# u-create studio

# Systemhandbuch

**Originalbetriebsanleitung**

Weidmüller

Dokument Nr.:
Dateiname: u_create_shde.pdf
Seitenanzahl: 146



**Weidmüller Interface GmbH & Co. KG** Klingenbergstraße 26, 32758 Detmold, Germany, Telefon: +49 5231 14-0, Fax: +49 5231 14-292083

## Änderungsverzeichnis

| Version | Datum | Änderung in Kapitel | Beschreibung | Geändert von |
|---|---|---|---|---|
| 00 | 06-2019 | - | Neu erstellt | hli, hasl |
| 01 | 08-2019 | Zugriff auf Flash-Speichermedium | Aktualisiert | hasl |
| 1.02 | 04-2020 | Diverse Kapitel | Aktualisiert auf neue Version | hasl |
| 1.03 | 06-2020 | Lizenzierung, Modbus Server | Gerätetausch aktualisiert, neues Kapitel | hasl |
| 1.04 | 08-2021 | Software Units, Div. Kapitel | Kapitel entfernt, aktualisiert auf neue Version | hasl |
| 1.05 | 11-2021 | Diverse Kapitel | Hotifx, Firmwareupdate, DevAdmin aktualisiert | hasl |

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Zweck des Dokuments

Dieses Dokument beschreibt den Aufbau von u-control.

Weiters werden hier die Montage und der Einbau, die Verkabelung und die Bedienung und Anzeigen der Baugruppen beschrieben.

Die Installation und Konfiguration wird soweit beschrieben, um ein betriebsfertiges System zu erhalten. "Betriebsfertig" bedeutet, das System bzw. die jeweiligen CPU-Baugruppen sind bereit zum Laden der Kundenapplikation.

***Information***

*Dieses Handbuch richtet sich nicht an Endkunden! Die für Endkunden notwendigen Sicherheitshinweise müssen vom Maschinenbauer oder Systemanbieter in die Betriebsanleitung für Endkunden in der jeweiligen Landessprache übernommen werden!*

## 1.2 Voraussetzungen

Das Systemhandbuch richtet sich an alle, die ein System bestehend aus u-create studio in Verbindung mit einem UC20-SL2000-EC oder UC20-SL2000-EC-CAN einsetzen oder den Einsatz eines solchen Systems planen.

Nur Elektrofachkräfte nach VDE 1000-10 dürfen mit Hilfe dieses Systemhandbuches ein System installieren und warten.

| Zielgruppe | Voraussetzung an Wissen und Können |
|---|---|
| Sicherheitsingenieur für "Funktionale Sicherheit" | Technische Grundausbildung (Fachschule, Ingenieur-Ausbildung oder entsprechende Berufserfahrung).<br>Kenntnisse über:<br>● Die Grundlagen der funktionalen Sicherheit<br>● Alle einschlägigen Normen und Sicherheitsvorschriften der Maschine/ Anlage, insbesondere auch Kenntnisse über Validierung nach EN ISO 13849-2.<br>● Spezielle Gefahrenpotentiale der Maschine/Anlage und des Produktionsprozesses<br>● Konkrete Schutzmaßnahmen zur Vermeidung der maschinenspezifischen Gefährdungen (aus Gefahren- und Risikoanalyse)<br>● Vertiefte Kenntnisse der nationalen Unfallverhütungsvorschriften |

| Zielgruppe | Voraussetzung an Wissen und Können |
|---|---|
| Projektierer | Technische Grundausbildung (Fachhochschule, Ingenieur-Ausbildung oder entsprechende Berufserfahrung),<br>Kenntnisse über:<br>● die Arbeitsweise einer SPS,<br>● aktuell gültige Sicherheitsvorschriften,<br>● die Applikation. |
| Elektromonteur | Elektrotechnische Fachausbildung (nach branchenüblichen Ausbildungsrichtlinien).<br>Kenntnisse über:<br>● aktuell gültige Sicherheitsvorschriften,<br>● Verdrahtungsrichtlinien,<br>● Schaltpläne<br>● Systemanalyse und Fehlerbehebung,<br>● fachgerechtes Herstellen elektrischer Anschlüsse nach nationalen und internationalen Vorschriften. |
| Programmierer | Technische Ausbildung (Fachhochschule, Ingenieur-Ausbildung oder entsprechende Berufserfahrung).<br>Kenntnisse über:<br>● die Arbeitsweise einer SPS,<br>● aktuell gültige Sicherheitsvorschriften,<br>● Programmierung einer SPS (IEC61131). |
| Inbetriebnehmer | Technische Grundausbildung (Fachhochschule, Ingenieur-Ausbildung oder entsprechende Berufserfahrung),<br>Kenntnisse über:<br>● aktuell gültige Sicherheitsvorschriften,<br>● die Arbeitsweise der Maschine oder Anlage,<br>● grundlegende Funktionen der Applikation,<br>● Systemanalyse und Fehlerbehebung,<br>● die Einstellmöglichkeiten an den Bedienvorrichtungen. |
| Servicetechniker | Technische Grundausbildung (Fachhochschule, Ingenieur-Ausbildung oder entsprechende Berufserfahrung),<br>Kenntnisse über:<br>● die Arbeitsweise einer SPS,<br>● aktuell gültige Sicherheitsvorschriften,<br>● die Arbeitsweise der Maschine oder Anlage,<br>● Diagnosemöglichkeiten,<br>● systematische Fehleranalyse und -behebung |

## 1.3 Bestimmungsgemäßer Gebrauch

u-control darf nur für die in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit empfohlenen bzw. zugelassenen Fremdgeräten verwendet werden.

Alle Baugruppen von u-control wurden unter Beachtung der einschlägigen Sicherheitsnormen entwickelt, gefertigt, geprüft und dokumentiert. Bei Beachtung der für den bestimmungsgemäßen Gebrauch beschriebenen Anweisungen und sicherheitstechnischen Hinweise gehen deshalb vom Produkt im Normalfall keine Gefahren in Bezug auf Sachschäden oder für die Gesundheit von Personen aus.

Das System ist zum Einbau in einen Schaltschrank bestimmt und darf nur folgendermaßen benutzt werden:

- Nur im Industriebereich
- Bestimmungsgemäß
- In technisch einwandfreiem Zustand
- Im Originalzustand ohne eigenmächtige Veränderungen.

Zugelassen sind die in der produktbegleitenden Dokumentation beschriebenen Umbauten oder Veränderungen.

## 1.4 Hinweise zu diesem Dokument

***Information***

*Beachten Sie auch die den Baugruppen beiliegende Dokumentation.*

### 1.4.1 Inhalt des Dokuments

- Überblick über das System
- Erklärung aller Schnittstellen und Anzeigen der Geräte
- Allgemeine Hinweise zur Montage
- Betriebsverhalten
- Beschreibung der Software Installation
- Beschreibung von Software-Schnittstellen und Services
- Diagnose und Instandhaltung
- Tutorials

### 1.4.2 Im Dokument nicht enthalten

- Programmieranleitung
- Applikationsdiagnose
- Firmwarebeschreibung

## 1.5 Weiterführende Dokumentation

Beim Einsatz von u-remote I/O-Modulen beachten Sie unbedingt auch das Handbuch Remote-I/O-System u-remote. Die Dokumentation der Tools und Bibliotheken ist jeweils als Onlinehilfe integriert.

## 1.6 Einsatzländer und Zulassungen

Normen und Prüfwerte, die das Produkt einhält und erfüllt, finden Sie in den Kapiteln "Technische Daten" und "EG-Richtlinien und Normen". Die EG-Richtlinien zur Produktkonformität entnehmen Sie bitte der Konformitätserklärung.

# 2 Sicherheitshinweise

## 2.1 Darstellung

Im Handbuch finden Sie an verschiedenen Stellen Hinweise und Warnungen vor möglichen Gefahren. Die verwendeten Symbole haben folgende Bedeutung:

**GEFAHR!**

bedeutet, dass Tod oder schwere Körperverletzung eintreten werden, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

**WARNUNG!**

bedeutet, dass Tod oder schwere Körperverletzung eintreten können, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

**VORSICHT!**

bedeutet, dass leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

**Achtung**

bedeutet, dass ein Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

**ESD**

Mit dieser Warnung wird auf die möglichen Folgen beim Berühren von elektrostatisch empfindlichen Bauteilen hingewiesen.

***Sicherheitsinformation***

*Beschreibt wichtige sicherheitsrelevante Anforderungen oder informiert über wesentliche sicherheitstechnische Zusammenhänge.*

***Information***

*Kennzeichnet Anwendungstipps und nützliche Informationen. Es sind keine Informationen enthalten, die vor einer gefährlichen oder schädlichen Funktion warnen.*

## 2.2 Allgemeine Sicherheitshinweise

Die Dokumentation enthält die nötige Information zur Planung des Einsatzes von u-create studio in Verbindung mit einer UC20-SL2000-EC oder einer UC20-SL2000-EC-CAN.

Die Kenntnis und das einwandfreie Umsetzen der in den Unterlagen enthaltenen Information ist Voraussetzung für die erfolgreiche Planung und für gefahrlose Installation, Inbetriebnahme und Instandhaltung von Automatisierungssystemen. Nur qualifiziertes Personal verfügt über das erforderliche Fachwissen, um die in diesen Dokumenten enthaltenen Anweisungen richtig zu interpretieren und umzusetzen.

Aus Gründen der Übersichtlichkeit sind hier nicht sämtliche Details zu allen Ausführungen der beschriebenen Produkte angeführt und es kann nicht jeder Anwendungsfall berücksichtigt werden. Sollten Sie weitere Informationen benötigen, dann fordern Sie diese bitte bei Weidmüller an.

Für die Installation, die Inbetriebnahme und die Instandhaltung von u-create-Produkten sind die jeweiligen Teile der Dokumentation zu beachten.

## 2.3 Personensicherheit

**WARNUNG!**

Bei unqualifizierten Eingriffen in die Steuerung kann dadurch ausgelöstes Fehlverhalten der Maschine/Anlage Körperverletzungen oder Sachschäden verursachen. Nur entsprechend qualifiziertes Personal darf deshalb Eingriffe am Steuerungssystem vornehmen.

## 2.4 Sicherheitsanweisung zur Projektierung

**VORSICHT!**

- Die in den weiteren mitgelieferten Handbüchern enthaltenen Anweisungen müssen in jedem Fall genau befolgt werden. Andernfalls können Gefahrenquellen geschaffen oder ins u-create integrierte Schutzbeschaltungen unwirksam gemacht werden.
- Unabhängig von den in diesen Handbüchern gegebenen Sicherheitshinweisen sind die dem jeweiligen Einsatzfall entsprechenden Sicherheits und Unfallverhütungsvorschriften zu beachten.
- Es sind Vorkehrungen zu treffen, dass nach Spannungseinbrüchen und Spannungsausfällen ein unterbrochenes Programm ordungsgemäß wieder aufgenommen werden kann. Dabei dürfen auch kurzzeitig keine gefährlichen Betriebszustände auftreten.
- Überall wo im Automatisierungssystem auftretende Fehler Personenschaden oder großen Materialschaden verursachen könnten, müssen zusätzliche externe Maßnahmen getroffen werden, die auch im Fehlerfall einen sicheren Betriebszustand des Gesamtsystems gewährleisten.

# 3 Systemübersicht

Das Einsatzgebiet von u-control ist die allgemeine Maschinen- und Anlagenautomatisierung. Das System ist modular konzipiert und kann den jeweiligen funktionalen Anforderungen entsprechend aufgebaut werden.

Es setzt sich aus folgenden Komponenten zusammen:

- Baukasten mit anreihbaren I/O-Baugruppen (u-remote) für verschiedene Anwendungen.
- Toolsuite für Automatisierungsaufgaben, bestehend aus Tools für Konfiguration, Programmierung, Visualisierung sowie zur Inbetriebnahme und Diagnose. Zur Unterstützung bei der Programmierung werden Software-Bibliotheken und Schnittstellen zur Verfügung gestellt.

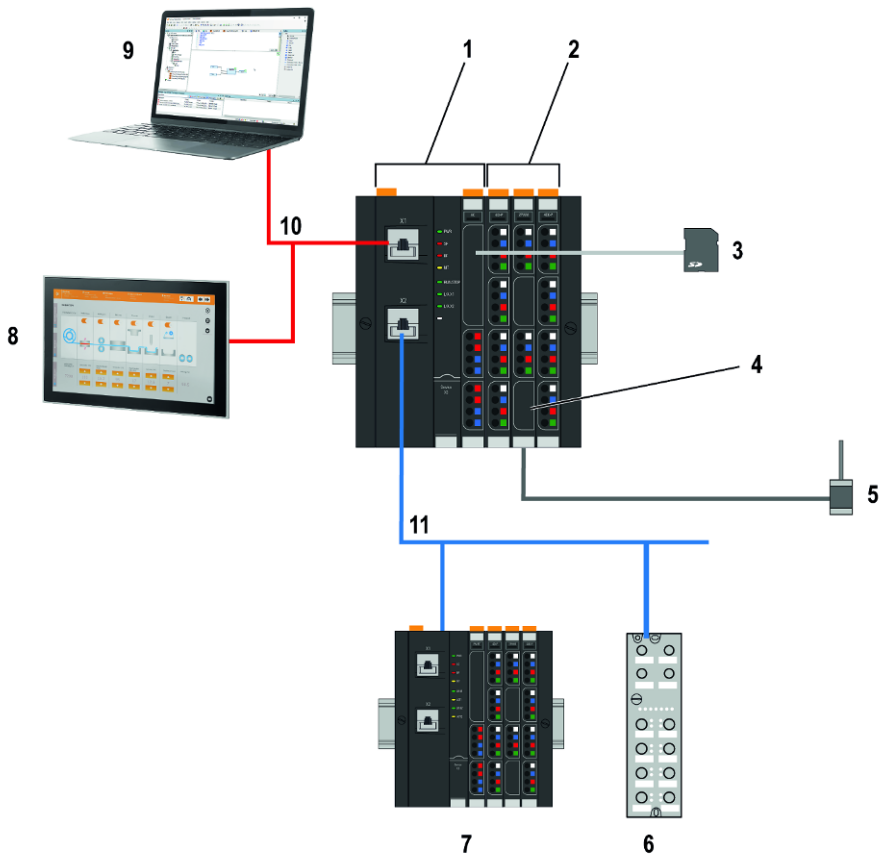



Abb. 3-1: Beispiel für einen Systemaufbau

| | |
|---|---|
| 1 ... Steuerung | 2 ... UR20-Module |
| 3 ... SD-Karte | 4 ... PWM-Modul |
| 5 ... DC-Motor | 6 ... UR67-Modul |
| 7 ... UR20-Station | 8 ... Touch-Panel |
| 9 ... Tool-Suite | 10 ... Ethernet |
| 11 ... Feldbus | |

***Information***

*Die in diesem Handbuch abgebildeten Systemkomponenten sind Beispielgrafiken. Die von Ihnen verwendeten Geräte können sich in ihrem Aussehen unterscheiden. Beachten Sie dazu die mitgelieferten Projektierungshandbücher der Geräte.*

## 3.1 Hardware Aufbau

Geräte werden entsprechend ihres Signaltyps, z.B. über analoge oder digitale Eingabe- oder Ausgabebaugruppen, über Positionierbaugruppen oder über Schnittstellenbaugruppen, etc. angeschlossen.

CPU-Baugruppen und I/O-Baugruppen müssen im Schaltschrank eingebaut werden. Ihr Gehäuse bietet ausschließlich mechanischen Schutz, die EMV-Schirmung erfolgt durch konstruktive innere Gestaltung. Bei größeren Entfernungen zwischen den Signalgebern ist eine dezentrale Gruppierung von I/O-Baugruppeninseln möglich. Diese können über Buskoppler mit der CPU-Baugruppe verbunden werden.

Die Bedien- und Anzeigegeräte können an geeigneter Stelle, abgesetzt an der Maschine/Anlage angeordnet werden.

Beim Systemstart vergleicht das Laufzeitsystem die aktuelle Hardwarekonfiguration (Ist-Konfiguration) mit der im u-create studio-Projekt gespeicherten Hardwarekonfiguration (Soll-Konfiguration). Abweichungen der Konfiguration oder fehlerhafte Baugruppen können über die Abfrage des Modulstatus in der IEC-Applikation identifiziert werden. Die Reaktion wird dort programmatisch festgelegt (z.B. Fehlerausgabe auf die Visualisierung, eingeschränkte Funktionalität bei optionalen I/O-Baugruppen,...).

## 3.2 Software Aufbau

Folgende Grafik zeigt den Aufbau der Software im u-create System.

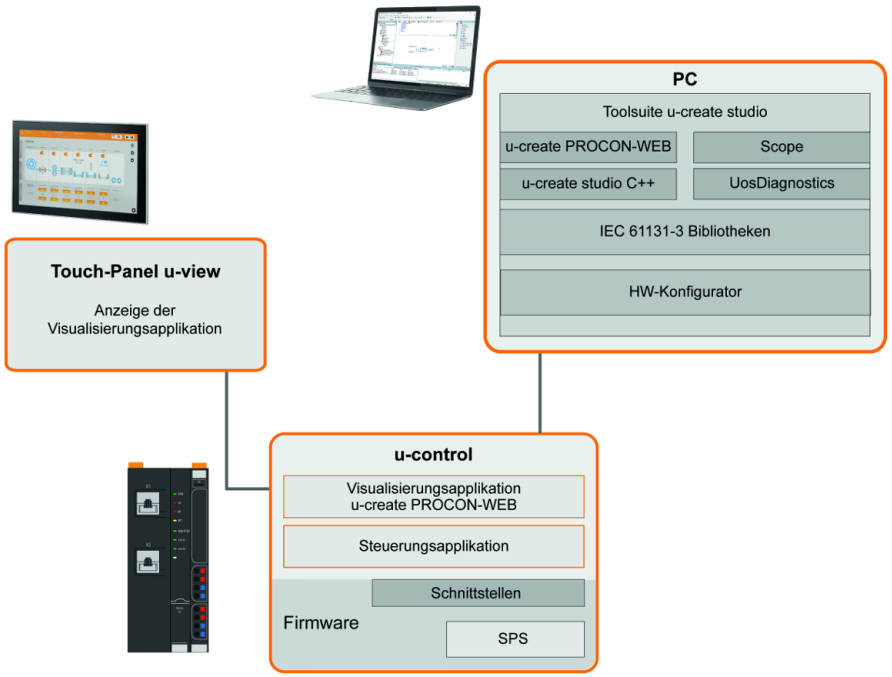



Abb. 3-2: SW-Übersicht

Die Steuerungsapplikation wird auf der Steuerung ausgeführt. Weiters stehen Bibliotheken und Schnittstellen zur Verfügung, über die auf die Steuerung programmatisch zugegriffen werden kann.

Auf einem PC wird die Toolsuite installiert. Die Programme dienen zum Parametrieren, Programmieren und Diagnostizieren.

Die Software der Steuerung basiert auf einem Debian (Linux) Betriebssystem mit speziell zugeschnittenem Umfang.

Das System setzt sich neben dem Linux-Basissystem aus einer Vielzahl an Paketen zusammen und ist damit individuell konfigurierbar. Neben einer Vielzahl öffentlich erhältlicher Standard-Pakete können Sie auch eigene Pakete erzeugen und in Ihr System einbinden. Eine detaillierte Beschreibung der verfügbaren Standard-Pakete entnehmen Sie bitte der Debian-Dokumentation unter http://www.debian.org/doc/. Unter http://www.debian.org/distrib/packages finden Sie eine inoffizielle Auflistung von Paketen und ihrer Paketquellen.

Folgende Laufzeitlizenzen unterstützen die Einbindung eigener Pakete:

- U-CREATE-STUDIO-RT-OLAC
- U-CREATE-STUDIO-RT-OLC

***Information***

*Die Versionsbezeichnung Ihres u-create-Systems entnehmen Sie bitte den mit der Version mitgelieferten Releasenotes!*

Das verwendete Laufzeitsystem auf der Steuerung entspricht dem IEC Standard nach 61131-3. Folgende Themenbereiche werden abgedeckt:

- IEC-Programmierung: Alle Sprachen
- C-Programmierung

## 3.3 Netzwerkaufbau

Die IP-Adresse des Netzwerks wird typischer Weise von einem Netzwerkadministrator vorgegeben. Bei der ersten Inbetriebnahme der Steuerung mit u-create studio ist die IP-Adressierung der Geräte nicht aufeinander abgestimmt. Die Netzwerkeinstellungen des PCs oder der Steuerungen müssen daher angepasst werden.

Die Kommunikationsverbindung zwischen PC und Steuerung kann entweder über eine direkte Verbindung über ein Ethernet-Kabel zwischen der Ethernet-Schnittstelle des PCs und der Onboard-Ethernet Schnittstelle der Steuerung oder über einen Switch erfolgen.

Ist ein DHCP-Server im Netzwerk vorhanden, kann die Netzwerkadresse automatisch bezogen werden. Ist kein DHCP-Server vorhanden oder besteht direkte Verbindung zwischen Steuerung und PC wird die Netzwerkadresse fix vom Benutzer vergeben.

