▲ Hochschule Harz
Hochschule für angewandte Wissenschaften

automotive engineering iav

Bachelorarbeit

Studiengang CDHAW MT Dual

Entwicklung eines Data-Logging-Systems für Industriesteuerungen

Yizhen Li

Matrikelnummer 27322

Erster Prüfer:	Prof. Dr. Bernhard Zimmermann
Zweiter Prüfer:	Dr.-Ing. Bennet Luck
Arbeit vorgelegt am:	24.08.2020
Durchgeführt bei:	IAV GmbH
	TP-D81
	Rockwellstraße 16
	38518 Gifhorn
Betriebliche Betreuer:	Dipl.-Ing. (FH) André Patrick Hoppe M. Eng.
Anschrift der Verfasserin:	Yizhen LI
	Seilerstraße 21. 38440 Wolfsburg
	liyizhen233@gmail.com

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich zur Anfertigung der vorliegenden Arbeit keine anderen als die angegebenen Quellen und Hilfsmittel und keine nichtgenannte fremde Hilfe in Anspruch genommen habe. Mir ist bewusst, dass eine falsche Versicherung rechtliche Konsequenzen hat.

gilt.

Ort:

Datum:

Unterschrift:

▲ Hochschule Harz
Fachbereich Automatisierung und Informatik

Thema und Aufgabenstellung der Bachelorarbeit
BA AI 58/2020

für Yizhen Li

Entwicklung eines Data-Logging-Systems für Industriesteuerungen

Die IAV GmbH entwickelt Applikationssoftware, die auf Industriesteuerungen laufen, für unterschiedliche Anwendungen, unter anderem Echtzeitbetriebsführungen für Windenergieanlagen (Stand-alone-Anwendung). Um das Verhalten der Applikationen zu analysieren und Störungsanalyse zu betreiben, ist es erforderlich bestimmte Daten aufzuzeichnen. Moderne Industriesteuerungen unterschiedlicher Hersteller kommen zum Einsatz.

Ziel dieser Bachelorarbeit ist es, ein geeignetes Data-Logging-System zu entwerfen. Das Data-Logging-System soll von der Applikationssoftware erzeugte Daten erfassen und diese Daten über einen längeren Zeitraum speichern. Datenverluste sind während des Data Logging Vorgangs zu vermeiden. Da Industriesteuerungen unterschiedlicher Hersteller zum Einsatz kommen, soll das Data-Logging-System möglichst plattformunabhängig sein. Analyseprogramme wie Matlab sollen auf die geloggten Daten zugreifen können. Mit den Analyseprogrammen werden aufgezeichnete Daten in vom Anwender spezifizierten Zeiträumen betrachtet.

Diese Bachelorarbeit beinhaltet folgende Teilaufgaben:

- Anforderungsanalyse an ein Data Logging System
- Erstellen eines Pflichtenheftes → Arbeitsplan
- Ausarbeitung eines Data-Logging-Konzepts
    - Recherche von Data-Logging-Möglichkeiten
    - Untersuchung und Bewertung der Data-Logging-Möglichkeiten
    - Erarbeiten einer geeigneten Schnittstelle zwischen aus Simulink generierten Programmobjekten und der Logging-Funktionalität → Data-Pipelining von der Produktivsoftware (komplexe Betriebsführung) zum Logging-System
    - Ablagekonzept der geloggten Daten, z. B. Datenbanken
    - Konzept zum lückenlosen Abruf der Messreihen mit hoher Verfügbarkeit

- Data-Logging PC
    - Konzepterstellung eines Softwaretools zur Aufbereitung der Logging-Daten
        - Abruf der von der Industriesteuerung aufgenommenen Messreihen (z.B. tragbare Datenträger)
        - Zusammenfügen der aufgenommenen Messreihen
        - Archivieren der Messreihen

- Prototypische Realisierung des Data-Logging-Konzepts an einer Beckhoff-Industriesteuerung
    - Export der Data-Logging-Testumgebung auf eine Industriesteuerung
    - Realisierung einer ausgewählten Data-Logging-Möglichkeit
    - Erprobung an einer Data-Logging-Testumgebung

Prof. Dr. Zimmermann
Erstprüfer

Dr. Bennet Luck
Zweitprüfer

Abstract

Die zur Steuerung einer Maschine verwendete Anwendung läuft auf einer Industriesteuerung. Um die Performance der Anwendung, wie z.B. die Stabilität, die Reaktionszeit der Anwendung oder die Ursache einer Störung zu analysieren, müssen Daten von der Industriesteuerung aufgezeichnet werden.

Im Rahmen dieser Bachelorarbeit wird die Erstellung von einem Data-Logging-System mittels Software-SPS von Beckhoff für einen Testanwendungsfall mit ein Simulink Model als Datenquelle untersucht. Dieses System kann Daten aus der SPS-Anwendungen aufzeichnen und für einen längeren Zeitraum speichern.

Verschiedene Data-Logging-Varianten für SPSen werden betrachtet und verglichen. Anschließend werden eine mit XML in IEC 61131-3 Programmierte Data-Logging-Funktion und eine andere in C++ selbstprogrammierte Data-Logging-Funktion erstellt und in einer Testanwendung untersucht.

Inhaltverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

IPC	Industrie-PC
SPS	Speicherprogrammierbare Steuerung
CPU	Central Processing Unit
FDR	flight data recorder
XAE	eXtended Automation Engineering
XAR	eXtended Automation Runtime
SDK	Software Development Kit
ODK	Open Data Kit
FB	Funktionsblock, Funktionsbaustein
FUP	Funktionsplan
KOP	Kontaktplan
AWL	Anweisungsliste
ST	Strukturierter Text
MQTT	Message Queuing Telemetry Transport
OPCUA	Open Platform Communications Unified Architecture
XML	Extensible Markup Language
IO	Input/Output

verwendete Software

Windows Professional 7

Microsoft Visual Studio 2017 Professional (x86)

MATLAB R2019a

TwinCAT 3.1 – eXtended Automation Engineering (XAE) version 3.1.4024.10

TwinCAT 3.1 – eXtended Automation Runtime (XAR) version 3.1.4024.10

Beckhoff TF 6421 XML Server version 3.1.26.0

verwendete Hardware

Beckhoff cx 2040-0130 Embedded-PC mit Windows Embedded Standard 7 P (64-Bit)

1 Einleitung

Die zur Steuerung einer Maschine verwendete Anwendung läuft auf einer Industriesteuerung. z.B. eine Anwendung zur Steuerung von Windkraftanlagen. Sie erhält Daten, z.B. die Windgeschwindigkeit und die Generatordrehzahlen, von den an die Industriesteuerung angeschlossenen Sensoren. Die Daten werden in der Anwendung analysiert und die an der Industriesteuerung angeschlossenen Aktuatoren werden betätigt, z.B. die Drehzahl der Turbine wird gesteuert. Anwendungen werden in Simulink erstellt und laufen auf der Industriesteuerung.

Die verwendeten Industriesteuerungen sind Speicherprogrammierbare Steuerungen (SPS). Eine SPS ist eine digitale Betriebssteuerung mit einem Mikroprozessor zur automatischen Steuerung. Sie ist echtzeitfähig und besteht aus Komponenten wie der CPU, dem Befehls- und Datenspeicher, der Eingabe- und Ausgabeschnittstelle, der Stromversorgung und Digital-Analog-Wandlern (vgl. [1]). Die Firmen Beckhoff, Siemens und Phoenix Contact sind Hersteller von SPSen.

Um die Performance der Anwendung, wie z.B. die Stabilität, die Reaktionszeit der Anwendung oder die Ursache einer Störung zu analysieren, müssen Daten von der Industriesteuerung aufgezeichnet werden. Die aufzuzeichnenden Daten umfassen die Ein- und Ausgänge der Industriesteuerung und in der Anwendung berechnete Daten. Diese Daten werden auf einem Speicher außerhalb der Steuerung zur späteren Anwendung langfristig gespeichert.

Das Ziel dieser Abschlussarbeit ist es, ein geeignetes Data-Logging-Systems zu entwerfen. Dieses System kann Daten aus der SPS-Anwendungen aufzeichnen und für einen längeren Zeitraum speichern. Die gespeicherten Daten können von einer Analysesoftware wie MATLAB verwendet werden.

Die Abschlussarbeit umfasst folgende Punkte:

- Grundlagen des Data-Loggings
- Anforderung an das Data-Logging-System
- Untersuchung und Vergleich von Varianten zur Umsetzung eines Data-Logging-Systems
- Aufbau des Data-Logging-Systems
- Test des entwickelten Data-Logging-Systems
- Auswertung des Data-Logging-Systems

2 Grundlagen des Data-Loggings

„Data-Logging“ ist ein englischer Begriff und heißt in der deutschen Übersetzung „Datenerfassung“. Laut [2] sind Daten „durch Beobachtungen, Messungen, statistische Erhebungen u. a. gewonnene Werte“. Erfassung bedeutet laut [2]: „in einen Computer einzugeben“. Beim Data-Logging werden Daten von einem Gerät oder System über einen bestimmten Zeitraum, in einem definierten Zeitintervall gesammelt. Die gesammelten Daten dienen als Grundlage für Datenauswertungen.

Das Data-Logging wird in zwei Varianten unterteilt: manuelles Data-Logging und automatisches Data-Logging.

2.1 Manuelles Data-Logging

Die einfachste Art des Data-Loggings besteht darin, die Daten manuell aufzuzeichnen. Messwerte werden von einem analogen oder digitalen Messgerät erzeugt. Im definierten Zeitintervall werden die von dem Messgerät erzeugten Messwerte aufgezeichnet, dabei werden die vom Messgerät angezeigten Messwerte abgelesen und diese Werte auf Papier aufgezeichnet oder in einen Computer eingegeben.

Ein Beispiel für das manuelle Data-Logging ist die Durchführung eines physikalischen Experiments. Wenn die versuchsdurchführende Person im physikalischen Experiment die Zeitabhängigkeit einer physikalischen Größe ermitteln möchte, muss die physikalische Größe in regelmäßigen Zeitabständen gemessen und aufgezeichnet werden. Mit Hilfe dieser aufgezeichneten Zeitreihe wird die Zeitabhängigkeit der physikalischen Größe bestimmt.

Der Vorteil der manuellen Datenerfassung ist, dass kein zusätzliches Gerät zum Aufzeichnen von Daten benötigt wird. Die manuelle Datenerfassung hat jedoch Nachteile. Erstens ist die Messgenauigkeit nicht hoch. Zu der Messabweichung des Messgeräts kommt die Messabweichung des Menschen hinzu. Die Messabweichung und die Wiederholgenauigkeit des Menschen sind von Mensch zu Mensch unterschiedlich. Zweitens ist die manuell aufgezeichnete Datenmenge durch den Menschen begrenzt. Das Zeitintervall kann nicht beliebig klein werden (weniger als eine Sekunde), weil das Ablesen und Aufschreiben der Werte einen bestimmten Zeitaufwand hat,

der nicht unterschritten werden kann. Die Anzahl der parallel bzw. zeitgleich aufgezeichneten Messwerte kann nicht beliebig viel werden, da der Mensch nicht gleichzeitig viele Messwerte ablesen und aufschreiben kann. Eine kontinuierliche Langzeitmessung in sehr kurzen Zeitintervallen ist für den Menschen eine einfache, monotone Tätigkeit. Wenn der Mensch diese einfache und monotone Tätigkeit nicht machen würde, hätte er Zeit um kompliziertere Tätigkeiten auszuführen.

2.2 Automatisches Data-Logging

Weil das manuelle Data-Logging Nachteile in Messabweichung, Wiederholgenauigkeit, Begrenzung der Datenmenge (Zeitintervall, parallele Messwerte) und Monotonie der Tätigkeit hat, wird sie durch ein automatisches Data-Logging ersetzt. Beim automatischen Data-Logging übernimmt ein Gerät selbstständig das Ablesen und Aufzeichnen der Messwerte, welches beim manuellen Data-Logging vom Menschen ausgeführt wird.

Durch die Verwendung eines Data-Logging-Geräts fallen die von Menschen verursachten Messabweichungen und Wiederholungenauigkeiten weg. Weiterhin ist ein Data-Logging-Gerät in der Lage eine größere Datenmenge als der Mensch zu verarbeiten. Wenn ein Data-Logging-Gerät verwendet wird, kann der Mensch andere Tätigkeiten ausführen.

Ein Beispiel für das automatische Data-Logging ist ein Messschreiber. Ein Messschreiber ist ein Messgerät, das den zeitlichen Verlauf der Messgröße direkt auf Papier in ein Liniendiagramm aufzeichnet.

Wie in Abbildung 1 dargestellt, besteht ein Messschreiber aus einem Zeiger, einem Schreibstift, der mit dem Zeiger verbunden ist, und einer Papierbahn, die Rollen geführt ist. Der Zeiger bewegt sich in Abhängigkeit vom Messwert auf der Messwert-Achse. Die Zeit-Achse wird durch die Geschwindigkeit der Papierbahn bestimmt. Die Messwert-Achse und Zeit-Achse bilden ein Diagramm (vgl. [3]).

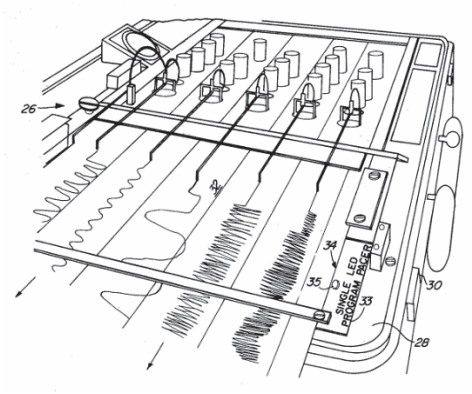


Abbildung 1 Messschreiber [4]

Ein Messschreiber zeichnet die Messwerte auf Papier auf. Eine elektronische Datenverarbeitung ist beim Messschreiber (elektromechanisches Data-Logging) nicht möglich. Wenn die Messwerte elektronisch weiterverarbeitet werden sollen, wird ein sogenannter Datenlogger verwendet.

Ein Datenlogger ist ein elektronisches Gerät und besteht aus einem Mikroprozessor, einem internen Speicher zur Datenspeicherung und einer Schnittstelle zu Sensoren. Datenlogger zeichnen Daten über einen bestimmten Zeitraum von einem eingebauten oder externen Sensor auf und speichern die Daten auf einem Speichermedium (vgl. [5]).

Wenn die aufzuzeichnenden Daten bereits in digitaler Form vorliegen, z. B. von einer Maschinensteuerung, können Software Datenlogger verwendet werden, um die Daten aufzuzeichnen. Software Datenlogger sind Programme, die auf einem IPC[1] laufen und die Daten von der Maschinensteuerung erhalten, z.B. SPS-ANALYSER pro 6 von der Firma Autem (vgl. [6]).

SPS[2]-Hersteller wie die Firmen Beckhoff, Siemens und Phoenix Contact bieten Funktionen zur Umsetzung eines Dataloggings an. In dieser Arbeit werden diese Funktionen der SPS-Hersteller ausführlicher untersucht und eine Datalogging-Anwendung für SPSen erstellt.

[1] IPC: Industrie-PC
[2] SPS: Speicherprogrammierbare Steuerung

2.3 Anwendungsbeispiele des Data-Loggings

Das Data-Logging wird in unterschiedlichen Anwendungen eingesetzt, z. B. Wetterstationen, Abgasmessung von Verbrennungsanlagen, Erfassung von Fahrzeugdaten, Flugschreiber. In diesem Abschnitt werden zwei Anwendungen vorgestellt: Wetterstation und Flugschreiber.

2.3.1 Wetterstation

Eine Wetterstation ist eine Zusammenstellung verschiedener Messgeräte, die zur Messung meteorologischer Größen und damit der Wetterbeobachtung an einem bestimmten Ort und der Klimaforschung dienen (vgl. [7]).

Abbildung 2 Wetterstation von TESTEM [8]

Wetterstationen enthalten unterschiedliche Messgeräte zur Messung der Wetterbedingungen. Diese Messgeräte bestehen aus Thermometer zur Messung der Lufttemperatur, Windrichtungs- und Windgeschwindigkeitsmesser, Hygrometer zur Messung der Luftfeuchtigkeit, Barometer zur Messung des Luftdrucks und Niederschlagsmesser. Weitere Messgeräte erfassen Bodenfeuchtigkeit, Bodentemperatur und Sonnenstrahlung (UV-Strahlung, Sonnenstunden). Wetterstationen enthalten einen Rechner, der Daten sammelt und zur Weiterverarbeitung ausgibt.

2.3.2 Flugschreiber

Ein Flugschreiber (allgemein bekannt als Black Box) ist ein Aufzeichnungsgerät in Flugzeugen, das relevante Flugdaten und Flugzeugparameter sowie Gespräche im Cockpit während des Fluges speichert. Nach einem Flugzeugunfall wird der Unfallverlauf mit Hilfe der gespeicherten Daten analysiert. Der Flugschreiber ist in „flight data recorder“ (FDR) (vgl. [9]) und „cockpit voice recorder“ (vgl. [10]) unterteilt.

Abbildung 3 Flight Data Recorder [9]

Der „flight data recorder“ empfängt Eingaben in einem bestimmten Datenrahmen von dem Flugdatensammelmodul. Er zeichnet wichtige Flugparameter wie Flughöhe und Geschwindigkeit auf, sowie Steuerungs- und Aktuatorstatus, Triebwerksinformationen und Zeitstempel. Normalerweise wird jeder Parameter mehrmals pro Sekunde aufgezeichnet. Wenn sich die Daten jedoch schnell ändern, werden diese abnormalen Daten mit einer höheren Frequenz gespeichert. Die meisten FDRs zeichnen kontinuierlich Daten für ungefähr 17-25 Stunden auf [9].

3 Anforderung an das Data-Logging-System

Die Systemanforderungen teilen sich in zwei Hauptpunkte auf, zum einen in die Anforderungen an den Signalaufzeichnungsprozess und zum anderen in die Anforderungen an das Zielsystem.

Die Anforderungen an den Signalaufzeichnungsprozess beinhalten die zu verarbeitende Datenmenge und die Zykluszeit der Signalaufzeichnung. Die Anforderungen an das Zielsystem umfassen die verwendete Hardware und Software sowie die Steuerungsanwendung. Als Steuerungsanwendung wird eine Testanwendung erstellt, um das Data-Logging-System zu testen.

3.1 Anforderung an den Signalaufzeichnungsprozess

Von einem SPS-Programm werden die aufgezeichneten Signale zur Verfügung gestellt. Die Zykluszeit der aufgezeichneten Signale, die Anzahl der aufgezeichneten Signale und die Größe der Signalwerte sind anhängig von der Anwendung.

Eine weitere Anforderung an das Data-Logging-System ist, dass die aufgezeichneten Daten zwischengespeichert werden und mit Hilfe eines tragbaren Datenträgers (Festplatte, USB-Stick, etc.) abgeholt werden.

3.2 Anforderung an das Zielsystem

Als Steuerung wird ein Embedded-PC von Beckhoff mit TwinCAT3 Runtime verwendet, als Engineering Software wird TwinCAT3 Engineering benutzt und als Testanwendung wird ein in Simulink erstelltes Heizsystem, bestehend aus Regler und Modell, verwendet.

3.2.1 Hardware: CPU-Modul cx2040 von Beckhoff

Die Anwendungen, für die das Data-Logging-System vorgesehen ist, enthalten hochmoderne Reglungen, die in MATLAB/Simulink entwickelt werden. SPS-Programme werden in der Regel mit einer IEC 61131-3 konformen Programmiersprache erstellt. Um Programmieraufwand zu vermeiden, wird eine Steuerung benötigt, auf der die in MATLAB/Simulink erstellte Reglung laufen kann.

Die SPS-Hersteller Beckhoff und Siemens unterstützen MATLAB/Simulink. Bei Siemens wird eine ODK fähige CPU benötigt, wie die CPU 1518 ODK oder der SIMATIC S7-1500 Software Controller[3]. Bei Beckhoff wird ein Industrie-PC oder ein Embedded-PC der CX Serie mit dem Betriebssystem Windows 7 und entsprechenden embedded Versionen oder Windows 10 von Microsoft benötigt[4].

Der Testaufbau zum Testen des Data-Loggings verwendet die Beckhoff Steuerung cx 2040-0130.

3.2.1.1 Hardwareaufbau des Moduls cx2040

Das CPU-Modul cx2040 ist ein Embedded-PC, das bedeutet, das CPU-Modul cx2040 ist ein PC in SPS-Bauform. Das cx2040 CPU-Modul verfügt über eine Intel-Core-i7-quadcore-CPU mit 2,1 GHz Taktfrequenz. Das Modul beinhaltet auch einen 4 GB RAM als Arbeitsspeicherplatz (vgl. [11]).

Die Grundkonfiguration von cx2040 Moduls besitzt:

- zwei mehrpolige Anschlüsse für Erweiterungsmodule
- zwei RJ45-Ethernet-Schnittstellen zur Verbindung mit lokalen Netzwerken, Internet oder EtherCAT
- eine DVI-I-Schnittstelle für einen Monitor oder ein Panel
- vier USB-Schnittstellen für Peripheriegeräte, zum Beispiel Tastatur, Maus oder USB-Speicherstick
- ein Kartensteckplatz für industrielle Speicherkarten (CFast-Karte) (vgl. [11])

[3] vgl. [51]
[4] vgl. [42]

3.1.1.2 Systemaufteilung des Moduls cx2040

Als Betriebssystem kommt Windows Embedded Standard 7 Pro zum Einsatz. Als Soft-SPS[5] wird die TwinCAT3 Runtime verwendet (vgl. [11]).

Der Embedded-PC ist in einen Echtzeitteil und einen Nicht-Echtzeitteil unterteilt. Der Echtzeit-Teil ist der TwinCAT3 Runtime und wird zum Ausführen von SPS-Programmen verwendet. Der Nicht-Echtzeit-Teil ist das Windows-Betriebssystem, auf dem Programme wie Microsoft Excel oder selbst erstellte Programme laufen können.

3.2.2 Software: TwinCAT 3

Die Entwicklungssoftware TwinCAT3 von Beckhoff unterteilt sich in eXtended Automation Engineering (XAE) und eXtended Automation Runtime (XAR) (vgl. [12]). XAE wird als Entwicklungssystem/Bearbeitungsumgebung verwendet, um SPS-Programme zu erstellen und zu debuggen. XAR wird als Ausführungssystem verwendet, um die SPS-Programme auszuführen. Der in XAE geschriebene SPS-Quellcode wird vom Compiler in Maschinencode kompiliert und dann in XAR ausgeführt. Der Systemaufbau der Software TwinCAT3 ist in Abbildung 4 dargestellt.

[5] Soft-SPS: Programm, die auf PC laufen, mit den gleichen Eigenschaften der Hardware-SPS

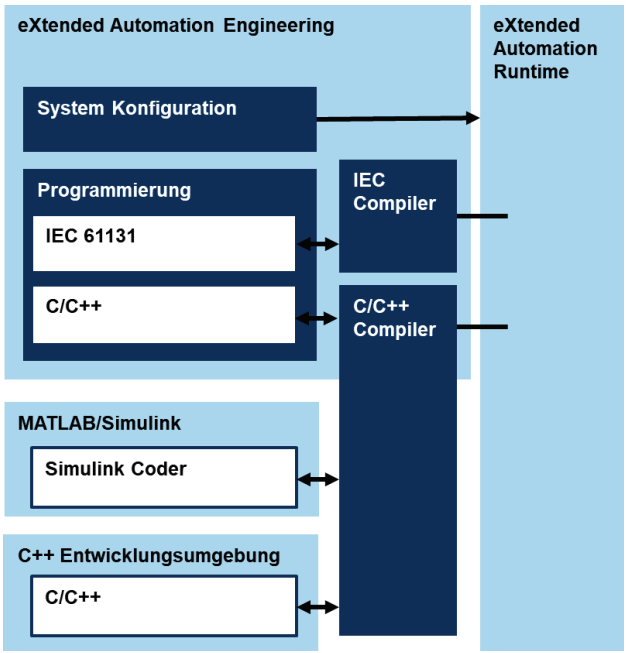


Abbildung 4 TwinCAT Systeminformation (vgl. [12])

TwinCAT3 eXtended Automation Engineering (XAE)

TwinCAT3 (TC3) eXtended Automation Engineering ist die Entwicklungsumgebung von Beckhoff für SPS-Programme. Im Gegensatz zu anderen Entwicklungsumgebungen wie die Vorgängerversion TwinCAT2 von Beckhoff ist TwinCAT3 XAE keine eigenständige Software. TwinCAT3 wird in Microsoft Visual Studio als Softwarepaket integriert (vgl. [12]).

