CODESYS STORE

CSV Utility

Die Bibliothek CSV Utility ermöglicht das Lesen und Schreiben von CSV Dateien mit einer CODESYS Steuerung.

Die Bibliothek ‚CSV Utility' ist jetzt Teil des Produktes IIoT Libraries SL und ist als Einzelprodukt nicht mehr verfügbar.

Produktbeschreibung

Die Bibliothek CSV Utility stellt Funktionsbausteine zum Lesen und Schreiben von CSV Dateien zur Verfügung. Die Bibliothek besteht aus den Komponenten `CSVReader` (Lesen von CSV Dateien) und `CSVWriter` (Schreiben von CSV Dateien). Sämtliche Funktionsbausteine können klassisch oder objektorientiert verwendet werden. In einem Bespielprojekt wird die Verwendung der Bibliothek ausführlich demonstriert.

Lesen einer CSV Datei

Im Ordner `CSVReader` befinden sich die Komponenten zum Lesen von CSV Dateien. Die Initialisierung und die Übergabe des Dateipfades erfolgt über den Funktionsbaustein `CSVReaderInit`. Die Werte der CSV Datei können einzelnen, zeilenweise oder komplett gelesen werden. Das Lesen der Werte erfolgt über die Funktionsbausteine `NextLine`, `NextElement` oder `ReadAll`. Die Verwendung wird in dem Beispielprojekt `CSVUtilityExamples.project` in der Applikation `CSVReaderExample` veranschaulicht. Die maximale Elementlänge muss größer oder gleich der maximalen Spaltenlänge der CSV Datei sein.

Schreiben einer CSV Datei

Im Ordner `CSVWriter` befinden sich die Komponenten zum Schreiben von CSV Dateien. Das Speichern der Daten erfolgt in drei Schritten. Im ersten Schritt wird die Bibliothek initialisiert, dabei wird der Pfad und der Dateiname übergeben. Im nächsten Schritt werden die Daten in einem Puffer abgelegt, im dritten Schritt wird der Puffer als CSV Datei geschrieben.

Klassische Programmierung

Die Funktionsbausteine für die klassische funktionsbausteinorientierte Programmierung liegen im Ordner `FunctionBlock`. Zu jedem Funktionsbaustein wurde zusätzlich ein Visualisierungsobjekt angelegt, das in weiteren Projekten verwendet werden kann. Ein Beispiel dazu ist im Projekt `CSVUtilityExamples.project` in der Applikation `CSVWriterCFCExample` zu finden.

Objektorientierte Programmierung

Im Ordner `Objects` liegen die Funktionsbausteine und Interfaces für die objektorientierte Programmierung. Ein Beispiel dazu ist im Projekt `CSVUtilityExamples.project` in der Applikation `CSVWriterSTExample` zu finden.

Funktionsbausteine im Ordner FunctionBlocks

Die Funktionsbausteine in diesem Ordner sind für eine asynchrone Verarbeitung ausgelegt und sind von dem Funktionsbaustein `CBM.ETrigA` abgeleitet.

Initialisierung:

Die Initialisierung erfolgt über den Funktionsbaustein `Init`. Hier kann der Speicherpfad und der Dateiname übergeben werden.

Hinzufügen von Werten:

Werte können über die Funktionsbausteine `AddXXX` in den Puffer geschrieben werden. Eine neue Zeile kann über den Baustein `NewLine` hinzugefügt werden, eine Datei wird mit `NewFile` angelegt.

Speichern:

Das Speichern des Puffers erfolgt über den Baustein `WriteFile`. Die Werte werden im Standardformat des

jeweiligen Datentyps geschrieben.

Visualisierungen:

Zu jedem Funktionsbaustein existiert eine Visualisierung der Ein- und Ausgänge.

Hinweis zur Verwendung der Bausteine:

Sämtliche Funktionsbausteine im Ordner `FunctionBlocks` reagieren auf eine ansteigende Flanke am Eingang `xExecute`. Bitte beachten Sie, dass diese Operationen asynchron ausgeführt werden und die Ausgänge `xDone`, `xError`, `eError` entsprechend abgefragt werden müssen. Die Abarbeitung muss nicht in einem Zyklus erfolgen! (Verwendung siehe auch CAA Richtlinien in der CODESYS Online Hilfe)

Funktionsbaustein CSVWriter (Ordner Objects)

Der Funktionsbaustein `CSVWriter` stellt grundlegende Methoden zum Speichern von CSV Dateien zur Verfügung.

Hinzufügen von Werten:

Beim Aufruf einer der Add-Methoden, wird der Wert in den Puffer geschrieben. Eine neue Zeile wird angehängt, sobald die entsprechende Methode aufgerufen wird. Ist der Puffer vollgeschrieben, wird ein Fehler zurückgegeben. Sobald Daten gespeichert wurden, besteht wieder Platz für neue Werte.

- `AddXXX`: Diese Methoden fügen die Daten am Ende des Puffers ein.
- `NewLine`: Hängt eine neue Zeile an.

Speichern:

- Save: Speichert den Wert des Puffers als CSV Datei.

Zu Beginn muss die `InitSave` Methode aufgerufen werden, um eine neue Datei anzulegen oder eine vorhandene Datei zu öffnen.

Weitere Methoden:

- `GetFileSize`: Diese Methode gibt die Größe der aktuellen Datei zurück.
- `NewFile`: Die Methode erzeugt eine neue Datei (mit einem gegebenen oder automatisch generierten Namen).

Fehler:

Alle Methoden besitzen ein Fehlerfeld, in dem ein möglicher Fehler vom Typ `CSV_ERROR` gespeichert wird.