***Information***

*Steuerung und Tools unterstützen kein NAT (Network Address Translation). Diese Einschränkung ist vor allem bei einem Betrieb über VPN-Tunnel, Firewalls zu beachten!*

*Wird eine Firewall im System verwendet, muss Port 1744 für UDP Verkehr (Einkommend sowie ausgehend) freigeschaltet werden. Ansonsten ist keine Verbindung zur Steuerung möglich.*

*Bei Fragen zum Netzwerkbetrieb wenden Sie sich bitte an Ihren Netzwerkadministrator.*

Der PC im Netzwerk kann nur mit der Steuerung kommunizieren. Eine Kommunikation zu den weiteren Komponenten des Systems ist in dieser Netzwerk-Anordnung nicht möglich.

Die IP-Adressen können beim Erstellen eines Zielsystems ("Erstelle Target") selbst festgelegt werden. Folgende Werte sind standardmäßig voreingestellt.

| Name | Adresse |
|---|---|
| Steuerung (X1): | 192.168.101.100 |
| Subnetzmaske: | 255.255.255.0 |
| Gateway: | 192.168.101.x |
| u-view Panel: | 192.168.101.x |
| USB Schnittstelle: | 192.168.227.1 |

***Information***

*Werden mehrere Panels verwendet, müssen für diese unterschiedliche IP-Adressen eingestellt werden.*

***Information***

*Um die IP-Adresse eines Panels zu ändern muss sich dieses in einem Netzwerk befinden.*

Die IP-Adressen aller angehängten I/O-Baugruppen werden im u-create studio mittels EoE-Einstellungen eingestellt (siehe Onlinehilfe).

***Information***

*Es können maximal 3 Handbediengeräte gleichzeitig verwendet werden.*

### Verbindung Service-PC - Systemkomponenten

Damit ein Service-PC auch mit weiteren Komponenten kommunizieren kann, muss sich dieser ebenfalls im Maschinennetzwerk befinden. Dazu wird der Service-PC entweder über einen Switch oder direkt an dem X1 Port der Steuerung angeschlossen und muss eine zum Netzwerk passende IP-Adresse erhalten.

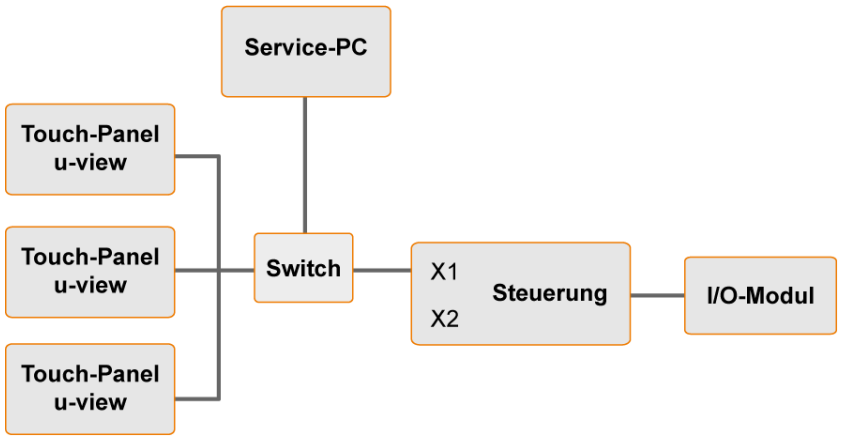



Abb. 3-3: Service-PC im Maschinennetzwerk

## 3.4 u-control - CPU-Baugruppen

| Bezeichnung | Leistungsdaten | Schnittstellen |
|---|---|---|
| UC20-SL2000-EC | Dual Core A9, 512 MB RAM, 8 GB EMMC | • u-remote Bus<br>• EtherCAT Master |

| Bezeich-nung | Leistungsdaten | Schnittstellen |
|---|---|---|
| UC20-SL2000-EC-CAN | Dual Core A9, 512 MB RAM, 8 GB EMMC | • u-remote Bus<br>• EtherCAT Master<br>• CANopen Interface |

Weitere Informationen entnehmen Sie bitte dem u-control Handbuch oder dem Produktkatalog (Bereich Automatisierung).

## 3.5 Busse

In den folgende Abschnitten werden die vom u-create System unterstützten Busse aufgelistet und näher beschrieben.

Folgende Master-Feldbus Schnittstellen stehen je nach Variante der Steuerung zur Verfügung:

- EtherCAT
- CANopen
- Modbus TCP/IP

Folgende Slave-Feldbus-Schnittstellen stehen je nach Variante der Steuerung zur Verfügung:

- Modbus TCP/IP
- CANopen

### 3.5.1 EtherCAT

Die EtherCAT-Schnittstelle kann zur Anbindung von Slave-Geräten (z.B. Antriebe, I/O-Module) verwendet werden.

***Information***

*Die Schnittstelle X2 kann optional auch als Ethernet-Schnittstelle verwendet werden. Stellen Sie sicher, dass auf der UC20-SL2000 ein Bootloader der Version 1.13 oder höher installiert ist. Sie können über das Firmware-Upgrade-Tool ein Upgrade durchführen. Bei Fragen wenden Sie sich bitte an support.automation@weidmueller.com.*

### 3.5.2 CAN

Bei Nutzung der CAN-Schnittstelle ist darauf zu achten, UC20-SL2000-EC-CAN mit CAN-Controller einzusetzen.

### 3.5.3 Modbus TCP/IP

Modbus ist ein offenes Kommunikationsprotokoll für den Datenaustausch zwischen einem Master, der die Daten anfordert oder schreibt, und Slaves, die die Daten bereitstellen oder entgegen nehmen.

Das Modbus Protokoll ist spezifiziert für serielle Verbindungen (Modbus RTU) und für Ethernet (Modbus TCP). Typische Einsatzgebiete sind Übertragung von Werten bei zeitunkritischen Aufgaben. Die Daten werden über eine vom Slave-Hersteller definierte Modbus-Tabelle in einzelnen Registern (16bit Werte) oder Coils (diskrete 1bit Werte) bereitgestellt.

### Konfiguration

Um mit dem Modbus Server auf der Steuerung kommunizieren zu können, muss dieser in u-create studio konfiguiert werden. Nähere Informationen siehe u-create studio Onlinehilfe.

## 3.6 u-create studio Toolsuite

Die Tools verbinden sich alle über Ethernet mit der Steuerung. Sie dienen der Konfiguration, Parametrierung, Programmierung, Debuggen und zur Diagnose der Steuerung. Folgende Tools werden dabei zu Verfügung gestellt:

- **Konfiguration und Programmierung mit u-create studio:**

  Die Programmierumgebung u-create studio dient zum Konfigurieren, Parametrieren, Programmieren und zum Debuggen von Steuerungsapplikationen.

- **C/C++ Programmerstellung mit u-create studio C++:**

  u-create studio C++ dient zur Erstellung und Diagnose/Debuggen einer Steuerungsapplikation in den Programmiersprachen C und C++.

- **Diagnose mit u-create studio Scope:**

  u-create studio Scope dient zur Beobachtung, Aufzeichnung und Visualisierung der Werte beliebiger Variablen. Im Unterschied zu einem Debugger ist keine Unterbrechung des Programmablaufs notwendig.

In den nachfolgenden Kapiteln werden die Tools näher beschrieben.

### 3.6.1 u-create studio

Die Entwicklungsumgebung u-create studio bietet eine komfortable Benutzeroberfläche mit folgenden Funktionen:

- Konfiguration und Parametrierung des u-create-Systems (mit der Steuerungskonfiguration)
- Programmierung nach EN 61131-3
- Integrierte Bausteinbibliotheken (siehe Bibliotheksbeschreibungen)
- Bibliotheksverwalter zur Einbindung weiterer Bibliotheken
- Debugging- und Diagnose-Funktionen (Programmablauf testen, Variablen beobachten und ändern, Fehlersuche).

## Die Programmiersprachen

u-create studio bietet alle 5 in der EN 61131-3 genormten Programmiersprachen. Jede dieser Sprachen hat bestimmte Eigenschaften, die sich zur Lösung bestimmter Aufgaben besonders gut eignen.

| Programmiersprache | Art | Beschreibung |
|---|---|---|
| Strukturierter Text (ST) | Textuelle Programmiersprache | Der "Strukturierte Text" kommt den für den PC genutzten Programmiersprachen wie Pascal und C am nächsten. Er besteht aus einer Reihe von Anweisungen, die wie in Hochsprachen bedingt (”IF..THEN..ELSE”) oder in Schleifen (WHILE..DO) ausgeführt werden können. |
| Funktionsbaustein Sprache (FUP) | Grafische Programmiersprachen | Ermöglicht die Programmierung mit Logiksymbolen. Sie ist besonders für Verknüpfungssteuerungen geeignet. |
| Freigraphischer Funktionsplan (CFC) | | Zusätzlich gibt es auf Basis des Funktionsplans den freigraphischen Funktionsplan (CFC), bei dem die Elemente frei platziert und Rückkopplungen direkt eingefügt werden können. |
| Kontaktplan (KOP) | | Der Kontaktplan wurde aus dem Stromlaufplan entwickelt. Die Darstellung eines KOP-Programms ist daher der Darstellung eines Stromlaufplans ähnlich - bezogen auf die Darstellung logischer Verknüpfungen. |
| Ablaufsprache (AS) | | Die "Ablaufsprache" ist eine Kette von Steuerungsschritten, die durch Weiterschaltbedingungen (Transitionen) verbunden sind. |

## Bibliotheksverwalter

Zur Erleichterung der Programmierung ermöglicht u-create studio projektunabhängig verwendbare Objekte wie Bausteine, Deklarationen und Visualisierungen in Bibliotheken zu organisieren. Hierzu steht ein Bibliotheksverwalter zur Verfügung, mit dem Sie Bibliotheken einbinden und einsehen können.

### 3.6.2 u-create studio C++

Mit u-create studio C++ können C bzw. C++ Programme erstellt und in ein Weidmüller-System eingebunden werden.

u-create studio C++ bietet auch die Möglichkeit die erstellten C bzw. C++ Programme, die auf der Steuerung abgearbeitet werden, debuggen zu können. Dabei läuft ein Debugserver direkt auf der Steuerung, die Bedienoberfläche zum Debuggen befindet sich aber im u-create studio C++.

Das u-create System bietet folgende zusätzliche Möglichkeiten der C-Programmierung:

- **Zyklische C-Funktionen**

  Ermöglicht es C-Funktionen zu integrieren. Dabei besteht die Möglichkeit je eine Funktion beim Hochlauf, beim Beenden bzw. beim Starten und Stoppen des Systems einzuhängen. Für die zyklische Abarbeitung können eine oder mehrere Tasks konfiguriert werden (inkl. Prioritäten, Watchdog,…). Für den Zugriff auf Dienste der Steuerung und das I/O-System werden Bibliotheken, welche bereits im u-create studio C++ eingebunden sind, zur Verfügung gestellt.
  Im u-create studio C++ dient der Projekttyp "Extended C Runtime Project" als Beispiel. Darin sind eine zyklische Callback Funktion und die Event Callback-Funktionen für die Zustandsübergänge der Steuerung und deren Konfiguration dargestellt.

- **Schnelle Regelung in C**

  Ähnlich dem zyklischen C-Funktionen kann hier eine C-Funktion eingehängt werden. Allerdings wird diese zu einem fixen Zeitpunkt ausgeführt. Ermöglicht das schnelle Lesen, Bearbeiten und Ausgeben von Werten innerhalb eines Zyklus. Funktionsaufrufe bei Init/Start/Stop/Exit der Steuerung können ebenfalls eingehängt werden.
  Im u-create studio C++ dient der Projekttyp "Fast Control C Project" als Beispiel. Darin sind die zyklische Callback-Funktion für die schnelle Regelungsaufgabe und die Event Callback-Funktionen für die Zustandsübergänge der Steuerung und deren Konfiguration dargestellt.

- **Eigenständiges C-Laufzeitsystem**

  Ermöglicht die Einbindung eines C-Laufzeitsystems ins System, das unabhängig läuft und über das Device Service in den Steuerungsablauf integriert werden kann. Alle Tasks müssen vollständig ausprogrammiert werden. Es besteht vollständiger Zugriff auf POSIX/Linux und allen von Weidmüller zur Verfügung gestellten Schnittstellen.
  Im u-create studio C++ dient z.B. der Projekttyp "Custom Realtime Runtime System" als Beispiel. Darin sind der strukturelle Aufbau eines eigenständig laufenden C-Laufzeitsystems und die Interaktion mit dem Steuerungsablauf dargestellt. Weitere Beispiele mit unterschiedlichen Aufgaben sind: "Executable Communication with the Controller" und "TCP Server Application".

- **Integrieren von C Funktionen in IEC**

  Ermöglicht das Aufrufen von C-Funktionen aus der IEC. Die zyklische Abarbeitung läuft somit im Kontext des IEC-Programmes.
  Im u-create studio C++ dient z.B. der Projekttyp "CoDeSys IEC Functions in C" als Beispiel. Darin ist der C-Anteil, der aus der IEC aufgerufen werden kann angeführt.

### 3.6.3 u-create studio Scope

u-create studio Scope dient zur Beobachtung, Aufzeichnung und Visualisierung der Werte beliebiger Variablen in Software-Komponenten. Dies können Firmwarekomponenten oder Applikationsprogramme sein. Im Unterschied zu einem Debugger ist keine Unterbrechung des Programmablaufs notwendig. Tabellen und Diagramme erlauben die Visualisierung der aufgezeichneten Variablenwerte.

Das Tool besitzt folgende Eigenschaften:

- Es können Variablen und Arrays in Firmwarekomponenten und Teachtalk- bzw. IEC-Programmen aufgezeichnet werden. Dazu müssen die Variablen beim Sampler registriert werden.
- Variablen können vom Scope aus aktiviert oder deaktiviert werden. Nur aktivierte Variablen werden aufgezeichnet.
- Mit Hilfe von Ereignissen kann das Auftreten eines bestimmten Zustandes im Zielprogramm gemeldet werden.
- Verwandte Ereignisse werden zu Klassen zusammengefasst. Jedes Ereignis gehört zu einer Klasse.
- Der Aufzeichnungsstart kann über eine Triggerung festgelegt werden. Der Trigger kann sich auf einen Variablenwert und/oder ein bestimmtes Ereignis beziehen. Pre- und Posttriggerung sind möglich.
- Darstellung der gesammelten Daten in verschiedenen Diagrammen und Ansichten: Variablenmonitor, Ereignismonitor, X-t Diagramm, X-Y Diagramm.

***Information***

*Bei der Aufzeichnung von Daten durch u-create studio Scope kommt es zu einer geringen zeitlichen Verzögerung im Programmablauf.*

Nähere Informationen siehe Dokumentation von u-create studio Scope.

## 3.7 Bibliotheken und Schnittstellen

Im Folgenden werden die Software-Schnittstellen, die das System zur Verfügung stellt, aufgelistet.

### 3.7.1 IEC-Standardbibliotheken

Folgende IEC Standardfunktionen werden im u-create studio zur Verfügung gestellt:

- IEC61131-3 Standardbausteine
- Zugriff auf Systemfunktionen
- Zugriff auf das Meldesystem
- Diagnosefunktionalitäten

Nähere Informationen siehe Onlinehilfe von u-create studio.

### 3.7.2 u-create studio Bibliotheken

Durch die spezifischen Bibliotheken stehen unter anderem folgende Funktionalitäten zur Verfügung:

- Bausteine zum Zugriff auf das Meldesystem und Variablen in der Steuerung
- Bausteine zur Buskommunikation
- Bausteine für Diagnosemöglichkeiten

Nähere Informationen siehe Onlinehilfe von u-create studio.

### 3.7.3 OPC UA Schnittstelle

Das u-create System bietet die Möglichkeit, über eine OPC UA Schnittstelle auf IEC-Variablen der Steuerung zugreifen zu können. Dazu wird ein OPC UA Client, der auf dem PC installiert sein muss, benötigt. Im Internet werden OPC UA Clients zum Download angeboten.

Um mit dem OPC UA Client eine Verbindung mit dem OPC UA Server auf der Steuerung herzustellen, muss zuerst die IP-Adresse oder der Hostname der Steuerung mit der Portnummer `4842` im Client eingegeben werden. Wurde der Server gefunden, muss der Endpunkt mit dem Namen "None - None" gewählt werden. Danach kann die Verbindung zum Server aufgebaut werden und die vom OPC UA Server zur Verfügung gestellten Daten (Variablen, Funktionen, ...) werden angezeigt. Eine detaillierte Beschreibung entnehmen Sie der Hilfe des verwendeten OPC UA Clients.

Die vom OPC UA Server gelieferten Server- bzw. Endpunkt-URLs enthalten immer den Hostnamen. Aus diesem Grund muss es dem OPC UA Client möglich sein, den Hostnamen auflösen zu können. Einige OPC UA Clients bieten diese Funktionalität an. Ist dies nicht der Fall, muss manuell auf dem PC die Datei `C:\Windows\System32\drivers\etc\hosts` um die gewünschte Steuerung erweitert werden. Dazu wird eine neue Zeile mit der IP-Adresse und dem Hostnamen der Steuerung hinzugefügt (z.B. `10.150.48.40  <Hostname der Steuerung>`).

### 3.7.4 REST-API (Representational state transfer)

Das u-create System bietet die Möglichkeit einer Visualisierungsanbindung über die REST-API (Representational state transfer) Schnittstelle. Die Schnittstellenbeschreibung steht als OpenAPI Spezifikation im YAML-Format zur Verfügung.

Die API ist über einen Internet-Browser über folgende Adresse erreichbar:

`<Steuerungs IP-Adresse>/api`

Über diese Adresse wird die aktuelle Version der REST-API angezeigt. Durch Anhängen der Version an die Adresse wird die unterstützte Schnittstelle angezeigt (z.B. `192.168.1.1/api/v3`)

Folgende Services werden angeboten und als Schnittstellendatei aufgelistet:

- `configuration.yaml`: Durchführen von Grundkonfigurationen (z.B. Netzwerkzugriff, Datum und Uhrzeit, ..)

- `device-service.yaml`: Ausführen von Grundkommandos der Steuerung (z.B. Starten, Stoppen, ...)
- `diagnostics.yaml`: Erhalten von Diagnoseinformationen (z.B. Statusreport, ...)
- `license-text.yaml`: Auslesen der "Open Source" Lizenztexte der Steuerung
- `licensing.yaml`: Durchfühen von Lizenzierungen (z.B. Aktivieren von Lizenzen, Lizenzübersicht, ...)
- `report-system.yaml`: Anbindung an das Meldesystem (z.B. Fehler auslesen, Fehler quittieren, ...)
- `update-service.yaml`: Durchführen eines Backups bzw. Wiederherstellen dessen
- `variables.yaml`: Lesen und Schreiben von freigegebenen Variablen

Durch Anhängen des jeweiligen Dateinamens an die Url können die Funktionen und eine Beschreibung dazu ausgelesen werden (z.B. `192.68.1.1/api/v3/configuration/configuration.yaml`)

# 4 Betriebsverhalten

## 4.1 Hochlauf

Der Hochlauf einer CPU gliedert sich in folgende drei Hauptabschnitte:

- Hochlauf Bootsystem
- Hochlauf Steuerungsfirmware
- Hochlauf Applikation

Der Hochlauf der Steuerung startet automatisch, sobald die CPU-Baugruppe mit Spannung versorgt wird. Voraussetzungen für den Hochlauf sind:

- Spannungsversorgung wurde korrekt angeschlossen,

# 5 Software Installation

Zur besseren Skalierbarkeit und zum Schonen von Speicherplatzreservern wurde das u-create studio in folgende Setups eingeteilt:

- Deliverable_Engineering_Environment: Setup zur Installation der Entwicklungsumgebungen (u-create studio, Codemeter)
  - Setup_SimulationService (optional): Setup zur Installation der Steuerungs-Simulation (Simulation Service, VBox)
- Deliverable_Server_Setup: Setup zu Installation der Server-Komponenten
- Deliverable_FirmwareUpdate: Tool zur Durchführung eines Firmware Updates der Steuerung

***Information***

*Wurde eine Komponente bereits installiert, unterstützt die automatische Installation keine Neu-Installation dieser Komponente. In diesem Fall muss die jeweilige Komponente mit Hilfe des individuellen Produkt-Setups oder über das Betriebssystem deinstalliert werden.*

Sytemvoraussetzungen: Standard PC mit Windows 10®

Beim Setup "Deliverable_Engineering_Environment" handelt es sich um die Installation der Entwicklungsumgebungen. Beim Setup "Deliverable_Server_Setup" handelt es sich um Server-Komponenten, auf die jede Entwicklungsumgebung zugreifen muss. Daher gibt es folgende Möglichkeiten einer Installation:

- Einzelplatz-Installation
- Netzwerk-Installation (empfohlene Installation)

**Einzelplatz-Installation**

Bei der Einzelplatz-Installation werden alle Setups auf dem PC ausgeführt. Die Entwicklungsumgebungen sowie die Server-Komponenten werden am selben PC installiert.

**Netzwerk-Installation**

Bei der Netzwerk-Installation werden die Entwicklungsumgebungen am jeweiligen PC installiert. Die Server-Komponenten werden einmalig auf einem Server im Netzwerk installiert und stehen somit allen Entwicklungsumgebungen zur Verfügung. Diese Installation bietet den Vorteil, die Installationsdauer und den Speicherbedarf bei der Installation der Entwicklungsumgebungen am PC deutlich zu reduzieren. Ebenfalls wird eine Mehrfach-Installation der selben Server-Komponenten (in einem Netzwerk) vermieden.