TC3 ist modular aufgebaut. Im Basismodul werden SPS-Programme mit IEC 61131-3[6] Programmiersprachen programmiert. Es gibt optionale Module z.B. zur Einbindung von MATLAB/Simulink und C++[7].

SPS-Programme können mit IEC 61131-3 Programmiersprachen, C++ oder als Kombination von IEC 61131-3 Programmiersprachen und C++ realisiert werden. Beckhoff stellt Bibliotheken zur Verfügung, welche Programmfunktionen für die SPS beinhalten. Einige Bibliotheken davon sind lizenzpflichtig, z.B. die Bibliothek Tc2_XmlDataSrv (vgl. [13]).

[6] IEC 61131-3: internationale Norm für SPS-Programmiersprachen
[7] C++: Programmiersprache

TwinCAT3 eXtended Automation Runtime (XAR)

Die TwinCAT 3 Runtime Komponente läuft auf einen Windowsrechner parallel zum Betriebssystem und hat Kommunikationsschnittstellen zum Betriebssystem. Sie bietet eine Echtzeitumgebung[8] und funktioniert wie eine Hardware-SPS. Das SPS-Projekt wird mit TC3 XAE in die Runtime hochgeladen ähnlich wie bei einer Hardware-SPS. Diese Form von SPS wird auch Software-SPS genannt.

Das in TC3 XAE erstellte SPS-Projekt kann in verschiedene Tasks aufgeteilt werden. Diese Tasks werden zyklisch abgearbeitet. TwinCAT3 unterstützt auch Multicore-CPUs. Das bedeutet unterschiedliche TwinCAT-Tasks können verschiedenen Kernen einer CPU zugeordnet werden. Damit wird die Leistung von Multicore-Industrie- und Embedded-PCs optimaler ausgeschöpft im Vergleich zu anderen IPC-Herstellern wie Siemens, welche für die Soft-SPS nur ein Kern vorsehen (vgl. [12] [14]).

3.2.3 Testanwendung

Die Testanwendung, mit der das Data-Logging-System getestet wird, besteht aus zwei Teilen, einem Heizsystem als Datenlieferant und einem Analysesystem zur Datenverarbeitung. Das Heizsystem erzeugt Daten, welche vom Data-Logging-System erfasst und gespeichert werden. Das Analysesystem entnimmt Daten aus dem Data-Logging-System und bereitet diese Daten auf [15].

3.2.3.1 Datenquelle Heizsystem

Das Heizsystem ist in der Abbildung 5 skizziert. Das Heizsystem ist in zwei Teilsystemen untergliedert, einem Wassertanksystem und einem Raumsystem. Das Wassertanksystem regelt die Heizwassertemperatur und versorgt das Raumsystem mit Heizwasser. Das Raumsystem be-

[8] Das Betriebssystem Windows ist nicht echtzeitfähig

kommt Wärme durch das Heizwasser und gibt diese Wärme über einen Heizkörper an die Raumluft ab. Das Regelsystem sorgt dafür, dass die Raumtemperatur und die Warmwassertemperatur vorgegebene Sollwert annehmen.

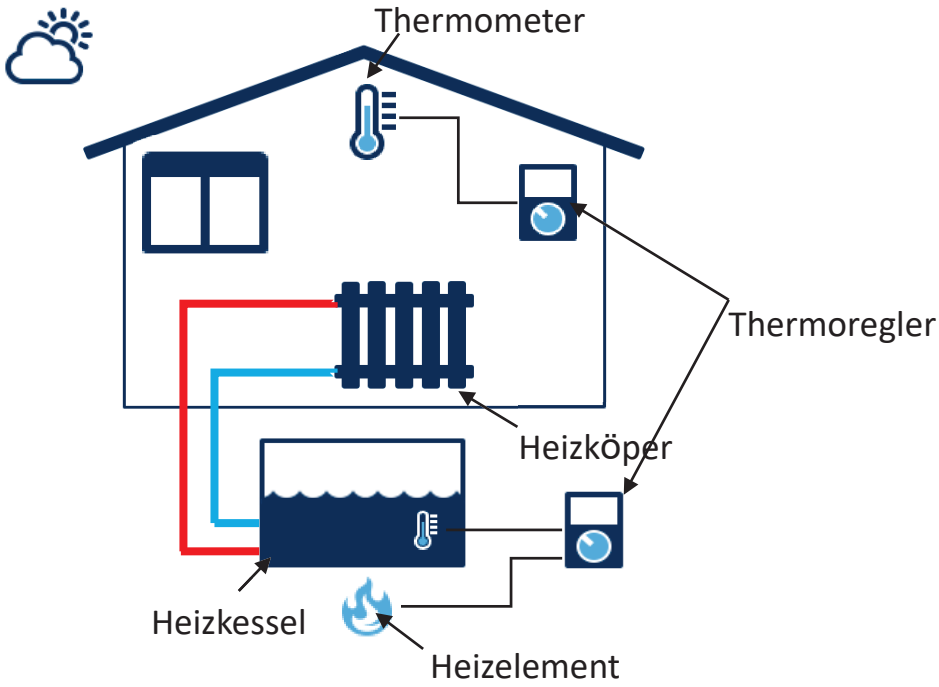


Abbildung 5 Skizze des Heizsystems (vgl. [15])

Das System wird in MATLAB/Simulink erstellt (siehe Abbildung 6). Die Eingangsgrößen des Heizsystems sind Außentemperatur „aussen_Temp“, die Sollgrößen der Raumtemperatur „set_Wasser_Temp“ und der Heizwassertemperatur „set_Raum_Temp“. Die Ausgangsgrößen des Testsystems sind die Istgrößen der Raumtemperatur „T_Wasser“ und der Heizwassertemperatur „T_Raum“ (vgl. [15]).

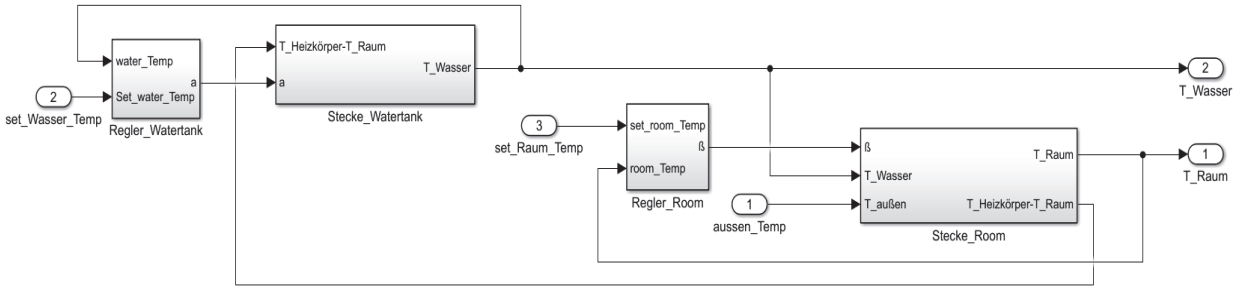


Abbildung 6 Aufbau des Heizsystems in Simulink (vgl. [15])

Mit dem Data-Logging-System sollen Eingangsgrößen, Stellgrößen und Ausgangsgrößen des Testsystemmodells aufgezeichnet werden. Deswegen wird das Testsystemmodell erweitert. Neben den Ausgangsgrößen werden auch die Eingangsgrößen und Stellgrößen als Ausgangssignalen nach außen geführt.

Darüber hinaus wird ein zweiter Raum hinzugeführt. Dieser Raum und der erste Raum teilen sich den gleichen Wassertank. Beide Räume sind unabhängig voneinander und haben jeweils einen eigenen Raumtemperaturregler. Die Temperatur-Sollwerte der beiden Räume können unabhängig voneinander eingestellt werden. Die Eingangsgrößen, Stellgrößen und Ausgangsgrößen dieses Raums werden auch als Ausgangssignale nach außen geführt und auch vom Data-Logging-Systems aufgezeichnet.

Die Ausgangssignale des erweiterten Heizsystems setzen sich aus der Raumtemperatur der beiden Räume „T_Raum1“ und „T_Raum2“, der Heizwassertemperatur „T_Wasser“, der Außentemperatur „setAußenT“, den Raumtemperatursollwerten der beiden Räume „setRaumT1“ und „setRaumT2“, dem Heizwassertemperatursollwert „setWasserT“, der Stellgröße „a“ vom Wassertankregler und den Stellgrößen der beiden Räume „ß1“ und „ß2“ zusammen (siehe Abbildung 7).

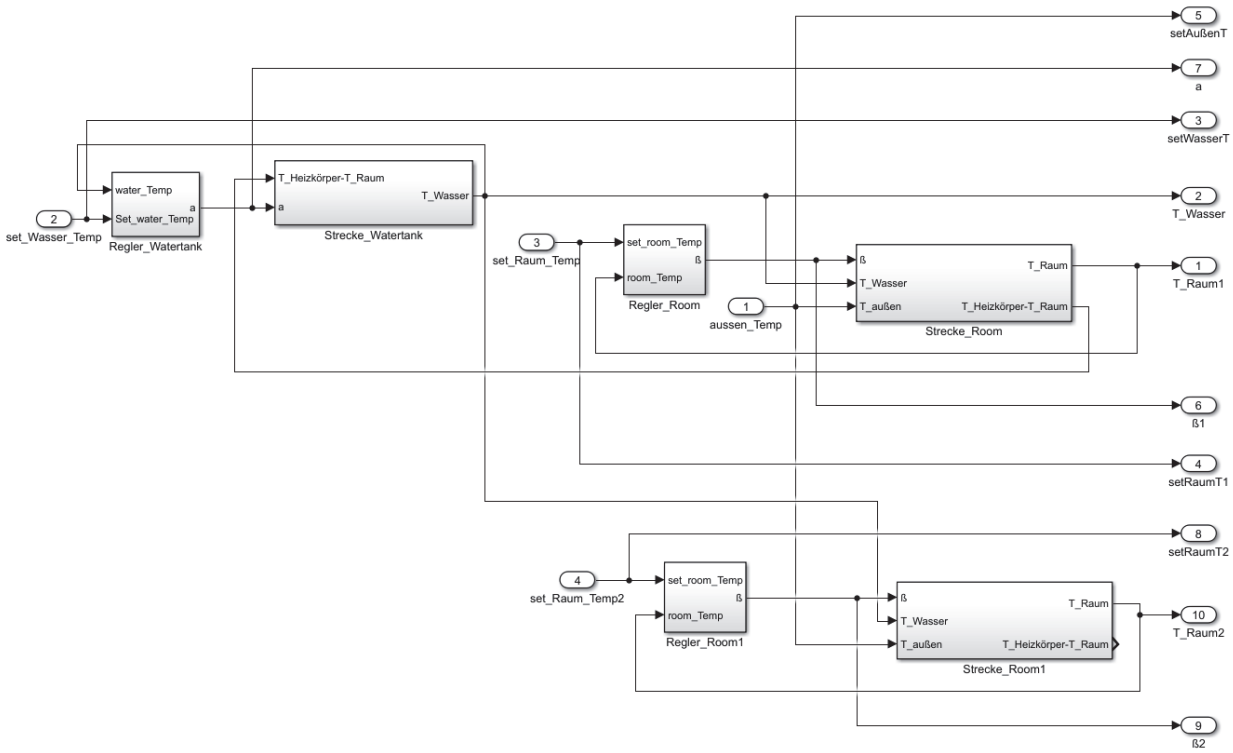


Abbildung 7 Aufbau des erweiterten Heizsystems in Simulink

In der Testanwendung wird eine Zykluszeit von zehn Millisekunden eingestellt[9]. Das bedeutet alle zehn Millisekunden wird ein neuer Datensatz vom Heizsystem erzeugt. Ein Datensatz besteht aus einem Zeitstempel und den aufzuzeichnenden Ausgangssignalen des Testsystems. Jedes aufzuzeichnende Ausgangssignal ist acht Byte groß.

3.2.3.2 Analyseprogramm

Das Analysesystem wird in Matlab entwickelt und als Nachbearbeitungsteil des Daten-Logging-Systems verwendet. Die vom Data-Logging-System erzeugten Dateien werden für die Verarbeitung mit dem Analyseprogramm ins mat-Dateiformat umgewandelt. Das Analyseprogramm ruft diese mat-Dateien ab und erstellt grafische Anzeigen der aufgezeichneten Signale, z.B. ein Temperatur-Zeit-Diagramm wie in Abbildung 8 dargestellt.

[9] Zykluszeit: Zeitintervall, indem das SPS-Programm neu abgearbeitet wird

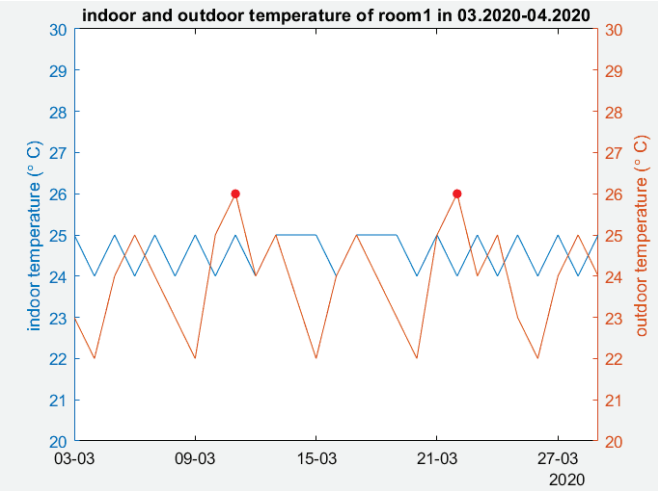


Abbildung 8 Graphische Anzeige des Analyseprogramms in MATLAB (vgl. [15])

4 Varianten des Data-Logging-Systems

Es gibt verschiedene Data-Logging-Varianten für SPSen: externe Datenlogger-Geräte, Datalogging-Funktionalitäten von SPS-herstellern, Software-Datenlogger und in hochsprachenprogrammierte, selbsterstellte Data-Logging-Funktionen. In diesem Kapitel werden die verschiedene Data-Logging-Varianten für SPSen betrachtet und hinsichtlich Wirtschaftlichkeit, Entwicklungsschwierigkeiten, Skalierbarkeit und Dokumentation verglichen. Anschließend wird eine Auswahl getroffen, welche dieser Varianten genauer untersucht wird.

4.1 Externe Datalogger

Eine Variante besteht darin, ein externes Datenlogger-Gerät an der SPS anzuschließen. Es gibt Datenlogger-Geräte mit integrierter Speicherkarte, wie z.B. der PLC Datalogger „FC-LOG PLC Datalogger“ von der Firma InterconnectingAutomation (vgl. [16]).

Der FC-LOG PLC Datalogger hat sowohl eine RS-232- als auch eine RS-485-Schnittstelle. Er wird über die RS-232- bzw. die RS-485-Schnittstelle mit einer SPS verbunden. Es können die Kommunikationsprotokolle Modbus RTU oder DirectNet verwenden werden. Die von der SPS bereitgestellten Daten werden geloggt und im CSV-Dateiformat gespeichert. Die geloggten Signalwerten werden mit Zeitstempeln versehen.

Die geloggten Daten werden auf der internen microSD-Karte gespeichert und sind gegen Datenverlust durch Stromausfall gesichert. Das Gerät wird in der Regel mit ein 4 GB microSD-Karten ausgeliefert. Es können microSD-Karten mit einer Speichergröße bis 32 GB verwendet werden.

Der FC-LOG PLC Datalogger kann nicht nur für die Datenaufzeichnung von SPSen, sondern auch von Modbus-fähigen Feldgeräten verwendet werden (vgl. [17]). In Abbildung 9 ist der FC-LOG PLC Datalogger dargestellt.

Abbildung 9 FC-LOG PLC Datalogger (vgl. [17])

Neben Datenlogger mit integrierter Speicherkarte gibt es auch Datenlogger-Geräte ohne integrierter Speicherkarte. An diesen Datenlogger-Geräten gibt es Schnittstellen zum Anschließen von mobilen Datenträgern wie USB-Sticks. Ein Beispiel für solche Geräte ist das Gerät „Datensicherung S7-SPS PN-Port“ von der Firma Process-Informatik Entwicklungsgesellschaft (vgl. [18]).

Dieses Gerät funktioniert ähnlich wie der FC-LOG PLC Datalogger. Der Unterschied besteht darin, dass das Gerät keinen internen Speicher hat und nur eine USB-Schnittstelle und eine Schnittstelle für eine microSD-Karte besitzt. Um die Verbindung zu der SPS herzustellen, wird ein RJ45 Profinet-Port verwendet. Die Aufnahmefunktion startet nach dem Einsetzen des USB-Sticks bzw. der microSD-Karte. Die Konfiguration, in welcher die aufzuzeichnenden Daten eingestellt werden, wird über den integrierten Webserver vorgenommen (vgl. [18]). In Abbildung 10 ist das Gerät „Datensicherung S7-SPS PN-Port“ dargestellt.

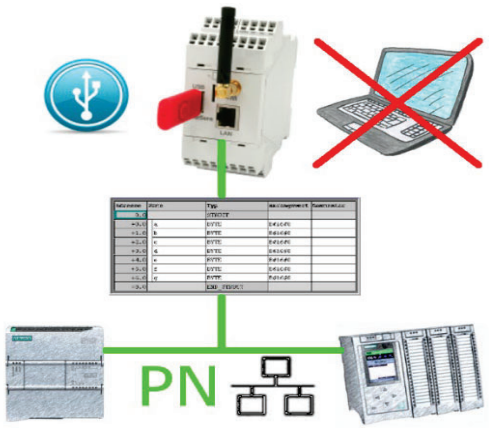


Abbildung 10 Datensicherung S7-SPS PN-Port (vgl. [18])

Eine andere Variante besteht darin, Daten mit Hilfe von bestimmten HMI-Panels aufzuzeichnen, z.B. von der Firma Siemens. Diese HMI-Panels werden sowohl als Kurzzeitspeicher als auch als Schnittstelle zu externen Speichern verwendet. In der Programmiersoftware für das HMI-Panel (z.B. TIA Portal WinCC von Siemens) wird vorgegeben, welche Signale aufgezeichnet werden. Die aufzuzeichnenden Daten werden von der SPS zum HMI-Panel übermittelt und beim Siemens-HMI-Panel in CSV-Dateien auf dem internen Speicher gespeichert (vgl. [19]).

Wenn der interne Speicher voll ist, müssen die Datei auf einen tragbaren Datenträger (z.B. ein USB-Stick) übertragen werden.

4.2 Interne Data-Logging-Variante

SPS-Hersteller bieten Data-Logging-Funktionen für Hardware-SPSen und auf IPC laufende Software-SPSen. Wenn ein IPC mit einer Hardware-SPS verbunden ist, kann das Data-Logging auch durch Software von Fremdanbietern realisiert werden. Eine weitere Variante besteht darin, eine Data-Logging-Funktion selbst zu programmieren, z.B. in C++. Diese Datalogging-Varianten laufen auf dem IPC und es werden keine zusätzlichen Geräte benötigt. Daher werden sie als interne Data-Logging-Variante bezeichnet.

4.2.1 SPS-Funktionalitäten von SPS Hersteller

Die von SPS-Herstellern bereitgestellten Methoden für Data-Logging werden in Methoden für Hardware-SPSen und Methoden für Software-SPSen unterteilt. Als Zielsystem wird die Software-SPS TC3 XAR von Beckhoff verwendet. Daher werden in diesem Kapitel Data-Logging-Funktionen für Software-SPSen beschrieben. Data-Logging-Funktionen für Hard-SPSen sowie Data-Logging-Funktionen für SPSen von Siemens und Phoenix Contact werden kurz erläutert.

4.2.1.1 Funktionen für Soft-SPSen von Beckhoff

Die von Beckhoff bereitgestellten Data-Logging-Methoden für SPS-Programme, die mit IEC 61131-3 Programmiersprachen erstellt worden sind, werden in vier Typen unterteilt:

- Data-Logging mit CSV-Datei
- Data-Logging mit XML-Datei
- Data-Logging mit Datenbank-Datei
- Data-Logging mit Analytics Logger

Beckhoff stellt zwei Funktionsblöcke [10] zur Verfügung für Data-Logging mit CSV-Datei: FB_CTRL_Log_DATA und CSVMemBufferWriter.

FB_CTRL_Log_DATA

Der Funktionsblock FB_CTRL_Log_DATA stammt aus der Bibliothek TwinCat PLC Lib controller Toolbox. Er funktioniert nur auf IPC-Systemen und speichert Dateien im IPC-Dateisystem ab. Dieser Funktionsblock ermöglicht die Erstellung einer Log-Datei im CSV-Format, in der maximal zehn Signale aufgezeichnet werden können. Die vom Benutzer angegebenen Signalbezeichnungen werden in die erste Zeile dieser Datei geschrieben. Die Signalwerte werden in den folgenden Zeilen in gleichen Zeitintervallen geschrieben. Das Zeitintervall wird mit dem Parameter tLogCycleTime angegeben (vgl. [20]).

CSVMemBufferWriter

Der Funktionsblock CSVMemBufferWriter stammt aus der Bibliothek TwinCat PLC Utilties. Mit diesem Funktionsbaustein werden Datensätze in einem externen Buffer im CSV-Format generiert.

Der Inhalt des Buffers wird dann mit dem Funktionsbaustein für Datei-Schreibzugriffe in eine Datei geschrieben. Dieser Funktionsblock kann keine Dateien öffnen und schließen und nicht in Dateien schreiben. Daher muss dieser Funktionsblock in Verbindung mit dateibezogenen Funktionsblöcken wie FB_FileOpen (zum Öffnen von Dateien), FB_FileClose (zum Schließen von Dateien), FB_FilePuts und FB_FileWrite (zum Schreiben in Dateien) verwendet werden (siehe Tabelle 1). Eine CSV-Datei kann im Text- oder Binärmodus gelesen oder geschrieben werden (vgl. [21]).

[10] Funktionsblock: auch als Funktionsbaustein genannt

Tabelle 1 Vergleich von CSV-Datei in Textmodus und Binärmodus (vgl. [21])

	Textmodus	Binärmodus
Funktionsbaustein für den Dateischreibzugriff	FB_FilePuts	FB_FileWrite
Unterstützte SPS Variablentypen, die direkt geschrieben/gelesen werden	T_MaxString (STRING mit 255 Zeichen)	Beliebige Datentypen
Maximale Datensatzlänge	Auf 253 Zeichen begrenzt	theoretisch unbegrenzt

XML-Server

Beckhoff bietet die Funktion XML-Server für den Zugriff auf XML-Dateien. Die Funktion besteht aus einem gleichnamigen XML-Server und der Bibliothek Tc2_XmlDataSrv. Der XML-Server wird auf den IPC installiert. Die Kommunikation zwischen SPS und XML-Server erfolgt mit den Funktionsblock FB_XmlSrvWrite aus der Bibliothek Tc2_XmlDataSrv, damit die aufzuzeichnenden Variablenwerte in die XML-Datei geschrieben werden. Die Dateien werden im IPC-Dateisystem gespeichert (vgl. [13]).

TC3 Datenbankserver

Beckhoff bietet die Funktion TC3 Datenbankserver für den Zugriff auf Datenbanken innerhalb des IPCs an. Der TC3 Datenbankserver ermöglicht den Datenaustausch zwischen dem SPS und verschiedenen Datenbanksystemen. Der TC3 Datenbankserver unterstützt ASCII- und XML-Datenbanken, serverbasierte Datenbanksysteme wie MS SQL, MS SQL Compact, Oracle, MongoDB und dateibasierte Datenbanken wie MS Access, SQLite.

Die TC3 Datenbankserver-Funktion besteht aus einem Datenbankserver, einem Konfigurator und der Bibliothek Tc3_Database. Nach der Installation des TC3 Datenbankservers müssen die für die Kommunikation mit jeder Datenbank erforderlichen Parameter über den Konfigurator festgelegt werden. Anschließend werden mit den von der Bibliothek bereitgestellten Funktionsblöcken eine Datenbankverbindung hergestellt, eine Tabelle erstellt, Daten in die Tabelle geschrieben und aus der Tabelle gelesen. Die Datenbank wird mit Hilfe der Funktionsblöcke auch aktualisiert oder gelöscht (vgl. [22]).

