with ControlLogix Controllers.

### Advantages of RSLinx Enterprise over RSLinx Classic:

Both Rockwell Automation data servers, RSLinx Enterprise and RSLinx Classic, have the ability to create tag groups when communicating with the Controllogix controller, but RSLinx Enterprise does have advantages which can affect performance.

RSLinx Enterprise has the following advantages:

* For HMI Applications, tag groups are made inactive rather than destroyed when navigating screens. So, if a screen is revisited, tag groups only need to be reactivated, not recreated. This reduces the amount of time necessary for screen updates which increases performance.
* The Multi-item trend driver (which optimizes the amount of memory used for tag retrieval) provides efficiency for all data types and uses fewer tag groups (or trend objects). Introduced in RSLinx Enterprise v5.21, it results in higher throughput, less loading on the controller, and less traffic on the wire/network.

The interaction between the data server and controller is crucial to performance for FactoryTalk applications.

## Controllogix Controller parameters which affect FactoryTalk® Client Applications:

The primary parameters in the ControlLogix Controller that can significantly impact data throughput are memory and task structure.

### Memory consumed by RSLinx (Runtime)

When a Rockwell Automation data server requests data from a Logix controller, the ControlLogix Controller will work with the data server to create and optimize tag groups. Tag groups in the controller are commonly referred to as Trend Objects. As the controller builds tag groups, additional Data and Logic memory is consumed. It is important to remember that this memory is consumed at runtime and will not be reflected in the RSLogix 5000 memory estimation that is available in the controller properties window. The example below shows an example of this memory consumption.

The screen shots below show the memory consumption of a controller before and after 5000 DINT tags are placed on scan using the RSLinx Live Data Test Client.

**BEFORE AFTER**

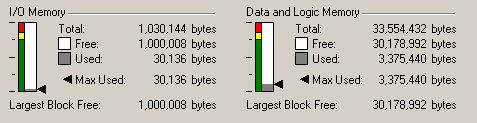
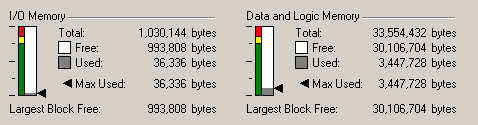
 

Figure 4 – Runtime Memory Consumption

Roughly 70,000 bytes are consumed in the controller when the 5000 tags are on scan. This is due to the tag groups that must be created to service the request from the data server. As the number of tags on scan increases, the amount of memory consumed will also increase. Additionally, when online with RSLogix 5000 or making online edits, runtime memory will be consumed. It is critical to take this into consideration when selecting the correct controller for an application. A good rule of thumb is to ensure 20% of memory in the controller is available at runtime.

### Task Structure

The Logix family of controllers utilizes a pre-emptive multitasking operating system that gives the user the ability to configure and schedule user tasks based on application requirements. Logix allows a maximum of 32 user tasks that can be a combination of continuous, periodic, and/or event tasks. Only one continuous task can exist in an application.

The controller task structure is extremely important when designing an application for optimized RSLinx communication with a Logix controller. Tasks must be configured and scheduled carefully to allow time in the controller to service communications. There are multiple ways to reserve communication time in the controller. These techniques are discussed below.

#### Periodic tasks

If only periodic / event tasks are present in application, the controller will service RSLinx communication when it is not executing logic in a task.

Figure 5 shows an example of how the communication task is executed when using only a periodic or event task. The communication task is only executed when the periodic or event task is NOT executing. If the periodic or event task is not executing and there is no demand for communication, the controller will be in a null state. During this time, the controller is waiting to be triggered to either execute logic or service communication.

Null time can be monitored using many tools including the Logix 5000 Task monitor tool, which will be discussed later in the document. This is a great way to get an indication of the available resources in the controller to service additional class 3 communications. Care should be taken if multiple tasks are present in an application. Be sure to configure them in a way to ensure enough time is available for the communication task.

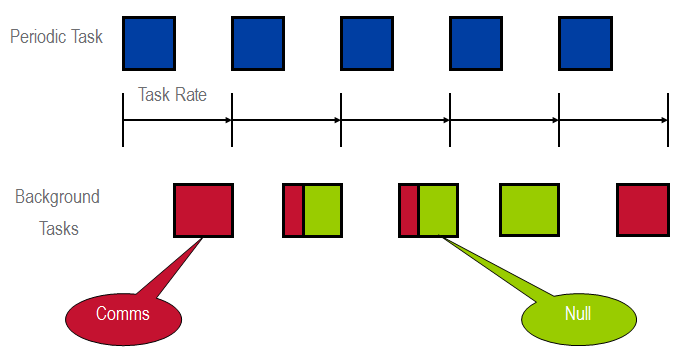


Figure 5 – Periodic Task Operation

#### Continuous tasks

If a continuous task is present in an application, the System Overhead Time Slice (SOTS) is used to adjust the amount of time that the controller will execute the communication task.

