**PLC Reader Application Documentation**

Contents:

**1.Introduction**

Overview of the PLC Reader Application and its purpose in industrial automation.

Scope of the application in terms of interfacing with PLCs, data extraction, storage, and analytics.

Goals of the PLC Reader Application, including improving efficiency, reducing costs, and facilitating better decision-making.

**2.PLC Communication and Display Documentation**

Explanation of classes and methods for managing connections with PLCs and displaying connection status.

Description of key methods, such as establishing PLC connections, reading data, and updating the user interface.

Details on the event methods and their functionalities, such as connecting/disconnecting PLCs and updating data views.

**3.Reports Generation**

Overview of the Reports Generation component for creating PDF and CSV reports of PLC data.

Description of the GeneratePdfReport method, including data retrieval, PDF generation, and file naming conventions.

Explanation of the GetImageDataFromResource method for retrieving embedded image data.

Details on event handlers for generating PDF and CSV reports and their interactions with user input.

**4.Databases Documentation**

Overview of the Databases region and its role in interacting with a SQLite database.

Description of key methods, such as creating the database table, saving data, and retrieving time and value data.

Explanation of the database schema, including column names and their respective data types.

**5.Graphs Documentation**

Introduction to the Graphs region and its purpose in visualizing data stored in the SQLite database.

Description of the event method for creating line charts based on user input.

Explanation of the CreateChart method, including chart configuration, data retrieval, and plotting data points.

Details on the chart control used and its properties for displaying the line chart.

By organizing the documentation into sections, users can easily navigate to the desired topic and find relevant information about the PLC Reader Application's functionalities and usage.

**-Introduction**

The Programmable Logic Controller (PLC) Reader Application is a Windows Forms application designed to interact with and extract data from PLCs. In the complex world of industrial automation, PLCs form the backbone of many systems, controlling a vast array of processes. However, their utility is only as good as the ability to understand, analyze, and act on the data they generate. That's where the PLC Reader Application comes in.

**-Scope**

The PLC Reader Application is designed to interface with PLCs, extract valuable data, store that data for analysis, and then present the user with insightful analytics. This provides a comprehensive solution that enables users to effectively monitor, control, and optimize their industrial processes.

While many applications may only allow for simple read/write operations or barebones data visualization, the PLC Reader Application goes above and beyond. It provides users with robust capabilities for real-time monitoring, historical data storage, and in-depth analytics.

**-Overview**

This application interacts directly with PLCs, utilizing the S7.Net library to facilitate communication. The PLC Reader Application is capable of establishing and managing multiple PLC connections simultaneously, allowing for broad system visibility.

The data collected from the PLCs is stored in a local database, ensuring that historical data is available for analysis. This is crucial for identifying trends, diagnosing problems, and optimizing system performance.

**-Goals**

The primary goals of the PLC Reader Application are to:

--**Improve efficiency**: By providing real-time data and historical trends, the PLC Reader Application allows users to identify inefficiencies and bottlenecks in their systems, enabling them to make changes that optimize system performance.

**--Reduce costs**: With better visibility into system performance, users can identify and address issues more quickly, reducing downtime and maintenance costs.

**--Facilitate better decision-making**: Armed with robust analytics and insights, users can make more informed decisions about their systems, improving overall operational effectiveness.

**-Functionalities**

The PLC Reader Application boasts several key functionalities:

**--PLC Communication and Display**: The application can establish connections with multiple PLCs, read data from these PLCs, and display this data in real-time.

**--Data Storage**: The application can store PLC data in a local database, preserving historical data for future analysis.

**--Data Analysis and Reporting**: The application provides robust data analysis capabilities, allowing users to generate reports that highlight key performance indicators and trends.

**--Graphical Visualization**: The application provides graphical visualization of data, making it easier to understand trends and patterns.

By combining these functionalities into a single, user-friendly application, the PLC Reader Application serves as a powerful tool for anyone working with PLCs and industrial automation systems.

**PLC Communication and Display Documentation**

This region contains classes and methods for managing connections with Programmable Logic Controllers (PLCs), displaying connection status, and reading data from PLCs.