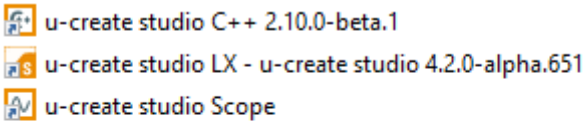
## Installation "Deliverable_Engineering_Environment"

Mit diesem Setup werden die benötigten Entwicklungsumgebungen am PC installiert. Gehen Sie wie folgt vor:

1) Öffnen des Installationsverzeichnisses.
2) Öffnen des Unterordners "Setup".

3) Ausführen der Anwendung "setup.exe". Der Installationsdialog wird geöffnet.
4) Folgen der Anweisungen im Installationsdialog.
5) Durch Drücken von "Install" wird der Installationsvorgang gestartet.
6) Durch Drücken von "Finish" wird die Installation abgeschlossen.

Alle Komponenten wurden installiert. Am Desktop befindet sich eine Verknüpfung auf ein Verzeichnis, aus dem alle notwendigen Komponenten gestartet werden können.




***Information***

*Während des Installationsvorgangs müssen alle laufenden Applikationen geschlossen sein.*

***Information***

*Falls die Installation fehlschlägt, deaktivieren Sie vorhandene Virenscanner temporär. Sprechen Sie die temporäre Deinstallation mit Ihrer IT ab.*

## Optional: Installation "Setup_SimulationService"

Mit diesem Setup wird das optionale Simulationsservice (Steuerungssimulation) am PC installiert.

***Information***

*Befindet sich bereits eine installierte Version der Software "VBox" am PC, die älter ist, als die zu installierende Version, wird diese vollständig deinstalliert und durch die mitgelieferte Version ersetzt. Die benötigte Version kann den mitgelieferten Releasenotes entnommen werden.*

Gehen Sie wie folgt vor:

1) Öffnen des Installationsverzeichnisses.
2) Öffnen des Unterordners "Setup".
3) Ausführen der Anwendung "Setup_SimulationService.exe". Der Installationsdialog wird geöffnet.
4) Folgen der Anweisungen im Installationsdialog.
5) Durch Drücken von "Install" wird der Installationsvorgang gestartet.

6) Durch Drücken von "finish" wird die Installation abgeschlossen.

Der Dienst "SimulationService" wurde installiert. Die Simulation kann nun verwendet werden.

## Installation "Deliverable_Server_Setup"

Mit diesem Setup werden die benötigten Server-Komponenten installiert. Gehen Sie wie folgt vor:

1) Öffnen des Installationsverzeichnisses.
2) Öffnen des Unterordners "Setup".
3) Ausführen der Anwendung "setup.exe". Der Installationsdialog wird geöffnet.
4) Folgen der Anweisungen im Installationsdialog.
5) Durch Drücken von "Install" wird der Installationsvorgang gestartet.
6) Durch Drücken von "finish" wird die Installation abgeschlossen.

Alle Komponenten wurden installiert.

Wurden die Komponenten auf einem Server im Netzwerk installiert (Netzwerk-Installation), muss noch der Zugriff darauf im u-create studio konfiguriert werden. Gehen Sie dazu wie folgt vor:

1) Starten von u-create studio
2) Öffnen von "Tools - Options"
3) Auswahl von "Software Service"

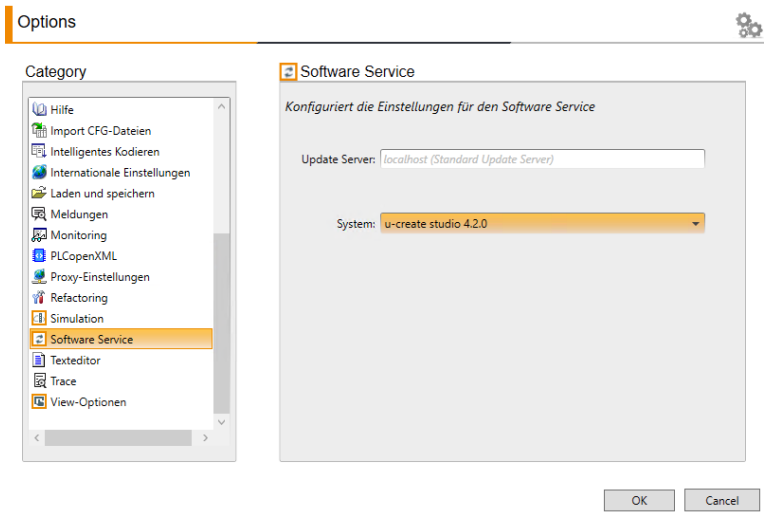



Abb. 5-4: Software Service

4) Einstellen der IP-Adresse des Servers und die gewünschte Software Version.

Die Konfiguration ist abgeschlossen.

## 5.1 Firmware auf Gerät installieren

Auf einigen Geräten (z.B. Steuerung) ist im Auslieferzustand außer der hardwarenahen Software keine Firmware installiert. Diese Firmware (ohne hardwarenahe Software) muss mittels u-create studio auf ein Speichermedium geladen und mit diesem auf dem Gerät installiert werden.

Dazu wird ein leeres Speichermedium (FAT32-formatiert) mit einem Speicher von mind. 2 GB (max. 32 GB) benötigt, das am PC angesteckt ist.

Für weiterführende Informationen siehe Online-Hilfe von u-create studio im Kapitel "Firmware auf Gerät laden ".

# 6 Konfiguration

Mit der Konfiguration wird die bei der Hardwaremontage erfolgte Zusammenschaltung der Baugruppen und Module in u-create studio nachgebildet.

Grundlage der Steuerungskonfiguration sind die Gerätebeschreibungen. Diese beschreiben die Grundkonfiguration, die vorgibt, welche Parametriermöglichkeiten zur Verfügung stehen. Die Steuerungskonfiguration erlaubt das Anbinden von I/O-Modulen, wie auch von Feldbus-Modulen. Je nach installierten Gerätebeschreibungen stehen weitere Elemente zur Verfügung.

Die verfügbaren I/O-Kanäle werden in der u-create studio-Steuerungskonfiguration angezeigt und können in der Applikation verwendet werden.

***Information***

*Nähere Informationen sowie die genaue Parameterbeschreibung der einzelnen Module entnehmen Sie der Onlinehilfe von u-create studio.*

Die Netzwerkkonfiguration der Steuerung ist auch über die Service-App "DevAdmin" möglich (siehe 10 Device Administration (DevAdmin)).

## 6.1 Datenaustausch zwischen Steuerung und Visualisierung

Wenn, wie allgemein bei Visualisierungen vorgesehen, Prozesswerte aus der Steuerung angezeigt oder Parameter aus der Steuerungsapplikation durch den Maschinenbediener verändert werden sollen, so kann dies via IEC-Systemvariablen erfolgen.

Für den Austausch der Systemvariablen zwischen Steuerung und Visualisierung stellt u-create studio die sogenannte Symboldatei zur Verfügung. Diese enthält die zum Austausch vorgesehenen Variablen des u-create studio Projekts.

Details zu Konfiguration bzw. Datenaustausch können der jeweiligen Onlinehilfe (u-create studio) entnommen werden.

# 7 Programmerstellung

Die Steuerung arbeitet nach folgendem Prinzip: "Eingänge lesen, Verarbeitung, Ausgänge schreiben". u-control führt bei der Abarbeitung jeder mit u-create studio definierten Task folgende Aufgaben aus:

- Eingänge lesen: Am Anfang des Zyklus werden die Ist-Zustände der Eingänge gelesen und ins Prozessabbild der Eingänge geschrieben.
- Verarbeitung: Das Anwenderprogramm wird ausgeführt. Der Soll-Zustand der Ausgänge wird in das Prozessabbild kopiert.
- Ausgänge schreiben: Am Ende des Zyklus werden die im Prozessabbild gespeicherten Soll-Zustände der Ausgänge auf die physischen Ausgänge übertragen.

Weiters besteht die Möglichkeit, mehrere Applikationen parallel oder geschachtelt zu erstellen und auf der Steuerung zu starten. Ein Variablenaustausch zwischen den Applikationen ist dabei möglich. Für jede angelegte Applikation kann eine eigene Symbolkonfiguration angelegt werden.

Details zu Programmierung können der Onlinehilfe von u-create studio entnommen werden.

## 7.1 Task-Prioritäten

Bei der Programmerstellung können in der IEC Abarbeitungsprioritäten für einen Task vergeben werden. Standardmäßig ist für einen Task Priorität 10 eingestellt.

In nachfolgenden Tabellen wird die Zuordnung der IEC-Task zum abgebildeten Linux-System Task beschrieben.

**Resevierte Echtzeittasks**

| Priorität Linux | Beschreibung |
|---|---|
| 99 - 82 | Tasks dieser Priorität sind dem System vorbehalten |

**Hochpriore Echtzeittasks (Standardpriorität 10)**

| Priorität Linux | Priorität IEC |
|---|---|
| 78 | 1 |
| 75 | 2 |
| 72 | 3 |
| 69 | 4 |
| 66 | 5 |
| 63 | 6 |

**Niedrig priore Echtzeittasks**

Diese Tasks können bei hoher Auslastung verdrängt werden und laufen im Linux Scheduler "SCHED_FIFO".

| Priorität Linux | Priorität IEC |
|---|---|
| 60 | 7 |
| 57 | 8 |
| 54 | 9 |
| 51 | 10 |
| 48 | 11 |
| 46 | Reserviert für UOS Task |
| 45 | 12 |
| 42 | 13 |
| 39 | 14 |
| 36 | 15 |

### Tasks außerhalb des Echtzeitkontextes

Diese Tasks laufen im Linux Scheduler "SCHED_OTHER".

| Priorität Linux | Priorität IEC |
|---|---|
| 10 ("nice level") | 16 - 31 |
| 0 ("nice level") | Standard Linux Prozesse |

## 7.2 Schnelle Regelung

Die schnelle Regelung wird verwendet, um die Systemreaktionszeit zu vermindern. Dies bedeutet, das Lesen der Eingänge, Verarbeiten der Daten und Schreiben der Ausgänge benötigt weniger Zyklen.

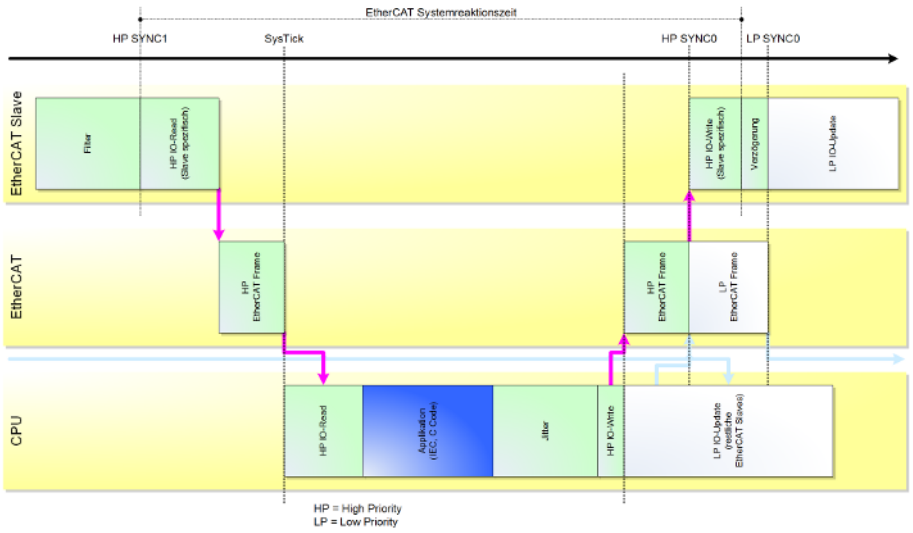



Abb. 7-5: Timing Diagramm

Die Konfiguration der schnellen Regelung erfolgt im u-create studio (siehe Onlinehilfe "EtherCAT-Slave konfigurieren").

## 7.3 Device Service

Das DeviceService ist ein Software Dienst, der einzelne Programme auf der Steuerung verwaltet, steuert und überwacht. Das DeviceService kann Informationen (z.B. Fehlermeldungen) der Programme sammeln und weitergeben.

Das DeviceService läuft als unabhängiger Prozess und startet automatisch, sobald das Betriebssystem vollständig hochgefahren ist.

Jedes Programm, das vom DeviceService verwaltet werden soll, muss auf der Steuerung als `DeviceServiceItem` definiert werden.

### 7.3.1 Definiertes Zustandsmodell

Das DeviceService ist verantwortlich für Zustandsübergänge einzelner Programme, die auf einem Gerät laufen.

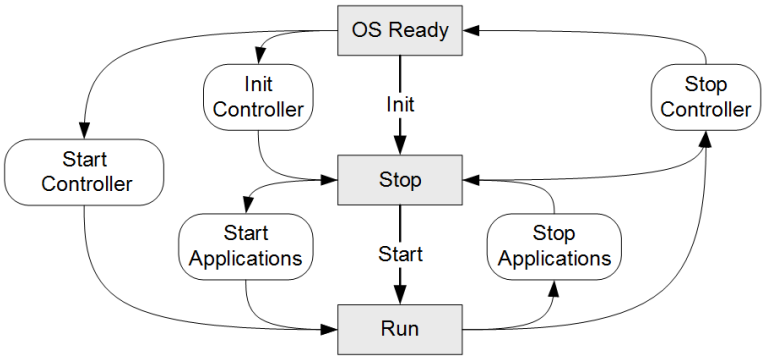



Abb. 7-6: DeviceService Zustandsmodell

#### Zustände

| Zustand | Beschreibung |
|---|---|
| OsReady | Das Betriebssystem ist bereit und das Device Service verfügbar. Programme nicht initialisiert. |
| Stop | Programme werden ausgeführt, Applikationen laufen nicht. |
| Run | Programme und Applikationen werden ausgeführt. |

Zustandsübergänge werden ausschließlich von Softwarekomponenten, welche auf dem Gerät laufen, von Bedienelementen des Geräts oder von einem Remote-Client (z.B. Programmiertool) ausgelöst.

Die folgenden Kommandos können vom DeviceService für die Zustandsänderung mehrerer Programme, die auf dem aktuellen Gerät laufen, durchgeführt werden:

| Kommando | Kommando-typ | Beschreibung |
|---|---|---|
| Start Controller | Statusübergang | Diese Zustandsübergang wird automatisch ausgeführt, sobald das Gerät gestartet wird. Der Zustand wechselt von "OsReady" zu "Run". |
| Stop Controller | Statusübergang | Der Zustand wechselt zu "OsReady". |
| Start Applications | Statusübergang | Der Zustand wechselt von "Stop" zu "Run". |
| Stop Applications | Statusübergang | Der Zustand wechelt von "Run" zu "Stop". |
| Init Controller | Statusübergang | Der Zustand wechelt von "OsReady" zu "Stop". |
| Shutdown Controller | Spezieller Statusübergang | Schließt alle laufenden Softwareanwendungen auf dem Gerät und hält es an. |
| Reboot Controller | Spezieller Statusübergang | Schließt alle laufenden Softwareanwendungen auf dem Gerät und startet es neu. |
| Delete Applications | Übergreifender Statusübergang | Löscht alle Anwendungen auf dem Gerät ("auf Werkseinstelungen zurücksetzen"). |
| Restart Controller | Kombinierter Statusübergang | Durchführen von "Stop Controller" und anschließendem "Start Controller". |
| Restart Applications | Kombinierter Statusübergang | Durchführen von "Stop Applications" und anschließendem "Start Applications". |

### 7.3.2 Device Service Item

Ein Device Service Item repräsentiert ein Programm, das auf dem aktuellen Gerät installiert ist. Dazu wird eine Beschreibungsdatei mit dem Namen `<itemname>.item` im Verzeichnis `/opt/deviceservice/items.d/` angelegt. Die Datei muss in Lua (http://www.lua.org) erstellt werden. Das Programm selbst muss ein definiertes Zustandsmodell mit definierten Übergängen implementieren.

#### 7.3.2.1 Definiertes Zustandsmodell Device Service Item

Jedes Programm (`DeviceServiceItem`) muss ein folgendes definiertes Zustandsmodell implementieren, um vom DeviceService verwaltet zu werden.

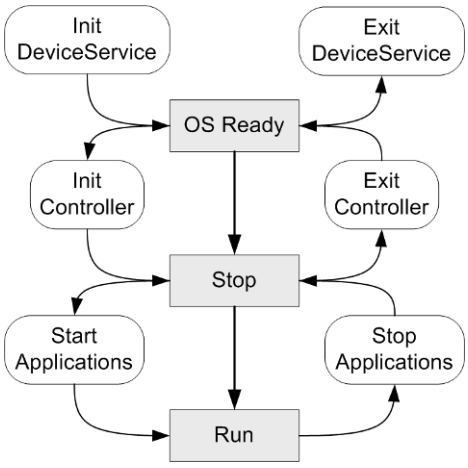



Abb. 7-7: DeviceService Item Zustandsmodell

Zustandsübergänge im Device Service Zustandsmodell werden an alle Device Service Items weitergeleitet. Dadurch können diese alle notwendigen Aktionen ausführen, um das jeweilige Programm in den gewünschten Zustand zu bringen.

### Zustände

Das Programm muss auf folgende Zustandsübergänge reagieren:

| Zustandsübergang | Beschreibung |
|---|---|
| Init DeviceService | Das DeviceService wird gestartet und liest alle angelegten Programme (DeviceService Items) ein |
| Init Controller | Das Programm muss alle notwendigen Initialisierungen durchführen und anschließend in den Zustand Stop wechseln |
| Start Applications | Das Programm beginnt die Abarbeitung und muss in den Zustand Run wechseln |
| Stop Applications | Das Programm beendet die Abarbeitung und muss in den Zustand Stop wechseln |
| Exit Controller | Das Programm muss alle in "Init Controller" durchgeführten Initialisierungen beenden und in den Zustand OS-Ready wechseln. |
| ExitDeviceService | Das DeviceService beendet alle Programme und sich selbst |

Kommandos, die an die Programme weitergeleitet werden, sind in Kapitel "Definiertes Zustandsmodell" zu finden.

## 7.3.2.2 Erstellen einer Beschreibungsdatei

Jedes Programm, dass vom DeviceService verwaltet werden soll, muss durch eine Beschreibungsdatei die folgende Elemente und die oben beschriebenen Funktionen enthält, angelegt werden.

| Name | Beschreibung |
|---|---|
| name | Name des Programms (`DeviceServiceItem`) |
| variables | Globale Variablen des Programms |
| defaultvariables | Globale Variablen, deren Standardwerte überschrieben werden können |

Das System stellt folgende globale Variabeln (defaultvariables) zur Verfügung, deren Standardwerte vom Programm überschrieben werden können:

| Globale Variable | Standardwert | Beschreibung |
|---|---|---|
| systemPath | | Verzeichnis, wo die Steuerungssoftware abgelegt ist. |
| applPath | /opt/kecontrolapplication/ | Verzeichnis, wo Applikationen abgelegt und geladen werden |
| workPath | | Verzeichnis, in das die Applikation Daten ablegen kann. |
| autorestart | false | Definiert, ob nach einem Error das System neu startet |
| starepPathTmp | /tmp/ | Verzeichnis, wo die Dateien des Statusreports abgelegt werden |
| maxStarepCount | 5 | Anzahl der gespeicherten Statusreports |
| autostart | true | Definiert, ob nach einem Starten der Steuerung die Programme sofort ausgeführt werden sollen |
| autostartTimeout | 5 | Zeit, in der der Autostart abgebrochen kann [s] |

Jedes `DeviceServiceItem` kann Konfigurationsvariablen exportieren oder von anderen `DeviceServiceItem` exportierte Konfigurationsvariablen verwenden. Variablen werden exportiert, indem sie zu den globalen Variablen `variables` hinzugefügt werden:

```
variables = {
    someVar = 12,
    otherVar = 'text'
}
```

Beispieldatei `example.item.ex`:

```
item = {
     name = "example"

    variables ={
    }

    defaultvariables = {
    }
    ---------TESTTRANSITION-----------------
        testtransition = function(transition)
        --Init DeviceService--
        if transition == "Init DeviceService" then
            return true
        --Exit DeviceService--
        elseif transition == "Exit DeviceService" then
```

```
        return true
    --Init Controller--
    elseif transition == "Init Controller" then
        return true
    --Exit Controller--
    elseif transition == "Exit Controller" then
        return true
    --Start Applications--
    elseif transition == "Start Applications" then
         return true
    --Stop Applications--
    elseif transition == "Stop Applications" then
        return true
    end
    return false
end

, --- DOTRANSITION ---------------------------
dotransition = function(transition, step)
    --Init DeviceService
    if transition == "Init DeviceService" then
        if step == 1 then
        elseif step == 2 then
        elseif step == 3 then
        elseif step == 4 then
        elseif step == 5 then
        elseif step == 6 then
        elseif step == 7 then
        elseif step == 8 then
        elseif step == 9 then
        elseif step == 10 then
        end
--Exit DeviceService--
elseif transition == "Exit DeviceService" then
        if step == 1 then
        elseif step == 2 then
        elseif step == 3 then
        elseif step == 4 then
        elseif step == 5 then
        elseif step == 6 then
        elseif step == 7 then
        elseif step == 8 then
        elseif step == 9 then
        elseif step == 10 then
        end
--Init Controller--
elseif transition == "Init Controller" then
        if step == 1 then
        elseif step == 2 then
        elseif step == 3 then
        elseif step == 4 then
        elseif step == 5 then
        elseif step == 6 then
        elseif step == 7 then
        elseif step == 8 then
        elseif step == 9 then
        elseif step == 10 then
        end
--Exit Controller--
elseif transition == "Exit Controller" then
        if step == 1 then
        elseif step == 2 then
        elseif step == 3 then
        elseif step == 4 then
        elseif step == 5 then
        elseif step == 6 then
        elseif step == 7 then
        elseif step == 8 then
        elseif step == 9 then
        elseif step == 10 then
        end
--Start Applications--
elseif transition == "Start Applications" then
```

```
            if step == 1 then
            elseif step == 2 then
            elseif step == 3 then
            elseif step == 4 then
            elseif step == 5 then
            elseif step == 6 then
            elseif step == 7 then
            elseif step == 8 then
            elseif step == 9 then
            elseif step == 10 then
            end
    --Stop Applications
    elseif transition == "Stop Applications" then
            if step == 1 then
            elseif step == 2 then
            elseif step == 3 then
            elseif step == 4 then
            elseif step == 5 then
            elseif step == 6 then
            elseif step == 7 then
            elseif step == 8 then
            elseif step == 9 then
            elseif step == 10 then
            end
        end
    , --- DOCHECK ----------------------------------
    docheck = function()
    end

    , --- DOSTATEREPORT ----------------------------
    dostatereport = function(stareppath)
    end

    , --- DOCLEARRETAIN ----------------------------
    doclearretain = function()
    end

    , --- DODELETEAPPLICATIONS ---------------------
    dodeleteapplications = function()
    end
}
```

### 7.3.3 Definiertes Betriebsverhalten

Jedes `DeviceServiceItem` (Programm) kann folgende Funktionen zur Verfügung stellen:

| Funktion | Eingänge | Rückgabewert | Beschreibung |
|---|---|---|---|
| testtransition | Statusübergang | true / false | Dient zur Prüfung, ob der gewünschte Zustandsübergang vom Program durchgeführt werden kann. |
| dotransition | Statusübergang, Stepnummer | - | Führt einen Schritt einer Zustandsüberführung durch. |
| docheck | - | | Prüft, ob das Programm korrekt arbeitet. |
| dostatusreport | Pfad | | Hinzufügen von Dateien zum aktuell ausgelösten Statusreport. |
| doclearretain | - | | Löschen der Retaindaten der Applikationen, die vom Programm verwaltet werden. |
| dodeleteapplication | - | | Löschen von Applikationen, die vom Programm verwaltet werden. |

Das DeviceService ruft vor jedem Zustandsübergang die Funktion `testtransition` mit dem gewünschten Zustandsübergang auf. Damit wird geprüft, ob es dem `DeviceServiceItem` möglich ist, seinen Zustand zu ändern. Die Funktion muss entweder `true` oder `false` zurückliefern. Ein anderer Wert ist nicht erlaubt.