Analytics Logger

Mit dem TF3500 TwinCAT Analytics Logger ist es möglich, die Prozessdaten und Variablen der SPS-Programme synchron zu den Taskzyklen zu erfassen. Der TwinCAT Analytics Logger kann

entweder als MQTT[11]-Client arbeiten und die Daten regelmäßig an einen MQTT-Nachrichten-Broker[12] übertragen (MQTT-Modus) oder die Daten lokal in einer binären Datei auf der Festplatte des PC-Systems speichern (File-Modus). Nachdem das Attribut „TcAnalytics“ vor die aufzuzeichnenden Variablen gesetzt wird, wird automatisch ein neuer Stream zum Analytics Logger hinzugefügt. In dem Stream können die aufzuzeichnenden Variablen durch Checkboxen ausgewählt werden. Die maximale Dateigröße wird im Analytics Logger eingestellt. Die Dateien werden unter C:\TwinCAT\3.1\Boot\Analytics gespeichert. Dieser Dateispeicherpfad kann nicht geändert werden (vgl. [23]).

4.2.1.2 Funktionen für Soft-SPSen von Siemens

Die Engineering-Software TIA Portal WinCC Advanced von Siemens unterstützt die Einbindung vom Microsoft SQL Datenbankserver, um die Werte in der SQL-Datenbank direkt von der Engineering-Software zu ändern und Variablen mit der WinCC Runtime in einer Datenbank aufzuzeichnen. Der Speicherort und der Variablenname werden in WinCC konfiguriert. Die archivierten Variablenwerte werden in der zugewiesenen Datenbank gespeichert (vgl. [24]).

4.2.1.3 Funktionen für Hardware-SPSen von Siemens

Siemens stellt eine Datenprotokollanweisungsfunktion für Hardware-SPSen zur Verfügung, mit der Datenwerte vom SPS-Programm zur Laufzeit in Log Dateien gespeichert werden. Die CPU speichert diese Dateien im Flash-Speicher im Standard-CSV-Format (vgl. [1]). Die Datenprotokollanweisungsfunktion kann auch auf Software-SPSen verwendet werden.

[11] MQTT: Message Queuing Telemetry Transport, offenes Netzwerkprotokoll für Maschine-zu-Maschine-Kommunikation
[12] MQTT-Nachricht-Broker: MQTT-Server

4.2.1.4 Funktionen für Hardware-SPSen von Phoenix Contact

Phoenix Contact bietet eine DataLogger-Service-Komponente. Diese Komponente überträgt Echtzeitdaten pro Taskzyklus vom IO-Port oder von globalen Variablen der SPS in eine Datenbank. Die zu archivierenden Daten werden in eine SQLite-Datenbank geschrieben. Es stehen zwei Modi zur Verfügung: „endless mode“ und „save on mode“. Im Modus „endless mode“ werden Variablen kontinuierlich aufgezeichnet und im Modus „save on mode“ werden Variablen nur aufgezeichnet, wenn sich der Variablenwert ändert. Wenn der Variablenwert gleich bleibt, werden sie mit „NULL“ aufgezeichnet.(vgl. [25]).

4.2.1.5 Data-Logging mit OPC UA

OPC UA (Open Platform Communications Unified Architecture) ist ein internationales, standardisiertes Kommunikationsprotokoll, mit dem Daten unabhängig von Hersteller und Plattform ausgetauscht werden können (vgl. [26]). Der integrierte TwinCAT3 OPC UA-Server von Beckhoff bietet eine standardisierte Kommunikationsschnittstelle für den Zugriff auf Variablen oder Symbolwerte aus der TwinCAT-Laufzeit (vgl. [27]). Software von Drittanbietern (normalerweise ein OPC UA-Client) stellt eine Verbindung zum Server her, um Variablenwerte zu lesen.

Siemens bietet wie bei Beckhoff auch einen OPC UA-Server für SPSen an. Das minimale Abtastintervall des OPC UA-Servers beträgt 100ms. Weitergehende Informationen sind in [1] beschrieben. Phoenix Contact bietet ebenfalls einen OPC UA-Server für SPS an. Das minimale Abtastintervall dieses Servers beträgt auch 100ms wie bei Siemens (vgl. [25]).

Ein Ende der OPC UA-Kommunikation ist ein OPC UA-Server auf der SPS und das andere Ende ist ein OPCU UA-Client (vgl. [26]). Der OPC UA-Client fragt Variablenwerte vom OPC UA-Server der SPS ab.

4.2.2 Data-Logging-Software von Drittanbietern

Es gibt Software von Drittanbietern, die für die Datenerfassung geeignet sind, z.B. die von „Open Automation Software“ (OAS) angebotene Software OAS Platform (vgl. [28]). Mit dieser Software

können Daten von einer SPS oder einem IPC aufgezeichnet werden. Daten, die von Anwendungen mit hoher Datenaktualisierungsrate generiert werden, werden mit Open Automation Software gelesen und aufgezeichnet. OAS zeichnet die gepufferten Daten mit einem Zeitstempel in einer Datenbank oder einer CSV-Datei im Tabellenformat auf. Die unterstützten Datenbanktypen sind SQL Express, Microsoft Azure, Oracle, mySQL, MariaDB, SQLite, MongoDB, PostgreSQL und Cassandra. Daten mit einem Zeitintervall von einer Millisekunde können aufgezeichnet werden. (vgl. [28])

4.2.3 in C++ selbsterstellte Data-Logging-Anwendung

Die Engineering Software TwinCAT3 von Beckhoff unterstützt sowohl SPS-Programme, die mit einer IEC 61131-3 Programmiersprache programmiert werden, als auch C++-Programme (vgl. [12]). Mit winAC RTX ODK von Siemens können ebenfalls C++-Programme auf der Software-SPS winAC RTX ausgeführt werden (vgl. [29]). Mit Hilfe von PLCnext Technology C++ Toolchain von Phoenix Contact ist es möglich, C++-Programme in der Steuerung AXC 2152 auszuführen (vgl. [30]).

Für die Ausführung von SPS-Programmen in TwinCAT3 ist eine PLC-Lizenz erforderlich und für die Ausführung von C++-Programmen eine TwinCAT C++-Lizenz. Für die Verwendung des MATLAB/Simulink-Moduls ist ebenfalls die TwinCAT C++-Lizenz erforderlich. Damit keine XML-Server Lizenz benötig wird, wird eine Data-Logging-Funktion in C++ erstellt. Die in MATLAB/Simulink erstellte Regleranwendung und die Data-Logging-Funktion teilen sich die TwinCAT C++ Lizenz.

Als Dateityp der Data-Logging-Dateien kann das Binärformat verwendet werden.

4.3 Vergleich der Data-Logging-Varianten

Zunächst werden die Varianten externe Datenlogger, Funktionalitäten von SPS-Herstellern und Data-Logging-Software von Drittanbieter miteinander verglichen. Die Vor- und Nachteile dieser Varianten sind in Tabelle 2 aufgeführt.

Tabelle 2 Vor- und Nachteile der Data-Logging-Varianten

	Vorteile	Nachteile
Externe Datenlogger	-entwickeltes Produkt -Keine Programmierung erforderlich, wird konfiguriert -nach Konfiguration keine Änderung erforderlich	-finanzieller Aufwand für Anschaffung -zusätzliches Gerät -eingeschränkte Funktionalität
Funktionalitäten von SPS-Herstellern	-hohe Anwendungsflexibilität durch Programmierung -umfangreiche Dokumentationen von SPS-Herstellern -keine Kenntnisse in Hochsprachenprogrammierung (C++) erforderlich	-finanzieller Aufwand für Lizenzen erforderlich -Programmieraufwand erforderlich -herstellerabhängig
Data-Logging-Software von Drittanbietern	-entwickeltes Produkt -Keine Programmierung erforderlich, wird konfiguriert	-finanzieller Aufwand für Anschaffung -eingeschränkte Funktionalität -(IPC erforderlich)
C++ Programm mit Data-Logging-Funktion	-hohe Anwendungsflexibilität durch Programmierung -umfangreiche Dokumentationen von SPS-Herstellern (Schnittstellen zum Dateisystem) -herstellerunabhängig in der Kernfunktion	-finanzieller Aufwand für Lizenzen erforderlich -Programmieraufwand erforderlich -Kenntnisse in Hochsprachenprogrammierung (C++) erforderlich

Für alle vier Varianten ist ein finanzieller Aufwand erforderlich für die Anschaffung des Geräts oder der Lizenz. Bei der Variante externe Datenlogger müssen zusätzliche Geräte zur SPS in der Anlage bzw. Maschine integriert werden. Dies beansprucht zusätzlichen Bauraum im Schaltschrank. Bei der Variante Software von Drittanbietern Funktionen gibt es eine niedrigere Anwendungsflexibilität im Vergleich zu den Varianten „Funktionalitäten von SPS-Herstellern“ und „C++ Programm mit Data-Logging-Funktion“, da diese unter Berücksichtigung der Anforderungen individuell erstellt werden.

Auf Grund der hohen Anwendungsflexibilität der Varianten „Funktionalitäten von SPS-Herstellern“ und „C++ Programm mit Data-Logging-Funktion“ werden diese Varianten weiter untersucht.

Die Funktionen von Beckhoff, die in Kapitel 4.2.1.1 beschrieben sind, werden miteinander verglichen. In Tabelle 3 sind die Vor- und Nachteile der verschiedenen Funktionen aufgeführt.

Tabelle 3 Vor- und Nachteile der Funktionen von Beckhoff

	Vorteile	Nachteile
CSV	-einfache Struktur -vergleichsweise großes Verhältnis von Nutzdatenmenge zur Gesamtdatenmenge der Datei -CSV-Standard kann von vielen anderen Anwendungen gelesen werden	-keine direkte Beziehung zwischen Datenbezeichnung und Datenwert -Anzahl der Datensätze in Textmode begrenzt -Im Binärmode sind die aufgezeichneten Daten nicht menschenlesbar
XML	-direkte Beziehung zwischen Datenbezeichnung und Datenwert -hierarchische Struktur -Metadaten	-vergleichsweise kleines Verhältnis von Nutzdatenmenge zur Gesamtdatenmenge der Datei
Datenbank	-Tabellenform -Beziehungen zwischen Daten und Tabellen können dargestellt werden	-serverbasierte Datenbanken brauchen ein Serverprogramm -Dateien sind schwer zu übertragen - Speicherplatzanforderung ist vergleichsweise groß -Aufbereitung der Daten zur Weiterverarbeitung erforderlich
OPC UA	-plattformunabhängig -Unterstützung von SPS-Herstellern	-kann Dateien nicht direkt erzeugen, muss durch Programmierung gelöst werden -OPC UA-Client erforderlich

Da die Daten in der XML-Datei im Vergleich zur CSV-Datei in einer hierarchischen Struktur gespeichert werden, wird der Variablenname zum Abfragen und Aufzeichnen der Daten verwendet. Sowohl die Variablennamen als auch die Variablenwerte werden aufgezeichnet. Außerdem ist der Speicherplatz einer XML-Datei im Vergleich zu einer vergleichbaren Datenbank kleiner. Daher wird eine Data-Logging-Funktion mit XML in IEC 61131-3 Programmiersprache erstellt

und untersucht. Bei der OPC UA Variante sind Programmieraufwände sowohl für die Kommunikation zwischen SPS und OPC UA-Client, als auch für den Dateizugriff (Dateien erzeugen, Daten darin schreiben, Dateien öffnen und schließen) erforderlich. Da ein IPC verwendet wird, ist eine Kommunikation zwischen SPS und Betriebssystem mittels OPC UA nicht zwingend erforderlich. Auf Grund dessen wird die OPC UA Variante nicht weiteruntersucht.

5 Aufbau des Data-Logging-Systems

In diesem Abschnitt werden zunächst der Datenfluss und die Struktur des Data-Logging-Systems beschrieben. Danach werden die Architektur und die Programmelemente erstellt. Das Data-Logging-Programm wird dann implementiert.

5.1 Datenfluss des Data-Logging-Systems

Eine Hardware-SPS ist echtzeitfähig und hat einen vergleichsweise kleinen Speicherplatz. Die CPU 1518-4 PN/DP MFP (SIMATIC S7-1500 SPS) von Siemens verwendet als Speicherplatz eine SD-Karte, welche maximal 32 GB groß ist (vgl. [31]).

Wenn bei einer Zykluszeit von zehn Millisekunden ein Datensatz mit zehn Datenwerten zu je acht Byte erstellt wird, werden pro Sekunde 8 KB Daten erzeugt. Das bedeutet nach 46 Tagen werden 32 GB Daten erzeugt. Bei einem Datensatz aus 10.000 Werten zu je acht Byte wird nach elf Stunden eine Datenmenge von 32 GB erzeugt. Die Speicherkarte muss dann ausgetauscht werden, um die Daten lesen zu können.

Bei der Verwendung einer SD-Karte als Speicher ist auch die Lebensdauer der Speicherkarte zu berücksichtigen. Jedes Byte einer 32GB Speicherkarte von Siemens kann max. 100.000 Mal geschrieben werden (vgl. [32]).

Da das SPS-Programm und die -Daten gleichzeitig auf der SD-Karte gespeichert werden, muss die Ausführung des SPS-Programms beendet werden, wenn die SD-Karte ausgetauscht wird, um die aufgezeichneten Daten zu lesen. In diesem Fall muss auch das SPS-Programm auf die neue SD-Karte übertragen werden. Das ist nicht akzeptabel, da die Maschine dafür in den Stillstand gesetzt werden muss, und nicht produzieren kann.

Deshalb ist ein IPC als Kurzzeitdatenspeicher und eine Datenexportschnittstelle der Hardware-SPS erforderlich. IPCs verfügen häufig über einen größeren Speicherplatz als Hardware-SPSen. Die Speicherkapazität kann durch Anschließen von externen Festplatten erweitert werden. Die Daten im IPC werden als Dateien im Dateisystem gespeichert.

IPCs von Beckhoff sind in einen Echtzeitteil (Software-SPS) und einen Nicht-Echtzeitteil (Windows-Betriebssystem) unterteilt (siehe Kapitel 3.2.1).

Eine Software-SPS funktioniert wie eine Hardware-SPS. Das SPS-Programm, läuft auf der Software-SPS, erhält Daten von Sensoren, berechnet Variablen (Daten) und gibt Daten an Aktuatoren aus. Diese Daten sind die Datenquelle der Data-Logging-Funktion. Im Testsystem wird das Steuerprogramm vom Simulink-Modell als Datenquelle importiert.

Aus den Anforderungen des Data-Logging-Systems werden die aufgezeichneten Daten zuerst im Nicht-Echtzeitteil des IPCs zwischengespeichert, bevor diese Daten zum Langzeitspeicher übertragen werden (siehe Kapitel 3.1). Daher wird im IPC ein Kurzzeitspeicher benötigt, um alle Datensätze zu speichern, die während jedes Zyklus für die Übertragung zum Langzeitspeicher erzeugt werden.

Aus Abbildung 11 ist ersichtlich, dass der Datenfluss im Data-Logging-Systems die folgenden Prozesse durchläuft:

1. Die Daten kommen aus der Datenquelle und werden im Speicherplatz 1 des Buffers in jedem Zyklus gespeichert.

2. Gleichzeitig werden die Daten aus dem Speicherplatz 2 des Buffers abgerufen und durch das Logging-Programm in Dateien im IPC-Windowsspeicherplatz gespeichert.

3. Die Dateien im IPC-Windowsspeicherplatz werden auf einen tragbaren Datenträger übertragen.

4. Die auf den tragbaren Datenträger übertragenen Dateien werden im Langzeitspeicherplatz gespeichert und von dort vom Analyseprogramm abgerufen.

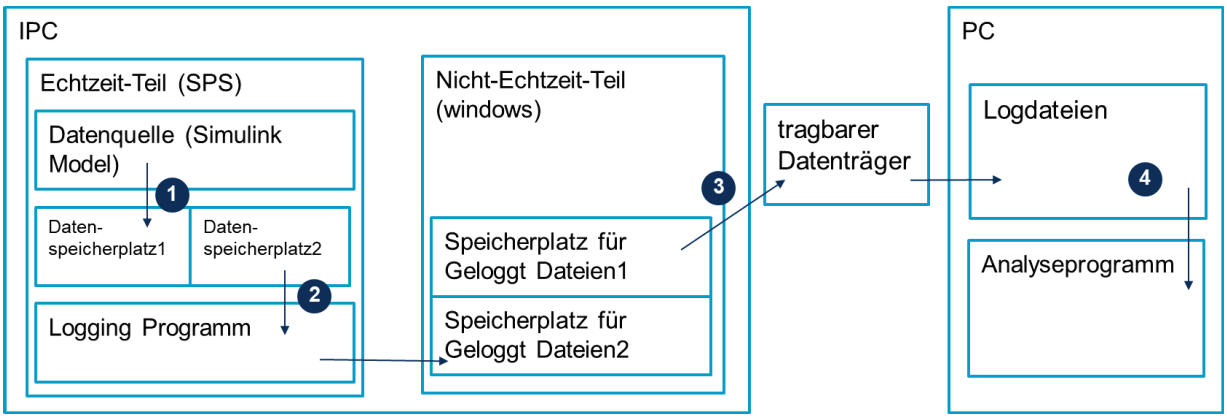


Abbildung 11 Datenfluss des Data-Logging-Systems

5.2 Form des Dateneinlesens von der Datenquelle

In jedem Taskzyklus generiert die Datenquelle einen neuen Datensatz. Die aufzuzeichnenden Daten sind die vom Steuerprogramm erzeugten Eingangssignale, berechneten Werte und Ausgangssignale, die für eine weitere Verwendung gespeichert werden sollen.

Es gibt zwei Arten Daten aufzuzeichnen: die Eine besteht darin Daten kontinuierlich aufzuzeichnen und die andere besteht darin, Daten nur dann aufzuzeichnen, wenn ein Ereignis eintritt (eventbasiert) z.B. bei Störungen oder abnormale Werte.

Kontinuierliches Data-Logging

Wenn kontinuierliche Daten benötigt werden, um das Langzeitverhalten der SPS-Anwendung zu bewerten, wird der kontinuierliche Aufzeichnungsmodus ausgewählt. Der Logging-Vorgang wird dauerhaft fortgesetzt und Daten werden kontinuierlich geloggt, so lange bis der kontinuierliche Aufzeichnungsmodus abgewählt wird oder die Steuerung ausgeschaltet wird.

Eventbasiertes Data-Logging

Wenn Daten im Zeitraum um ein Ereignis benötigt werden, wird der eventbasierte Aufzeichnungsmodus ausgewählt. Ein Ereignis kann ein ungewolltes Maschineverhalten sein (Störung).

Die eventbasierte Art hat drei Varianten:

1. Das Event steht am Ende der Aufzeichnung. Diese Art wird verwendet, um die Ursachen der Störung zu analysieren.
2. Das Event steht am Anfang der Aufzeichnung. Diese Art wird verwendet, um das Verhalten des Systems nach der Störung zu beobachten.
3. Das Event steht im Mitte der Aufzeichnung. Diese Art wird verwendet, um sowohl die Ursachen der Störung zu analysieren als auch das Verhalten des Systems nach der Störung zu beobachten.

Während des normalen Betriebs werden diese Werte nicht aufgezeichnet. In der entwickelten Data-Logging-Funktion werden 20 Datensätze (zehn Millisekunden je Datensatz) vor und nach dem Ereignis aufgezeichnet. Die Datensatzmenge kann nach Bedarf angepasst werden.

Im Programm wird die dritte Art ausgewählt.

5.3 Buffer

Ein Buffer ist ein Datenspeicherplatz im Echtzeitsystem, welches in Abbildung 11 dargestellt ist. Da derselbe Speicherplatz zu gleichen Zeit nicht gelesen und geschrieben werden kann, wird ein zweiter Buffer benötigt.

5.3.1 Bufferstruktur

Der Buffer wird als Datensatzarray realisiert. Bei der kontinuierlichen Data-Logging-Funktion enthält das Datensatzarray 100 Elemente, das bedeutet die erzeugte Datei enthält 100 Datensätze. Bei der eventbasierten Data-Logging-Funktion enthält dieses Datensatzarray 20 Elemente, das heißt die erzeugte Datei enthält auch 20 Datensätze.

Diese Datensatzarray-Buffer wird im SPS-Programm als Array vom Typ Struktur[13] umgesetzt. Im Testbeispiel wird ein Datensatz aus einem Zeitstempel (Timestamp) und zehn Signalen zusammengesetzt, welche von Simulink Model in Abbildung 7 zu Verfügung gestellt werden. Diese Signale sind die Heizwassertemperatur (watertemp), die Außentemperatur (setoutsidetemp), der Heizwassertemperatursollwert (setwatertemp), die Raumtemperatur der beiden Räume (roomtemp1 und roomtemp2), die Raumtemperatursollwerte der beiden Räume (setroomtemp1 und setroomtemp2), die Stellgröße vom Wassertankregler (a) und die Stellgrößen der beiden Räume (b und b2). Die Struktur dieses Datensatzarrays „Buffer[n]“ ist in Abbildung 12 dargestellt.

[13] Struktur: Anordnung von beliebigen Datentypen

```
TYPE Datensatz :
STRUCT
    Timestamp : STRING ;
    watertemp : LREAL;
    setoutsidetemp : LREAL;
    setwatertemp : LREAL;
    roomtemp1 : LREAL;
    setroomtemp1 : LREAL;
    roomtemp2 : LREAL;
    setroomtemp2 : LREAL;
    a : LREAL;
    b1 : LREAL;
    b2 : LREAL;
END_STRUCT
END_TYPE
```

Abbildung 12 Struktur des Datensatzarrays im SPS-Programm

5.3.2 Lesen/Schreiben des Buffers

Im Buffer wird jeder Datensatz in eine Position geschrieben. Das bedeutet der erste Datensatz wird in die erste Position des Buffers geschrieben, der zweite Datensatz wird in die zweite Position des Buffers geschrieben usw. Der 20. Datensatz wird in die 20. bzw. die letzte Position des Buffers geschrieben. Danach wird der 21. Datensatz wieder in die erste Position des Buffers geschrieben. Wenn ein Ereignis eintritt, bilden die 20 Datensätze im Buffer den Zeitraum (0,2 Sekunden) vor dem Ereignis ab (siehe Abbildung 13). Beim Lesen des Buffers wird der gesamte Buffer als ein Array vom Data-Logging-Programm gelesen.

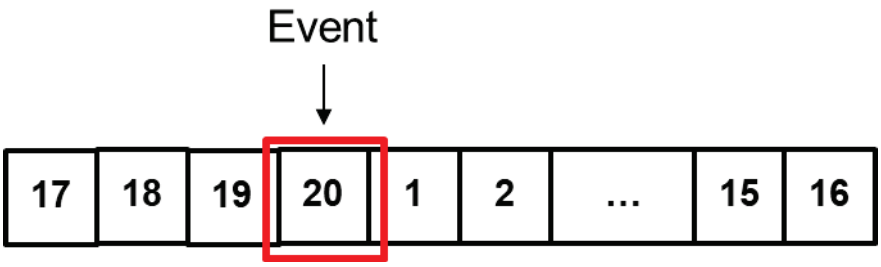


Abbildung 13 Buffer in eventbasierter Modus

5.4 Data-Logging-Funktion

Nachdem die Data-Logging-Varianten in Kapitel 4.3 verglichen worden sind, wird die Variante mit dem XML-Server (TwinCAT PLC) und die Variante mit dem in C++ selbsterstellten Programm (TwinCAT C++) ausgewählt, um eine Data-Logging-Funktion in TwinCAT3 zu realisieren.

5.4.1 Variante 1: Data-Logging-Funktion in TwinCAT PLC

TwinCAT3 unterstützt die Realisierung von einen oder mehreren SPS-Projekten auf einer CPU. Steuerprogramme werden im SPS-Projekt geschrieben, um die gewünschten Funktionen zu erreichen (vgl. [12]). Ein SPS-Programm kann in den IEC 61131-3 Programmiersprachen FUP[14], KOP[15], AWL[16], ST[17] erstellt werden. Die ST-Sprache wird in der folgenden Programmierung verwendet.

In dieser Variante wird die von Beckhoff bereitgestellte SPS-Funktion XML-Server (Server und Bibliothek) verwendet, um die Data-Logging-Funktion zu erstellen.

5.4.1.1 TwinCAT3 Funktion XML Server

Diese Data-Logging-Funktion verwendet die von Beckhoff bereitgestellte Funktion TF6421 „TC3 XML-Server“. TF6421 besteht aus 2 Komponenten: dem XML Server und der SPS-Bibliothek Tc2_XmlDataSrv (vgl. [13]).