Figure 6 shows an example of how the System Overhead Time Slice (SOTS) operates in the controller. The SOTS percentage is based on a 10ms window of execution time.

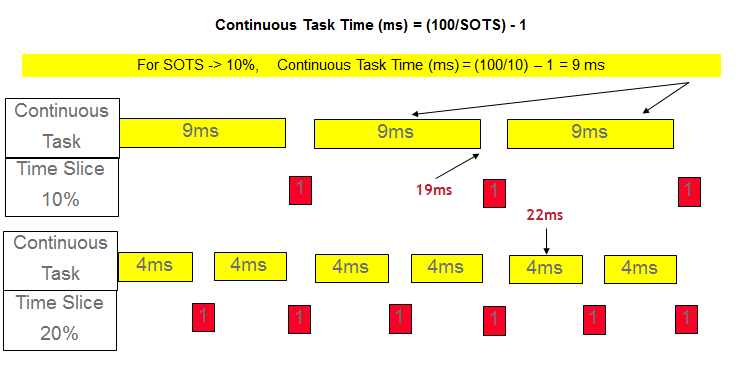


Figure 6 – Continuous Task Operation

In the first example, let’s assume we have 18ms of raw logic to execute. If the SOTS is set to 10%, the controller will execute the continuous task for 9ms and then execute the communication task for 1ms. This sequence will be repeated until the continuous task is completed and then it will begin again. With the SOTS at 10%, the continuous task took 19 ms total to execute 18 ms of logic.

The second example shows the same configuration only the SOTS is set to 20%. In this case, the continuous task will execute for 4ms before executing the communication task for 1ms. This process is repeated until the continuous task is complete. With the SOTS at 20%, the continuous task took 22ms to execute 18 ms of logic. It is important to remember that if a communication load is present and the SOTS is increased, the continuous task scan time will also increase.

When using a continuous task, it is possible to view the amount of SOTS that is available for future class 3 communication loading. By default, if the controller does not require the full 1ms of a SOTS interruption, the controller will return to and continue executing the continuous task.

This setting can be temporarily changed to force the controller to wait until the full 1ms SOTS interrupt is complete before returning to the continuous task. This change can be made on the ‘Controller Properties’ window on the ‘Advanced Tab’ under ‘During unused System Overhead Time Slice’; change the selection from ‘Run Continuous Tasks’ to ‘Reserve for System Tasks, eg Communications’(See Figure 7). When using the Logix 5000 Task Monitor Tool (discussed later) in conjunction with this setting, the unused SOTS will appear as null time.

**Warning**: This setting is ONLY to be used temporarily to view available SOTS. After the SOTS has be analyzed, change the setting back to “Run Continuous Task”. This option should NEVER be left on during normal production.

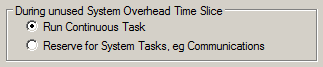


Figure 7 – Verifying SOTS with Continuous Task

## System Performance Monitoring and Maintenance Tools

The tools and utilities for performance monitoring and maintenance have been separated into 3 categories:

* + Logix
  + FactoryTalk
  + Network

### Logix

#### RSLogix5000 Programming Software

In RSLogix5000, the Controller and Module properties can be used as a tool to monitor and maintain a Logix controller. Figure 8 shows examples of the controller and module property windows.

The Controller properties window contains various tabs which are very useful:

* The Major and Minor fault tabs provide fault codes, descriptions, code location, and fault reset capabilities
* The memory tab provides total I/O and Data/Logic Memory Usage
* The Redundancy tab provides redundancy status, partner status, and partner fault information.

The Module properties window can provide useful information such as:

* Module Connection Status
* Module Fault Status (Many fault codes are highlighted and explained in RSLogix5000 Help)
* Detailed Module Information
* In some cases, Comms port diagnostics
* Module specific information

Figure 8 – Controller and Module property Windows

Figure 9 provides examples of Task and Property windows within RSLogix 5000 programming software.

The Task properties window can provide valuable information for monitor and maintenance such as:

* + Elapsed Scan Time: Total amount of time required for a task to complete execution (including all interrupts)
  + Elapsed Time Between Triggers: The amount of time between task execution triggers.
  + Task Overlap Count: The number of times that a task trigger was invoked and the task was not finished with the previous execution scan.

The Program properties window provides similar scan time information, however there are some subtle differences:

* + The scan time displayed is the total execution time. This is defined as the total amount of “raw” time the program took to execute. NO interrupts are factored into this calculation.

Figure 9 - Program and Task Property Windows

#### Logix5000 Task Monitor Tool

The Logix5000 Task Monitor tool provides an overview of the activities that a Logix5000 controller is executing. It has divided the activities into the following 8 categories:

* User Tasks
* Performance
* Connections
* Messages
* RSLinx OP
* Comms
* Modules (Local Comms Modules)
* Redundancy (when enabled).