Danach wird vom DeviceService die Funktion `dotransition` mit dem durchzuführenden Zustandsübergang und einer Stepnummer aufgerufen. Jedes `DeviceServiceItem` muss den Zustandsübergang durchführen. Tritt dabei ein Fehler auf, muss das `DeviceServiceItem` die Lua Standardfunktion `error` aufrufen, um die Fehlermeldung damit beim DeviceService eintragen zu können. Die Funktion `dotransition` darf keinen Rückgabewert besitzen.

Das DeviceService prüft auf Fehlermeldungen. Trat bei einem `DeviceServiceItem` ein Fehler auf, wird das gesamte System niedergefahren. Je nach Konfiguration fährt die Steuerung automatisch wieder in den Zustand Run hoch.

Das DeviceService prüft zyklisch durch Aufruf der Funktion `docheck`, ob das registrierte `DeviceServiceItem` korrekt arbeitet.

Der Statusreport ist eine Sammlung von Informationen von allen Programmen, die auf der Steuerung zu einem bestimmten Zeitpunkt laufen. Wenn ein Statusreport ausgelöst wird, werden alle `DeviceServiceItem` informiert, damit sie ihre Informationen an den Statusreport anhängen können.

### 7.3.4 Statusreport

Der Statusreport ist eine Sammlung an Informationen der Software-Teilsysteme, die zu einem bestimmten Zeitpunkt auf einem Gerät laufen. Wird ein Statusreport ausgelöst, werden alle `DeviceServiceItem` informiert, damit diese Informationen zum Statusreport hinzufügen können.

### 7.3.5 Co-Routine Modell

`DeviceServiceItem` Callbacks werden als Lua Co-Routinen ausgeführt. Dadurch ist es möglich, die Ausführung eines `DeviceServiceItem` durch Aufruf von `coroutine.yield()` zu verzögern. Dies soll einer Verzögerungsschleife im Lua-Script oder Zwischenlösungen wie das Aufrufen von Sleep-Funktionen mittels Bibliothek des Betriebssystems vorgezogen werden.

Andere Co-Routinen-Funktionen dürfen nicht verwendet werden.

# 8 Lizenzierung

Weidmüller Produkte müssen teilweise lizenziert werden, um den vollen Funktionsumfang nutzen zu können. Standardmäßig werden die Produkte mit einer "Trial-Lizenz" ausgeliefert. Das bedeutet, der komplette Funktionsumfang eines Produktes ist für eine bestimmte, befristete Zeit voll nutzbar. In dieser Zeit muss eine gültige Lizenz bei Weidmüller angefordert werden, um die gekauften Funktionen dauerhaft nutzen zu können. Dazu kontaktieren Sie bitte Weidmüller.

Folgende Lizenzarten sind möglich:

- Einzelplatzlizenz für Softwareprodukte am PC. Diese gelten nur für den PC, wo das Softwareprodukt installiert wurde.
- Laufzeitlizenz für Softwareprodukte auf dem Gerät. Diese gelten nur für die Geräte, wo die Software installiert wurde.
- Trial-Lizenz: Lizenz mit begrenzter Laufzeit

Für eine Lizenzierung eines Weidmüller Produkts ist immer ein gültiges Lizenz-Ticket und ein PC mit einer Internetverbindung notwendig.

***Information***

*Bewahren Sie Ihr Lizenz-Ticket immer auf! Es wird zwingend für eine Aktivierung der Lizenz bzw. Deaktivierung/Neuaktivierung bei Gerätetausch benötigt.*

Um das Software-Produkt auf dem aktuell installierten Gerät (PC oder Steuerung) zu lizenzieren, muss die Lizenz (vom Lizenz-Ticket) aktiviert werden. Eine Lizenz kann nur einmal aktiviert werden. Danach ist die Software an das Gerät gebunden und kann nur dort im vollständigen Umfang genutzt werden. Ist im Servicefall ein Gerätetausch notwendig, kann bei einem PC-Tausch die aktivierte Lizenz wieder freigegeben und danach neu aktiviert werden. Bei einem Gerätetausch ist eine Rückgabe der Lizenz nicht möglich, die Lizenz kann jedoch auf dem neuen Gerät wiederhergestellt werden. Dies ist bis zu 2 x möglich. Danach muss Weidmüller kontaktiert werden.

## 8.1 Trial-Lizenz

Standardmäßig werden alle Produkte mit einer Trial-Lizenz ausgeliefert. Diese Lizenz ist auf 30 Tage befristet. Innerhalb dieser Zeit können alle Funktionen des Produkts vollständig genutzt werden. Nach Ablauf dieser Zeit ist die Nutzung nicht oder nur mehr eingeschränkt möglich. Zur vollständigen Nutzung wird eine neue, gültige Lizenz benötigt, die bei Weidmüller angefordert werden kann.

Eine Trial-Lizenz kann nur einmal verwendet werden. Auch wenn das Produkt deinstalliert und neu installiert wird, ist eine Nutzung ohne gültige Lizenz nicht mehr möglich.

### 8.1.1 Aktivierung einer Trial-Lizenz (Einzelplatzlizenz)

Bei Verwendung eines Produktes am PC ist keine Aktivierung der Trial-Lizenz notwendig. Nach der Installation des Produktes ist die Trial-Lizenz automatisch aktiviert und das Produkt kann ab dem Installationszeitpunkt 30 Tage verwendet werden.

### 8.1.2 Aktivierung einer Trial-Lizenz (Laufzeitlizenz)

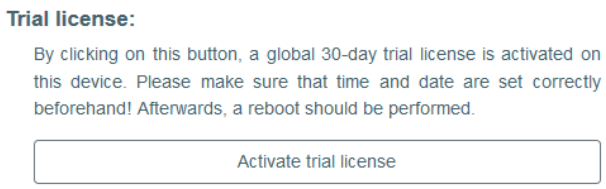
Bei Verwendung eines Geräts muss die Trial-Lizenz manuell aktiviert werden. Andernfalls kann das Produkt nicht oder nur unvollständig genutzt werden. Eine Aktivierung ist nur über den Zugang "DevAdmin" möglich.

***Information***

*Vor der Aktivierung muss das Datum und die Uhrzeit am Gerät korrekt eingestellt sein! Andernfalls kann es sein, dass die Lizenz bei Aktualisierung des Datums sofort ungültig wird.*

Zum Aktivieren der Trial-Lizenz gehen Sie wie folgt vor:

1) Starten von "DevAdmin"
2) Herstellen einer Verbindung zum gewünschten Gerät.
3) Wechseln auf den Karteireiter **Licensing ► Activation**.




4) Klicken auf **Activate trial license**.

Die Trial-Lizenz wurde aktiviert und das Produkt kann 30 Tage vollständig genutzt werden.

## 8.2 Aktivierung Einzelplatzlizenz (Software am PC)

Folgende Möglichkeiten zur Aktivierung einer Lizenz am PC stehen zur Verfügung:

- Online-Aktivierung

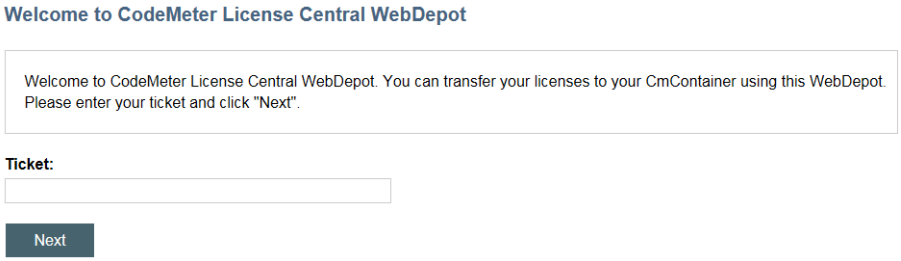
Zum Aktivieren der Lizenz ist das Lizenzverwaltungstool "CodeMeter" der Firma "WIBU Systems" (https://www.wibu.com/de.html) am PC des zu lizenzierenden Tools notwendig. Das Tool wird automatisch beim Setup mitinstalliert. (Sollte das Tool manuell deinstalliert worden sein, kann es auf der Hersteller-Seite wieder heruntergeladen und installiert werden.)

## Online-Aktivierung

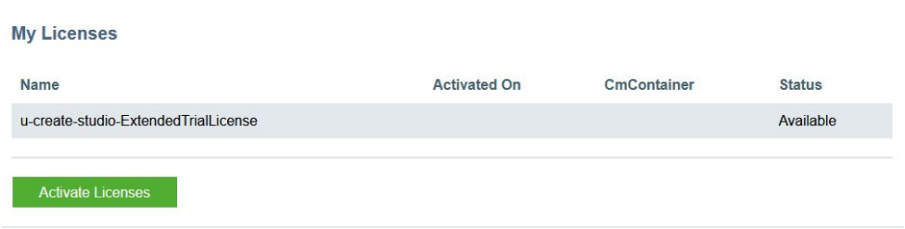
Hat der PC mit dem installierten, zu lizenzierenden Tool eine Internetverbindung, dann ist eine Online-Aktivierung möglich. Die Aktivierung erfolgt auf diesem PC.

Gehen Sie dazu wie folgt vor:

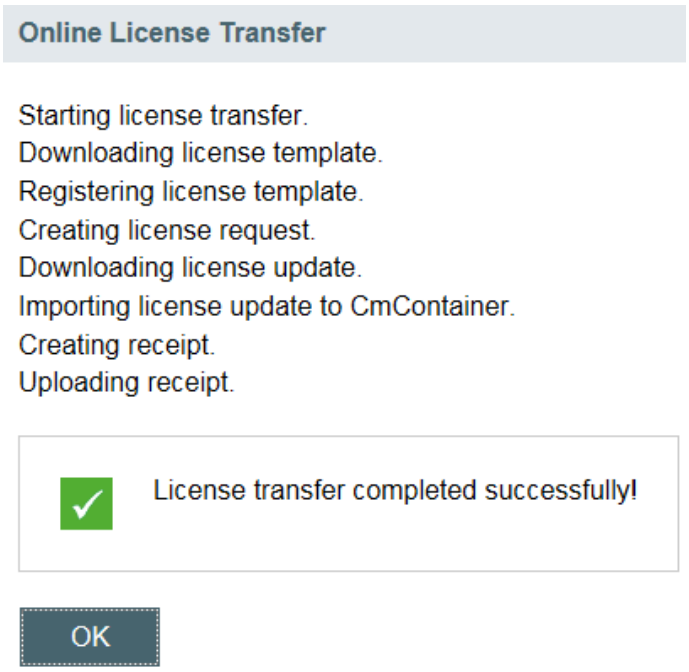
1) Öffnen von "WebDepot" durch Eingabe des Links, der am Lizenz-Ticket angegeben ist, in einem Internet-Browser am PC
2) Eingeben des Ticket-Codes und klicken auf **Next**.




3) Klicken auf **Activate Licenses**.




4) Folgender Dialog öffnet sich.




5) Bestätigen des Dialogs mit OK.

Die Lizenz wurde erfolgreich aktiviert.

## 8.3 Aktivierung Laufzeitlizenz (Software am Gerät)

Folgende Möglichkeiten zur Aktivierung einer Lizenz am Gerät stehen zur Verfügung:

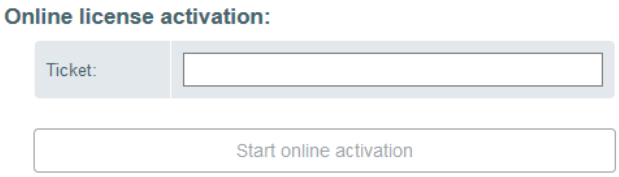
- Online-Aktivierung über "DevAdmin"

### Online-Aktivierung über "DevAdmin"

Für diese Online-Aktivierung wird ein PC mit Internetverbindung benötigt. Das Gerät, auf dem die zu lizenzierende Software installiert ist, benötigt keine Internetverbindung, muss jedoch über eine Netzwerkverbindung mit dem PC verbunden sein.

Zum Aktivieren der Lizenz gehen Sie wie folgt vor:

1) Verbindung herstellen zwischen PC und Gerät mittels **DevAdmin**.
2) Klicken auf **Licensing ► Activation**.




3) Eingeben des Ticketcodes.
4) Klicken auf **Start online activation**.

Die Lizenz wurde am Gerät wurde erfolgreich aktiviert.

***Information***

*Es wird empfohlen, ein Backup des Geräts zu sichern.*

## 8.4 Lizenzstatus und Lizenzübersicht

### Einzelplatzlizenzen

Der Status einer Software-Lizenz kann durch Klicken des Icons in der Taskleiste abgefragt werden. Weiters kann eine Lizenzübersicht mittels Link zum **WebAdmin** aufgerufen werden. Im Karteireiter **Lizenz-Monitor** sind die Lizenz-Anzahl, die Anzahl der belegten und verfügbaren Lizenzen dargestellt. Im Karteireiter **Container** befindet sich eine Lizenz-Übersicht aller Lizenzen und zugehöriger Gültigkeitsdauer.

### Laufzeitlizenzen

Um den Status einer Laufzeitlizenz (Softwarelizenz am Gerät) abzufragen, muss mittels "DevAdmin" auf das Gerät zugegriffen werden. Im Karteireiter **Licensing ► Overview** ist eine Übersicht aller Lizenzen und deren Gültigkeitsdauer sowie der Lizenztyp dargestellt.

## 8.5 Gerätetausch

Ist im Servicefall ein Gerätetausch notwendig, kann bei einem PC-Tausch die aktivierte Lizenz wieder freigegeben und danach neu aktiviert werden. Bei einem Gerätetausch ist eine Rückgabe der Lizenz nicht möglich, die Lizenz kann jedoch auf dem neuen Gerät wiederhergestellt werden. Dies ist bis zu 2 x möglich. Danach muss Weidmüller kontaktiert werden.

### 8.5.1 Tausch PC (Einzelplatzlizenz)

Folgende Möglichkeiten, bei denen eine bereits aktivierte Lizenz auf einen neuen PC übertragen werden kann, stehen zur Verfügung:

- Umziehen der Lizenz (Rehost): Der bestehende PC ist noch funktionstüchtig. Die Lizenz kann umgezogen werden, d.h. auf dem alten/bestehenden PC zurückgegeben und auf dem neuen PC wieder aktiviert werden.
- Reaktivierung der Lizenz (Restore): Der bestehende PC ist nicht mehr funktionstüchtig, eine Rückgabe der Lizenz ist nicht möglich.

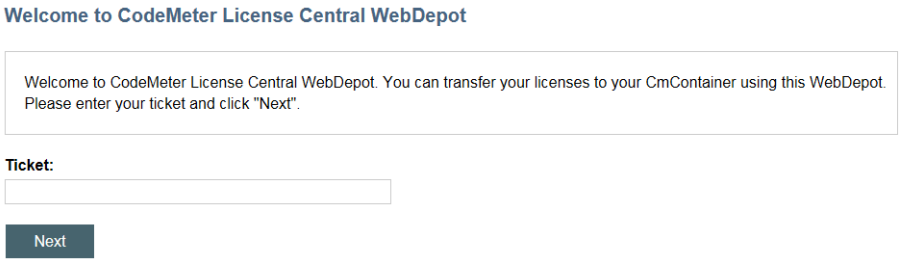
In jedem Fall müssen die PCs über eine Internetverbindung verfügen.

#### Umziehen der Lizenz (Rehost)

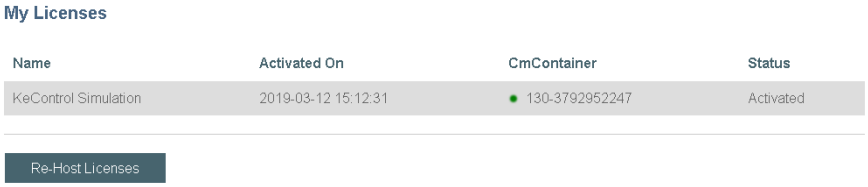
Bei dieser Möglichkeit wird eine Lizenz von einem alten, funktionsfähigen PC auf einen neuen PC übertragen. Beide PCs haben Internetzugang.

Um eine bereits aktivierte Software-Lizenz auf einen anderen PC zu übertragen, gehen Sie wie folgt vor:

1) Öffnen von "WebDepot" durch Eingabe des Links, der am Lizenz-Ticket angegeben ist, in einem Internet-Browser am alten PC.
2) Eingeben des Ticket-Codes und klicken auf **Next**.




3) Klicken auf **Re-Host Licenses**




4) Wenn notwendig, selektieren der Lizenz für die Deaktivierung.
5) Klicken auf **Deactivate Selected Licenses Now**.
6) Bestätigen des Dialogs mit OK.

Die eingegebene Lizenz steht wieder für die Lizenzierung zur Verfügung und kann wie in Kap. "Lizenzierung einer Software am PC" beschrieben verwendet werden.

### Reaktivierung der Lizenz (Restore)

In diesem Fall ist der PC mit aktivierter Lizenz nicht mehr funktionsfähig und eine Rückgabe der Lizenz ist dadurch nicht möglich. Der neue PC verfügt über einen Internetzugang.

Um die Lizenz wieder verwenden zu können kontaktieren Sie bitte zuerst den Weidmüller Service. Der Weidmüller Service autorisiert die erneute Aktivierung der Lizenz.

Um die reaktivierte Lizenz auf einem neuen PC zu aktivieren gehen Sie wie folgt vor:

1) Öffnen von "WebDepot" durch Eingabe des Links, der am Lizenz-Ticket angegeben ist, in einem Internet-Browser.
2) Eingeben des Ticket-Codes und klicken auf **Next**.
3) Klicken auf **Re-Store Licenses**.
4) Akzeptieren der Restore-Bedingungen.
5) Klicken auf **Restore Selected Licenses Now**.

Die reaktivierte Lizenz wurde erfolgreich auf dem PC aktiviert.

***Information***

*Systembedingt wird auch immer der Button **Re-Host Licenses** im Webdepot abgebildet. Durch Klicken auf diesen Button wird eine Fehlermeldung ausgelöst, d.h. **Re-Host Licenses** kann **nicht** zum **Reaktivieren** der Lizenz verwendet werden.*

## 8.5.2 Tausch Gerät (Runtime-Lizenz)

Muss ein Gerät aufgrund eines Defektes getauscht werden, muss auf dem neuen Gerät ein Backup des alten Geräts wiederhergestellt werden. Erst danach kann die bereits aktivierte Lizenz auf das neue Gerät übertragen werden. Dazu steht folgende Möglichkeit zur Verfügung:

- Online-Restore einer Lizenz mittels DevAdmin

Die Lizenz kann insgesamt nur 2 x auf ein neues Gerät übertragen werden.