Der TwinCAT XML-Server bietet eine Schnittstelle zu XML-Dateien, mit der ein Schreib- und Lesezugriff auf XML-Dateien realisiert wird. Die SPS-Bibliothek bietet vier verschiedene Funktionsbausteine mit Lese- oder Schreibfunktionen: FB_XmlSrvWrite, FB_XmlSrvRead, FB_XmlSrvWriteName und FB_XmlSrvReadName. Nach dem Installieren des XML-Servers auf den IPC erfolgt

[14] FUP: Funktionsplan
[15] KOP: Kontaktplan
[16] AWL: Anweisungsliste
[17] ST: Strukturierter Text

die Kommunikation zwischen dem SPS-Programm und dem XML-Server mit Hilfe der Funktionsbausteine.

Funktionsbaustein: FB_XmlSrvWrite

Um die Variablenwerte in der SPS zyklisch in eine XML-Datei zu schreiben, wird der Funktionsblock FB_XmlSrvWrite verwendet. Die Ein- und Ausgänge des Funktionsblocks sind im Anhang C dargestellt. In ST werden die Eingänge eines Funktionsblocks unter VAR_INPUT deklariert (siehe Anhang C) und die Ausgänge eines Funktionsblocks werden unter VAR_OUTPUT (siehe Anhang C).

XML-Server-Performancetest

Die Bufferschreib- und die Bufferleseoperationen werden zyklisch ausgeführt. Die Zeit zum Aufzeichnen von Daten aus dem Buffer in einer XML-Datei muss kürzer sein als die Zykluszeit des Bufferschreibvorgangs. Auf diese Weise wird sichergestellt, dass alle im vorherigen Zyklus eingegebenen Werte innerhalb eines Schreibzyklus in der XML-Datei aufgezeichnet werden und im nächsten Zyklus nicht überschrieben und verloren werden.

Aus der Dokumentation von Beckhoff ist die Performance des Servers nicht ersichtlich, daher sind Tests erforderlich, um festzustellen, ob der Server die Geschwindigkeitsanforderung für das Bufferschreiben erfüllt.

Ablauf des Testverfahrens

Der Ablauf des Testverfahrens ist in Abbildung 14 dargestellt. Zuerst werden 100 Datensätze in den Buffer geschrieben. Jeder Datensatz besteht aus einem Zeitstempel und zehn Datenwerte. Die Systemzeit wird zum Beginn und zum Ende des Buffer-Schreibvorgangs erfasst und daraus die Dauer des Buffer-Schreibvorgangs berechnet.

Nach Abschluss des Schreibvorgangs wird der Inhalt des Buffers durch Aufrufen der Funktion XmlSrvWrite in das XML-Dokument geschrieben. Die Systemzeit wird zum Beginn und zum Ende des Schreibens der XML-Datei erfasst. Die Dauer für das Schreiben der XML-Datei wird daraus berechnet.

Die erzeugte XML-Datei wird auf Vollständigkeit geprüft. Die beiden berechneten Zeitdauern werden verglichen, um festzustellen, ob die Laufzeit des XML-Servers den Anforderungen entspricht.

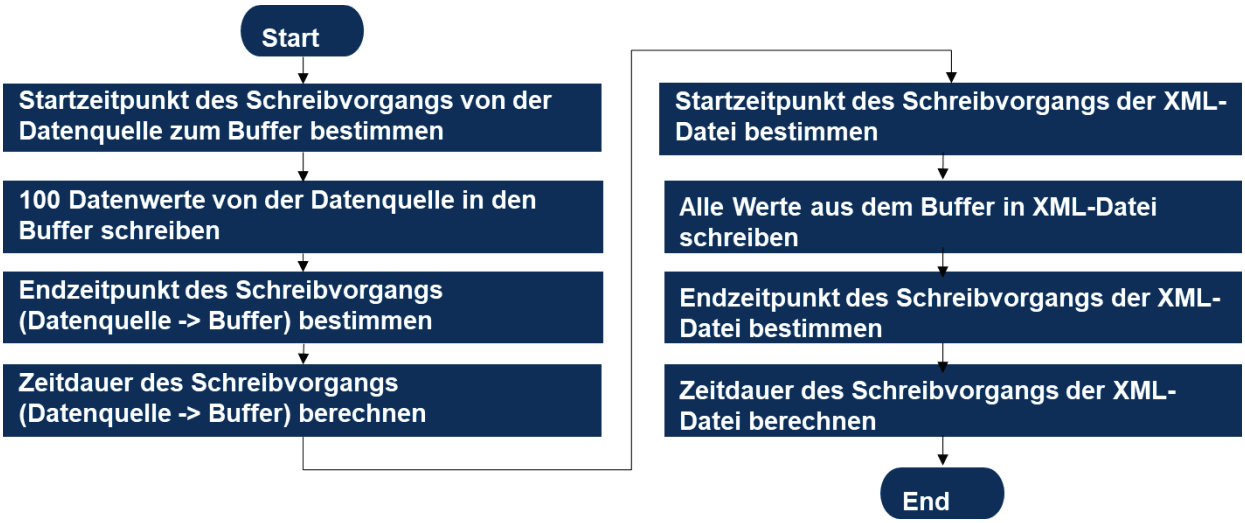


Abbildung 14 Flussdiagramm des XML-Server-Testverfahrens

Funktionen und –befehle

Die Beziehung zwischen den Funktionsbausteinen ist in Abbildung 15 dargestellt und wird im Folgenden erläutert.

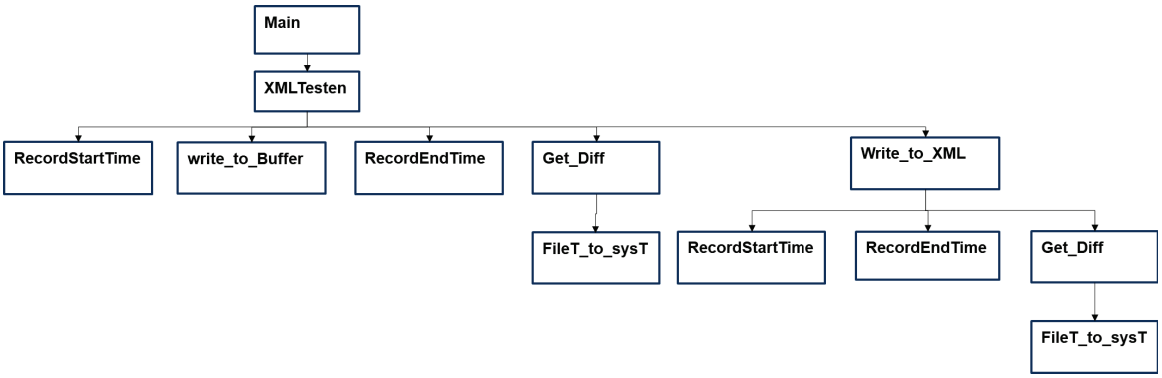


Abbildung 15 aufrufende Beziehung zwischen den Funktionsbausteinen in XMLServer-Testprogramm

XMLTesten: Der Programmbaustein „XMLTesten“ funktioniert als Hauptprogramm für das Testen des XML-Servers. Aus diesem Baustein werden die Funktionesbausteine „RecordStartTime“, „write_to_Buffer“, „RecordEndTime“, „Get_Diff“ und „Write_to_XML“ nacheinander ausgeführt.

RecordStartTime: Mit dem Funktionsbaustein „RecordStartTime“ wird der Anfangszeitpunkt des Schreibvorgangs ermittelt. Nachdem eine ansteigende Flanke erkannt wird, die den Start des

Schreibvorgangs darstellt, wird der Zeitstempel des Betriebssystems durch Aufruf der Standardfunktion FB_getsystemtime() ausgelesen, welche Teil der Beckhoff PLC-Bibliothek Tc2_Utilities ist. Der Zeitstempel ist ein 64-bit Integer Wert mit einer Genauigkeit von 100ns ab 1.1.1601 00:00. Die Ausgänge sind die niederwertigeren und höherwertigeren 4 Byte des Zeitstempels BeginLo und BeginHi (siehe Anhang A).

write_to_Buffer: In dem Funktionsblock „write_to_Buffer“ werden Werte in einen Datensatz in den Buffer geschrieben.

RecordEndTime: Der Endzeitpunkt des Schreibvorgangs erfolgt mit dem Funktionsbaustein „write_to_Buffer“. Nachdem eine ansteigende Flanke am Eingang erkannt wird, die das Ende des Bufferschreibprozesses (write_to_Buffer) darstellt, wird der Zeitstempel des Betriebssystems zu diesem Zeitpunkt ähnlich wie beim Funktionsbaustein „RecordStartTime“ ermittelt. Die Ausgänge dieses Funktionsbausteins sind die niederwertigeren und höherwertigeren 4 Byte des Zeitstempels EndLo und EndHi.

Get_Diff: Mit dem Funktionsbaustein wird die Differenz von BeginLo und EndLo sowie BeginHi und EndHi berechnet. Hierzu wird der Funktionsbaustein FB_FileT_to_sysT aufgerufen.

FileT_to_sysT: Der Funktionsbaustein „FileT_to_sysT“ konvertiert die Zeitdifferenz von dem FILETIME-Format (64-bit Integer) in das SYSTEMTIME-Format (Datentyp: TIMESTRUCT). TIMESTRUCT besteht aus den acht WORD-Variablen: wYear, wMonth, wDayofWeek, wDay, wHour, wMinute, wSecond, wMiliseconds.

Write_to_XML: In dem Funktionsblock „Write_to_XML“ wird der Funktionsblock „FB_XmlSrvWrite“ aufgerufen, um Daten vom Buffer in die XML-Datei zu schreiben und den Start- und Endzeitpunkt des XML-Schreibvorgangs zu bestimmen.

Testergebnis

Da in jedem Taskzyklus (auf 10ms eingestellt) ein Datensatz geschrieben wird, beträgt die Zeit für den Schreibvorgang des Buffers 100 Zyklen * 10ms/Zyklus = 1s. Die Messung ergibt, dass der Schreibvorgang eine Sekunde dauert und dem berechneten Wert entspricht. Weiterhin wird für das Schreiben der 100 Datensätze in die XML-Datei eine Zeitdauer von 70 ms gemessen. Die Zeitdauer des Schreibens der XML-Datei ist geringer als die Zeitdauer des Schreibens der Daten in den Buffer. Dieses entspricht den Anforderungen.

5.4.1.2 Dateiennamen

Der Dateiname setzt sich aus einem Zeitstempel und einem Bezeichner zusammen. Die Struktur des Zeitstempels ist Jahr-Monat-Tag-Stunde-Minute-Sekunde-Millisekunde (JJJJ-MM-TT-SS-mm-ss-SSS). Der Bezeichner enthält Anwendungsinformationen wie die Maschinenbezeichnung, um die Dateien mit den aufgezeichneten Daten eindeutig einer Anwendung zuordnen zu können. Im Fall der Testanwendung wird der Bezeichner „TEST“ verwendet. Der Dateiname der XML-Datei sieht beispielhaft wie folgt aus: 2020-08-10-19-13-56-123TEST.xml.

5.4.1.3 XML-Dateistruktur

Die erzeugte XML-Datei besteht aus zwei Teilen: einem Kopfteil „Header“ und einem Datenteil „Datapart“. (siehe Abbildung 16)

Im Kopfteil „Header“ werden maschinenbezogene Informationen wie Maschinenbezeichnung bzw. Seriennummer, Maschinentyp, Kunde, etc. gespeichert. Die Informationen im Header werden in der Struktur XMLHeader in SPS-Programm hinzugefügt.

Der Datenteil „Datapart“ entspricht dem Inhalt des Buffers und besteht beim kontinuierlichen Data-Logging aus 100 Datensätzen, beim eventbasierten Data-Logging aus 20 Datensätzen. Die Datenstruktur des Datensatzes in der XML-Datei ist ähnlich wie die Datenstruktur des Buffers aufgebaut (vergleiche in Kapitel 5.3.1). Die Daten des Datensatzes entsprechen den Daten des Buffers. Alle Daten in der XML-Datei werden in der Form „<key> value </key>“ gespeichert. „key“ entspricht hierbei dem zugehörigen Variablennamen in der Bufferstruktur, z. B. „Timestamp“. „value“ ist der Variablenwert.

```
<datafromroom>
      <Header>
            <Maschineninformation>value</Maschineninformation>
            <Bestellerinformation>value</Bestellerinformation>
            <Herstellerinformation>value</Herstellerinformation>
      </Header>
      <datapart index="1">
            <Signal1>value</Signal1>
            <Signal2>value</Signal2>
            ……
            <Signal10>value</Signal10>
      </datapart>
      <datapart index="2">
            <Signal1>value</Signal1>
            <Signal2>value</Signal2>
            ……
            <Signal10>value</Signal10>
      </datapart>
      ……
</datafromroom>
```

Abbildung 16 XML-Dateienstruktur

5.4.1.4 Aufbau des SPS-Programms

Es gibt zwei Modi, die im Hauptprogramm zur Auswahl stehen: das kontinuierliche Data-Logging und das eventbasierte Data-Logging. Die entsprechenden Unterprogramme werden verwendet: kontinuierliches Data-Logging-Programm und eventbasiertes Data-Logging-Programm. Die Aufrufe der Funktionsbausteine sind in Abbildung 17 dargestellt.

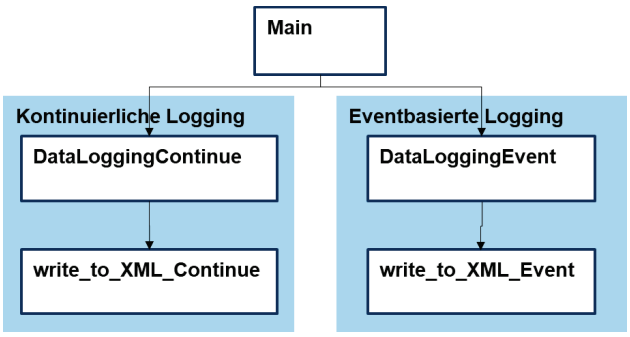


Abbildung 17 Unterprogramme des Data-Logging-Programms (Variante 1)

Aufbau des kontinuierlichen Data-Logging-Programms (Unterprogramm)

Der Programmablauf des kontinuierlichen Data-Logging-Programms ist in Abbildung 18 dargestellt. Zuerst werden 100 Datensätze in den Buffer1 geschrieben und jeder Datensatz bekommt einen Zeitstempel und 10 Werte. Nach Abschluss des Schreibvorgangs, wenn der Buffer gefüllt

ist, wird der Inhalt von Buffer1 durch Aufrufen der Funktion XmlSrvWrite in ein neues XML-Dokument geschrieben. Wenn Daten aus Buffer1 gelesen werden, dürfen gleichzeitig nicht Daten in Buffer1 geschrieben werden, um sicherzustellen, dass die richtigen Daten gelesen werden. Gleiches gilt für Buffer2.

Während des Schreibens der XML-Datei von Buffer1 werden Daten von der Datenquelle in Buffer2 geschrieben. Nachdem Buffer2 voll ist, wird der Inhalt von Buffer2 durch Aufrufen der Funktion XmlSrvWrite in ein neues XML-Dokument geschrieben. Während des Schreibens der XML-Datei von Buffer2 werden die neuen Datensätze in Buffer1 geschrieben.

Der Vorgang wird kontinuierlich wiederholt. Der Zeitstempel im Dateinamen stimmt mit dem Zeitstempel des letzten Datensatzes in der Datei überein.

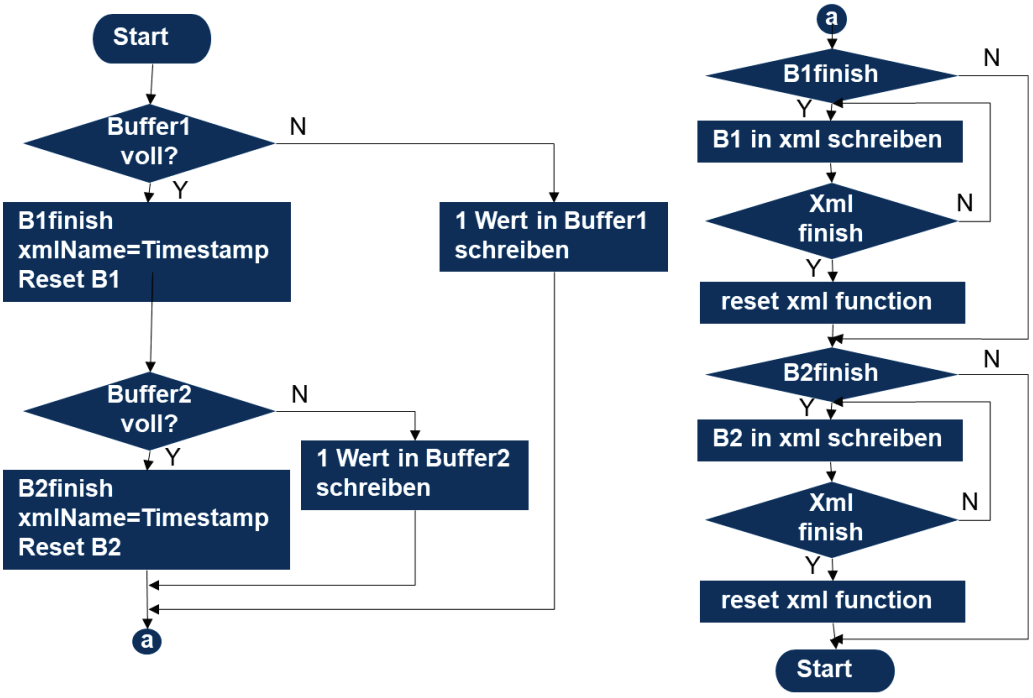


Abbildung 18 Flussdiagramm des Kontinuierlichen Data-Logging-Programms in Variante 1[18]

[18] B1=Buffer1, B2=Buffer2

Funktionsbausteine

DataLoggingContinue: Im Funktionsblock „DataLoggingContinue“ werden Werte durch Aufruf des Funktionsblocks „write_to_XML_Continue“ in den Buffer geschrieben. In jedem Zyklus wird DataLoggingContinue aufgerufen und ein Datensatz in Buffer geschrieben. Wenn der Buffer voll ist, wird das Flag bBufferFinish auf TRUE gesetzt. Die Variable LogInput verweist dann auf den anderen Buffer, in dem die neuen Datensätze dann geschrieben werden.

write_to_XML_Continue: Wenn bBufferFinish TRUE ist, wird der Funktionsbaustein write_to_XML_Continue aufgerufen. In diesem Funktionsblock wird FB_XmlSrvWrite zweimal aufgerufen. Beim ersten Mal wird der Inhalt vom Kopfteil in das XML-Dokument geschrieben und beim zweiten Mal wird der Inhalt des von LogInput verwiesenen Buffers in das XML-Dokument geschrieben. Das Ende der Bearbeitung von FB_XmlSrvWrite wird durch eine fallende Flanke des Ausgangs bBusy erkannt. In diesem Fall ist auch die Bearbeitung von write_to_XML_Continue beendet und der Ausgang WriteFinishFlag wird auf TRUE gesetzt.

Nachdem WriteFinishFlag gesetzt ist, werden bBufferFinish und WriteFinishFlag zurückgesetzt. write_to_XML_Continue ist für den nächsten Schreibvorgang bereit.

Aufbau des eventbasierten Data-Logging-Programms (Unterprogramm)

Der Programmablauf des eventbasierten Data-Logging-Programms ist in Abbildung 19 dargestellt. In jedem Zyklus wird ein Datensatz in den Buffer1 geschrieben. Die Kapazität dieses Buffers beträgt 20 Speicherplätze (pro Datensatz ein Speicherplatz). Nachdem ein Datensatz in den 20-ten Speicherplatz geschrieben worden ist, wird der nächste Datensatz in den ersten Speicherplatz des Buffers geschrieben. (Der Inhalt der Speicherplätze wird überschrieben.) Buffer1 wird solange überschrieben, bis ein Ereignis auftritt.

Wenn ein Ereignis auftritt, befinden sich im Buffer1 die 20 Datensätze, welche unmittelbar vor dem Ereignis liegen. Zu diesem Zeitpunkt werden die 20 Datensätze aus Buffer1 in den Buffer2 kopiert. Der Inhalt von Buffer2 wird mit der Funktion XmlSrvWrite in ein XML-Dokument geschrieben.

Während des Schreibens der XML-Datei von Buffer2 werden neue Datensätze in Buffer1 geschrieben. Nachdem Buffer1 wieder voll ist, das heißt 20 weitere Datensätze sind in Buffer1 geschrieben worden, werden diese 20 Datensätze in den Buffer2 übertragen. Anschließend wird der Inhalt von Buffer2 in ein neues XML-Dokument geschrieben. Diese 20 Datensätze sind die 20 Datensätze nach dem Ereignis. Während des Schreibens der XML-Datei von Buffer2 werden

neue Datensätze wieder in Buffer1 geschrieben. Es werden insgesamt zwei XML-Dateien pro Ereignis erzeugt.

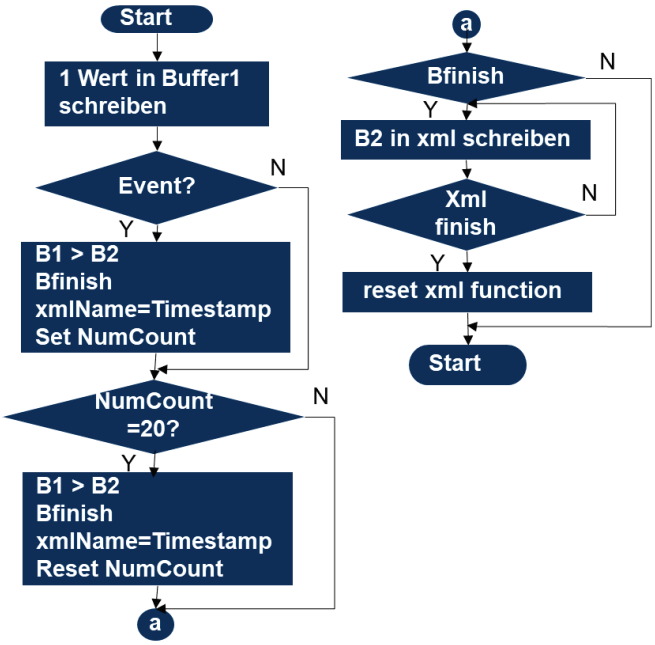


Abbildung 19 Flussdiagramm des eventbasierten Data-Logging-Programms in Variante 1[19]

Funktionsbausteine

DataLoggingEvent: In dem Funktionsblock „DataLoggingEvent“ werden Datensätze in den Buffer geschrieben und mit dem Funktionsblock FB_write_to_XML_Event der Bufferinhalt in eine XML-Datei geschrieben. In jedem Zyklus wird DataLoggingEvent aufgerufen und ein Datensatz in Buffer1 geschrieben. Wenn eine steigende Flanke am Eingang Event detektiert worden ist, werden die Datensätze von Buffer1 nach Buffer2 übertragen. Anschließend wird das Flag bBufferFinish auf TRUE gesetzt.

write_to_XML_Event: Wenn bBufferFinish TRUE ist, wird der Funktionsbaustein „write_to_XML_Event“ gestartet. Dieser Funktionsbaustein funktioniert ähnlich wie der Funktionsbaustein „write_to_XML_Event“ beim kontinuierlichen Data-Logging. Der Unterschied ist die Größe des aufzuzeichnenden Buffers. In write_to_XML_Event wird der Inhalt von Buffer2 in die XML-Datei geschrieben.

[19] B1=Buffer1, B2=Buffer2, B1>B2: von B1 zu B2 übertragen

5.4.2 Variante 2: Data-Logging-Funktion in TwinCAT C++

Nachdem ein neues C ++ - Projekt erstellt worden ist, werden die Ordner für die „source file“s und die „header file“s automatisch von der Entwicklungsumgebung erzeugt. Die Programmierung mit TwinCAT C++ entspricht der Programmierung mit C++.

5.4.2.1 Verwendete Schnittstellen

Beckhoff stellt Schnittstellen zur Verfügung, um die vom Anwender entwickelten C++-Module zu verwenden. Im Folgenden werden die Schnittstellen aufgeführt, die für die Data-Logging-Funktion erforderlich sind.