**-Classes**

**--PlcInfo**

This class encapsulates information about a PLC, including an instance of the PLC, its name, and a System.Windows.Forms.Timer associated with it. It has a constructor that takes these three parameters to create an instance of PlcInfo.

**-Methods**

**--button\_Connect\_PLC\_Click(object sender, EventArgs e)**

This event method is invoked when the "Connect PLC" button is clicked. It reads the CPU type, IP address, rack number, and slot number from the user interface, then creates a new Plc instance and attempts to open a connection to it. If the connection is successful, it updates the connection status label, starts a new Timer with a 1-second interval, and adds a new PlcInfo object to the plcs list. It also adds a new item to the listView\_ConnectedPLCs list view.

**--Timer\_TickForPlc(Plc plc, string plcName)**

This method is invoked on each tick of a PLC's Timer. If the PLC is connected, it reads data from the PLC for each item in the listView\_DB\_Locations list view and displays it in the dataGridView\_PLCValues data grid view. It also handles starting and stopping the saveDataTimer Timer.

**--button\_Disconnect\_Selected\_PLC\_Click(object sender, EventArgs e)**

This event method is invoked when the "Disconnect Selected PLC" button is clicked. It checks whether a PLC is selected in the listView\_ConnectedPLCs list view. If so, it stops and disposes of the Timer associated with the selected PLC, closes the PLC connection, removes the PLC from the plcs list, and removes it from the listView\_ConnectedPLCs list view. It also removes all rows associated with the PLC from the dataGridView\_PLCValues data grid view.

**--UpdateDataGridView(string plcName, string itemName, string db, string startByteAdr, string dataType, string valueRead, string unitOfMeasurement)**

This method updates the dataGridView\_PLCValues data grid view with the passed parameters. If a row with the same PLC name and item name is found, it updates the value read. If no such row is found, it adds a new row with the passed parameters.

--**button\_Add\_Click(object sender, EventArgs e)**

This event method is invoked when the "Add" button is clicked. It reads the name, DB, location, data type, and unit of measurement from the user interface, then adds a new item to the listView\_DB\_Locations list view with this information.

**--button\_Remove\_Click(object sender, EventArgs e)**

This event method is invoked when the "Remove" button is clicked. It checks whether an item is selected in the listView\_DB\_Locations list view. If so, it removes all rows associated with the selected item from the dataGridView\_PLCValues data grid view and removes the selected item from the listView\_DB\_Locations list view

**Reports Generation**

The Reports Generation component of the PLC Reader Application provides functionality for creating PDF and CSV reports of PLC data over a specified time period.

**- GeneratePdfReport(SysDateTime startTime, SysDateTime endTime)**

This method generates a PDF report of PLC data between the provided startTime and endTime. The report includes a timestamped list of item names and their corresponding values and units of measurement. The report also includes images of a ship and the Kronospan logo.

This method first reads data from the SQLite database and pivots the data table to create a new table with one row per timestamp and one column per unique item name. The PDF report is then created using the iTextSharp library, with the images and pivoted data table included.

The PDF report is saved to the user's desktop with the filename in the format PLCValues\_StartTime\_EndTime.pdf.

**- GetImageDataFromResource(string resourceName)**

This method retrieves image data from a specified resource embedded in the assembly. The resourceName parameter should be the name of the image file in the Resources folder.

**-button\_generateReport\_Click(object sender, EventArgs e)**

This event handler is triggered when the user clicks the "Generate Report" button. It reads the start and end times from the respective text boxes, and then calls the GeneratePdfReport method with these times.

**-button\_generateCSV\_Click(object sender, EventArgs e)**

This event handler is triggered when the user clicks the "Generate CSV" button. It reads the start and end times from the respective text boxes, and then calls the GenerateCSVReport method with these times.

**-GenerateCSVReport(SysDateTime startTime, SysDateTime endTime)**

This method generates a CSV report of PLC data between the provided startTime and endTime. The report includes a timestamped list of item names and their corresponding values and units of measurement.