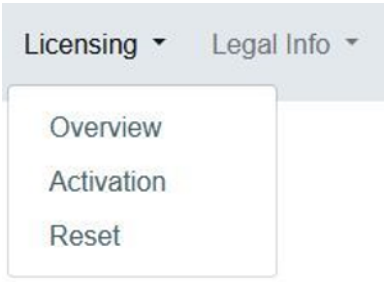
***Information***

*Nur zeitlich unbegrenzte aktivierte Lizenzen können übertragen werden. Befindet sich auf dem neuen Gerät bereits eine aktivierte Lizenz, wird diese vom Restore nicht überschrieben.*

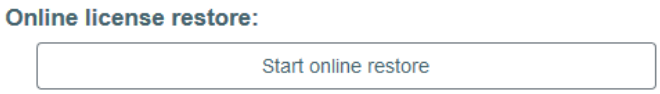
### Online-Restore einer Lizenz mittels DevAdmin

Für den Online-Restore einer Lizenz befindet sich ein PC mit Internetanschluss im Netzwerk des Geräts und ist hochgefahren. Das zu lizenzierende Gerät ist ebenfalls hochgefahren.

1) Verbindung herstellen zwischen PC und Gerät mittels **DevAdmin**.
2) Klicken auf **Licensing ► Activation**.




3) Klicken auf **License ► Online Restore**.
4) Klicken auf **Start license Restore process**.




5) Bestätigen des Dialogs mit **OK**.

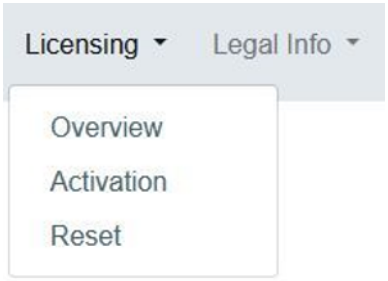
Die eingegebene Lizenz steht wieder für die Lizenzierung zur Verfügung und kann wie in Kap. "Lizenzierung einer Software auf einem Gerät" beschrieben verwendet werden.

## 8.6 Zurücksetzen der Lizenzinformationen

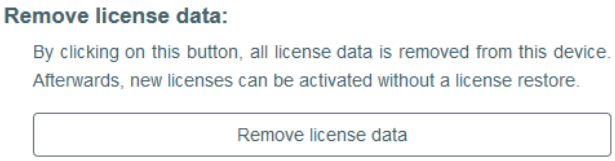
In bestimmten Fällen ist es notwendig alle Lizenzinformationen eines Geräts zurückzusetzen (z.B. beim Fehlschlagen eines Restores). Die vorher am Gerät verwendete Lizenz wird dadurch ungültig und kann nicht mehr reaktiviert werden.

Um alle Lizenzinformationen des Geräts zurückzusetzen, gehen Sie wie folgt vor:

1) Verbindung herstellen zwischen PC und Gerät mittels **DevAdmin**.
2) Klicken auf **Licensing ► Reset**.




3) Klicken auf **Remove license data**




4) Bestätigen des Dialogs mit **OK**.

Die Lizenzinformationen des Geräts wurden zurückgesetzt. Um das Gerät neu zu lizenzieren muss eine neue Lizenz erworben werden. Kontaktieren Sie dazu den Weidmüller Service.

# 9 Software Service

Das Tool Software Service wird automatisch mit u-create installiert. Nach erfolgreicher Installation erscheint im rechten Bereich der Windows-Taskleiste das Icon **Software Service**.

Das Software Service dient zum Ablegen der Komponenten, die auf die Steuerung geladen werden können.

Nähere Informationen zum Software Service kann in der Online-Hilfe nachgelesen werden.

## Systemanforderung

Der Port 1744 muss geöffnet sein. Ab Version 2.5.0 wird dieser Port von der Installation des Software Service automatisch über die Windows Firewall freigegeben. Der Benutzer benötigt dazu die notwendigen Windows-Rechte. Wenn Sie diese Rechte nicht besitzen, wenden Sie sich an den Administrator.

# 10 Device Administration (DevAdmin)

Befindet sich ein PC im Netzwerk der Steuerung, kann darauf durch Öffnen eines Internetbrowsers mit der Adresse `http://<IP-Adresse der Steuerung>` der Login-Dialog für die Weboberfläche **DevAdmin** geöffnet werden.

Mit folgenden Login-Daten wird der Benutzer angemeldet:

Username: Administrator

Password: tobechanged

Der **DevAdmin** besitzt folgende Karteireiter:

- **Diag.**: Informationen zur Steuerung lesen, Status- oder Crashreport auslösen
- **Config.**: Netzwerk-, Zeit- und Zeitzoneneinstellungen durchführen
- **Backup/Restore**: Sicherungskopie erstellen oder einspielen
- **System Overview**: Übersicht über die Geräte im Systemverbund
- **Licensing**: Lizenzierung durchführen, siehe 8 Lizenzierung
- **Legal Info**: Anzeige der (Open-Source-)Lizenztexte aller installierten Softwarepakete
- **Plugins**: Anzeige der installierten Erweiterungen

## 10.1 Karteireiter "Diag."

Dieser Karteireiter ist in folgende Bereiche unterteilt:

- Device Information
- Device State
- Statereport
- Crashreport

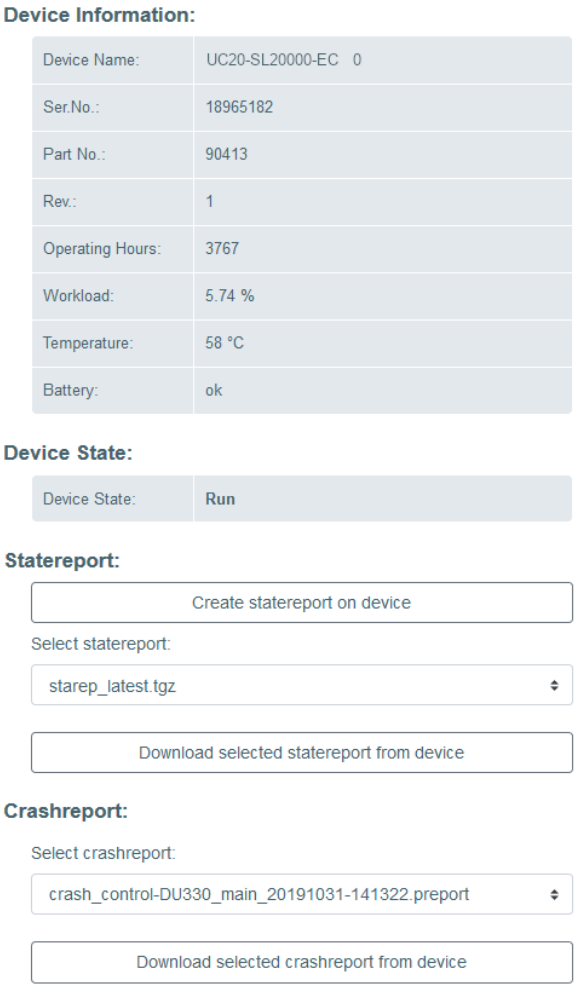



Abb. 10-8: DevAdmin - Diagnostics

## Bereich "Device Information"

Im diesem Bereich werden folgende Informationen über die verwendete Steuerung angezeigt:

| Information | Beschreibung |
|---|---|
| Device Name | Bezeichnung der Steuerung |
| Ser.No. | Seriennummer der Steuerung |
| Part No. | Materialnummer der Steuerung |
| Rev. | Revisionsnummer der Steuerung |
| Operating Hours | Betriebsstunden der Steuerung |
| Workload | Aktuelle Auslastung der Steuerung |
| Temperature | Aktuelle Temperatur der Steuerung |
| Battery | Status der Batterie |

### Bereich "Device State"

Im diesem Bereich werden folgende Informationen über die verwendete Steuerung angezeigt:

| Information | Beschreibung |
|---|---|
| Device State | Status der Steuerung |

### Bereich "Statereport"

In diesem Bereich kann mittels **Create statereport on device** ein Statusreport ausgelöst werden. Dieser wird auf der Steuerung unter `/masterdisk/protocol/statusreport` abgelegt. Der Statusreport enthält auch die Diagramme aus der Langzeitaufzeichnung (Karteireiter "Monitor").

Über das Dropdown-Menü kann ein Statusreport ausgewählt werden, der auf den PC geladen werden soll.

### Bereich "Crashreport"

Über das Dropdown-Menü kann ein Crashreport ausgewählt werden, der auf den PC geladen werden soll.

## 10.2 Karteireiter "Config."

Dieser Karteireiter ist in folgende Bereiche unterteilt:

- Network Hostname
- Interface Ethernet X1
- Time, Date and Timezone
- Software Service
- Software Unit Key

### Network Hostname

Im Bereich "Network Hostname" wird der Name der Steuerung eingegeben.

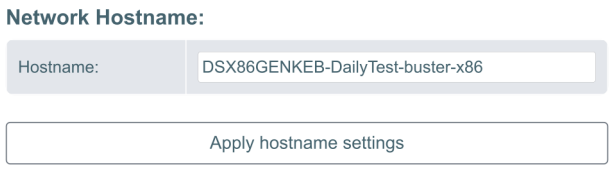



Abb. 10-9: Network Hostname

| Einstellung | Beschreibung |
|---|---|
| Hostname: | Name der Steuerung |
| Apply hostname settings | Der eingestellte Hostname der Steuerung wird übernommen. |

***Information***

*Bei Änderung des Hostnamens kann es zu Verbindungsproblemen der Steuerung kommen.*

## Interface Ethernet X1

In diesem Bereich wird die IP-Adresse der Ethernet-Schnittstelle X1 konfiguriert.

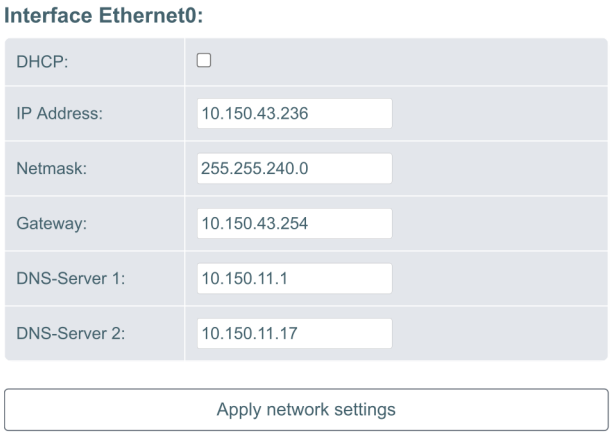



Abb. 10-10: Interface Ethernet X1

| Einstellung | Beschreibung |
|---|---|
| DHCP: | DHCP aktivieren oder deaktivieren |
| IP Address: | IP-Adresse der Steuerung |
| Netmask: | Einstellen der Subnetzmaske |
| Gateway: | Einstellen des Gateway |
| DNS-Server 1: | Einstellen des DNS-Server 1 |
| DNS-Server 2: | Einstellen des DNS-Server 2 |
| Apply network settings | Die Konfiguration der Ethernet-Schnittstelle X1 wird übernommen. |

## Time, Date and Timezone

Im Bereich "Time, Date and Timezone" werden Uhrzeit, Datum und Zeitzone eingestellt.

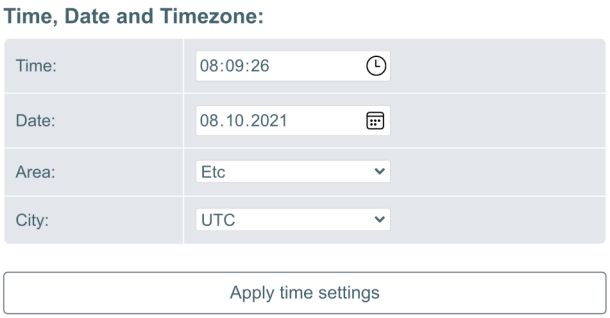



Abb. 10-11: Time, Date and Timezone

| Einstellung | Beschreibung |
|---|---|
| Time: | Uhrzeit einstellen |
| Date: | Datum einstellen |
| Area: | Region auswählen |
| City: | Stadt auswählen |
| Apply time settings | Einstellungen zu Uhrzeit, Datum und Zeitzone werden übernommen |

## Software Service

Im Bereich "Software Service" werden die Verbindungsdaten zum Software Service konfiguriert.

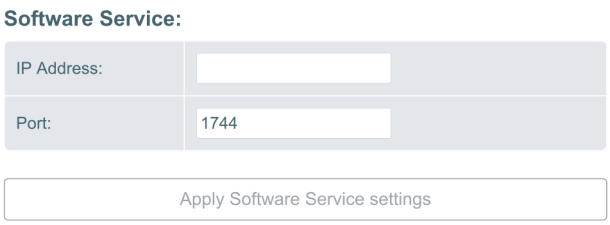



Abb. 10-12: Software Service

| Einstellung | Beschreibung |
|---|---|
| IP Adresse: | IP Adresse des PCs, auf dem das Software Service installiert ist |
| Port: | Netzwerk-Port des Software Service (Standard: 1744) |
| Apply Software Service settings | Verbindungseinstellungen werden übernommen und getestet |

***Information***

*Um die Verbindungseinstellungen zu übernehmen und zu testen, muss das Service am PC gestartet und ein Netzwerkzugriff möglich sein.*

### Software Unit Key

Im Bereich "Software Unit Key" wird der, zum Installieren von eigenen Software Units, benötigte Public Key auf die Steuerung übertragen.

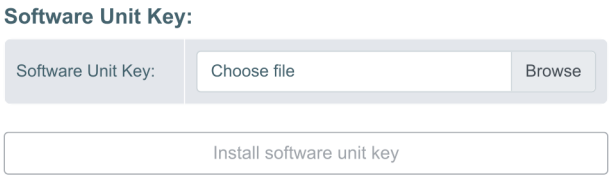



Abb. 10-13: Software Unit Key

| Einstellung | Beschreibung |
|---|---|
| Software Unit Key: | Public Key Datei auswählen |
| Install software unit key | Public Key wird auf der Steuerung installiert |

## 10.3 Karteireiter "Backup/Restore"

Der Karteireiter "Backup/Restore" ist in folgende Bereiche unterteilt:

- Backup
- Restore

### Backup

Im Bereich "Backup" können Sicherungskopien erstellt werden. Je nach Auswahl sind unterschiedliche Felder aktiv.

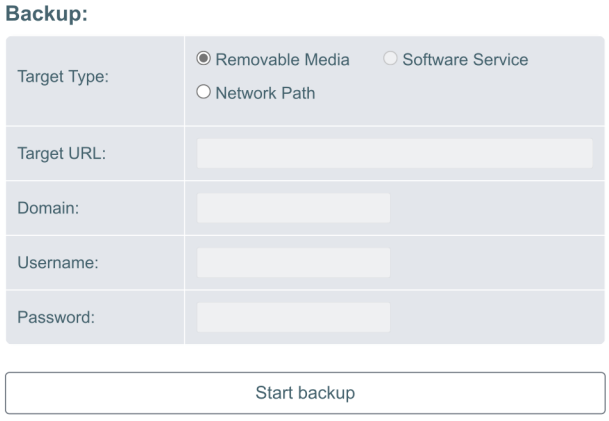



Abb. 10-14: Backup

| Einstellung | Beschreibung |
|---|---|
| Target Type: | • Removable Media: Wechseldatenträger<br>• Software Service (nur auswählbar, wenn das Software Service konfiguriert wurde)<br>• Network Path: Pfad zu einem Ablageort am Netzwerk |
| Target URL: | Pfad zu einem Ablageort im Netzwerk |
| Domain: | Domain (optional) |
| Username: | Benutzername |
| Password: | Passwort |
| Start backup | Sicherungskopie wird erstellt. Es erscheint eine Sicherheitsabfrage. Die Vorbereitung beginnt und das Gerät wird neu gestartet. Es werden keine weiteren Informationen über die Weboberfläche angezeigt. |

## Restore

Im Bereich "Restore" können Sicherungskopien wiederhergestellt werden. Je nach Auswahl sind unterschiedliche Felder aktiv.

***Information***

*Wird **Removable Media** ausgewählt, müssen die Sicherungskopien in einem Verzeichnis mit dem Namen **images** abgelegt werden.*

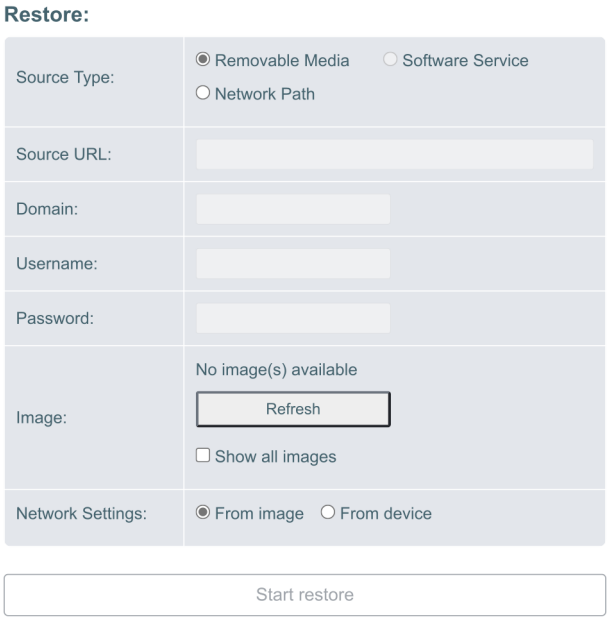



Abb. 10-15: Restore

| Einstellung | Beschreibung |
|---|---|
| Source Type: | • Removable Media: Wechseldatenträger<br>• Software Service (nur auswählbar, wenn das Software Service konfiguriert wurde)<br>• Network Path: Pfad zu einem Ablageort am Netzwerk |
| Source URL: | Pfad der Sicherungskopie am Netzwerk |
| Domain: | Domain (optional) |
| Username: | Benutzername |
| Password: | Passwort |
| Image: | Die gefundenen Sicherungskopien werden automatisch aufgelistet.<br>Refresh: Auswahlbox wird aktualisiert<br>Show all images: alle Sicherungskopien auflisten |
| Network settings: | • From image: Netzwerkeinstellungen werden von der zu wiederherstellenden Sicherungskopie übernommen<br>• From device: Aktuelle Netzwerkeinstellungen des Gerätes werden beibehalten |
| Start restore | Sicherungskopie wird wiederhergestellt. Es erscheint eine Sicherheitsabfrage. Die Vorbereitung beginnt und das Gerät wird neu gestartet. Es werden keine weiteren Informationen über die Weboberfläche angezeigt. |

## 10.4 Karteireiter "Plugins"

Im Karteireiter "Plugins" werden zusätzliche Informationen oder Funktionalitäten (z.B. Hilfe, Einstellungen, ...) zu optional installierten Softwarepaketen (siehe "Software Units") angezeigt. Ebenfalls können eigene Informationen oder Funktionalitäten, z.B. für selbst erstellten Softwarepaketen eingehängt werden.

### 10.4.1 Erstellen von eigenen Inhalten

Es gibt folgende Möglichkeiten, eigene Inhalte unter den Karteireiter Plugins einzubinden:

- Inhalte als HTML Dateien
- Inhalte, die auf einem eigenen Server gehostet werden

Die Dateien, die unter dem Karteireiter "Plugins" angezeigt werden sollen, müssen auf der Steuerung unter dem Verzeichnis `/opt/devadmin/plugins/<name>` abgelegt werden. Darin muss sich immer eine Konfigurationsdatei mit dem Namen `config.json` befinden. In dieser werden notwendige Konfigurationen angegeben.

***Information***

*Ein Datenaustausch zwischen DevAdmin und den eigenen Inhalten ist nicht möglich.*

## Aufbau der Konfigurationsdatei

Die Datei `config.json` enthält folgende Konfigurationseinträge.

| Eintrag | Typ / Wert | Beschreibung |
|---|---|---|
| `pluginType` | `iFrame` | Plugin-Typ |
| `iFrame.type` | `file` oder `link` | Art der Integration (`file`: HTML Dateien, `link`: Server) |
| `iFrame.path` | STRING | Bei `file`: Relative Pfadangabe von `index.html` ausgehend vom Basisverzeichnis |
| `iFrame.url` | STRING | Bei `link`: URL des Servers |
| `iFrame.port` | 0 - 65535 | Bei `link`: Port des Servers |
| `fullScreen` | BOOL | Noch nicht verfügbar |
| `displayName` | STRING | Anzeigename in der Dropdown-Liste unter "Plugins" |

## Inhalte als HTML Dateien

Die Startdatei muss den Namen `index.html` haben und in der Konfigurationsdatei angegeben werden. Aufbau der Konfigurationsdatei `config.json`:

```
{
    "pluginType": "iFrame",
    "iFrame": {
        "type": "file",
        "path": "html/index.html"
    },
    "fullScreen": false,
    "displayName": "Mein neues Plugin"
}
```

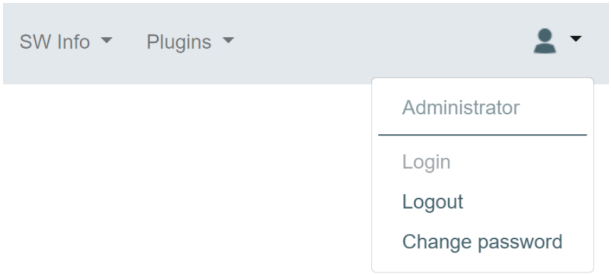
## Inhalte, die auf einem eigenen Server gehostet werden

Aufbau der Konfigurationsdatei `config.json`:

```
{
    "pluginType": "iFrame",
    "iFrame": {
        "type": "link",
        "url": "https://myserver.com/",
        "port": 1880
    },
    "fullScreen": false,
    "displayName": "Server Plugin"
}
```

## 10.5 Passwort für DevAdmin ändern

Rechts oben im **DevAdmin** befindet sich das Benutzermenü mit dem **Change password** Eintrag.