Schnittstelle TcFileAccess

Mit der Schnittstelle TcFileAccess wird über ein TwinCAT C ++ - Modul auf das Dateisystem des IPCs zugegriffen. Die Quelldatei muss die Unterstützungsdatei TcFileAccessInterfaces.h von Beckhoff aus dem Ordner „external dependencies“ enthalten(vgl. [33]). Die von der Schnittstelle bereitgestellten Methoden sind im Anhang D aufgeführt:

Wenn die Datei, auf die fileOpen : szFileName verweist, nicht vorhanden ist, wird die Datei automatisch generiert. Die Datentypen TcFileHandle und TcFileAccessMode werden in der von Beckhoff bereitgestellten Header-Datei TcFileAccessServices.h definiert. TcFileAccessMode wird im Programm in den folgenden Modi verwendet (siehe Tabelle 4).

Tabelle 4 Mode Code für TcFileAccessMode (vgl. [34])

Mode Name	Mode Code
amRead	0x00000001
amWrite	0x00000002
amAppend	0x00000004
amBinary	0x00000010
amText	0x00000020

Die Modi können überlagert werden. (Dies ist der Dokumentation nicht zu entnehmen). Im Programm werden die Kombination der Modi „amBinary“ und „amAppend“ verwendet. Das bedeutet, dass die als Zahlen definierten Daten in binärer Form in einer Datei gespeichert und bearbeitet werden. Neue Daten werden am Ende der Datei angehängt, wenn dieselbe Datei zum wiederholten Mal zum Schreiben geöffnet wird.

Schnittstelle ITcTask

Die Schnittstelle ITcTask wird verwendet, um den Zeitstempel, die Anzahl der Taskzyklen seit Taskstart und die Taskpriorität der TwinCAT-Task abzufragen. Die Methode GetCurrentSysTime von dieser Schnittstelle wird in diesem Programm verwendet, um den Zeitstempel für jeden Datensatz abzurufen.

5.4.2.2 Initialisierung des zyklischen Aufrufprogramms

Wenn ein C++ - Modul als Data-Logging-Funktion verwendet wird, überträgt das Programm in jedem Taskzyklus den aus der Datenquelle erhaltenen Wert in den Buffer. Es ist bei der Auswahl des TwinCAT C++-Moduls darauf zu achten, dass das Programm zyklisch aufgerufen wird.

Wenn beim Hinzufügen des neuen Twincat C++ - Moduls die Klasse „module with cyclic caller“ ausgewählt wird, ist bereits eine zyklische aufrufende Umgebung im erzeugten Instanz-Modul vorhanden. Das heißt, die Zustandsmaschine, die Schnittstelle ITcCyclicCaller und die Schnittstelle ITcCyclic werden durch die Entwicklungsumgebung initialisiert. In den folgenden Abschnitten werden die Funktionen dieser drei Komponente kurz beschrieben. Das spezifische Initialisierungsprogramm ist in der Klasse „module with cyclic caller“ erhalten.

Zustandsmaschine

Jedes Modul enthält eine Zustandsmaschine, die den Initialisierungsstatus des Moduls und den Status beim Starten und Beenden des Moduls beschreibt. Die Zustandsmaschine definiert die Zustände INIT, PREOP, SAFEOP und OP. Neben dem Status gibt es entsprechende Statusübergänge, in denen allgemeine oder modulspezifische Operationen ausgeführt werden, z. B. Modulerstellung, Parametrierung und Verbindungsaufbau mit anderen Modulen.

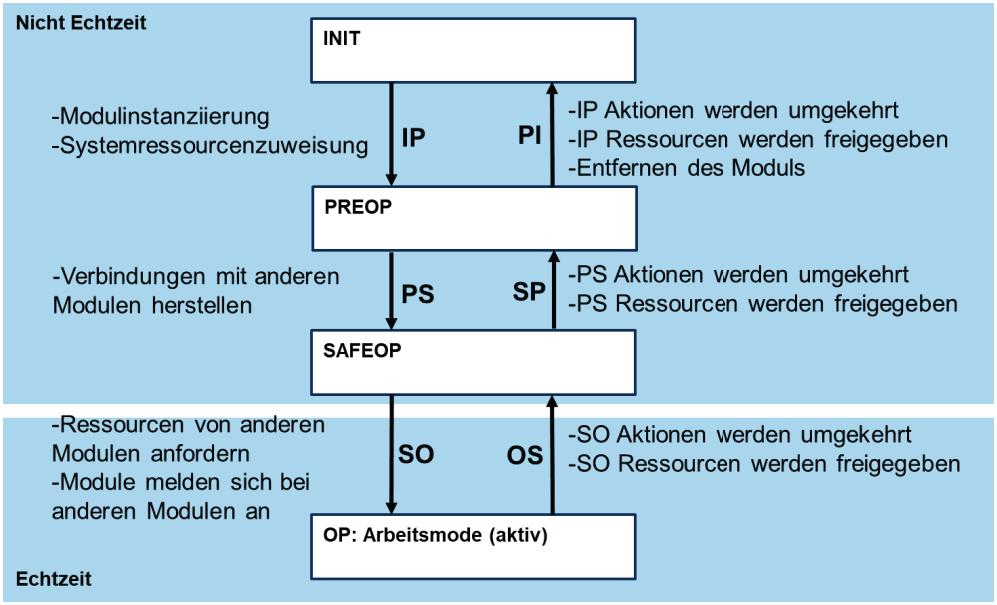


Abbildung 20 Status und Statusübergänge in Zustandsmaschine (vgl. [35])

Schnittstelle ITcCyclic

Die ITcCyclic-Schnittstelle wird einmal pro Taskzyklus aufgerufen. Sie enthält die Methode CycleUpdate. CycleUpdate wird normalerweise von einer TwinCAT Task aufgerufen, nachdem die Schnittstelle angemeldet worden ist. Der Hauptcode der zu implementierenden Funktion befindet sich in der CycleUpdate-Methode der Modulinstanz (vgl. [36]).

Schnittstelle ITcCyclicCaller

Ein Modul verwendet die ITcCyclicCaller-Schnittstelle, um seine ITcCyclic-Schnittstelle bei einem Task anzumelden und abzumelden. Nach dem Anmelden wird die CycleUpdate-Methode der Modulinstanz aufgerufen. Es wird normalerweise als letzter Initialisierungsschritt im Statusübergang zwischen SAFEOP und OP verwendet (vgl. [37]).

5.4.2.3 Dateienname

Der Dateiname setzt sich wie in Variante 1 (siehe Kapitel 5.4.1.2) zusammen. Der Dateiname besteht aus einem Zeitstempel und einem Bezeichnerteil. Die Struktur des Zeitstempels ist Jahr-

Monat-Tag-Stunde-Minute-Sekunde-Millisekunde. Der Bezeichnerteil enthält Anwendungsinformationen wie z. B. eine Gerätekennzeichnung. In der Test-anwendung wird für den Bezeichnerteil „TEST“ verwendet.

5.4.2.4 Dateienstruktur

Die generierte Datei (Daten-Datei) enthält keinen für Menschen lesbaren Inhalt wie ASCII-Zeichen, sondern Binärdaten. Wenn die zu loggenden Werte als double-Datentyp definiert worden sind, werden alle acht Byte des Werts in die Logging-Datei geschrieben. Je nach Aufnahmemodus enthält jede Datei 100 Datensätze im kontinuierlichen Mode oder 20 Datensätze im eventbasierten Mode. Die einzelnen Daten in jedem Datensatz werden weder mit Trennzeichen wie Komma noch mit Leerzeichen in die Binärdatei geschrieben. Zwei aufeinanderfolgende Datensätze werden ebenfalls nicht getrennt. Die Reihenfolge der Daten im Datensatz stimmt mit der Reihenfolge der Daten in der Bufferstruktur überein. Die Struktur des Buffers (Abbildung 21a) gibt die Datenreihenfolge der entsprechenden Binärdatei vor (Abbildung 21b).

```
struct st_Buffer
{
    double Timestamp;
    double setRoomT1;
    double setRoomT2;
    double setWaterT;
    double setOutT;
    double watertemp;
    double roomtemp1;
    double roomtemp2;
    double a;
    double b;
    double b2;
};
```

Datei
Timestamp/setRoomT1/setRoomT2/setWaterT/setOutT/watertemp/roomtemp1/roomtemp2/a/b/b2/Timestamp/setRoomT1/setRoomT2/setWaterT/setOutT/watertemp/roomtemp1/roomtemp2/a/b/b2/Timestamp/setRoomT1/setRoomT2/setWaterT/setOutT/watertemp/roomtemp1/roomtemp2/a/b/b2……

Abbildung 21 a) Struktur des Buffers in Variante 2 b) Datenreihenfolge in der generierten Datei in Variante 2

Der erste Wert (Timestamp) in jedem Datensatz stellt den Zeitstempel dar. Er ist ein „double“-Wert und zeigt die Zeit (UTC), die seit dem 1. Januar 1601 vergangen ist, in 100 Nanosekunden-Schritten an.

Gleichzeitig wird eine Datei für die Variablentags (Title-Datei) im gleichen Ordner generiert. Die Namen der aufzuzeichnenden Variablen werden als Zeichenkette, durch Kommas getrennt, in die

Variable „Title“ eingegeben. Wenn sich die aufzuzeichnenden Variablen nicht ändern, wird die Datei nur einmal generiert.

5.4.2.5 Aufbau des Logging Programms

Der Ablauf des Logging Programms ist in der Abbildung 22 dargestellt. Während des Schreibens wird ein Zeiger verwendet: Der Zeiger „Buffer_write“ zeigt auf den Buffer, dessen Inhalt in die Binärdatei geschrieben werden soll. Nach dem Start wird die Betriebsart „Kontinuierlich“ oder „Eventbasiert“ bestimmt und das entsprechende Unterprogramm ausgeführt. Das Unterprogramm wird verwendet, um den Buffer zu füllen und den Zeiger „Buffer_write“ zu setzen.

Wenn der Zeiger auf einen bestimmten Buffer zeigt, wird der Wert im Buffer in die Datei eingegeben und der Zeiger zu NULL zurückgesetzt. Wenn der Zeiger auf NULL zeigt, wird kein Wert aufgezeichnet und keine Operation ausgeführt.

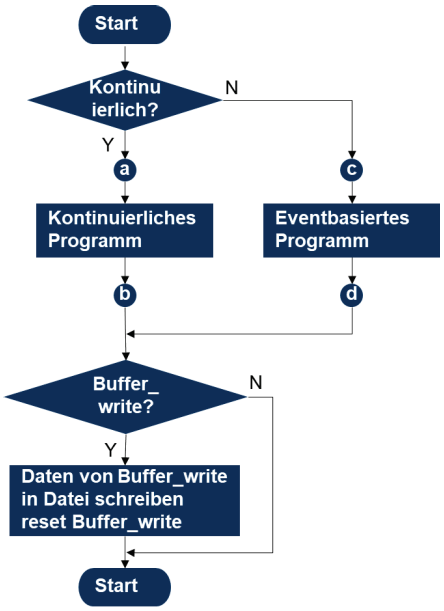


Abbildung 22 Flussdiagramm des gemeinsamen Data-Logging-Programms in Variante 2

Kontinuierliches Data-Logging

In diesem Unterprogramm werden zwei Buffer verwendet. Zusätzlich zu dem oben erwähnten Zeiger „Buffer_write“ gibt es auch einen Zeiger „Buffer_Fill“. Der Zeiger „Buffer_Fill“ zeigt auf den

Buffer, in den Daten geschrieben werden. Die Datensätze werden in den Buffer geschrieben, auf den der Zeiger „Buffer_Fill“ zeigt. Wenn dieser Buffer voll ist, zeigt der Zeiger „Buffer_write“ auf den Buffer, so dass im nächsten Schritt der Wert dieses Buffers in der Datei aufgezeichnet wird. Gleichzeitig zeigt der Zeiger „Buffer_Fill“ auf den anderen Buffer, so dass die Datensätze aus dem nächsten Zyklus in dem anderen Buffer gespeichert werden (siehe Abbildung 23).

Der Vorgang wird kontinuierlich wiederholt. Der Zeitstempel im Dateinamen stimmt mit dem Zeitstempel des letzten Datensatzes in der Datei überein.

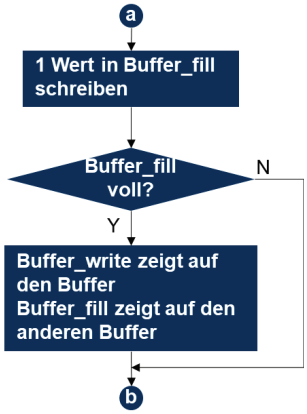


Abbildung 23 Flussdiagramm des kontinuierlichen Data-Logging-Programms in Variante 2

Eventbasiertes Data-Logging

In diesem Unterprogramm werden ebenfalls zwei Buffer verwendet.

Der Ablauf des Unterprogramms ist in Abbildung 24 dargestellt. In jedem Zyklus wird der Datensatz in den durch den Zeiger „Buffer_Fill“ angezeigten Puffer geschrieben. Die Kapazität des Buffers beträgt 20 Datensätze. Nachdem die 20 Datensätze voll sind, wird der Speicher vom Kopf des Buffers aus neu gestartet und die vorherigen Datensätze werden überschrieben. Wenn kein Ereignis auftritt, wird der Buffer1 immer übergeschrieben.

Wenn ein Ereignis auftritt, sind die Datensätze im Buffer, auf die „Buffer_Fill“ zeigt, in diesem Moment die 20 Datensätze direkt vor dem Ereignis. Zu diesem Zeitpunkt zeigt der Zeiger „Buffer_write“ auf diesen Buffer, so dass im nächsten Schritt der Wert dieses Buffers in der Datei aufgezeichnet wird. Gleichzeitig zeigt der Zeiger „Buffer_Fill“ auf den anderen Buffer, so dass die Datensätze aus dem nächsten Zyklus in dem anderen Buffer gespeichert werden.

Ein EventFlag wird gesetzt, was bedeutet, dass der Wert im Puffer innerhalb von 20 Zyklen nach dem Ereignis auch aufgezeichnet werden muss. Wenn 20 Werte in den Puffer geschrieben werden, zeigt der Zeiger „Buffer_write“ auf den Buffer und EventFlag wird zurückgesetzt.

Der Vorgang wird kontinuierlich wiederholt. Allgemeinen werden für jedes Ereignis zwei Dateien generiert. Eine Datei enthält 20 Datensätze vor dem Event und die andere enthält 20 Datensätze nach dem Event. Der Zeitstempel im Dateinamen stimmt mit dem Zeitstempel des letzten Datensatzes in der Datei überein.

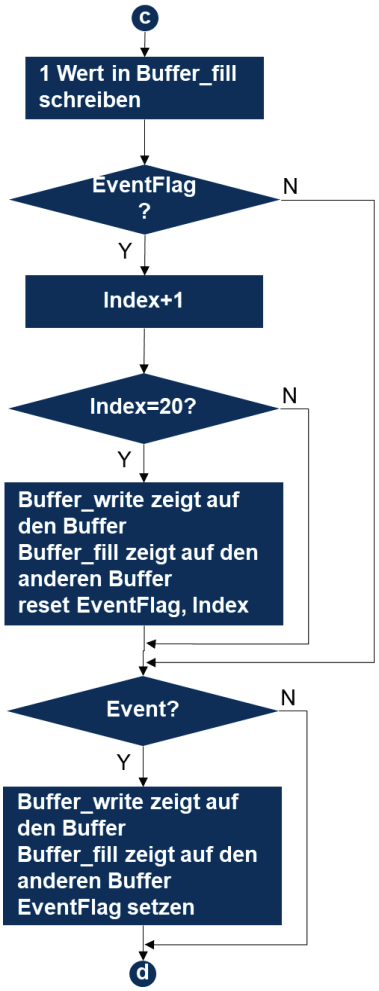


Abbildung 24 Flussdiagramm des Eventbasierten Data-Logging-Programms in Variante 2

5.4.2.6 Methoden und Klassen

Die im Programm verwendeten Klassen, Methoden und Instanzen werden in Tabelle 5 aufgeführt. Von Beckhoff bereitgestellte Klassen werden, soweit es sinnvoll ist, verwendet (CTcFsmFileWriter) bzw. abgewandelt, wenn es erforderlich ist (CTcFsmFileOpen) (vgl. [38]).

Tabelle 5 im C++ Programm verwendete Klassen, Methoden und Instanzen

Klasse	Methode	Objekte	Beschreibung
CTcAsyncBufferWritingModule	FillBuffer	Hauptprogramm	Werte in den Buffer schreiben
CTcFsmFileWriter	Init, Eval	m_fsmFileWriter	Daten-Datei erzeugen
		m_fsmFileTitleWriter	Title-Datei erzeugen
CTcFsmFileOpen	Init, Eval	m_fsmFileOpen	Öffnet die Datei
CTcFsmFileWrite	Init, Eval	m_fsmFileWrite	Schreibt eine Datei
CTcFsmFileClose	Init, Eval	m_fsmFileClose	Schließt eine Datei

Die aufrufende Beziehung zwischen den eigenen Methoden und der von Beckhoff bereitgestellten Schnittstellenmethode ist in Abbildung 25 abgebildet. Die dunkelblauen Blöcke sind die Schnittstellenmethoden von Beckhoff. Die weißen Blöcke sind die eigenen Methoden. GetSystemTime wird verwendet, um den Zeitstempel für jeden Datensatz zu erstellen. FillBuffer ist eine Funktion von CycleUpdate(), die verwendet wird, um Werte vom Simulink Modul in Buffer zu schreiben. Die aus der Klasse CTcFsmFileWriter erzeugten Instanzen m_fsmFileWriter und m_fsmFileTitleWriter werden verwendet, um Dateien zu erstellen. Darin werden Dateien geöffnet, Daten in Dateien geschrieben und Dateien geschlossen. Daher werden Methoden von den Instanzen m_fsmFileOpen, m_fsmFileWrite und m_fsmFileClose verwendet. In Init() werden Inputparameter für Eval() initialisiert und in Eval() die entsprechende Schnittstellenmethode von Beckhoff verwendet. Beispielsweise verwendet m_fsmFileOpen-Eval() die Methode FileOpen im Anhang D, um die Datei zu öffnen.

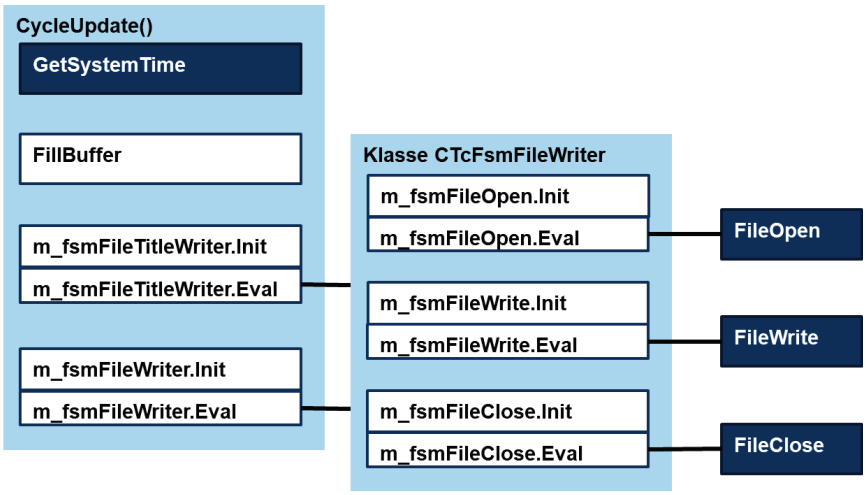


Abbildung 25 Beziehung zwischen den Methoden in Variante 2

5.4.3 Vergleich Variante 1 und 2

Die Variante 1 (XML-Server-Methode) wird verwendet, wenn im SPS-Projekt keine C++-Lizenz verwendet wird. Wenn im SPS-Projekt eine C++-Lizenz verwendet wird, wird Variante 2 (C++-Methode) verwendet.

Die Programmlogik der beiden Varianten in Abbildung 17, Abbildung 23 als kontinuierlichen Mode und in Abbildung 18, Abbildung 24 als eventbasierten Mode sind ähnlich. Der Unterschied spiegelt sich in der Programmiersprache, der Kompilierungsmethode, dem Format der generierten Datei und der Speichermethode der Daten in der Datei wider.

Die Hauptunterschiede zwischen den beiden Varianten sind in Tabelle 6 aufgeführt.

Das Matlab / Simulink-Modul und das C ++ - Modul von TwinCAT3 verwenden die gleiche Lizenz. Da das auf der Steuerung ausgeführte Steuerungsprogramm (Datenquelle der Data-Logging-Funktionalität) in den bei IAV entwickelten Regelungsanwendungen mit Simulink erstellt wird, wird die TC3 C ++ / MatSim Lizenz verwendet. Wenn die Datenerfassungsmethode in C ++ entwickelt oder ausgeführt wird, wird die gleiche TC3 C ++ / MatSim Lizenz benutzt.

Da die von Variante 2 generierten Binärdateien nicht direkt gelesen werden können und Signalnamen und Signalwerte in separaten Dateien gespeichert sind, müssen die Dateien aufbereitet werden, bevor sie weiterverwendet werden. Die von Variante 1 generierten Dateien werden direkt

mit dem integrierten Windows-Editor geöffnet und haben eine klare Datenstruktur und Variablennamen. Bei Bedarf wird die XML-Datei direkt geöffnet, um die Daten zu lesen.

Da zusätzliche Informationen in der XML-Datei gespeichert sind, ist die Datei relativ groß. Ein Vergleich mit gleichen Datensätzen ergibt eine Dateigröße der XML-Datei von 40 KB und eine Dateigröße der Binärdatei von 9 KB.

Tabelle 6 Vergleich Variante 1 und Variante 2

	Variante 1	Variante 2
Programmiersprache	Alle Sprachen von IEC 61131	C++
Compiler	IEC Compiler	C++ Compiler
Lizenz	TC3 PLC TC3 XML Server	TC3 C++/MatSim
Zusätzlicher Download	TC3 XML Server (kostenlos)	-
Dateityp	XML	binär
Dateigröße	relativ groß	relativ klein
Datenlesbarkeit	lesbar	Daten-Datei nicht lesbar Title-Datei lesbar
Title	automatisch erzeugt in XML-Datei	müssen manuell in Programm eingeben werden

Performancebezogene Vergleiche werden in Kapitel 6.4 erläutert.

5.5 Dateienübertragung zum PC (Langzeitspeicher)

Aus den Anforderungen des Systems werden Dateien vom IPC für einen bestimmten Zeitraum aufgezeichnet (siehe Kapitel 3.1). In diesem Teil müssen zwei Punkte berücksichtigt werden: die Medien, die auf physischer Ebene zur Dateienübertragung erforderlich sind, und die erforderliche Operation in IPC.

5.5.1 Hardware für Dateienübertragung

Ein Übertragungszyklus von einer Woche wird angenommen. Bei einer zehn Millisekunde betragenden Zykluszeit und 10 Werten pro Datensatz beträgt die Größe der jede Sekunde generierten XML-Datei 40 KB und die Größe der jede Sekunde generierten binären Datei 9 KB. Die maximale Größe aller in einer Woche generierten XML-Dateien beträgt 40KB/s*3600Sekunde/Stunde*24Stunde/Tag*7Tagen/Woche=24,19 GB. Als Übertragungsmedium wird ein USB-Stick oder eine mobile Festplatte mit einer Kapazität von mehr als 25 GB ausgewählt. Die maximale Größe aller in einer Woche generierten binär-Dateien beträgt 9KB/s*3600Sekunde/Stunde*24Stunde/Tag*7Tagen/Woche=5,44 GB. Als Übertragungsmedium wird ein USB-Stick oder eine mobile Festplatte mit einer Kapazität von mehr als 6 GB ausgewählt.

Die von Beckhoff angebotene Festplatte CX2900-0401 hat eine maximale Kapazität von 1 TB (vgl. [39]). Im kontinuierlichen Logging Mode, wenn jeder Datensatz 10 Werte enthält, speichert die Festplatte Daten als XML-Dateien über 41 Wochen und als Binär-Dateien 184 Wochen. Bei 1.000 Werten pro Datensatz speichert die Festplatte Daten über 69 Stunden und als Binär-Dateien 1,84 Wochen[20].