Beispielprojekt CSVUtilityExamples

Das Beispielprojekt `CSVUtilityExamples.project` enthält eine Beispielapplikation zum `CSVReader` und zwei Beispielapplikationen zum `CSVWriter`.

Applikation CSVReaderExample

Das Beispiel zeigt die drei Möglichkeiten, wie Werte einer CSV Datei gelesen werden können. In dem Programm `Prog` wird die Verwendung der Funktionsbausteine `CSVReaderInit`, `NextElement`, `ReadAll` und `NextLine` veranschaulicht. Der Inhalt der Datei `c:/temp/CSVReader.csv` wird in einer Visualisierung dargestellt.

Applikation CSVWriterCFCExample

Dieses Beispiel zeigt anhand eines CFC Beispiels, wie die Funktionsbausteine der Bibliothek `CSVWriter` verwendet werden können. Dabei wird auch die Verwendung der integrierten Visualisierung demonstriert.

Funktionsweise:

Das Programm `WriteValues` erzeugt eine CSV Datei im Ordner `c:/temp/CSVWriterCFCExampleData.csv` . Das Schreiben der Datei wird über die Visualisierung beim Klick auf den Button Click here to add angestoßen. Die Initialisierung erfolgt über den Baustein `CSV.Init`. Der Pfad und der Dateiname werden hier direkt übergeben. Die Bausteine `AddWord`, `AddString` und `NewLine` demonstrieren das Hinzufügen von Werten zu dem internen Puffer. Der Counter `ctu` zählt bei jedem Event die Zeilennummer hoch. Nach dem Hinzufügen einer neuen Zeile werden die Werte über den Funktionsbaustein `WriteFile` in eine CSV Datei geschrieben.

Applikation CSVWriterSTExample

Dieses Beispiel zeigt, wie Variablenwerte über einen objektorientierten oder der sequentiellen Weg in eine CSV Datei geschrieben werden können. Jeder Ablauf wird in zwei Tasks ausgeführt. Eine Task schreibt die Werte in den Puffer und die andere speichert die Daten aus dem Puffer in die Datei.

Funktionsweise:

- `ObjectOriented_Save` (Objektorientiertes Speichern):

  Dieses Programm ruft die `Save` Methode von `DataObject` auf, in welcher der Speichervorgang implementiert ist.

- `ObjectOriented_Write` (Objektorientiertes Schreiben):

  Dieses Programm ruft die `Write` Methode von `DataObject` auf, in welcher der Schreibvorgang implementiert ist.

- `DataObject`:

  Dieser Funktionsblock implementiert das Interface `ICSVObject`. Die `Write` Methode beinhaltet die lokalen Variablen, welche später gespeichert werden. Für jeden Datentyp wird die entsprechende `AddXXX` Methode aufgerufen. Nach einer gewissen Anzahl von Zeilen wird eine neue Datei mit anderen Werten erstellt. Sobald ein Fehler auftritt, wird der Error Status erreicht und nichts mehr geschrieben. Die `Save` Methode implementiert den Speichervorgang. Zu Beginn wird die Methode `InitSave` der `CSVWriter` Instanz aufgerufen. Danach muss nur noch die `Save` Methode aufgerufen werden.

  Um die folgenden Programme zu aktivieren, müssen die POUs in den Tasks geändert werden.

- `Sequential_Save` (Sequentielles Speichern):

  Der Speichervorgang muss ebenso mit der Methode `InitSave` gestartet werden. Nachdem die CSV Datei eine bestimmte Größe erreicht hat, wird eine Neue erzeugt. Wird kein Dateiname angegeben, so wird der bestehende Name mit angehängter Nummerierung verwendet.

- `Sequential_Write` (Sequentielles Schreiben):

  Es werden unterschiedliche Variablen in den Puffer geschrieben. Ist dieser voll, wird der Fehler `EndOfBuffer` zurückgegeben. Im Gegensatz zum obigen Schreiben, wird diesmal der Fehler ignoriert und der Puffer weiter beschrieben. Somit gehen einige Daten verloren, aber das Programm läuft weiter. Sobald der nächste Speichervorgang gestartet wurde, ist der Puffer wieder frei.

Allgemeine Informationen

Lieferant:
CODESYS GmbH
Memminger Straße 151
87439 Kempten
Deutschland

Support:
https://support.codesys.com

Artikelname:
CSV Utility
Artikelnummer:
2111000006
Vertrieb:
CODESYS Store
https://store.codesys.com

Lieferumfang:
CODESYS Package mit Bibliothek und Beispielprojekt

Systemvoraussetzungen und Einschränkungen

Programmiersystem	CODESYS Development System V3.5.14.0
Laufzeitsystem	CODESYS Control V3.5.14.0
Unterstützte Plattformen/ Geräte	Alle Hinweis: Verwenden Sie das Projekt Device Reader, um die von der Steuerung unterstützten Funktionen zu ermitteln. Device Reader ist kostenlos im CODESYS Store erhältlich.
Zusätzliche Anforderungen	-
Einschränkungen	-
Lizenzierung	Arbeitsplatzlizenz
Erforderliches Zubehör	CODESYS Key für CODESYS < 3.5.14.0

Bitte beachten Sie: Nicht alle CODESYS-Funktionen sind in allen Ländern verfügbar. Weitere Informationen zu diesen länderspezifischen Einschränkungen erhalten Sie unter sales@codesys.com.
Bitte beachten Sie: Technische Änderungen, Druckfehler und Irrtümer vorbehalten. Es gilt der Inhalt der aktuellen Online-Version dieses Dokuments.