This method first reads data from the SQLite database and pivots the data table to create a new table with one row per timestamp and one column per unique item name. The CSV report is then written to a file using the StreamWriter class.

The CSV report is saved to the user's desktop with the filename in the format PLCValues\_StartTime\_EndTime.csv.

By using these methods, users can easily generate PDF or CSV reports of PLC data for any specified time period, helping improve efficiency and facilitate better decision-making.

**Databases Documentation**

The Databases region contains methods for interacting with a SQLite database that stores PLC data.

**-Methods**

**--CreateDataTable()**

This method establishes a connection to a SQLite database and creates a table named PLCValues, if it does not exist. The table has columns for Id, PLCName, ItemName, DB, StartByteAdr, DataType, ValueRead, UnitOfMeasurement, and Timestamp.

**--listView\_DB\_Locations\_SelectedIndexChanged(object sender, EventArgs e)**

This event method is invoked when the selected item in listView\_DB\_Locations is changed. It populates form fields with data from the selected item.

**--CreateDatabaseFile()**

This method checks if a SQLite database file with the name PLCValues.db exists. If it doesn't, the method creates the file.

**--SaveDataToDatabase(string plcName, string itemName, string db, string startByteAdr, string dataType, string valueRead, string unitOfMeasurement)**

This method saves a record to the PLCValues table in the SQLite database. It inserts a new row with the provided PLC name, item name, DB, start byte address, data type, value read, unit of measurement, and the current timestamp.

**--SaveAllDataToDatabase(Plc plc, string plcName)**

This method loops over all the items in the listView\_DB\_Locations, reads the values from the PLC, and saves them to the SQLite database by invoking the SaveDataToDatabase method.

**--FetchTimeAndValueData(string itemName)**

This method fetches timestamped value data for a given item name from the PLCValues table. It returns a list of tuples, where each tuple contains a timestamp and a value.

**Database Schema**

The PLCValues table in the SQLite database has the following structure:

**Id**: An integer primary key that auto-increments.

**PLCName**: A text field that holds the name of the PLC from which the value was read.

**ItemName**: A text field that holds the name of the item.

**DB**: A text field that stores the DB number.

**StartByteAdr**: A text field that stores the start byte address.

**DataType**: A text field that stores the data type.

**ValueRead**: A text field that stores the value read from the PLC.

**UnitOfMeasurement**: A text field that stores the unit of measurement for the value.

**Timestamp**: A text field that stores the timestamp at which the value was read.

Graphs Documentation

The Graphs region contains methods for visualizing the data stored in the SQLite database as a line chart.

**-Methods**

**--buttonCreateChart\_Click**(object sender, EventArgs e)

This event method is invoked when the "Create Chart" button is clicked. It reads the ItemName from a textBox, connects to the SQLite database, and retrieves all rows with the specified ItemName, ordered by Timestamp. If data is found, a DataTable is loaded with the data and passed to the CreateChart method. If no data is found, a message box is displayed to the user.

**--CreateChart(DataTable dataTable)**

This method creates a line chart in the chart1 Chart control. It first clears any existing series, sets the X-axis label to "Timestamp" and the Y-axis label to "ValueRead", and configures the X-axis to display dates in the "yyyy-MM-dd HH:mm:ss" format. The X and Y axes are set to auto-scale based on the data points.

A new Series named "ValueRead" is created and added to the chart. The XValueType property of the series is set to DateTime, and the ChartType property is set to Line.

Next, the method loops through the rows of the passed DataTable, parsing the Timestamp and ValueRead values, and adds these as points to the series. Finally, the method forces the chart to recalculate the axis scales.

**Chart Control**

The chart control used in this region is the System.Windows.Forms.DataVisualization.Charting.Chart control. The chart is a line chart that plots the ValueRead (Y-axis) against the Timestamp (X-axis). The X-axis is labeled with dates in the "yyyy-MM-dd HH:mm:ss" format. The chart automatically adjusts the scale of the X and Y axes based on the data points.

.