Schnittstelle zur CPU

Für Beckhoff IPC: Nach der Beschreibung der Beckhoff-IPC-Hardware in Kapitel 3.2.1 verfügt die IPC-CPU über zwei mehrpolige Schnittstellen, die durch Anschließen an den HDD- / SSD-Einschub CX2550-0020 von Beckhoff an eine Festplatte angeschlossen werden (vgl. [40]). Nachdem das Speichermedium eingebaut ist, wird dieses automatisch von dem CPU-Grundmodul erkannt und verwendet. Maximal zwei Erweiterungsmodule CX2550-0020 können an das CPU-Grundmodul angesteckt werden. Dieses zusätzliche Speichermedium (HDD/SDD) wird als Übertragungsmedium verwendet.

Ein weiteres Übertragungsmedium ist der USB-Stick, der über die USB-Schnittstelle des IPC-Moduls verbunden und identifiziert wird.

[20] Eine Woche = sieben Tage

5.5.2 Ablauf der Dateienübertragung

Eine Möglichkeit besteht darin, dass nach dem Einstecken des Übertragungsmediums in den IPC die generierten Dateien auf das Medium übertragen werden und anschließend vom IPC gelöscht werden. Da diese Operation auf dem IPC ausgeführt wird, ist für diese Methode eine visuelle Benutzeroberfläche (Monitor) oder Remote-Control erforderlich.

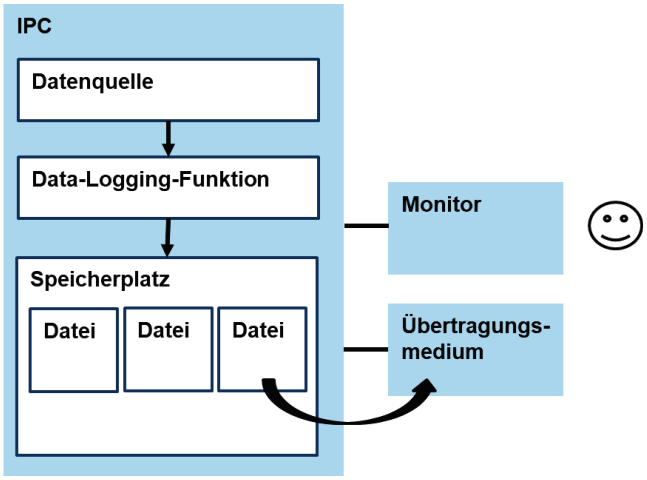


Abbildung 26 Dateienübertragung Möglichkeit: generierte Dateien werden in das Medium eingefügt

Eine andere Möglichkeit besteht darin, den Kurzzeitspeicher als Übertragungsmedium zu verwenden. Beispielweise werden zwei Erweiterungsmodule, das heißt zwei Festplatten, an den Beckhoff-IPC angeschlossen. Die Dateien der ersten Woche werden auf der ersten Festplatte aufgezeichnet. Nach einer Woche wird der Speicherort der Dateien geändert und die nachfolgenden Dateien werden auf der zweiten Festplatte aufgezeichnet. Zu diesem Zeitpunkt wird die erste Festplatte durch eine neue Festplatte ersetzt. Die ersetzte Festplatte ist das Übertragungsmedium.

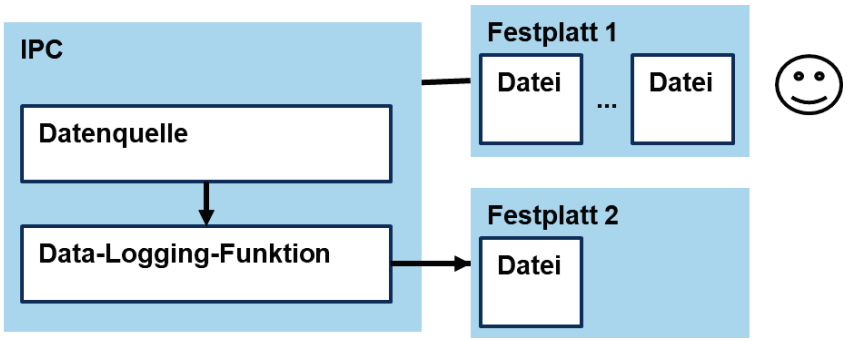


Abbildung 27 Dateienübertragung Möglichkeit: Kurzzeitspeicher als Übertragungsmedium zu verwenden

5.6 Aufbereitung der Data-Logging-Dateien

Die in der Logging-Funktionalität generierten Dateien enthalten für Menschen unlesbare Informationen. Um die Binärdatei verarbeiten zu können, müsste der Anwender den Aufbau dieser Dateien kennen, also die von der SPS gesammelten Daten, sowie Größe und Struktur der Daten. Deshalb werden die Binärdateien aufbereitet, damit der Inhalt der Binärdateien für den Anwender menschenlesbar wird. XML-Dateien sind menschenlesbar. Die aufgezeichneten Werte müssen von dem anderen Inhalt getrennt werden, damit diese Daten verwendet werden können.

5.6.1 XML-Dateien verwenden

Um Messwerte zu analysieren, wird in der Testanwendung Matlab verwendet. Die XML-Dateien werden in eine mat-Datei konvertiert.

Funktion xml2struct

Die Funktion xml2struct ist eine open-sources-Funktion von Mathwork-Dateiaustausch (vgl. [41]). Mit dieser Funktion wird eine XML-Datei in eine MATLAB-Struktur für den Zugriff auf die Daten konvertiert. Die gesamte Datei wird in eine Struktur umgewandelt, die untergeordneten Knoten in der XML-Datei werden in Elemente in der Struktur umgewandelt und das Strukturarray in der XML-Datei in Matlab in ein cell-array. Die im Knoten gespeicherten Daten werden mit dem Key-Wort "Text" indiziert und im String-Format gespeichert.

Aufbau Dateitypwechsel

Zum Konvertieren einer XML-Datei in eine mat-Datei sind insgesamt vier Schritte erforderlich wie in Abbildung 28 dargestellt: Die Datei wird importiert, die Daten in der Datei werden extrahiert, der Datenteil und der Zeitstempel werden in ein Formular eingegeben, der Header wird für das Formular gestellt und das Formular wird gespeichert.

Zuerst wird xml2struct verwendet, um alle XML-Dateien in den Ordner zu importieren, auf den der FilePath verweist und als Struktur gespeichert. Dann wird jede Struktur separat bearbeitet.

Datenteil: Nachdem der Datenteil herausgenommen worden ist, wird dieser von einer Zeichenfolge (string) in den Double-Datentyp konvertiert und in einem MATLAB-Formular übertragen. Die Anzahl der Spalten in der Tabelle entspricht der Anzahl der Signale (inklusiv Timestamp) in jedem Datensatz und die Anzahl der Zeilen der Anzahl der Datensätze. Nachdem der Datenteil gelesen worden ist, wird die Anzahl der Zeilen und Spalten der Tabelle automatisch festgelegt. Das heißt für verschiedene XML-Dateien braucht die Anzahl der Zeilen und Spalten der Tabelle nicht manuell geändert werden.

Zeitstempel: Nachdem der Zeitstempel herausgenommen worden ist, wird dieser von einer Zeichenfolge in den Datetime-Datentyp (als 'yyyy-MM-dd-HH:mm:ss.SSS' gezeigt) konvertiert und in das Formular eingetragen.

Header: Die Strukturfeldnamen werden herausgenommen und in das Formular als Variablennamen eingetragen.

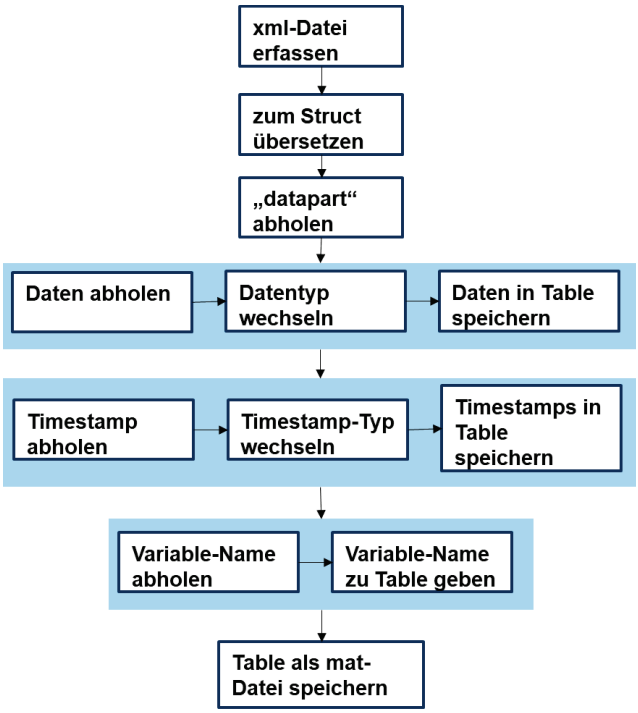


Abbildung 28 Ablauf der Dateitypumwandlung von XML zu mat

5.6.2 binär Dateien verwenden

Um Daten aus einer Binärdatei weiterverarbeiten zu können, werden diese in eine mat-Datei konvertiert.

Aufbau Dateitypumwandlung

Das C ++ - Programm erstellt zwei Dateitypen, Daten-Datei und Title-Datei, die separat verarbeitet werden. Der Ablauf des Konvertierens der Daten-Datei und Title-Datei in die mat-Dateien ist wie in Abbildung 29 dargestellt.

Daten-Datei: Da die Daten in dieser Datei vom Datentyp 8-Bit-Double sind und in binärer Form ohne Trennzeichen angeordnet sind, werden diese Daten über den Befehl „fread“ in ein Array mit benutzerdefinierter Größe eingelesen. Jede Zeile des Arrays ist ein Datensatz und die erste Spalte ist der Zeitstempel. Im Gegensatz zur Dateitypumwandlung von einer XML-Datei in eine mat-Datei wird die Anzahl der Zeilen und Spalten des Arrays nicht automatisch erkannt. Bei Binärdateien

mit unterschiedlicher Anzahl von Signalen und Datensätzen muss die Anzahl der Zeilen und Spalten des Arrays manuell im Programm eingegeben werden. Dieses Array wird in ein Formular umgesetzt.

Der Zeitstempel in der ersten Spalte wird über den Befehl „datetime“ in ein Standardzeitformat „Datetime“ (als 'yyyy-MM-dd-HH:mm:ss.SSS' gezeigt) konvertiert. Es ist zu beachten, dass nur MATLAB 2018 und spätere Versionen diese Zeitstempelkonvertierung unterstützen. Anschließend wird der Zeitstempel der ersten Spalte des Formulars vom Daten-Array eingetragen.

Title-Datei: Die Titlenamen von der Title-Datei werden herausgenommen und in das Formular als Variablen-Name eingetragen.

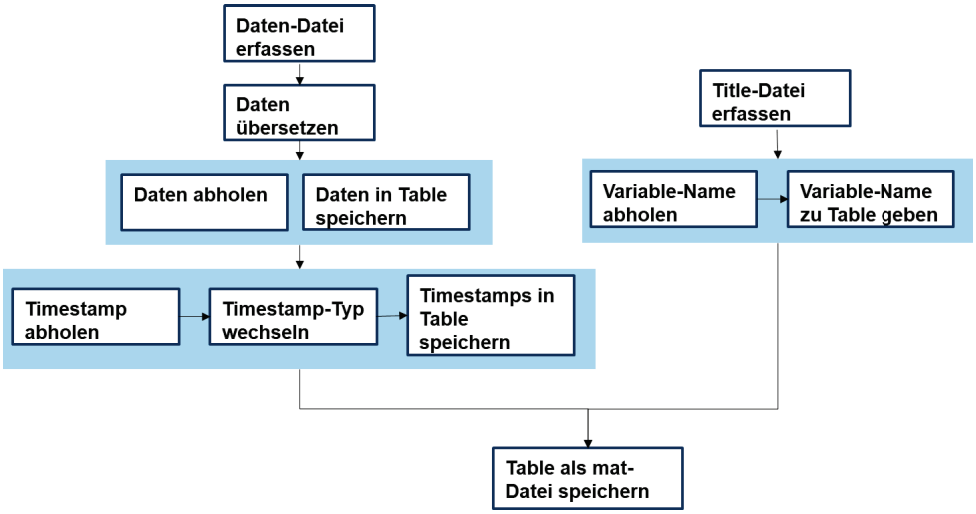


Abbildung 29 Ablauf der Dateitypumwandlung von binär zu mat

Abschließend wird das Formular in der mat-Datei gespeichert.

6 Testaufbau

Die entwickelten Data-Logging-Funktionen werden an einer Testumgebung untersucht. Der Testprozess und die Testergebnisse werden in diesem Kapitel beschrieben.

6.1 Versuchsaufbau

Der Versuchsaufbau ist in Abbildung 30 dargestellt. Die aufzuzeichnenden Variable wird aus dem von Simulink generierten Modul erhalten und die Werte der Variablen durch die Logging-Funktionalität in der XML-Datei aufgezeichnet. Dann wird die XML-Datei durch die in Kapitel 5.6 beschriebene Dateikonvertierung in eine mat-Datei mit einem Formularformat umgewandelt. Diese mat-Datei wird zur Datenanalyse in Matlab importiert.

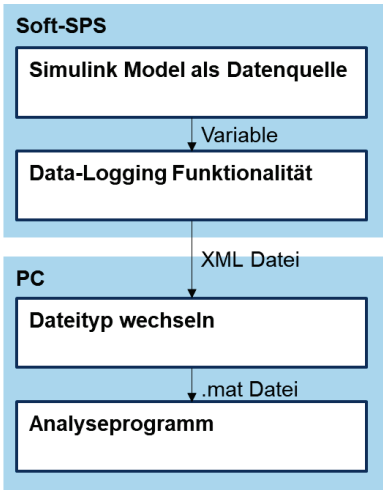


Abbildung 30 Versuchsaufbau

6.1.1 Simulink-Modul erstellen

Der Codegenerator von Simulink erzeugt entsprechenden C ++ - Code aus einem Simulink-Modell. TE1400 TwinCAT Target für MATLAB®/Simulink® arbeitet auf Basis des Codegenerators und generiert die entsprechenden Modulklassen des Simulink-Modells für das konfigurierte TwinCAT

Target (vgl. [42]). Die Eingangs- und Ausgangsschnittstellen des C++-Codes entsprechen denen des Simulink-Modells.

6.1.2 Schnittstelle zum Simulink-Modul

Gemäß der Beschreibung von TwinCAT 3 in Kapitel 3.2.2 wird ein SPS-Projekt, ein C++ - Projekt oder ein MATLAB/Simulink-Projekt als Modul in TwinCAT 3 generiert oder importiert. Wenn die Datenquelle ein Simulink-Modul ist und das Logging-Programm ein SPS-Modul oder ein C++ - Modul ist, wird eine Kommunikation zwischen den Modulen erforderlich.

Diese Kommunikation wird durch IO-Mapping erreicht. Dazu werden die Ein- und Ausgänge von Modulen miteinander verknüpft (siehe Abbildung 31). Dafür werden Daten im TMC-Editor angelegt, die entsprechende Ein-/Ausgänge beschreiben. Diese werden in dem TwinCAT Projekt dann verknüpft.

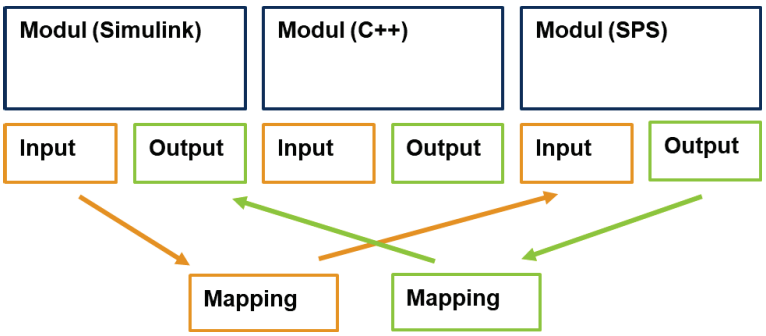


Abbildung 31 IO Mapping zwischen Modulen in TwinCAT (vgl. [43])

IO-Mapping ermöglicht die Datenkommunikation zwischen Modulen in einen Task und stellt die Datenkonsistenz sicher. Die Implementierungssprache (PLC, C++, Matlab) spielt dabei keine Rolle (vgl. [43]).

6.2 Testdurchführung

Das in Matlab generierte Heizsystem als Modulklasse-Datei wird zum Twincat-Konfigurationsordner hinzugefügt und in der TwinCAT 3 Entwicklungsumgebung instanziiert. Das Simulink-Modul

wird dem gleichen Task wie das SPS-Modul zugewiesen. Die Eingangs- und Ausgangsports werden automatisch generiert.

Nach dem Import der vom Testsystem generierten Module (vergleiche Abbildung 7) wird das resultierende Modul Object1 (room_heating_0508), wie in Abbildung 32 dargestellt, gebildet.

Die Ein- und Ausgangsports von diesem Modul entsprechen Eingangs- und Ausgangsgrößen des erweiterten Heizsystems (vergleiche Abbildung 7). Diese Größen sind die Raumtemperaturen T_Raum1 und T_Raum2, die Heizwassertemperatur T_Wasser, die Außentemperatur setAußenT, die Sollgrößen der Raumtemperatur der beiden Räume setRaumT1 und setRaumT2, die Sollgröße der Heizwassertemperatur setWasserT sowie die Stellgrößen a und ß der beiden Räume.

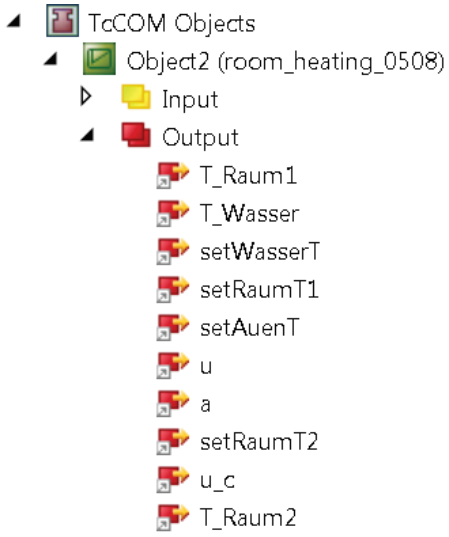


Abbildung 32 Ein- und Ausgangsports des Simulink-Moduls

6.2.1 IO-Mapping zwischen Simulink-Modul und SPS-Modul

Zuerst müssen die Eingangs- und Ausgangsports des SPS-Moduls generiert werden. Dazu müssen globale Variablen erstellt und Adressen für die Variablen festgelegt werden. Die Variable Raumtemperatur ist beispielweise die Ausgangsvariable im Simulink-Modul und die Eingangsvariable des Data-Logging-Programms. Alle Ausgangsvariablen in der globalen Variablentabelle erhalten eine allgemeine Adresse %Q* und alle Eingangsvariablen in der globalen Variablentabelle eine allgemeine Adresse %I* (siehe Abbildung 33).

Scope	Name	Address	Data type	Initialization
VAR_GLOBAL	**SetAussenTemp**	%Q*	LREAL	17
VAR_GLOBAL	**SetWasserTemp**	%Q*	LREAL	90
VAR_GLOBAL	**SetRaumTemp1**	%Q*	LREAL	20
VAR_GLOBAL	**SetRaumTemp2**	%Q*	LREAL	25
VAR_GLOBAL	**Traum1**	%I*	LREAL	
VAR_GLOBAL	**TWasser**	%I*	LREAL	
VAR_GLOBAL	**Traum2**	%I*	LREAL	
VAR_GLOBAL	**setAussenT**	%I*	LREAL	
VAR_GLOBAL	**setWasserT**	%I*	LREAL	
VAR_GLOBAL	**a**	%I*	LREAL	
VAR_GLOBAL	**b1**	%I*	LREAL	
VAR_GLOBAL	**b2**	%I*	LREAL	
VAR_GLOBAL	**setRaumT1**	%I*	LREAL	
VAR_GLOBAL	**setRaumT2**	%I*	LREAL	

Abbildung 33 Globale Variablen des SPS-Moduls

Ab diesem Zeitpunkt werden die Eingangs- und Ausgangsports des SPS-Moduls unter dem SPS-Projektordner angezeigt. Die Eingangs-/Ausgangsports des Simulink-Moduls und der Ausgangs-/Eingangsports vom SPS-Modul werden beim Einfügen einer Verknüpfung der Eingangs-/Ausgangsports vom Simulink-Modul zu den Ein- bzw. Ausgangsports vom SPS-Modul durch IO-Mapping verbunden.

6.2.2 IO-Mapping zwischen Simulink-Modul und C++-Modul

Die Parameter und Schnittstellen der Module werden in Klassenbeschreibungsdateien (*.tmc) beschrieben. Hier werden die Eingangs- und Ausgangsvariablen des Moduls hinzugefügt, geändert oder entfernt. Nachdem die neuen Eingabe- und Ausgabevariablen in der tmc-Datei erstellt und konfiguriert worden sind, werden die Eingangs- und Ausgangsports des SPS-Moduls unter dem SPS-Projektordner angezeigt. Die Eingangs-/Ausgangsports vom Simulink-Modul und die Ausgangs-/Eingangsports vom C++-Modul werden dadurch verbunden.

6.2.3 Verfahren des Tests

Das TwinCAT Projekt wird kompiliert und auf die TC3 Runtime heruntergeladen. Nach dem Start der TC3 Runtime wird das SPS-Programm bzw. C++-Modul ausgeführt.

Während des Betriebs werden die Zieldateien an dem durch sFilePath (FB_ XmlSrvWrite) im SPS-Programm oder szFileName (FileOpen) im C++-Programm angegebenen Speicherort erzeugt. Dieser Speicherort entspricht dem Kurzzeitspeicher.

Nach einer bestimmten Zeitdauer werden alle generierten Dateien zusammen in einen Zielordner übertragen. Dieser Zielordner entspricht dem Langzeitspeicher.

Nachdem der FilePath im Dateitypkonvertierungsprogramm in den Pfad der Zielordner eingetragen worden ist, wird die als Tabelle gespeicherte mat-Datei in diesem Ordner erzeugt. Diese Mat-Datei wird in das Analyseprogramm importiert und die Daten werden anschließend in Form von Liniendiagrammen angezeigt.

6.3 Testergebnisse des Versuchs

Nachdem der Logging-Funktionalität an das Testsystem angeschlossen worden ist, wird das Programm zur XAR heruntergeladen und gestartet. Die Daten werden durch die Data-Logging-Funktion in Dateien auf dem IPC aufgezeichnet, die erzeugten Dateien dann ins Analyseprogramm übertragen und dieses ausgeführt. Damit wird die Logging-Funktionalität in der Testumgebung getestet.

6.3.1 Variante 1

Das erzeugte PLC-Projekt wird auf TwinCAT 3 Runtime ausgeführt. Wenn nach dem Start im Hauptprogramm der kontinuierliche Modus ausgewählt worden ist, generiert das Programm automatisch jede Sekunde eine XML-Datei unter dem angegebenen Dateipfad (definiert durch sFilePath). Ein Beispiel ist in Abbildung 34 abgebildet. In den nachfolgenden Vorgängen werden die

Dateien, welche die Datensätze für einen definierten Zeitraum enthalten, durchsucht. Der Zeitstempel im Dateinamen entspricht dem Zeitstempel des letzten Datensatzes. Die Größe jeder Datei beträgt 40 KB.

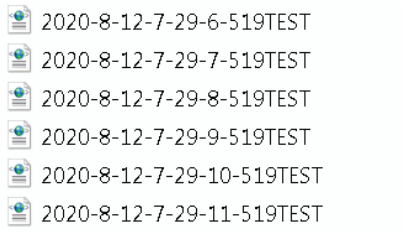


Abbildung 34 gespeicherte Dateien im Kontinuierlichen Mode in Variante 1

Die Datenstruktur einer XML-Datei (Data-Logging-Datei) ist in Abbildung 35 dargestellt. Die erzeugte XML-Datei besteht aus zwei Teilen: „Header“ und „Datapart“.