Wird dieser angeklickt, öffnet sich das Fenster, in dem das Passwort für den **DevAdmin** geändert werden kann.

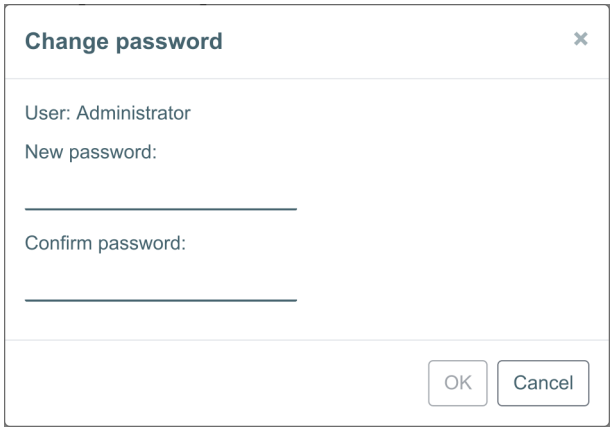



Abb. 10-16: Change password

| Einstellung | Beschreibung |
|---|---|
| User: | Benutzername |
| New password: | Neues Passwort eingeben |
| Confirm password: | Neues Passwort bestätigen |
| OK | Passwort wird übernommen |
| Cancel | Vorgang wird abgebrochen |

# 11 OPC UA Server

Der OPC UA Server kann als optionales Softwarepaket (siehe "Software Units") installiert werden. Dieses Kapitel beschreibt die Verwendung des OPC UA Servers auf Weidmüller Steuerungen mit Betriebssystem Linux.

OPC Unified Architecture (OPC UA) ist ein Standard zur herstellerunabhängigen Vernetzung von Geräten im Automatisierungsbereich. Damit stehen standardisierte Zugriffsmöglichkeiten auf die Steuerung zur Verfügung. Durch diesen Standard kann jedes Programm (z.B. ein OPC UA Client), das diesen Standard unterstützt, mit der Weidmüller Steuerung kommunizieren und auf Daten zugreifen. Allgemeine Informationen dazu siehe: www.opc-foundation.org

## Überblick

Der OPC UA Server läuft neben den Applikationen auf der Steuerung und nimmt OPC UA Client Verbindungen entgegen.

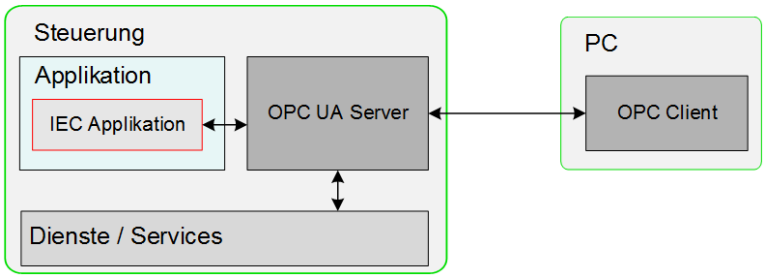



Abb. 11-17: Überblick

## Installation

Der OPC UA Server kann als optionales Paket über einen Target Wechseldatenträger auf der Steuerung installiert werden. Im Auswahldialog kann festgelegt werden, ob der OPC UA Server auch unverschlüsselte Verbindungen akzeptiert.

Weitere Informationen dazu siehe u-create studio Onlinehilfe.

# 11.1 Übersicht Varianten

Die folgende Tabelle gibt einen Überblick über unterstützten Funktionalitäten.

| Funktionalität | OPC UA Server |
|---|---|
| Sicherer Verbindungsaufbau (SecureChannel): | x |
| Anzahl gleichzeitiger Client Verbindungen (Sessions): | 5 |
| Benutzerauthentifizierung: | x |
| Services laut Tabelle unterhalb: | x |

| Funktionalität | OPC UA Server |
|---|---|
| Generischer Variablenbaum: | x |
| Erstellen eines Informationsmodells: | - |
| Unterstützung von kundenspezifischen Events und Methoden: | - |

*) Diese Funktionalitäten werden zur Zeit noch nicht unterstützt.

Die folgende Tabelle gibt einen Überlick über die vom OPC UA Server unterstützen Dienste (Service Sets).

| Service Sets laut OPC UA Spezifikation Part 4 | Unterstützte Dienste |
|---|---|
| Discovery | vollständig |
| SecureChannel | vollständig |
| Session | vollständig |
| Nodemanagement | nicht unterstützt |
| View | • Browse<br>• BrowseNext<br>• TranslateBrowsePathsToNodeIds |
| Query | nicht unterstützt |
| Attribute | • Read<br>• Write |
| Method | nicht unterstützt |
| MonitoredItem | • CreateMonitoredItems<br>• ModifyMonitoredItems<br>• SetMonitoringMode<br>• DeleteMonitoredItems |
| Subscription | • CreateSubscription<br>• ModifySubscription<br>• SetPublishingMode<br>• Publish<br>• Republish<br>• DeleteSubscriptions |

## 11.2 Verbindungsaufbau

Der OPC UA Server unterstützt einen Verbindungsaufbau von OPC UA Standard konformen Clients mittels TCP-IP Binärprotokoll. Die IP-Adresse der Steuerung ist gleichzeitig die IP-Adresse des OPC UA Servers. Der Port ist standardmäßig 4842. Applikationsspezifische Einstellungen wie Port, Applikationsname, Hersteller etc. können über die Server Konfiguration (in u-create studio) eingestellt werden.

Beispiel eines Verbindungsaufrufs:

```
opc.tcp://ServerUrl:Port
```

Im Internet stehen freie sowie lizenzpflichtige OPC UA Clients zur Verfügung, mit denen die Funktionalität des OPC UA Servers getestet werden kann. Die vom Discovery-Dienst zurückgelieferten Endpunkt-URLs des OPC UA Servers enthalten immer den Hostnamen der Steuerung. Der OPC UA Client muss den Hostnamen beim Verbinden auf die Server-IP auflösen können (z.B. mittels Host-Datei oder DNS).

***Information***

*Der Hostname darf nie mit einer Ziffer beginnnen, sonst kann es zu Problemen beim Server-Hochlauf oder beim Verbindungsaufbau kommen.*

Falls der Discovery-Dienst einen Fehlercode zurück gibt, besteht entweder ein Netzwerkproblem oder der Server konnte nicht korrekt gestartet werden. Genauere Fehlerquellen können der Server-Protokollierung entnommen werden, siehe 11.6 Protokollierung des Serverbetriebs.

Falls der OPC UA Server keine bzw. abgelaufene Lizenzen vorfindet, wird beim Verbindungsaufbau (`ActivateSessionRequest`) der OPC UA Fehler-Code `BadLicenseNotAvailable` zurückgegeben. Ein Verbindungsaufbau ist nicht möglich.

Der OPC UA Server unterstützt folgende Verbindungsarten:

- Unverschlüsselte Verbindung
- Verschlüsselte Verbindung mittels Zertifikat

## Unverschlüsselte Verbindung

Wurde zum Installationszeitpunkt konfiguriert, dass unverschlüsselte Verbindungen erlaubt sind, kann auf den OPC UA Server mit dem OPC UA Client über eine unverschlüsselte Verbindung zugegriffen werden. Die Anmeldung kann anonym oder mittels Benutzername und Passwort erfolgen. Ein anonymer Client hat vollen Zugriff auf alle Knoten und Netzwerkdaten.

In der Benutzerverwaltung wird standardmäßig der Benutzer "Administrator" mit dem Passwort "tobechanged" angelegt. Der OPC UA Server akzeptiert alle in der Benutzerverwaltung angelegten Benutzer.

***Information***

*Von der Verwendung einer unverschlüsselten Verbindungen im Betrieb wird abgeraten, da dies aufgrund möglicher unerlaubter Zugriffe ein großes Sicherheitsrisiko darstellt. Die unverschlüsselte Verbindung sollte daher nur zu Entwicklungszwecken verwendet werden.*

## Verschlüsselte Verbindung

Bei einer verschlüsselten Verbindung müssen sich sowohl OPC UA Server als auch OPC UA Client gegenseitig authentifizieren. Dies geschieht über ein "Public-Key"-Verfahren unter Verwendung von Zertifikaten. Als Ver-

schlüsselungsalgorithmus wird "Basic256Sha256" (Sign, Sign & Encrypt) verwendet. (Weiterführende Literatur über den Zertifikatsmechanismus siehe z.B. OPC Foundation Website.)

Folgende Verzeichnisse werden vom OPC UA Server automatisch auf der Steuerung im Verzeichnis `/opt/kecontrolapplication/OpcUa/PKI/CA` angelegt:

| Verzeichnis | Beschreibung |
|---|---|
| `/own` | Enthält den geheimen Server-Schlüssel und das Server-Zertifikat. |
| `/trusted/certs` | Enthält alle vertrauenswürdigen Client-Zertifkate. |
| `/trusted/crl` | Enthält alle widerrufenen Client-Zertifkate (Certificate Revocation List). Diese Clients können sich nicht mehr verbinden. |
| `/issuers/certs` | Enthält alle vertrauenswürdigen Zeritifkate, die zur Verifikation verwendet werden. |
| `/issuers/crl` | Enthält alle widerrufenen Zeritifkate (Certificate Revocation List), die zur Verifikation verwendet werden. |
| `/rejected` | Enthält alle vom Server zurückgewiesene Client-Zertifikate. Diese Clients können sich nicht verbinden. |

Der OPC UA Server erzeugt beim Hochfahren ein Server-Zertifikat (`uaserverc.der`). Anstelle des automatisch generierten Server-Zertifikats kann auch ein eigenes erzeugtes Zertifikat verwenden werden, das im Verzeichnis `/opt/kecontrolapplication/OpcUa/PKI/CA/own` abgelegt werden muss. Dieses Zertifikat muss jedem OPC UA Client, der sich beim OPC UA Server authentifizieren möchte, bekannt gemacht werden.

Nur Verbindungsanfragen von OPC UA Clients, deren Zertifikat im Verzeichnis `/trusted/certs` abgelegt ist, werden akzeptiert. Verbindet sich ein Client zum Server, dessen Zertifikat nicht bekannt ist, wird der Verbindungsaufbau abgelehnt und das abgewiesene Zertifikat im Verzeichnis `/rejected` abgelegt.

Nachdem sich Client und Server erfolgreich authentifziert haben, ist zusätzlich eine Benutzerauthentifzierung mit Benutzername und Passwort zwingend notwendig.

## 11.3 Server Konfiguration

In diesem Kapitel werden die Konfigurationen des OPC UA Servers beschrieben. Diese können über "Experteneinträgen" im u-create studio vorgenommen werden. Alle OPC UA Konfigurationen müssen im Bereich `[OpcUa]` angeführt werden. Konfigurationsänderungen werden erst nach einem Neustart der Steuerung wirksam.

### Applikationsspezifische Konfigurationen

Folgende applikationsspezifische Einstellungen können konfiguriert werden:

| Bezeichnung | Beschreibung |
|---|---|
| Port | Server-Port |
| ApplicationName | Applikationsname |
| ApplicationURI | Applikations-URI |
| ProductURI | Produkt-URI |
| ProductName | Produktname |
| ManufacturerName | Herstellername |
| ShowVariableTree | Anzeige generischer Variablenbaum (1 = aktiviert, 0 = deaktiviert) |
| GenericVariablesNodeName | Name des generischen Variablenbaums (Anzeige des Names in den Attributen "DisplayName", "BrowseName", "Description") |
| GenericEventsNodeName | Nur bei unverschlüsselter Verbindung sichtbar, interner Parameter |
| GenericFunctionNodeName | Nur bei unverschlüsselter Verbindung sichtbar, interner Parameter |

### Beispiel

```
[OpcUa]
[OpcUa.Application]
Port=4840
ApplicationName="SampleApplicationName"
ApplicationURI="SampleApplicationURI"
ProductURI="http://www.company.com/"
ProductName="OPC-UA Server"
ManufacturerName="Sample Manufacturer"
ShowVariableTree = 1 // 1 = activate, 0 = deactivate
GenericVariablesNodeName = "SampleNode"
```

## Konfigurationen "SecureChannel"

Bei einer verschlüsselten Verbindung wird der Verschlüsselungsalgorithmus Basic256Sha256 mit 2 verschiedenen Einstellungsmöglichkeiten (`Sign` und `Sign and Encrypt`) unterstützt. Alle Konfigurationswerte können über Integer Werte konfiguriert werden.

| Security Policy | Wert |
|---|---|
| Basic256Sha256 | 3 |

| Message Mode | Wert |
|---|---|
| Sign | 1 |
| Sign and Encrypt | 2 |

***Information***

*Die Verschlüsselungsalgorithmen "Basic256" und "Basic128Rsa15" werden nicht mehr unterstützt, da sie nicht mehr den aktuellen Sicherheitsanforderungen entsprechen.*

### Beispiel (standardmässige Einstellung)

```
[OpcUa]
// Policy 3 = Basic256Sha256
// MessageMode 1 = Sign, 2 = SignAndEncrypt
[OpcUa.Security.CommunicationChannel.Communication:0]
Policy=3
MessageMode=1
[OpcUa.Security.CommunicationChannel.Communication:1]
Policy=3
MessageMode=2
```

Es können weitere Kommunikationskanäle (`CommunicationChannel`) mit unterschiedlichen Sicherheitseinstellungen hinzugefügt werden. Jede weitere Verbindung wird mit aufsteigender Nummer deklariert. Die Nummer muss eindeutig sein. Für jede Verbindung müssen immer die Einträge `Policy` und `MessageMode` gesetzt werden.

## Konfiguration "Subscription"

Für den "Subscription"-Service können die unterstützten Zeitintervalle konfiguiert werden. Standardmäßig ist 50 ms für die minimale und 3 600 000 ms für die maximale Grenze eingestellt.

Mögliche Konfigurationseinträge:

| Name | Beschreibung |
|---|---|
| MinPublishingInterval | Minimalstes Zeitintervall in Millisekunden, Datentyp Double |
| MaxPublishingInterval | Maximalstes Zeitintervall in Millisekunden, Datentyp Double |

***Information***

*Das minimale Zeitintervall sollte nicht zu klein gewählt werden, da bei mehrerern Clients der OPC UA Server durch die Beantwortung dieser Anfragen eine hohe Auslastung erreichen kann.*

Beim Senden einer Anfrage (`CreateSubscription`) übergibt der Client ein gewünschtes Zeit-Intervall. Der Server überprüft, ob das geforderte Intervall innerhalb der konfigurierten Grenzen liegt. Ist dies nicht der Fall, wird der eingestellte Grenzwert vom Server zurückgegeben.

### Beispiel

```
[OpcUa.Subscription.Publishing]
MinPublishingInterval=100.50 // double
MaxPublishingInterval=10000.50 // double
```

Der Client fordert ein Intervall von 70 ms. Der Server antwortet, dass nur ein Zeitintervall von 100,5 möglich ist.

### Aufzeichnen von Werten (Samples)

Standardmäßig sind Aufzeichnungs-Intervalle von 50 ms, 100 ms, 250 ms, 500 ms, 1000 ms, 2500 ms und 5000 ms eingestellt. Die Intervalle können über die Konfiguration verändert werden, der Datentyp ist Integer:

```
[OpcUa.Subscription.Publishing]
[OpcUa.Subscription.Sampling.Intervals.Interval:0]
Interval=100 // int32 - ms
[OpcUa.Subscription.Sampling.Intervals.Interval:1]
```

```
Interval=20 // int32 - ms
[OpcUa.Subscription.Sampling.Intervals.Interval:2]
Interval=500 // int32 - ms
```

Es können mehrere Aufzeichnungs-Intervalle definiert werden, indem die Intervall Nummer erhöht wird. Jede Intervall Nummer muss eindeutig sein.

Beim Senden einer Anfrage (`CreateMonitoredItem`) übergibt der Client ein gewünschtes Aufzeichnungs-Intervall. Der Server überprüft, ob das geforderte Intervall innerhalb der konfigurierten Grenzen liegt. Ist dies nicht der Fall, wird der nähestmögliche Wert vom Server zurückgegeben.

**Beispiel 1 (Standardkonfiguration)**

Standardmäßige Aufzeichnungs-Intervalle: 50 ms, 100 ms, 250 ms, 500 ms, 1000 ms, 2500 ms und 5000 ms

Der Client fordert ein Intervall von 70 ms. Der Server antwortet, dass das überwachte Objekt (`MonitoredItem`) in 50 ms Zeitabständen aufgezeichnet wird.

**Beispiel 2 (Beispielkonfiguration)**

Konfigurierte Aufzeichnungs-Intervalle: 20 ms, 100 ms, 500 ms

Der Client fordert ein Sampling Intervall von 70 ms. Der Server antwortet, dass das das überwachte Objekt (`MonitoredItem`) in 100 ms Zeitabständen aufgezeichnet wird.

## 11.4 Generischer Variablenbaum

Der generische Variablenbaum dient zum Anzeigen der Systemvariablen der IEC-Applikation und kann über den Experteneintrag `ShowVariableTree` (im u-create studio) aktiviert bzw. deaktiviert werden. Standardmäßig ist der generische Variablenbaum deaktiviert.

```
[OpcUa]
[OpcUa.Application]
ShowVariableTree = 1 // 1 = activate, 0 = deactivate
```

Nach der Aktivierung des generischen Variablenbaums und einem Neustart der Steuerung befindet sich der generische Variablenbaum im Verzeichnis `Root/Objects` (NodeId: Namensraum URI = "KeControlOpcUa/KeControlVar", Numerische ID = 0, DisplayName = "KeControl Variables").

Der generische Variablenbaum ist ein Abbild des u-create studio Variablen Browsers. Die Verzeichnisstruktur besteht aus den Wurzelknoten APPL, in dem sich die freigegebenen IEC Variablen befinden, und SYS, welcher die Daten der Systemkonfiguration enthält. Mit Ausnahme von SYS.CAT sind alle Daten des Variablen Browsers auch im OPC UA Server sichtbar. Die Applikations- und Konfigurationsdaten werden mit Hilfe von Objekt-Knoten (`FolderType`) und Variablen-Knoten dargestellt. Die Knoten-ID der Knoten ist immer vom Typ String und wird aus aus dem Pfad der Daten aufgebaut. Alle Knoten werden im Namensraum "KeControlOpcUa/KeControlVar" angelegt.

**Beispiel**

Die boolsche Applikationsvariable `VarBool` unter APPL.system hat die Knoten-ID "APPL.system.VarBool" und den OPC UA Datentyp Boolean.

Arrays und Strukturen werden im generischen Variablenbaum mit Hilfe von Verzeichnissen und einzelnen Variablenknoten für die Elemente aufgebaut. Alle Variablenknoten bieten grundsätzlich Lese- und Schreibzugriffe an (`Attribut AccessLevel`). Ob Lese- und Schreiboperationen funktionieren, hängt von der darunterliegenden Datenquelle ab. Freigegebene Systemvariablen der IEC im Verzeichnis APPL unterstützen Lese- und Schreibzugriffe. Bestimmte Systemkonfigurationen können jedoch nicht überschrieben werden. Bei einem unerlaubten Schreibzugriff wird ein entsprechender Fehlercode zurückgegeben.

## 11.5 Erstellen eines Informationsmodells

Der OPC UA Server unterstützt die im Standard spezifizierte Möglichkeit zur Erstellung eines Informationsmodells über eine XML Datei. Das bedeutet, dass der OPC UA Standard Adressraum um ein kundenspezifisches Informationsmodell erweitert werden kann. Ebenfalls kann damit die Struktur, in der die Daten angezeigt werden sollen, definiert werden.

Alle OPC UA Informationen (Variablen, Datentypen, Methoden, ...) werden als Knoten dargestellt. Jeder Knoten ist einem Namensraum zugewiesen. Mit Hilfe von Namensraum und Knoten-ID erfolgt eine eindeutige Identifizierung. In der XML Datei können eigene Namensräume, Objekt- und Variableninstanzen definiert werden. Methoden, Typen (Referenz-, Objekt-, Variablen-, Datentypen) oder Ansichten werden nicht unterstützt.

Die Informationen müssen in genau einer XML Datei definiert werden und diese Datei muss auf der Steuerung im Verzeichnis `/appldisk/application/OpcUa/` abgelegt werden. In diesem Verzeichnis dürfen sich keine weiteren XML Dateien befinden. Beim Hochlauf des Servers wird die XML Datei ausgelesen und das definierte Informationsmodell angelegt. Anpassungen in der XML Datei werden erst nach einem Neustart der Steuerung wirksam.