Figure 10 shows an example of the ‘Users Tasks’ tab of the Logix 5000 task monitor tool. The Users Task tab provides vital information on task type, status, execution, scan times, overlaps, CPU Utilization, priority, and much more. Selecting the “enable program monitor” checkbox will display detailed program breakdowns of each task.

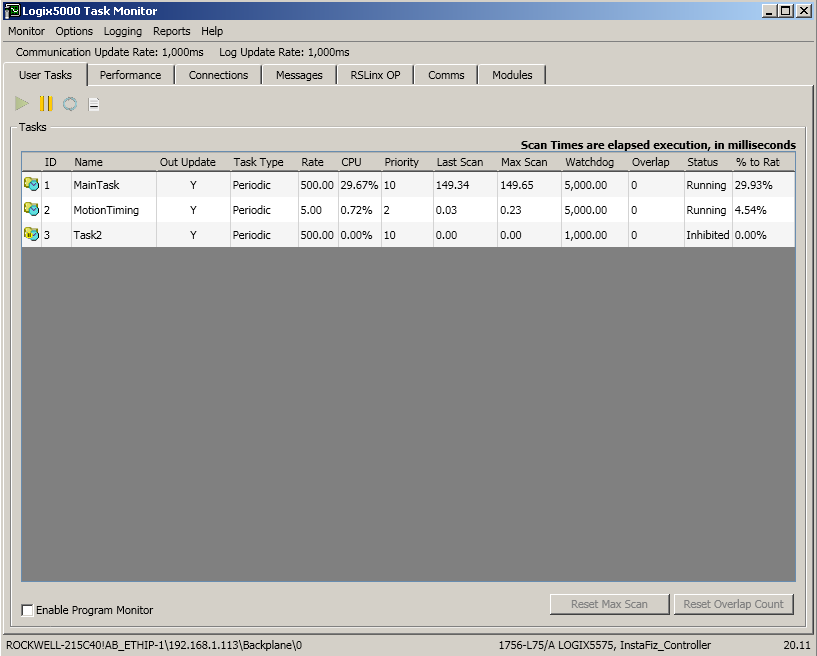


Figure 10 – User Tab of Logix 5000 Task Monitor Tool

Figure 11 shows an example of the ‘Performance’ tab of the Logix 5000 Task Monitor tool. The performance tab provides quick and easy access to the processes and utilization for a given controller.

* + Memory Usage – Viewed as a bar graph on the left (% of total memory used) or a detailed breakout at the bottom
  + CPU Utilization – Viewed as a bar graph on the left (Includes all processes) or see a breakout on the pie graph.
  + Communications – contains the controller utilization required for all class 3 communication (RSLinx, MSGs, etc.)
  + Null Time – Shows the percent of time the controller is idle. This information can be key in monitoring heavy FactoryTalk application and MSGing traffic. Note: For optimal FT application performance, continuous tasks must be designed to allow for controller null time.

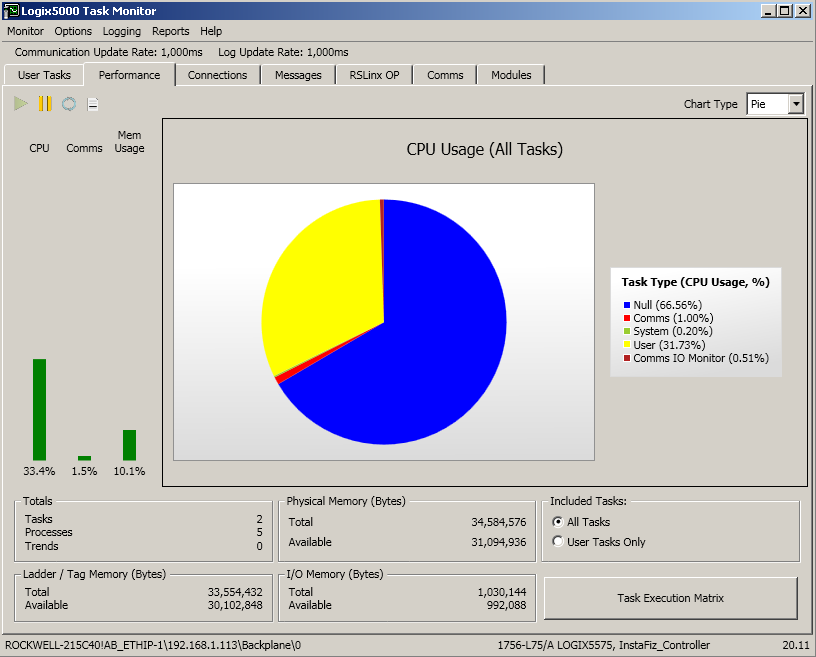


Figure 11 – Performance tab of Logix5000 task Monitor tool

The latest version of the Logix 5000 Task Monitor tool can be downloaded from Answer ID: [51737 - Logix 5000 Task Monitor Tool](https://rockwellautomation.custhelp.com/app/answers/detail/a_id/51737)