Im Testbeispiel sind die im Header gespeicherten Informationen das Maschinenmodell, der Steuerungshersteller und die Kundeninformationen. Die Variablen dieser Informationen sind vom Datentyp String. Im Datenbereich werden 100 Datensätze nacheinander gespeichert. Jeder Datensatz enthält Zeitstempel, Wassertemperatur, Raumtemperatur und weitere Signalwerte, die aus dem Simulink-Modell kommen. Die Zeitdifferenz zwischen jeweils zwei Datensätzen beträgt 10 ms.

```
<?xml version="1.0"?>
- <datafromroom>
  - <Header>
      <MaschineNummer>1234567890</MaschineNummer>
      <Manufacturer>Beckhoff</Manufacturer>
      <Maschinetype>cx2040.1.1.1</Maschinetype>
      <Customer>liyizhen23333</Customer>
    </Header>
  - <datapart index="1">
      <Timestamp>2020-08-12-07:29:06.529</Timestamp>
      <watertemp>20.274515382086</watertemp>
      <setoutsidetemp>17</setoutsidetemp>
      <setwatertemp>90</setwatertemp>
      <roomtemp1>17.0094347824005</roomtemp1>
      <setroomtemp1>20</setroomtemp1>
      <roomtemp2>17.0251594197347</roomtemp2>
      <setroomtemp2>25</setroomtemp2>
      <a>1</a>
      <b1>0.0269919023740183</b1>
      <b2>0.0719784063307156</b2>
    </datapart>
  - <datapart index="2">
      <Timestamp>2020-08-12-07:29:06.539</Timestamp>
      <watertemp>20.275451143927</watertemp>
      <setoutsidetemp>17</setoutsidetemp>
      <setwatertemp>90</setwatertemp>
      <roomtemp1>17.0094516468287</roomtemp1>
      <setroomtemp1>20</setroomtemp1>
      <roomtemp2>17.0252043915432</roomtemp2>
      <setroomtemp2>25</setroomtemp2>
      <a>1</a>
      <b1>0.0269984543941081</b1>
      <b2>0.0719958783842883</b2>
    </datapart>
  - <datapart index="3">
      <Timestamp>2020-08-12-07:29:06.549</Timestamp>
      <watertemp>20.2763868981072</watertemp>
      <setoutsidetemp>17</setoutsidetemp>
      <setwatertemp>90</setwatertemp>
      <roomtemp1>17.0094685176253</roomtemp1>
      <setroomtemp1>20</setroomtemp1>
      <roomtemp2>17.0252493803342</roomtemp2>
```


Abbildung 35 Datenstruktur XML-Datei

Wenn nach dem Start im Hauptprogramm der eventbasierte Modus ausgewählt ist, erstellt das Programm nach dem Event zwei XML-Dateien unter dem angegebenen Pfad (definiert durch sFilePath) (siehe Abbildung 36). Durch Ändern der Variablen „Event“ wird der Event-Trigger simuliert. Der Zeitstempel im Namen der ersten Datei entspricht dem Zeitpunkt des Ereignisses. In dieser Datei werden 20 Datensätze direkt vor dem Event gespeichert, das entspricht 200 ms vor dem Event. 20 Datensätze werden nach dem Event in der zweiten Datei aufgezeichnet. Der Name der Datei stimmt mit dem Zeitstempel des letzten Datensatzes überein. Die Größe jeder Datei beträgt 9 KB.

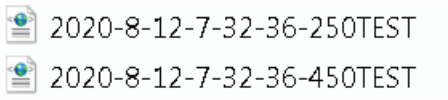


Abbildung 36 gespeicherten Dateien in Eventbasierten Mode in Variante 1

Die Datenstruktur der XML-Datei im eventbasierten Modus ist gleich wie in dem kontinuierlichen Modus.

Alle erstellten Dateien werden in denselben Ordner des MATLAB- Dateitypumwandlungsprogramms übertragen und das Programm wird ausgeführt. Dann werden die konvertierten mat-Dateien in diesem Ordner nacheinander erzeugt.

Nach der Konvertierung des Dateityps wird die XML-Datei zu einer von MATLAB lesbaren mat-Datei gewandelt, die nach dem Öffnen in MATLAB in Form einer Tabelle dargestellt wird. Im kontinuierlichen Modus beträgt die Größe der erstellten Mat-Datei 5 KB und im Eventbasierten Modus beträgt die Größe jeder erstellten Mat-Datei 2 KB. Das Ergebnis der Datei in Abbildung 35 nach der Konvertierung ist in Abbildung 37 dargestellt.

1	2	3	4	5	6	7	8	9	10	11
Timestamp	watertemp	setoutsidetemp	setwatertemp	roomtemp1	setroomtemp1	roomtemp2	setroomtemp2	a	b1	b2
2020-08-12-07:29:06.529	20.2745	17	90	17.0094	20	17.0252	25	1	0.0270	0.0720
2020-08-12-07:29:06.539	20.2755	17	90	17.0095	20	17.0252	25	1	0.0270	0.0720
2020-08-12-07:29:06.549	20.2764	17	90	17.0095	20	17.0252	25	1	0.0270	0.0720
2020-08-12-07:29:06.559	20.2773	17	90	17.0095	20	17.0253	25	1	0.0270	0.0720
2020-08-12-07:29:06.569	20.2783	17	90	17.0095	20	17.0253	25	1	0.0270	0.0720
2020-08-12-07:29:06.579	20.2792	17	90	17.0095	20	17.0254	25	1	0.0270	0.0721
2020-08-12-07:29:06.589	20.2801	17	90	17.0095	20	17.0254	25	1	0.0270	0.0721
2020-08-12-07:29:06.599	20.2811	17	90	17.0096	20	17.0255	25	1	0.0270	0.0721
2020-08-12-07:29:06.609	20.2820	17	90	17.0096	20	17.0255	25	1	0.0270	0.0721
2020-08-12-07:29:06.619	20.2829	17	90	17.0096	20	17.0256	25	1	0.0271	0.0721
2020-08-12-07:29:06.629	20.2839	17	90	17.0096	20	17.0256	25	1	0.0271	0.0722
2020-08-12-07:29:06.639	20.2848	17	90	17.0096	20	17.0257	25	1	0.0271	0.0722
2020-08-12-07:29:06.649	20.2857	17	90	17.0096	20	17.0257	25	1	0.0271	0.0722
2020-08-12-07:29:06.659	20.2867	17	90	17.0097	20	17.0257	25	1	0.0271	0.0722
2020-08-12-07:29:06.669	20.2876	17	90	17.0097	20	17.0258	25	1	0.0271	0.0722
2020-08-12-07:29:06.679	20.2886	17	90	17.0097	20	17.0258	25	1	0.0271	0.0722
2020-08-12-07:29:06.689	20.2895	17	90	17.0097	20	17.0259	25	1	0.0271	0.0723
2020-08-12-07:29:06.699	20.2904	17	90	17.0097	20	17.0259	25	1	0.0271	0.0723
2020-08-12-07:29:06.709	20.2914	17	90	17.0097	20	17.0260	25	1	0.0271	0.0723
2020-08-12-07:29:06.719	20.2923	17	90	17.0098	20	17.0260	25	1	0.0271	0.0723
2020-08-12-07:29:06.729	20.2932	17	90	17.0098	20	17.0261	25	1	0.0271	0.0723

Abbildung 37 MATLAB Formular aus XML-Datei

Diese Mat-Datei wird in das Analyseprogramm importiert und die Daten werden anschließend in Form eines Liniendiagramms angezeigt. Die für den Test verwendeten Daten sind die Daten, die beim Start des Heizsystems erzeugt werden.

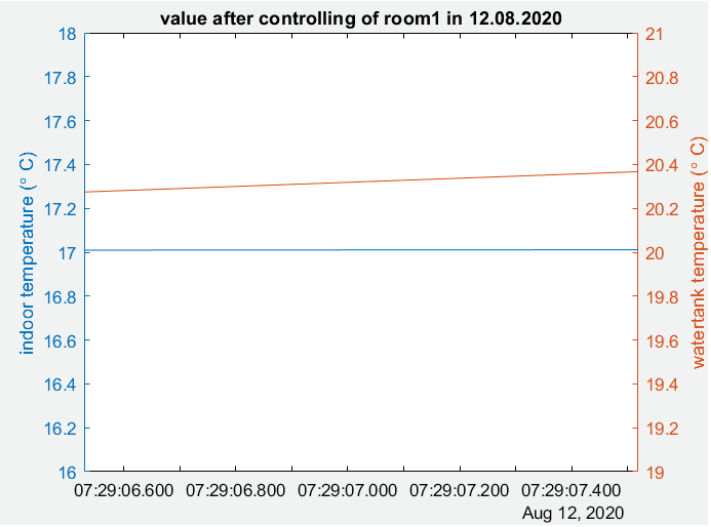


Abbildung 38 Diagramm erzeugt von MATLAB Formular in Variante 1

6.3.2 Variante 2

Das erstellte C++-Projekt wird auf der TwinCAT 3 Runtime ausgeführt. Im Programm entspricht die Variable „conti0_Event1“ der Modusauswahl, 0 ist der kontinuierliche Modus und 1 der event-basierte Modus. Der Speicherort der Datei wird durch den Parameter szFileName angegeben. In diesem Beispiel lautet dieser „% TC_BOOTPRJPATH% Timestamp + ATEST. txt“ (%TC_BOOT-PRJPATH% steht für den Boot-Ordner von TwinCAT).

Kontinuierlicher Modus: Die erzeugten Dateinamen ähneln denen von Variante 1 (siehe Abbildung 39). Der Zeitstempel im Dateinamen entspricht dem Zeitstempel des letzten Datensatzes und die Größe jeder Datei beträgt 9 KB. Wenn die erzeugte txt-Datei im Binärformat mit Notepad, dem Texteditor von Windows, geöffnet wird, wird der Dateiinhalt wie in Abbildung 40 angezeigt. Der Inhalt ist für Menschen nicht lesbar.

Abbildung 39 gespeicherten Dateien in Kontinuierlichen Mode in Variante 2

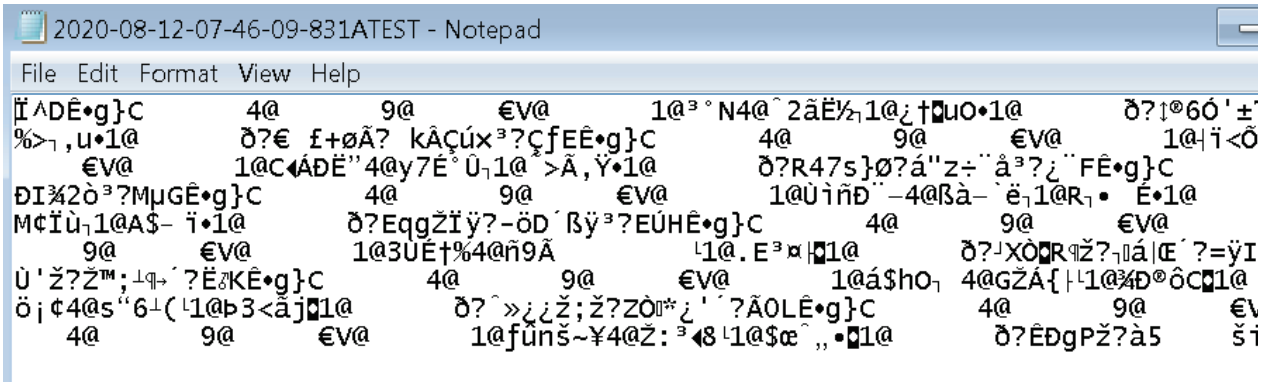


Abbildung 40 Binärdatei öffnet mit Notepad

Gleichzeitig wird eine Title- Datei erzeugt, wie in Abbildung 41 dargestellt. Da der Inhalt der Title-Datei als Zeichenkette gespeichert wird, ist er für Menschen lesbar. Die Größe der Title-Datei beträgt 2 KB.

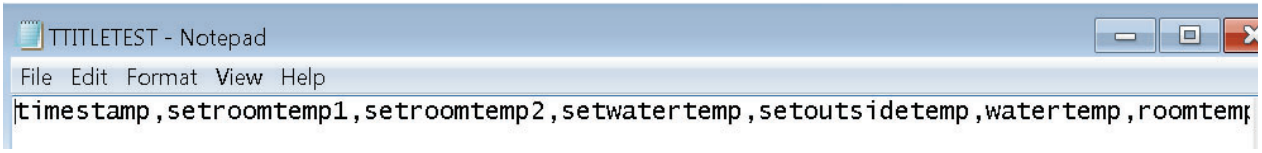


Abbildung 41 Title-Datei mit Notepad geöffnet

Eventbasierter Modus: Wie bei Variante 1 werden nach jedem Ereignis zwei Dateien erzeugt. In den Dateien werden die 20 Werte vor dem Ereignis und die 20 Werte nach dem Ereignis gespeichert. Der Name der Datei stimmt mit dem Zeitstempel des letzten Datensatzes überein. Die Größe jeder Datei beträgt 2 KB.

Alle generierten Dateien werden in denselben Ordner des MATLAB- Dateitypumwandlungsprogramms übertragen und das Programm ausgeführt. Dann werden die konvertierten Mat-Dateien in diesem Ordner nacheinander generiert.

Nach der Konvertierung des Dokumenttyps wird die binäre Datei in eine von MATLAB lesbaren mat-Datei gewandelt, die nach dem Öffnen in MATLAB in Form einer Tabelle dargestellt wird. Im kontinuierlichen Modus beträgt die Größe der generierten mat-Datei 5 KB und im eventbasierten Modus beträgt die Größe der generierten Mat-Datei 2 KB. Das Ergebnis der Datei in Abbildung 40 nach der Konvertierung ist gleich wie in Variante 1 (siehe Abbildung 42).

1 timestamp	2 setroomtemp1	3 setroomtemp2	4 setwatertemp	5 setoutsidetemp	6 watertemp	7 roomtemp1	8 roomtemp2	9 a	10 b	11 b2
2020-08-12 07:46:08.831	20	25	90	17	20.5598	17.0107	17.0286	1	0.0290	0.0773
2020-08-12 07:46:08.841	20	25	90	17	20.5607	17.0107	17.0286	1	0.0290	0.0773
2020-08-12 07:46:08.851	20	25	90	17	20.5617	17.0107	17.0287	1	0.0290	0.0774
2020-08-12 07:46:08.861	20	25	90	17	20.5626	17.0108	17.0287	1	0.0290	0.0774
2020-08-12 07:46:08.871	20	25	90	17	20.5635	17.0108	17.0288	1	0.0290	0.0774
2020-08-12 07:46:08.881	20	25	90	17	20.5645	17.0108	17.0288	1	0.0290	0.0774
2020-08-12 07:46:08.891	20	25	90	17	20.5654	17.0108	17.0289	1	0.0290	0.0774
2020-08-12 07:46:08.901	20	25	90	17	20.5663	17.0108	17.0289	1	0.0290	0.0774
2020-08-12 07:46:08.911	20	25	90	17	20.5673	17.0109	17.0290	1	0.0290	0.0775
2020-08-12 07:46:08.921	20	25	90	17	20.5682	17.0109	17.0290	1	0.0291	0.0775
2020-08-12 07:46:08.931	20	25	90	17	20.5691	17.0109	17.0291	1	0.0291	0.0775
2020-08-12 07:46:08.941	20	25	90	17	20.5700	17.0109	17.0291	1	0.0291	0.0775
2020-08-12 07:46:08.951	20	25	90	17	20.5710	17.0109	17.0292	1	0.0291	0.0775
2020-08-12 07:46:08.961	20	25	90	17	20.5719	17.0110	17.0292	1	0.0291	0.0775
2020-08-12 07:46:08.971	20	25	90	17	20.5728	17.0110	17.0293	1	0.0291	0.0776
2020-08-12 07:46:08.981	20	25	90	17	20.5738	17.0110	17.0293	1	0.0291	0.0776
2020-08-12 07:46:08.991	20	25	90	17	20.5747	17.0110	17.0294	1	0.0291	0.0776
2020-08-12 07:46:09.001	20	25	90	17	20.5756	17.0110	17.0295	1	0.0291	0.0776
2020-08-12 07:46:09.011	20	25	90	17	20.5766	17.0111	17.0295	1	0.0291	0.0776
2020-08-12 07:46:09.021	20	25	90	17	20.5775	17.0111	17.0296	1	0.0291	0.0777
2020-08-12 07:46:09.031	20	25	90	17	20.5784	17.0111	17.0296	1	0.0291	0.0777
2020-08-12 07:46:09.041	20	25	90	17	20.5794	17.0111	17.0297	1	0.0291	0.0777

Abbildung 42 MATLAB Formular aus Binärdatei

Diese Mat-Datei wird in das Analyseprogramm importiert und die Daten werden in Form eines Liniendiagramms angezeigt. Das Ergebnis ähnelt Variante 1.

Abbildung 43 Diagramm erzeugt von MATLAB Formular in Variante 2

6.4 Auswertung des Tests

Der Test ergibt, dass beide Varianten Data-Logging-Dateien mit den aufgezeichneten Signalwerten liefern. Die Datenübertragungsdauer (vom Buffer in die Logging-Datei) von zehn 4.000 kB Dateien (1.000 Werte pro Datensatz) beträgt weniger als 10 s, das heißt, für jede Datei weniger als eine Sekunde. Die Datenerzeugungsdauer beträgt eine Sekunde pro Logging-Datei. Das bedeutet, die Datenübertragung in eine Logging-Datei geht schneller als das Erzeugen dieser Daten. Somit wird die Anforderung an ein kontinuierliches Data-Logging erfüllt.

Bei gleicher Datenmenge ist die Größe der in Variante 1 erzeugten XML-Datei viermal so groß wie die in Variante 2 generierte Binärdatei. Die XML-Datei belegt mehr Speicherplatz im Kurzzeitspeicher, wodurch die maximale Datenmenge im Kurzzeitspeicher reduziert wird.

Wenn die Datei in das mat-Format konvertiert wird, wird die Dateigröße reduziert und die Ergebnisse der beiden Varianten sind gleich. Daraus folgt, wenn Daten im mat-Format, statt im XML-Format oder Binärformat im Langzeitspeicher gespeichert werden, wird die maximale Datenmenge im Langzeitspeicher gesteigert.

7 Weiterführende Schritte

Aus den Ergebnissen und dem Rückblick dieser Arbeit werden weitergehende Schritte identifiziert.

Die von Beckhoff angebotene Festplatte CX2900-0401 hat eine maximale Kapazität von 1 TB (vgl. [39]). Bei 1.000 Werten pro Datensatz speichert die Festplatte im kontinuierliches Modus Daten über 69 Stunden. Wenn diese Festplatte als Übertragungsmedium verwendet wird, müssen alle 69 Stunden Daten vom IPC übertragen werden. Diese Übertragungsfrequenz ist vergleichsweise häufig und es ist notwendig, ein Übertragungsmedium mit größerer Kapazität zu finden, das für den Beckhoff-IPC geeignet ist. Bei 1.000 Werten pro Datensatz wird ein 2,43 TB Speichermedium zum Speichern aller Datensätze für einen Zeitraum Woche benötigt.

Eine weitere Aufgabe besteht darin, eine Funktion für das Analyseprogramm zu entwickeln, mit der Logging-Dateien für einen bestimmten Zeitraum aus der Gesamtheit der Logging-Dateien herausgesucht und die Daten zusammengefügt werden. Dadurch müssen nicht alle vorhandenen Logging-Dateien geladen werden, sondern nur ein Teil davon.

Weil der Zeitstempel im Dateinamen den Zeitpunkt anzeigt, wann die Datei geschrieben wurde, stimmt der Zeitstempel im Dateinamen mit dem Zeitstempel des letzten Datensatzes in der Datei überein. Aber das ist nicht günstig, um Dateien zuordnen und auszuwählen. Eine weitere Verbesserung ist, wenn der Zeitstempel im Dateinamen der Zeitstempel des ersten Datensatzes in der Datei wäre.

Da das Programm ein von TwinCAT geschriebenes Modul ist, müsste es zur Änderung der Variablen oder zur Beendigung der Data-Logging-Funktion während der Laufzeit über TwinCAT betrieben werden. Die Möglichkeiten, das Programm auf andere SPS-Plattformen zu übertragen, wird im nächsten Absatz beschrieben.

Möglichkeit: auf andere Plattformen übertragen werden

Die beiden in Kapitel 5.4 beschriebenen Varianten eines Data-Logging-Systems sind für eine Beckhoff-SPS entwickelt worden. Um Anwendungen, die auf Steuerungen von Siemens oder Phoenix Contact laufen, analysieren zu können, bietet es sich an, die Data-Logging-Funktion auf diese Steuerungen zu übertragen. Bei Siemens gibt es den ODK 1500S FileServer für den Zugriff auf XML-, ASCII- Binary- und CSV-Dateien (vgl. [44]). Phoenix Contact stellt keine Funktionsblöcke wie die in Variante 1 verwendeten XML-Server von Beckhoff bereit, mit dem XML-Dateien

direkt erstellt werden könnten. Die Variante 1 ist daher aufwändig auf andere Plattformen zu übertragen.

Sowohl Siemens- als auch PhoenixContact-SPS unterstützen die C++-Programmierung. Sie verwenden Dateizugriffsschnittstellen und -methoden von verschiedenen Herstellern, um die in Variante 2 bereitgestellten Dateizugriffsschnittstellen von Beckhoff zu ersetzen und Variante 2 auf andere Plattformen zu übertragen.

Die Herausforderung bei der Übertragung auf andere Plattformen besteht darin, dass verschiedene SPS-Hersteller unterschiedliche Dateizugriffsschnittstellen bereitstellen. Wenn das auf Beckhoff geschriebene Programm übertragen werden soll, braucht die Verwendung des Buffers im Programm nicht verändert werden. Die Dateizugriffsschnittstelle muss jedoch angepasst werden. Wenn der SPS-Hersteller einen bestimmten Lese- und Schreibserver für Dateien bereitstellt (z. B. Beckhoffs XML-Server), sollte auch die Geschwindigkeit des vom Hersteller bereitgestellten Servers getestet werden.

8 Zusammenfassung

Im Rahmen dieser Bachelorarbeit wurde die Erstellung von einem Data-Logging-System mittels einer SPS von Beckhoff für einen Testanwendungsfall mit einem Simulink Model als Datenquelle untersucht. Dieses System kann Daten aus der SPS-Anwendungen aufzeichnen und für einen längeren Zeitraum speichern.

Ausgehend von einem Vergleich hinsichtlich Wirtschaftlichkeit, Entwicklungsschwierigkeiten, Skalierbarkeit und Dokumentation der vom Hersteller unterstützten Datenformate wurde eine mit XML in IEC 61131-3 Programmierte Data-Logging-Funktion und eine andere in C++ selbstprogrammierte Data-Logging-Funktion separat erstellt und diese untersucht. Wenn die XML Data-Logging-Funktion verwendet wird, werden XML-Dateien im IPC Dateisystem generiert. Wenn die C++ Data-Logging-Funktion benutzt wird, werden Binärdateien im IPC Dateisystem erzeugt. Die aufgezeichneten Dateien werden zum Langzeitspeicher übertragen und gespeichert. Im Langzeitspeicher werden die aufgezeichneten Dateien durch ein Aufbereitungsprogramm in MATLAB zu mat-Dateien umgewandelt, damit diese Daten weiterverwendet werden können.

Das Data-Logging-Programm ist an die Testanwendung angepasst. Wenn das Data-Logging-Programm in anderen Anwendungen benutzt werden soll, muss dieses an diese Anwendungen adaptiert werden.

9. Literaturverzeichnis

[1] Siemens, „SIMATIC S7 S7-1200 Automatisierungssystem,“ 11 2019. [Online]. Available: https://support.industry.siemens.com/cs/document/109772940/simatic-s7-s7-1200-automatisierungssystem?dti=0&lc=de-DE. [Zugriff am 02 08 2020].

[2] Duden, „Bedeutung-Erfassung,“ [Online]. Available: https://www.duden.de/rechtschreibung/erfassen#Bedeutung-3. [Zugriff am 02 08 2020].

[3] M. P. A.J. Martyr, „Data Handling, the Use of Modeling, and Post-Test Processing,“ in *Engine Testing*, version 4 Hrsg., Butterworth-Heinemann, 2012.

[4] Wikipedia Messschreiber , „Messschreiber Wiki,“ [Online]. Available: https://de.wikipedia.org/wiki/Messschreiber. [Zugriff am 03 08 2020].

[5] Dickson, „The Basics of Data Loggers,“ [Online]. Available: https://dicksondata.com/products/product-overview/data-loggers/. [Zugriff am 02 08 2020].

[6] Autem, „Autem SPS amalyser 6,“ [Online]. Available: https://www.autem.de/produkte/sps-analyzer-pro-6/. [Zugriff am 03 08 2020].

[7] A. Bott, „Wetterbeobachtungen,“ in *Synoptische Meteorologie*, Springer Berlin Heidelberg, 2016.

[8] TESTEM, [Online]. Available: https://testem.de/de/messtechnik/umweltmessysteme/wettersensoren/. [Zugriff am 04 08 2020].