Ist die XML Datei fehlerhaft (z.B. falsche Syntax, falsche Datentypzuordnung) startet der OPC UA Server nicht. Clients können sich nicht verbinden. Die Fehlerursache kann im System- Trace genauer untersucht werden. Siehe Kapitel 11.6 Protokollierung des Serverbetriebs.

### Namensräume

Ein Namensraum ist durch seine URI eindeutig identifiziert. Diese URI entspricht einem numerischen Index, der während des Hochlaufs vom OPC UA Server vergeben wird. Der Namensraum `"http://opcfoundation.org/UA/" (Index = 0)` wird bereits vom OPC UA Standard vorgegeben und enthält vordefinierte Datentypen und Knoten für eine OPC UA Serverdiagnose.

Die Beschreibung von Namensräumen erfolgt innerhalb des XML-Tags `<NamespaceUris>`. Darin können mehrer Namensräume mittels `<Uri>` angelegt werden. Folgende Attribute sind zulässig:

| Attribut | Beschreibung |
|---|---|
| `ns` | Mit diesem Attribut wird der numerische Index der Namensraum URI vergeben, der innerhalb der XML Datei verwendet wird. Nur der vom OPC UA Standard vordefinierte Namensraum 0, ist ohne explizite XML Konfiguration verwendbar.<br>Der numerische Index wird vom OPC UA Server zur Laufzeit neu vergeben. Dieser Index kann über das Objekt "NamespaceArray" ausgelesen werden. |
| `isDefaultNs` | Mit diesem Attribut kann innerhalb der Konfigurationsdatei ein Namensraum als Standardwert angegeben werden. Bei der Beschreibung von Instanzen muss dieser Namensraum nicht mehr explizit bei der Definition der NodeId angegeben werden. Es kann nur genau einen Standardnamensraum geben. |

In folgendem Beispiel werden zwei Namensräume angelegt:

```
<NamespaceUris>
    <Uri ns="1" isDefaultNs="1">www.company.com</Uri>
    <Uri ns="2">MyNamespace</Uri>
</NamespaceUris>
```

## Objekt- und Variableninstanzen

Um Variablen im OPC UA Client anzuzeigen, müssen Instanzen angelegt und mit den gewünschten Variablen der Steuerung verknüpft werden.

***Information***

*Zur Zeit werden nicht alle im OPC UA Standard definierte Datentypen unterstützt.*

Nachfolgende Tabelle beschreibt die unterstützten OPC UA Datentypen mit den jeweils entsprechenden IEC Datentypen. Für jeden Datentyp ist der eindeutige Knotenpfad angegeben, der beim Anlegen in der XML Datei verwendet werden muss. Der Knotenpfad besteht aus mehreren sogenannten "BrowseNames". BrowseName ist ein OPC UA Konzept, mit dessen Hilfe Knotenpfade angegeben werden können. Ein BrowseName besteht aus einem Namensraumindex und einem Namen. In der XML Definition werden diese durch einen Doppelpunkt getrennt. Die BrowseNames werden im XML mit einem Punkt getrennt zu einem Knotenpfad zusammengebaut. Die Knotenpfade für die Standard Datentypen sind vom OPC UA Standard vorgegeben und können mit einem Client aus dem Adressraum ausgelesen werden.

Der Knotenpfad für eine boolsche Variable ist zum Beispiel folgendermaßen definiert: `0:Types.0:DataTypes.0:BaseDataType.0:Boolean`. In nachfolgender Tabelle werden die Knotenpfade gekürzt dargestellt. Bei jedem Knotenpfad muss in der XML Datei folgende Zeichenkette vorne angefügt werden: `0:Types.0:DataTypes.0:BaseDataType.`

| OPC UA Datentyp | IEC Datentyp | Knotenpfad |
|---|---|---|
| Boolean | BOOL | `0:Boolean` |

| OPC UA Datentyp | IEC Datentyp | Knotenpfad |
|---|---|---|
| SByte | SINT | `0:Number.0:Integer.0:SByte` |
| Byte | BYTE, USINT | `0:Number.0:UInteger.0:Byte` |
| Int16 | INT | `0:Number.0:Integer.0:Int16` |
| UInt16 | WORD, UINT | `0:Number.0:UInteger.0:UInt16` |
| Int32 | DINT | `0:Number.0:Integer.0:Int32` |
| UInt32 | DWORD, UDINT | `0:Number.0:UInteger.0:UInt32` |
| Int64 | LINT | `0:Number.0:Integer.0:Int64` |
| UInt64 | LWORD, ULINT | `0:Number.0:UInteger.0:UInt64` |
| Float | REAL | `0:Number.0:Float` |
| Double | LREAL | `0:Number.0:Double` |
| String | STRING, WSTRING | `0:String` |
| DateTime | DATE_AND_TIME, DATE, TIME_OF_DAY | `0:DateTime` |

***Information***

*Die gemappten Datentypen zwischen OPC UA und IEC müssen übereinstimmen, ansonsten wird kein Knoten erzeugt.*

***Information***

*Die Instanzierung von Strukturvariablen oder komplexeren (Basis-)Datentypen wird nicht unterstützt.*

Für den Aufbau eines hierarchischen Modells können die OPC UA Datentypen `BaseObjectType (0:Types.0:ObjectTypes.0:BaseObjectType)` und `FolderType (0:Types.0:ObjectTypes.0:BaseObjectType.0:FolderType)` verwendet werden.

Alle Instanzen werden unter einem `Instances` Eintrag gesammelt. Dieser Eintrag benötigt ein Attribut `Parent`, dass den Wurzelknoten der nachfolgenden Instanzen als Knotenpfad definiert. Der Eintrag `Instances` kann mehrmals hintereinander definiert werden. Der Wurzelknoten muss existieren, sonst können keine Kinderknoten erzeugt werden.

Nachfolgendes Beispiel zeigt die Instanzierung von zwei Verzeichnissen innerhalb vom OPC UA Standard vordefinierten Knoten `Objects`. Die Knotenhierarchie wird durch die Hierarchie der XML Einträge bestimmt.

```
<Instances Parent="0:Objects">
    <Object
        DataType="0:Types.0:ObjectTypes.0:BaseObjectType.0:FolderType"
        NodeId="1:i=1000" BrowseName="Folder" DisplayName="Folder">

        <Object
            DataType="0:Types.0:ObjectTypes.0:BaseObjectType.0:FolderType"
            NodeId="1:i=1001" BrowseName="Subfolder" DisplayName="Subfolder">
```

```
            <!-- define further objects or variables .... -->

        </Object>
    </Object>
</Instances>
```

Für die Objektinstanzierung wird der XML Eintrag `Object`, für Variablen der XML Eintrag `Variable` verwendet. Folgende Attribute müssen für beide Tags angegeben werden:

- DataType
- NodeId
- BrowseName
- DisplayName

Die Beschreibung der Knoten-ID (NodeId) setzt sich immer aus einem Namensraumindex und der Angabe eines Identifizieres zusammen. Die Identifizierer müssen numerisch (i=xx) angegeben werden. Variablen benötigen zusätzlich einen Wert und optional ein Zugriffsrecht (Attribut `AccessLevel`). Der Wert der Variable kann entweder durch eine Konstante, definiert mit dem Attribut `ValueConstant` oder durch den Pfad einer Systemvariable, definiert durch das Attribut `ValuePath`, gesetzt werden.

Mögliche XML Attribute:

| Attribut | Beschreibung |
|---|---|
| `DataType` | Knotenpfad des gewünschten Datentyps für die Instanz. Der Knotenpfad der Typen kann gegebenenfalls mit Hilfe eines OPC UA Clients aus dem Adressraum ausgelesen werden. |
| `NodeId` | Legt die Knoten-ID der aktuellen Instanz fest, diese muss eindeutig sein. Diese Server-Version unterstützt ausschließlich numerische Knoten-ID. |
| `BrowseName` | Intern verwendeter Name für Knotenpfade. |
| `DisplayName` | Angezeigter Name beim OPC UA Client. |
| `AccessLevel` (nur Variablen) | Definiert die Zugriffsrechte für das Attribut `Value` einer Variable. Standardmäßig ist kein Zugriff erlaubt. Das Attribut wird als Hexadezimalzahl angegeben und als `AccessLevelType` - definiert in der OPC UA Spezifikation 3 - interpretiert. Beispiel: "0x01" = CurrentRead, "0x03" = CurrentRead und CurrentWrite. |
| `ValueConstant` (nur Variablen) | Zuweisung einer Konstante. |
| `ValuePath` (nur Variablen) | Pfad zu einer Systemvariable. Der Pfad kann nach erfolgreichem Login im u-create studio im Variablen Browser ausgelesen werden. |

In folgendem Beispiel wird eine OPC UA Variable vom Typ `Boolean` im Server-Adressraum instanziert und mit einer Variable auf der Steuerung verknüpft. Bei der Definition der Knoten-ID wurde im Beispiel auf die Angabe des Namensraumindex verzichtet. Damit wird der Standardnamensraum verwendet (falls im Eintrag `NamspaceUris` definiert).

```
<Variable DataType="0:Types.0:DataTypes.0:BaseDataType.0:Boolean"
    NodeId="i=1002" BrowseName="MyBool" DisplayName="MyBool"
    ValuePath="APPL.system.vBOOL" AccessLevel="0x01"/>
```

Anstelle eines Variablenpfads können auch Konstanten, die in der XML Datei definiert sind, verknüpft werden. Dabei werden alle oben angegebenen Datentypen, außer `DateTime`, unterstützt.

In folgendem Beispiel wird eine OPC UA Variable vom Typ `Boolean` instanziert und der Wert mit einer Konstanten verknüpft:

```
<Variable DataType="0:Types.0:DataTypes.0:BaseDataType.0:Boolean"
    NodeId="i=1003" BrowseName="MyBool" DisplayName="MyBool"
    ValuePath="true" AccessLevel="0x03"/>
```

### Arrays

***Information***

*Der OPC UA Server unterstützt nur eindimensionale Arrays mit Basisdatentypen und Variablenmapping (`Attribut ValuePath`).*

Für die Instanzierung von Arrays wird ein weiterer Eintrag `ArrayDimension` und folgendes zusätzliche Attribut benötigt:

| Attribut | Beschreibung |
|---|---|
| `ValueRank="1"` | Definiert die Anzahl der Array-Dimensionen. Es wird nur der Wert "1" unterstützt. |

In folgendem Beispiel wird das gesamte IEC Array im Server als OPC UA Array instanziert.

```
<Variable DataType="0:Types.0:DataTypes.0:BaseDataType.0:Boolean"
    NodeId="i=10" BrowseName="bool array" DisplayName="bool array"
    AccessLevel="0x03" ValueRank="1">
    <ArrayDimension ValuePath="APPL.system.vBoolArray"/>
<Variable/>
```

Optional kann ein Teilbereich eines Arrays für das Variablenmapping angegeben werden. Dafür sind folgende Attribute notwendig:

| Attribut | Beschreibung |
|---|---|
| `ArrayOffset` | Angabe des Startwerts des Array Teilbereichs (linker Index). |
| `ArrayLength` | Gewünschte Länge des Array Teilbereichs. |

In folgendem Beispiel wird ein Teilbereich des IEC Arrays im Server als OPC UA Array instanziert. In der IEC ist ein Array von 1 bis 100 definiert. Im OPC UA Server wird nur der Bereich 10 bis 15 gemappt.

```
<Variable DataType="0:Types.0:DataTypes.0:BaseDataType.0:Boolean"
    NodeId="i=10" BrowseName="bool array" DisplayName="bool array"
    AccessLevel="0x03" ValueRank="1">
    <ArrayDimension ArrayOffset="9" ArrayLength="5"
        ValuePath="APPL.system.vBoolArray"/>
<Variable/>
```

## Beispiel einer vollständigen XML Datei

```
<OpcUAInformationModel>
<!-- Declaration of namespace URIs -->
<NamespaceUris>
    <Uri ns="1" isDefaultNs="1">http://www.company.com</Uri>
```

```
    <Uri ns="2">MyNamespace</Uri>
</NamespaceUris>

<!-- Declaration of instances -->
<!-- add child instances to parent Root/Objects node -->
<Instances Parent="0:Objects">

<!-- create folder My Variables -->
<Object DataType="0:Types.0:ObjectTypes.0:BaseObjectType.0:FolderType"
NodeId="1:i=1000" BrowseName="My Variables" DisplayName="My Variables">

<!-- add a UInt8 variable to My Variables folder -->
<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:UInteger.0:Byte"
NodeId="1:i=1003" BrowseName="MyUInt8" DisplayName="MyUInt8"
ValuePath="APPL.system.vUSINT" AccessLevel="0x03"/>
<!-- variable is read- and writeable -->

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:UInteger.0:UInt16"
NodeId="1:i=1004" BrowseName="MyUInt16" DisplayName="MyUInt16"
ValuePath="APPL.system.vUINT" AccessLevel="0x03"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:UInteger.0:UInt32"
NodeId="1:i=1005" BrowseName="MyUInt32" DisplayName="MyUINt32"
ValuePath="APPL.system.vUDINT" AccessLevel="0x03"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:UInteger.0:UInt64"
NodeId="1:i=1006" BrowseName="MyUInt64" DisplayName="MyUInt64"
ValuePath="APPL.system.vULINT" AccessLevel="0x03"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Integer.0:SByte"
NodeId="1:i=1008" BrowseName="MyInt8" DisplayName="MyInt8"
ValuePath="APPL.system.vSINT" AccessLevel="0x01"/>
<!-- variable is only readable -->

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Integer.0:Int16"
NodeId="1:i=1009" BrowseName="MyInt16" DisplayName="MyInt16"
ValuePath="APPL.system.vINT" AccessLevel="0x01"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Integer.0:Int32"
NodeId="1:i=1010" BrowseName="MyInt32" DisplayName="MyInt32"
ValuePath="APPL.system.vDINT" AccessLevel="0x01"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Integer.0:Int64"
NodeId="1:i=1011" BrowseName="MyInt64" DisplayName="MyInt64"
ValuePath="APPL.system.vLINT" AccessLevel="0x01"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Float"
NodeId="1:i=1013" BrowseName="MyFloat" DisplayName="MyFloat"
ValuePath="APPL.system.vREAL" AccessLevel="0x03"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Double"
NodeId="1:i=1014" BrowseName="MyDouble" DisplayName="MyDouble"
ValuePath="APPL.system.vLREAL" AccessLevel="0x03"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Boolean"
NodeId="1:i=1016" BrowseName="MyBool" DisplayName="MyBool"
ValuePath="APPL.system.vBOOL" AccessLevel="0x03"/>

<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:DateTime"
NodeId="1:i=1017" BrowseName="MyDateTime" DisplayName="MyDateTime"
ValuePath="APPL.system.vDATE_AND_TIME" AccessLevel="0x03"/>
```

```
<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:String"
NodeId="1:i=1018" BrowseName="MyString" DisplayName="MyString"
ValuePath="APPL.system.vSTRING" AccessLevel="0x03"/>

<!-- string constant -->
<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:String"
NodeId="1:i=1020" BrowseName="String Constant" DisplayName="String Constant"
ValueConstant="My test string constant" AccessLevel="0x03"/>

<!-- int32 constant-->
<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Integer.0:Int32"
NodeId="1:i=1021" BrowseName="Int32 Constant" DisplayName="Int32 Constant"
ValueConstant="12" AccessLevel="0x03"/>

<!-- float constant -->
<Variable
DataType="0:Types.0:DataTypes.0:BaseDataType.0:Number.0:Float"
NodeId="1:i=1022" BrowseName="Float Constant" DisplayName="Float Constant"
ValueConstant="12.47" AccessLevel="0x03"/>

<!-- create a subfolder for arrays -->
<Object DataType="0:Types.0:ObjectTypes.0:BaseObjectType.0:FolderType"
NodeId="1:i=1023" BrowseName="1-dim Arrays" DisplayName="1-dim Arrays">

<!-- boolean array - block mapping -->
<Variable DataType="0:Types.0:DataTypes.0:BaseDataType.0:Boolean"
NodeId="i=1024" BrowseName="bool array" DisplayName="bool array"
AccessLevel="0x03" ValueRank="1"> <!-- ValueRank must always be 1 -->
<!-- ArrayOffset and ArrayLength are optional -->
<ArrayDimension ArrayOffset="0" ArrayLength="2"
ValuePath="APPL.system.vBoolArray"/>
</Variable>
</Object>
</Object>
</Instances>
</OpcUAInformationModel>
```

## Rollenbasiertes Berechtigungsmodell

Der OPC UA Server unterstützt die Vergabe von Zugriffsberechtigungen für Benutzer auf Basis von Benutzerrollen. Diese Zugriffsberechtigungen gelten nur für die in der XML Datei definierten Knoten und bei Anmeldung des Clients mit Benutzername und Passwort. Ein anonymer Client hat weiterhin vollen Zugriff auf alle Knoten.

In der Benutzerverwaltung müssen sowohl Benutzer und Rollen angelegt, als auch den Benutzern eine oder mehrere Rollen zugewiesen werden. Die Rollennamen im XML müssen mit den Namen in der Benutzerverwaltung übereinstimmen. Wenn sich ein OPC UA Client mit dem Server verbindet, greift dieser auf die Benutzerverwaltung zu, um die Authentifizierungsdaten und Rollen des Clients auszulesen. Im XML werden den angelegten Rollen Berechtigungen zugewiesen. Diese Berechtigungen gelten nur für die vom XML instanzierten Knoten. Falls in der Benutzerverwaltung bereits Rechte für Rollen vergeben wurden, werden diese vom OPC UA Server nicht berücksichtigt. Der OPC UA Server überprüft vor jeder Anfrage, ob die Rollen des Benutzers die Berechtigungen für den Zugriff erfüllen.

Bei gewünschten Zugriffen auf Werte von Variablenknoten wird zuvor zusätzlich die am jeweiligen Variablenknoten eingestellte Zugriffsberechtigungen (Attribut `AccessLevel`) überprüft. Wenn der Wert eines Variblenknotens

z.B. generell nicht schreibbar ist, kann der Client keinen Schreib-Befehl durchführen, selbst wenn die Berechtigungen aufgrund der Rollenzuordnung gegeben sind.

### Globale Berechtigungen

Vor der Instanzierung von Variablen und Objekten (XML Eintrag `Instances`) müssen Rollen und Berechtigungen global vergeben werden. Die Berechtigungen werden als Bitmaske definiert und entsprechen dem Zugriffstyp (`PermissionType`), der in der OPC UA Spezifikation Teil 3 definiert ist.

Ein Rollenname darf mit terminierender Null nur maximal 128 Zeichen lang sein. Maximal 30 verschiedene Rollen werden unterstützt. Die Berechtigungen werden als Hexadezimalzahl interpretiert. Das Präfix "0x" ist optional. Groß- und Kleinschreibung der Hexzahl (0xAABB bzw. 0xaabb) wird nicht berücksichtigt.

### Beispiel

Folgender Codeausschnitt zeigt die Definition von 2 Rollen mit Berechtigungen. Die Rolle "Observer" kann nur den Baum durchlaufen und lesend auf die instanzierten Knoten zugreifen. Die Rolle "Operator" kann zusätzlich schreiben.

```
<DefaultRolePermissions>
    <RolePermission RoleName="Observer" Permissions="0x21"/> <!-- Bit 0 -
Browse, Bit 5 - Read -->
    <RolePermission RoleName="Operator" Permissions="0x61"/> <!-- Bit 0 -
Browse, Bit 5 - Read, Bit 6 - Write -->
</DefaultRolePermissions>
```

Ein Benutzer ohne Rollen bzw. mit Rollen die im XML nicht global definiert wurden, hat nur Zugriff auf den Standardadressraum von OPC UA und den Variablenbaum. Das gleiche gilt, wenn keine globalen Rollen im XML definiert werden. Falls ein Benutzer mehreren Rollen hat, werden alle Berechtigungen der Rollen berücksichtigt.

### Knotenbasierte XML Berechtigungen

Die global deklarierten Berechtigungen einer Rollen können pro Knoten erweitert oder eingeschränkt werden.

Folgendes Beispiel überschreibt die Berechtigung der Rolle "Operator" für den Variablenknoten "Var1". Die Rolle "Operator" kann in der globalen Berechtigungsdefinition "Browse", Lese und Schreibzugriffe durchführen. Für den Variablenknoten "Var1" wird nun der Zugriff auf Browse und Lesen beschränkt.

```
<Variable
    DataType="0:Types.0:DataTypes.0:BaseDataType.0:Boolean" NodeId="i=5000"
BrowseName="Var1" DisplayName="Var1" AccessLevel="0x03" ValueRank="1">
    <NodeRolePermissions>
        <RolePermission RoleName="Operator" Permissions="0x21"/>
    </NodeRolePermissions>
</Variable>
```

## 11.6 Protokollierung des Serverbetriebs

Der OPC UA Server protokolliert Informationen und Fehler über den Hochlauf und die Ausführung zur Laufzeit. Diese Informationen werden entweder in die allgemeine Protokollierung der Steuerung integriert und somit im Trace-Monitor (im u-create studio) angezeigt oder vom Server als eigene Datei verwaltet. Diese Datei wird auf der Steuerung im Verzeichnis `/opt/kecontrol/protocol/OpcUa/` abgelegt. Vor allem für die Fehlersuche bei der Informationsmodellerstellung (fehlerhaftes Variablenmapping, Tippfehler, bereits vorhandene NodeIds etc.) können die hier ausgegebenen Meldungen hilfreich sein.