[9] T. Bailey, „Flight Data Recorders: Build to Survive,“ AVIONICS NEWS, 01 2006. [Online]. Available: http://aea.net/AvionicsNews/ANArchives/FlightDataRecordersJan06.pdf. [Zugriff am 02 08 2020].

[10] G. Walker, „Redefining the incidents to learn from: Safety science insights acquired on the journey from black boxes to Flight Data Monitoring,“ Heriot-Watt University,, 02 06 2017. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S0925753517309086?via%3Dihub. [Zugriff am 02 08 2020].

[11] Beckhoff Automation, „Product overview cx20x0,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/cx2000_hw/36028799690476299.html&id=3487737817067410475. [Zugriff am 02 08 2020].

[12] Beckhoff Automation, „TwinCAT 3 Product Overview,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_overview/4275768971.html&id=5986876456246657686. [Zugriff am 02 08 2020].

[13] Beckhoff Automation, „TC3 XML Server,“ [Online]. Available: https://infosys.beckhoff.com/index.php?content=../content/1031/tf6421_tc3_xml_server/27021597951346955.html&id=7508111809458367441. [Zugriff am 02 08 2020].

[14] Siemens, „SIMATIC S7-1500 CPU 150xS,“ 02 2015. [Online]. Available: https://support.industry.siemens.com/cs/document/109249299/simatic-s7-1500-cpu-150xs?dti=0&lc=de-WW. [Zugriff am 10 08 2020].

[15] Y. Li, „Testsystem für Data-Logging-System, Bachelorpraktikumsbericht,“ IAV, 06.2020.

[16] InterConnectingAutomation, „PLC Logger,“ [Online]. Available: https://www.interconnectingautomation.com/plclogger. [Zugriff am 2020 08 02].

[17] InterConnectingAutomation, „FC-LOG USB Data Logger and Mass Storage Unit,“ 21 09 2016. [Online]. Available: https://www.interconnectingautomation.com/sites/default/files/FC-LOG-DataSheet.pdf. [Zugriff am 02 08 2020].

[18] Process Informatik Entwicklungsgesellschaft mbH, „Datensicherung S7-SPS PN-Port auf USB-Stick,“ [Online]. Available: https://www.process-informatik.de/7000_datensicherung-s7-sps-pn-port-auf-usb-stick.html/?lang=de. [Zugriff am 02 08 2020].

[19] Siemens, „Long-term archivingand display of process data with WinCC V15.1 RT Advanced and SIMATIC Panels,“ 03 2019. [Online]. Available: https://cache.industry.siemens.com/dl/files/071/109477071/att_980801/v3/109477071_LongTermDataArchives_en_v20.pdf. [Zugriff am 02 08 2020].

[20] Beckhoff Automation, „FUNCTION_BLOCK FB_CTRL_LOG_DATA,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclibcontrollertoolbox/html/tcplclibcontoller_ctrl_log_data.htm&id=2538166337691639609. [Zugriff am 02 08 2020].

[21] Beckhoff Automation, „Example: Writing/reading of CSV file,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclibutilities/html/tcplclibutilities_csv_sample.htm&id=5453132095136970524. [Zugriff am 02 08 2020].

[22] Beckhoff Automation, „TC3 Database Server,“ [Online]. Available: https://infosys.beckhoff.com/index.php?content=../content/1031/tf6420_tc3_database_server/index.html&id=8330845658550886667. [Zugriff am 02 08 2020].

[23] Beckhoff Automation, „TF 3500 TC3 data logger,“ [Online]. Available: https://infosys.beckhoff.com/index.php?content=../content/1031/tf3500_tc3_analytics_logger/index.html&id=4500759640326969526. [Zugriff am 06 08 2020].

[24] Siemens, „Wie können Variablen und Meldungen mit WinCC Advanced V15 in eine SQL-Datenbank archiviert werden?,“ 04 2018. [Online]. Available: https://cache.industry.siemens.com/dl/files/098/61886098/att_941310/v3/61886098_WinCC_Adv_Archive_SQL_Database_de.pdf. [Zugriff am 02 08 2020].

[25] Phoenix Contact, „User manual AXC F 1152,“ 04 12 2019. [Online]. Available: https://www.phoenixcontact.com/online/portal/de/?uri=pxc-oc-itemdetail:pid=2404267&library=dede&pcck=P-21-14-01&tab=5&selectedCategory=ALL. [Zugriff am 02 08 2020].

[26] OPC Foundation, „OPC UA,“ [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/. [Zugriff am 02 08 2020].

[27] Beckhoff Automation, „TF 6100 TC3 OPC UA,“ [Online]. Available: https://infosys.beckhoff.de/index.php?content=../content/1031/tf6100_tc3_opcua/index.html&id=3624913834516646834. [Zugriff am 02 08 2020].

[28] OAS, „Log Buffered Data from a PLC or Controller,“ [Online]. Available: https://openautomationsoftware.com/data-destinations/log-buffered-data-from-a-plc-or-controller/. [Zugriff am 02 08 2020].

[29] Siemens, „WinAC RTX ODK V3.0,“ [Online]. Available: https://support.industry.siemens.com/cs/document/7088366/neu%3A-winac-rtx-odk-v3-0?dti=0&lc=de-WW. [Zugriff am 06 08 2020].

[30] Phoenix Contact, „Software für Steuerung - AXC F 2152,“ [Online]. Available: https://www.phoenixcontact.com/online/portal/de?uri=pxc-oc-itemdetail:pid=2404267&library=dede&tab=5&requestType=qr&productId=2404267#softw.

[31] Siemens, „Manual S7-1500 CPU 1518-4 PN/DP,“ 12 2014. [Online]. Available: https://cache.industry.siemens.com/dl/files/632/81164632/att_40252/v1/s71500_cpu_1518_4_pndp_manual_en-US_en-US.pdf. [Zugriff am 06 08 2020].

[32] Siemens, „How do you calculate the service life of a memory card of the S7-1500/S7-1200 and ET 200?,“ [Online]. Available: https://support.industry.siemens.com/cs/document/109482591/how-do-you-calculate-the-service-life-of-a-memory-card-of-the-s7-1500-s7-1200-and-et-200-?dti=0&lc=en-WW. [Zugriff am 06 08 2020].

[33] Beckhoff Automation, „c/c++ Interface ITcFileAccess,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_c/72057595006142091.html&id=6936551464357775429. [Zugriff am 02 08 2020].

[34] Beckhoff Automation, „Method ITcFileAccess: FileOpen,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_c/36028797987186699.html&id=1834189196969220111. [Zugriff am 02 08 2020].

[35] Beckhoff Automation, „TwinCAT module state machine,“ [Online]. [Zugriff am 02 08 2020].

[36] Beckhoff Automation, „c/c++ Interface ITcCyclic,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_c/72057594148683019.html&id=4642795910018037679. [Zugriff am 02 08 2020].

[37] Beckhoff Automation, „c/c++ Interface ITcCyclicCaller,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_c/81064793403426955.html&id=414813139677506670. [Zugriff am 02 08 2020].

[38] Beckhoff Automation, „C++ Beispiel: FileIO-Write,“ [Online]. Available: https://infosys.beckhoff.com/index.php?content=../content/1031/tc3_c/5404319564105690 7.html&id=2857292988539513773. [Zugriff am 02 08 2020].

[39] Beckhoff Automation, „Systemzubehör CX2900,“ [Online]. Available: https://www.beckhoff.de/default.asp?embedded_pc/cx2900_xxxx.htm. [Zugriff am 02 08 2020].

[40] Beckhoff Automation, „CX2550-0020 | Erweiterungsmodul für CX20xx,“ [Online]. Available: https://www.beckhoff.de/default.asp?embedded_pc/cx2550_0020.htm. [Zugriff am 02 08 2020].

[41] W. Falkena, „xml2struct für MATLAB,“ [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/28518-xml2struct. [Zugriff am 02 08 2020].

[42] Beckhoff Automation, „TC3 Target for MATLAB/Simulink,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/te1400_tc3_target_matl ab/index.html&id=7328785815492855617. [Zugriff am 02 08 2020].

[43] Beckhoff Automation, „Module-to-module communication,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_c/27021600255260 939.html&id=4913669680486956516. [Zugriff am 02 08 2020].

[44] Siemens, „ODK 1500S FileServer,“ 10 2015. [Online]. Available: https://support.industry.siemens.com/cs/document/109479497/simatic-odk-1500s-fileserver-?dti=0&lc=en-WW. [Zugriff am 11 08 2020].

[45] Beckhoff Automation, „FB_XmlSrvWrite,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tf6421_tc3_xml_server/ 27021597951346955.html&id=7215426458055596947. [Zugriff am 20 08 2020].

[46] Beckhoff Automation, „c/c++ Interface ITcTask,“ [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_c/81064793403441 803.html&id=2957684578221096252. [Zugriff am 02 08 2020].

[47] Beckhoff Automation, „TwinCAT Zeitquellen,“ [Online]. Available: https://infosys.beckhoff.com/index.php?content=../content/1031/ethercatsystem/2469114379.html&id=. [Zugriff am 02 08 2020].

[48] H. W. (.-s. K. Jang, „PLC data log module with external storage for storing PLC log data and method for storing PLC log data in the same“. USA Patent 9037780, 20 03 2012.

[49] Siemens, „On a PC with WinCC flexible, how can you use a script to save and read out data in a text file?,“ 14 07 2015. [Online]. Available: https://support.industry.siemens.com/cs/document/26106418/on-a-pc-with-wincc-flexible-how-can-you-use-a-script-to-save-and-read-out-data-in-a-text-file-?dti=0&lc=en-WW. [Zugriff am 02 08 2020].

[50] AutomationDirect, „PLC Data Logging / Web-based Monitoring Software,“ [Online]. Available: https://www.automationdirect.com/adc/overview/catalog/software_products/plc_data_logging_-z-_web-based_monitoring_software. [Zugriff am 02 08 2020].

[51] Siemens, „SIMATIC Target 1500S: Aufruf von Simulink®-Modellen,“ [Online]. Available: https://support.industry.siemens.com/cs/document/109482830/simatic-target-1500s%3A-aufruf-von-simulink%C2%AE-modellen?dti=0&lc=de-WW. [Zugriff am 04 08 2020].

[52] Siemens, „Prozessdaten erfassen und überwachen mit der SIMATIC S7-1200 / S7-1500 (Data Logging),“ 07 2019. [Online]. Available: https://support.industry.siemens.com/cs/document/64396156/prozessdaten-erfassen-und-%C3%BCberwachen-mit-der-simatic-s7-1200-s7-1500-(data-logging)?dti=0&lc=de-DE. [Zugriff am 05 08 2020].

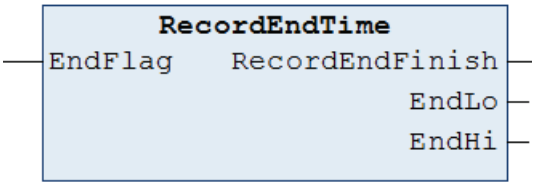
[53] Unified Automation, „UaExpert – ein vollausgestatteter OPC UA Client,“ [Online]. Available: https://www.unified-automation.com/de/produkte/entwicklerwerkzeuge/uaexpert.html. [Zugriff am 05 08 2020].

[54] Siemens, „SIMATIC S7-1500 S7-PLCSIM Advanced,“ [Online]. Available: https://support.industry.siemens.com/cs/document/109773484/simatic-s7-1500-s7-plcsim-advanced-?dti=0&lc=en-WW. [Zugriff am 09 08 2020].

Anhang

A Funktionsbausteine für der XML-Server-Performancetest

A1 RecordEndTime



Mit dem Funktionsbaustein „RecordEndTime“ wird der Endzeitpunkt des Schreibvorgangs (zu Buffer) ermittelt.

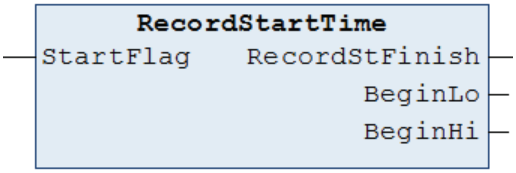
```
FUNCTION_BLOCK RecordEndTime
VAR_INPUT
    EndFlag: BOOL := FALSE;
END_VAR
VAR_OUTPUT
    RecordEndFinish : BOOL :=FALSE;
    EndLo: UDINT;
    EndHi: UDINT;
END_VAR
```

EndFlag: Mit einer positiven Flanke an EndFlag wird der Baustein aktiviert.

EndLo: die niederwertigeren 4 Byte des Zeitstempels

EndHi: die hochwertigeren 4 Byte des Zeitstempels

A2 RecordStartTime



Mit dem Funktionsbaustein „RecordStartTime“ wird der Anfangszeitpunkt des Schreibvorgangs (zu Buffer) ermittelt.

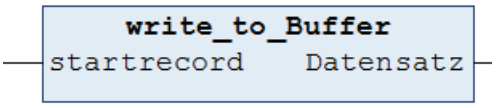
```
FUNCTION_BLOCK RecordStartTime
VAR_INPUT
    StartFlag: BOOL := FALSE;
END_VAR
VAR_OUTPUT
    RecordStFinish : BOOL :=FALSE;
    BeginLo: UDINT;
    BeginHi: UDINT;
END_VAR
```

StartFlag: Mit einer positiven Flanke an StartFlag wird der Baustein aktiviert

StartLo: die niederwertigeren 4 Byte des Zeitstempels

StartHi: die hochwertigeren 4 Byte des Zeitstempels

A3 write_to_Buffer



In dem Funktionsblock „write_to_Buffer“ werden Werte in einen Datensatz in den Buffer geschrieben.

```
FUNCTION_BLOCK write_to_Buffer
VAR_INPUT
    startrecord : BOOL;
END_VAR
VAR_OUTPUT
    Datensatz : st_Datensatz;
END_VAR
```

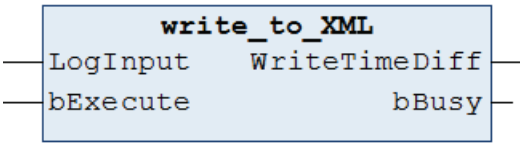
startrecord: Wenn startrecord TRUE ist, wird der Baustein aktiviert

Datensatz: Output Datensatz

Struktur st_Datensatz:

```
TYPE st_Datensatz :
STRUCT
    Timestamp : STRING ;
    roomnum : REAL;
    temp : REAL;
    signal3 : REAL;
    signal4 : REAL;
    signal5 : REAL;
    signal6 : REAL;
    signal7 : REAL;
    signal8 : REAL;
END_STRUCT
END_TYPE
```

A4 write_to_XML



In dem Funktionsblock „Write_to_XML“ wird der Funktionsblock „FB_XmlSrvWrite“ aufgerufen, um Daten vom Buffer in die XML-Datei zu schreiben und den Start- und Endzeitpunkt des XML-Schreibvorgangs zu bestimmen.

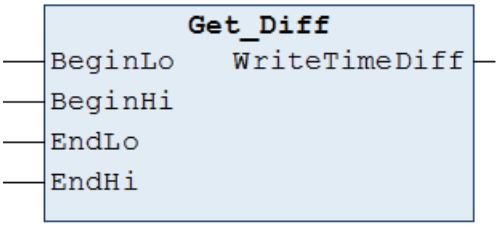
```
FUNCTION_BLOCK write_to_XML
VAR_INPUT
    LogInput : ARRAY[1..100] OF st_Datensatz;
    bExecute         : BOOL := FALSE;
END_VAR
VAR_OUTPUT
    WriteTimeDiff: ST_TimeDiff;
    bBusy : BOOL;
END_VAR
```

LogInput: Buffer zu loggen in die XML-Datei

bExecute: Mit einer positiven Flanke an bExecute wird der Baustein aktiviert.

WriteTimeDiff: verwendete Zeit für das Schreiben der XML-Datei

A5 Get_Diff



Mit dem Funktionsbaustein wird die Zeitdifferenz von BeginLo und EndLo sowie BeginHi und EndHi berechnet.

```
FUNCTION_BLOCK Get_Diff
VAR_INPUT
    BeginLo: UDINT;
    BeginHi: UDINT;
    EndLo: UDINT;
    EndHi: UDINT;
END_VAR
VAR_OUTPUT
    WriteTimeDiff: ST_TimeDiff;
END_VAR
```

EndLo: die niederwertigeren 4 Byte des Endzeitstempels

EndHi: die hochwertigeren 4 Byte des Endzeitstempels

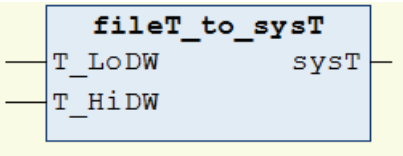
StartLo: die niederwertigeren 4 Byte des Startzeitstempels

StartHi: die hochwertigeren 4 Byte des Startzeitstempels

Struktur ST_TimeDiff:

```
TYPE ST_TimeDiff :
STRUCT
    sSecond : WORD;
    mMillisecond : WORD;
END_STRUCT
END_TYPE
```

A6 fileT_to_sysT



Der Funktionsbaustein „FileT_to_sysT“ konvertiert die Zeitdifferenz von dem FILETIME-Format (64-bit Integer) in das SYSTEMTIME-Format (Datentyp: TIMESTRUCT).

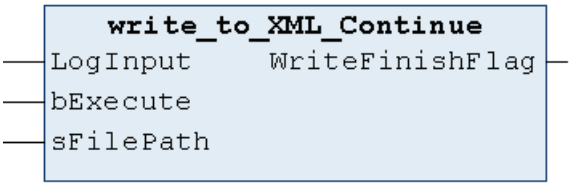
```
FUNCTION_BLOCK fileT_to_sysT
VAR_INPUT
    T_LoDW : DWORD;
    T_HiDW : DWORD;
END_VAR
VAR_OUTPUT
    sysT : TIMESTRUCT;
END_VAR
```

T_LoDW: die niederwertigeren 4 Byte des wechselnden Zeitstempels

T_HiDW: die hochwertigeren 4 Byte des wechselnden Zeitstempels

B Funktionsblock für Variante 1

B1 write_to_XML_Continue



In diesem Funktionsblock wird der Inhalt vom Kopfteil und der Inhalt des von LogInput verwiesenen Buffers in das XML-Dokument geschrieben.

```
FUNCTION_BLOCK write_to_XML_Continue
VAR_INPUT
    LogInput     : ARRAY[1..100] OF Datensatz;
    bExecute     : BOOL := FALSE;
    sFilePath    : T_MaxString := 'C:\temp_Yizhen\Test.xml';  //disable for CE
  //sFilePath    : T_MaxString := '\Hard Disk\Test.xml';      //enable for CE
END_VAR
VAR_OUTPUT
    WriteFinishFlag : BOOL;
END_VAR
```

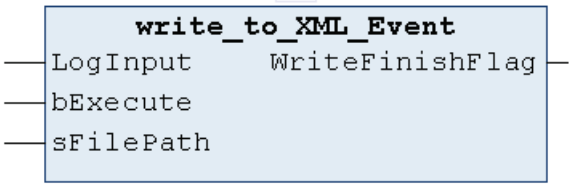
LogInput: Beim Aufruf von FB_write_to_XML_Continue verweist LogInput auf den Buffer, der gelesen werden soll

bExcute: Bei einer steigenden Flanke von bExecute wird der XML-Dateischreibprozess gestartet.

sFilePath: sFilePath enthält den vollständigen Dateipfad der zu erstellenden XML-Datei inklusive dem Dateinamen.

WriteFinishFlag: Wenn die Bearbeitung von write_to_XML_Continue beendet wird der Ausgang WriteFinishFlag auf TRUE gesetzt.

B2 write_to_XML_Event



In diesem Funktionsblock wird der Inhalt vom Kopfteil und der Inhalt des von LogInput verwiesenen Buffers in das XML-Dokument geschrieben.

```
FUNCTION_BLOCK write_to_XML_Event
VAR_INPUT
    LogInput    : ARRAY[1..20] OF Datensatz;
    bExecute    : BOOL := FALSE;
    sFilePath   : T_MaxString := 'C:\temp_Yizhen\Test.xml'; //disable for CE
  //sFilePath   : T_MaxString := '\Hard Disk\Test.xml';     //enable for CE
END_VAR
VAR_OUTPUT
    WriteFinishFlag : BOOL;
END_VAR
```

LogInput: Beim Aufruf von FB_write_to_XML_Continue verweist LogInput auf den Buffer, der gelesen werden soll

bExcute: Bei einer steigenden Flanke von bExecute wird der XML-Dateischreibprozess gestartet.

sFilePath: sFilePath enthält den vollständigen Dateipfad der zu erstellenden XML-Datei inklusive dem Dateinamen.

WriteFinishFlag: Wenn die Bearbeitung von write_to_XML_Continue beendet wird der Ausgang WriteFinishFlag auf TRUE gesetzt.

C Funktionsbaustein: FB_XmlSrvWrite [45]

- VAR_INPUT

Portname	Funktion
sNetId	Netzwerkadresse des TwinCAT 3 XML-Servers. Für den lokalen Rechner bleibt sNetId leer.
ePath	TwinCAT-Systempfad auf dem Zielgerät (IPC)
nMode	Modus des Schreibverhaltens. Für XmlSrvWrite gibt es die Modi XMLSRV_SKIPMISSING und XMLSRV_ADDMISSING. Im Modus XMLSRV_SKIPMISSING werden nur die bereits vorher in der XML-Datei existierenden Elemente eines SPS-Symbols geschrieben. Im Modus XMLSRV_ADDMISSING werden in der XML-Datei fehlende Element der XML-Datei hinzugefügt.
pSymAddr	Adresse der aufzuzeichnenden SPS-Variablen
cbSymSize	Größe der aufzuzeichnenden SPS-Variablen
sFilePath	Der Pfad der zu öffnenden XML-Datei. XML-Dateien werden nur auf das lokale Dateisystem des Rechners gespeichert. (sFilePath kann nur den Pfad zu lokalen Dateien enthalten)
sXPath	Adresse des Tags im XML-Dokument
bExecute	Mit einer positiven Flanke an bExecute wird der Baustein aktiviert.
tTimeout	Maximale Zeit zum Ausführen des Funktionsblocks

- VAR_OUTPUT

Portname	Funktion
bBusy	bBusy wird bei der Aktivierung des Funktionsbausteins gesetzt, wird nach erfolgreiche Bearbeitung oder im Fehlerfall rückgesetzt.
bError	Wenn während der Übertragung ein Fehler auftritt, wird bError gesetzt.
nErrId	TC3 XML Server Fehlernummer

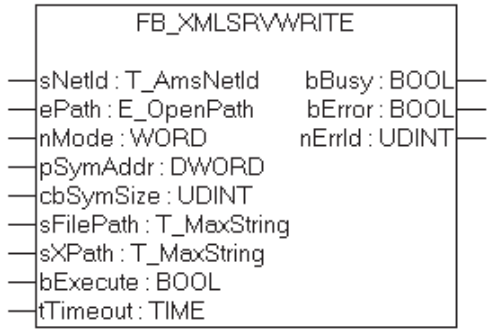


Abbildung C Funktionsbaustein FB_XMLSRVWRITE

D verwendete Methode in Variante 2

Tabelle D1 Von Schnittstelle TcFileAccess bereitgestellten Methoden (vgl. [33])

Name	Verwendung	Parameter
FileO-pen	Öffnet eine Datei	-szFileName: PCCH, Input, der Name der zu öffnenden Datei -AccessMode: TcFileAccessMode, Input, Art des Zugriffs auf die Datei -phFile: TcFileHandle, Output, zurückgegebener Datei-Handle
FileWrite	Schreibt in eine Datei	-hFile: TcFileHandle, Input, verweist auf die zuvor geöffnete Datei -pData: PVOID, Input, Speicherort der zu schreibenden Daten -cbData: PVOID, Input, Größe der zu schreibenden Daten (Größe des Speichers hinter pData) -pcbRead: PUINT, Output, Größe der geschriebenen Daten
Fil-eClose	Schließt eine Datei	-phFile: TcFileHandle, Input, zurückgegebener Datei-Handle

Tabelle D2 Von Schnittstelle ITcTask bereitgestellten Methoden (vgl. [46])

Name	Verwendung	Parameter
GetCurrentSys-Time	Abfrage der Zeit in Intervallen von 100 Nanosekunden seit dem 1. Januar 1601	-pSysTime: PLONGLONG, Output, die aktuelle Systemzeit zu Beginn des Taskzyklus