Die Protokollierung kann über Experteneinträgen (im u-create studio) konfiguriert werden. Ist eine generelle Protokollierung der Steuerung vorhanden, wird nur der Parameter `TraceMask` berücksichtigt. Falls eine eigene Datei verwaltet wird, kann zusätzlich der Parameter `TraceFileSizeKbMax` angeben werden.

Der Wert der `TraceMask` konfiguriert die Granularität der Ausgaben und ist als Bitfeld ausgeführt. Standardmäßig ist die Konfiguration auf `TraceMask = 5` eingestellt. Der einzustellende Wert ergibt sich durch bitweises addieren von folgenden Teilmasken:

| Bit | Teilmaske | Protokollierung |
|---|---|---|
| 0 | 1 | Hochlaufinformationen- und fehler, standardmäßig konfiguriert |
| 1 | 2 | Genauere Informationen und Fehlerausgaben beim Hochlauf |
| 2 | 4 | Laufzeitfehler (interne Fehlschläge z.B. Speicherengpass, unerwartete Typen für Lesen/Schreiben etc.), standardmäßig konfiguriert |
| 3 | 8 | Detailausgaben über Laufzeitfehler |

Die `TraceMask` wird als Dezimalzahl interpretiert. Um alle Ausgaben zu aktivieren, kann die `TraceMask` auf den Wert -1 gesetzt werden.

Der Parameter `TraceFileSizeKbMax` gibt die maximale Dateigröße in kB an. Der Standardwert ist 4096 kB. Ist die Maximalgröße erreicht, wird einmalig eine weitere Protokolldatei mit Suffix `.1` erstellt. Anschließend werden die Protokolldateien jeweils überschrieben.

### Beispiel

```
[OpcUa]
[OpcUa.Trace]
TraceMask = 15
TraceFileSizeKbMax = 4096
```

# 12 Node-RED

Node-RED kann als optionales Softwarepaket (siehe Software Units) installiert werden. Es ist ein von IBM entwickeltes grafisches Entwicklungswerkzeug. Damit können Funktionsbausteine, sogenannte "Nodes", durch Ziehen von Verbindungen in einer gewissen Reihenfolge aneinander gereiht werden. Jeder Funktionsbaustein hat eine festgelegte Aufgabe. Es können Daten verarbeitet und zwischen den Funktionsbausteinen übermittelt werden.

Der grafische Editor von Node-RED kann über "DevAdmin" im Karteireiter "Plugins" gestartet werden. Weiterführende Informationen zur Funktionalität von Node-RED ist im Internet zu finden. Siehe auch https://nodered.org/.

# 13 LongtermDiagnosticMonitor

Der LongtermDiagnosticMonitor kann als optionales Softwarepaket (siehe Software Units) installiert werden und bietet die Möglichkeit einer Langzeitaufzeichnung von Daten auf der Steuerung. Über den "Monitor" im DevAdmin (Karteireiter Plugins) können die Daten angezeigt werden.

## 13.1 Monitor

Die Anzeige ist in folgende Bereiche unterteilt:

- Problems
- Groups
- Categories

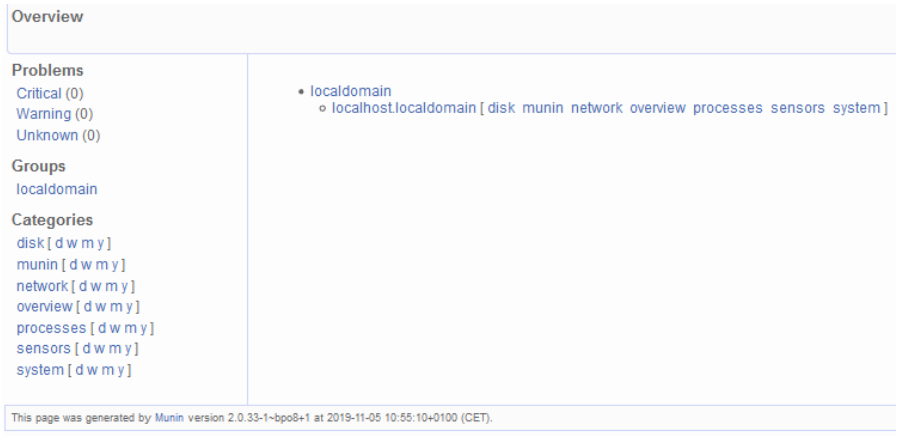



Abb. 13-18: DevAdmin - Monitor

### Problems

In diesem Bereich werden die aufgetretenen Störungen unterteilt in ihre Dringlichkeit angezeigt.

### Groups

In diesem Bereich sind alle verfügbaren Diagramme aufgelistet und können ausgewählt werden.

### Categories

In diesem Bereich sind die Kategorien für die Langzeitaufzeichnungen aufgelistet. Nach Auswählen einer Kategorie werden die dazugehörigen Diagramme angezeigt. Die Diagramme werden ebenfalls beim Auslösen eines Statusreports mitgespeichert.

Neben der Kategorie kann über den entsprechenden Buchstaben **d w m y** die Tagesübersicht **d**, Wochenübersicht **w**, Monatsübersicht **m** oder Jahresübersicht **y** aufgerufen werden. Durch Klicken auf ein Diagramm öffnet sich ein Überblick mit 4 Diagrammen bestehend aus der Tages-, Wochen-, Monats- und Jahresübersicht.

Folgende Diagramme können angezeigt werden:

| Kategorie | Diagramm | Beschreibung |
|---|---|---|
| **disk** | Average latency for /dev/sda | Durchschnittliche Wartezeit (Latenz) für Zugriffe auf das Verzeichnis `/dev/sda`. Die Wartezeit zeigt an, wie ausgelastet das System ist. 1 Sekunde Wartezeit bedeutet eine Auslastung von 100 %. |
| | Disk IOs per device | Ein- und Ausgaben am Datenträger pro Gerät |
| | Disk latency per device | Wartezeit am Datenträger |
| | Disk throughput for /dev/sda | Durchsatz des Datenträgers für das Verzeichnis `/dev/sda` [in Byte pro Sekunde] |
| | Disk utilization for /dev/sda | Auslastung des Datenträgers für das Verzeichnis `/dev/sda`. Wenn eine Ein-/Ausgabe 1 Sekunde dauert, ist der Datenträger zu 100 % ausgelastet. |
| | IOs for /dev/sda | Anzahl von Ein-/Ausgaben (I/O-Anforderungen) pro Sekunde und ihre durchschnittliche Größe [in kB] (im Diagramm entspricht 1 kB = 1000 Byte). |
| | Throughput per device | Durchsatz von Ein-/Ausgaben für das Verzeichnis `/dev/sda` |
| | Utilization per device | Auslastung des Verzeichnisses `/dev/sda` [in Prozent] |
| **munin** | Munin processing time | Verarbeitungsdauer der vier verschiedenen Munin-Prozesse (munin update, munin graph, munin html, munin limits) [in Sekunden] |
| | Munin update | Die von Munin benötigte Zeit für das Sammeln der aufgezeichneten Daten [in Sekunden] |
| **network** | Firewall Throughput | Durchsatz der Firewall [empfangene und weitergeleitete Pakete pro Sekunde] |
| | Netstat | TCP-Aktivität aller Netzwerkschnittstellen. Anzahl von aktiven, passiven, fehlgeschlagenen, neugestarteten und aktuell durchgeführten TCP-Verbindungen pro Sekunde. |
| | ETH0 errors | Anzahl von Fehlern, Paketverlusten und Kollisionen auf der ETH0-Netzwerkschnittstelle |
| | ETH0 traffic | Datenverkehr der ETH0-Netzwerkschnittstelle [in Bits pro Sekunde] |
| | ETH1 errors | Anzahl von Fehlern, Paketverlusten und Kollisionen auf der ETH1-Netzwerkschnittstelle |
| | ETH1 traffic | Datenverkehr der ETH1-Netzwerkschnittstelle [in Bits pro Sekunde] |

| Kategorie | Diagramm | Beschreibung |
|---|---|---|
| **overview** | CPU utilization | Gesamtauslastung der CPU pro Kern [in Prozent] |
| | Disk lifetime | Verbleibende Lebensdauer des Flash-Speichers [in Prozent] |
| | Disk space | Belegter und freier Speicherplatz auf dem Flash-Speicher [in MB] |
| | PLC baselib heap memory statistics | Belegter und freier Speicherplatz auf dem “Base-Lib Heap-Speicher” [in MB] |
| | RAM usage | RAM-Auslastung unterteilt in belegt, frei und verfügbar [in MB] |
| | System task performance on core n | Verteilung der Rechenleistung (pro Kern) auf die verschiedenen Prozessgruppen [in Prozent] |
| | Temperature information | Temperatur von CPU und Mainboard [in Grad Celsius] |
| | Uptime of PX-Package | Betriebszeit des Steuerungspackages (PX-Package) [in Tagen] |
| **processes** | Fork rate | Anzahl der neu gestarteten Prozesse [pro Sekunde] |
| | Number of threads | Anzahl der Threads |
| | Processes | Anzahl der Prozesse unterteilt in ihren Zustand |
| | Processes priority | Anzahl der Prozesse unterteilt in ihre Wichtigkeit |
| | VMstat | Nutzung der CPU-Zeit durch Prozesse [in Milli = Tausendstel] |
| **system** | Available entropy | Anzahl an verfügbaren Zufallszahlen |
| | CPU usage | CPU-Nutzung unterteilt in die verschiedenen Prozessgruppen |
| | File table usage | Anzahl der geöffneten Dateien und maximale Anzahl an geöffneten Dateien |
| | Individual interrupts | Anzahl an Interrupts unterteilt in die Art des Interrupts |
| | Inode table usage | Anzahl der geöffneten Inodes |
| | Interrupts and context switches | Anzahl der Interrupts und Kontextwechsel |
| | Load average | Durchschnittsauslastung, zeigt die Anzahl der unmittelbar auszuführenden Prozesse an |
| | Logged in users | Eingeloggte Benutzer |
| | Memory usage | Anzeige, wofür der Arbeitspeicher verwendet wird [in Bytes] |
| | Swap in/out | Transfer zwischen Arbeitsspeicher und Swap-Speicher |
| | Uptime | Betriebszeit |

# 14 Modbus Server

Der Modbus Server kann als optionales Softwarepaket (siehe Software Units) installiert werden und bietet die Möglichkeit über das Protokoll "Modbus TCP" bestimmte Variablen der Steuerung zu lesen und zu schreiben.

Die Variablen können in u-create studio über die Expert-Konfiguration definiert werden. Weiters können dort die Zugriffrechte und der Umrechnungsfaktor einer Variable festgelegt werden.

Folgende Schnittstelle wird unterstützt:

- Ethernet, Port 502

### Basis Bibliothek

Die Modbus Server Implementierung basiert auf **libmodbus-3.1.4** unter LGPL v2.1+ Lizenz.

libmodbus – A Modbus library for Linux, Mac OS X, FreeBSD, QNX and Win32, http://libmodbus.org/

### Test-Software Empfehlung

QModMaster, https://sourceforge.net/projects/qmodmaster/

Python modbus-tk

## 14.1 Funktionsbeschreibung

Der Modbus Server ist ein ausführbares Programm, das automatisch mit der Steuerung gestartet wird. Der Modbus Server der Steuerung fungiert als Server und bearbeitet die Anfragen eines Clients. Der externe Client hat die Möglichkeit Werte zu lesen bzw. auch Werte abzuändern.

### Funktionen

Die folgenden Funktionen werden vom Modbus Server unterstützt:

| Funktionscode | Name |
|---|---|
| 0x01 | Read Coils |
| 0x02 | Read Discrete Inputs |
| 0x03 | Read Holding Registers |
| 0x04 | Read Input Registers |
| 0x05 | Write Single Coil |
| 0x06 | Write Single Register |
| 0x0F | Write Multiple Coils |
| 0x10 | Write Multiple Registers |
| 0x16 | Mask Write Register |
| 0x17 | Read / Write Multiple Registers |

Alle anderen Funktionsaufrufe werden mit einer Fehlermeldung beantwortet.

## Fehlermeldungen

| Fehlername | Fehlernummer | Beschreibung |
|---|---|---|
| MODBUS_EXCEPTION_ILLEGAL_FUNCTION | 1 | Modbus Funktionscode nicht unterstützt |
| MODBUS_EXCEPTION_ILLEGAL_DATA_ADDRESS | 2 | Variable für Register / Coil Zugriff ist nicht konfiguriert |
| MODBUS_EXCEPTION_ILLEGAL_DATA_VALUE | 3 | • Anzahl für Zugriff auf mehrere Register / Coils ist außerhalb der Spezifikation<br>• Anzahl der Bytes im empfangenen Modbus Kommando ist ungültig |
| MODBUS_EXCEPTION_SLAVE_OR_SERVER_FAILURE | 4 | • Modbus Server Konfiguration ungültig<br>• Steuerung läuft nicht<br>• Keine Variable für zu manipulierendes Register / Coil zugeordnet<br>• Manipulation mehrerer Register / Coils über den konfigurierten Adressbereich hinaus<br>• Keine Rechte für Schreibzugriff<br>• Kein gültiger Zugriff auf verschränkte Register z.B. nur 1 Register bei Variablen-Mapping auf 2 Register wird gelesen<br>• Mapping eines nicht unterstützten Variablentypes |

## Datendarstellung

Modbus unterstützt nur 16 Bit Register. Ein Register kann daher nur Ganzzahlwerte ohne Dezimalstellen darstellen. Werte, die als Kommazahl dargestellt werden, müssen Ganzzahlwerte ohne Vorzeichen größer als $2^{16}$ (65535) oder Ganzzahlwerte mit Vorzeichen größer / kleiner als $2^{15}$ (+/- 32768) über einen Umrechnungsfaktor skaliert oder auf mehrere Modbus-Register aufgeteilt werden.

Coils arbeiten Bitweise. Ist der Variablenwert gleich 0, ist die Coil nicht gesetzt. Ist der Wert ungleich 0, gilt das Coil als gesetzt. Beim Schreiben von Coils wird nur 0 oder 1 auf die Applikationsvariable geschrieben.

## 14.2 Konfiguration

Die Konfiguration erfolgt über "Expert Entries" im u-create studio-Projekt.

Die Konfiguration für den Modbus Server besteht aus den folgenden Abschnitten:

- Grundsätzliche Konfiguration der Schnittstelle
- Konfiguration der Holding-Register (optional)
- Konfiguration der Input-Register (optional)
- Konfiguration der Coils (optional)
- Konfiguration der Input-Coils (optional)

Jeder Abschnitt besteht aus einem Basis-Knoten in eckiger Klammer und Sub-Einträgen. Die Sub-Einträge können entweder einzelne Parameter sein oder wieder ein Sub-Knoten in eckiger Klammer. Am Beginn eines Sub-Knotens steht der übergeordnete Knoten gefolgt von einem Punkt als Trennzeichen. Jeder Eintrag muss in einer neuen Zeile stehen, es können aber Leerzeilen eingefügt werden. Einrückungen am Anfang werden ignoriert. Am Ende einer Zeile kann beginnend mit `//` ein Kommentar-Text eingefügt werden.

### 14.2.1 Konfiguration Schnittstelle

Die Bezeichnung des Basis-Knotens lautet `[ModBus]`.

Folgende Parameter können für die Schnittstelle konfiguriert werden.

Auswahl der Schnittstelle:

- Konfigurationseintrag `enableRTU = 0`

***Information***

*Aktuell wird ausschließlich Modbus-TCP unterstützt. Modbus-RTU muss deaktiviert werden.*

Konfiguration der Modbus TCP Schnittstelle:

- Bezeichnung des Sub-Knotens `[ModBus.TCP]`
- Konfigurationseintrag `IP`: IP-Adressen-Filter für den Server-Zugriff
  `IP="0"` erlaubt den Zugriff von jedem Client
  `IP="xxx.xxx.xxx.xxx"` erlaubt nur den Zugriff des Clients mit der eingetragenen IP-Adresse
- Konfigurationseintrag `Port`: Port, der vom Modbus Server geöffnet wird
  `PORT = 502` ist der Standardport
- Konfigurationseintrag `TIMEOUT`: Timeout in s für Überwachung einer zyklischen Kommunikation mit einem externen Client.
  Dieser Parameter wird derzeit nicht genutzt.

**Beispiel Konfiguration - Schnittstelle**

```
[ModBus]
    enableRTU = 0
    [ModBus.TCP]
```

```
IP = "0" // jeder Client wird zugelassen
PORT = 502 // Standard Port
```

### 14.2.2 Konfiguration der Prozessdaten

Für die Konfiguration der Prozessdaten werden folgende Basis-Knoten verwendet:

| Name | Beschreibung |
|---|---|
| `[ModBusReg]` | Basisknoten für Holding Register |
| `[ModBusCoil]` | Basisknoten für Coils |
| `[ModBusInputReg]` | Basisknoten für Input Register (Read Only Daten) |
| `[ModBusInputCoil]` | Basisknoten für Input Coils (Read Only Daten) |

Die Datenzuordnung erfolgt dann in Sub-Knoten.

Folgende Parameter können für die Konfiguration von Registern und Coils des Modbus Servers gesetzt werden:

| Parameter | Beschreibung |
|---|---|
| Modbus Register / Coil Adressenzuordnung | • Die Adresse wird beim entsprechenden Knoten als Index angegeben, z.B. `[ModBusReg.Adr:X]`<br>• Die Nummerierung beginnt bei 1 und läuft bis maximal 65535 (0xFFFF)<br>Es können Lücken in der Zuordnung konfiguriert werden. |
| Variablenname | • Wird als Sub-Eintrag für den entsprechenden Knoten angegeben z. B. `Variablenname = "APPL.Application.GVL.gint16_ro"`<br>• Es muss der vollständige Variablenpfad angegeben werden (bis zum Wurzelknoten "APPL")<br>• Variablen müssen in der "Symbol Configuration" des u-create studio-Projekts die Zugriffsrechte gesetzt bekommen, damit sie für den Modbus Server verwendbar sind (Access Rights --> Lesen, Schreiben, Schreiben / Lesen)<br>• Folgende Variablentypen können für den Modbus Server freigegeben werden:<br>  • 8-Bit Datentypen: SINT, USINT, BYTE ohne Einschränkung<br>  • 16-Bit Datentypen: INT, UINT, WORD ohne Einschränkung<br>  • 32-Bit Datentypen: DINT, UDINT, DWORD, REAL: über Doppel-Zuordnung bei Registern oder über Umrechungsfaktor und beschneiden auf INT Zahlenbereich<br>  • 64-Bit Datentypen: LREAL mit Umrechnungsfaktor und beschneiden auf INT Zahlenbereich<br>Andere Variablentypen werden durch den Modbus Server nicht unterstützt und führen bei Zugriff auf das zugeordnete Register / Coil zu Error Code `MODBUS_EXCEPTION_SLAVE_OR_SERVER_FAILURE`. |

| Parameter | Beschreibung |
|---|---|
| Schreibberechtigung | • Wird als Sub-Eintrag für den entsprechenden Knoten angegeben, z. B. `WritePermission = 0`<br>• Wert 0: nur Lesezugriff, Wert 1: Schreib- und Lesezugriff<br>• Für Input-Register und Input-Coils wird die Einstellung ignoriert. Hier ist nur Lesezugriff per Definition möglich. |
| Umrechnungsfaktor | • Wird als Sub-Eintrag für den entsprechenden Knoten angegeben z. B. `Factor = 1.0`<br>• Die u-create studio-Applikationsvariable wird bei Lesezugriff mit diesem Faktor multipliziert, bevor der Wert über Modbus zurück geschickt wird.<br>• Der über Modbus empfangene Wert wird mit diesem Faktor dividiert, bevor bei Schreibzugriffen die u-create studio-Applikationsvariable beschrieben wird. |

### Beispiel Konfiguration - Basic

```
[ModBus]
    enableRTU = 0

    [ModBus.TCP]
    IP = "0"
    PORT = 502

//Konfiguration Holding Register
[ModBusReg]
//Beispiel Lesezugriff 16 Bit Variable
    [ModBusReg.Adr:1]
            Factor = 1.0
    Variablenname = "APPL.Application.GVL.gint16_rw"
        WritePermission = 1

//Konfiguration Input Register
[ModBusInputReg]
    [ModBusInputReg.Adr:1]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.gint16_ro"
    WritePermission = 0

//Konfiguration Coil
[ModBusCoil]
    [ModBusCoil.Adr:1]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.gint8_rw"
        WritePermission = 0

//Konfiguration Input Coil
[ModBusInputCoil]
    [ModBusInputCoil.Adr:1]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.gint8_ro"
        WritePermission = 0
```

## 14.2.3 Erweiterte Konfiguration

Register mit Variablen, die auf Datentypen mit mehr als 16-Bit aufbauen, müssen entweder über den Umrechnungsfaktor skaliert und dann entsprechend beschnitten werden oder auf mehrere Register aufgeteilt werden.

Für eine Aufteilung wird der Variablenname in zwei Registerkonfigurationen eingetragen. Zugriff auf Variablen, die auf zwei Register (Doppelregister) aufgeteilt wurden, sind nur in einem einzelnen Lese- / Schreibzugriff über Functioncodes 0x03, 0x04 / 0x10 möglich.

### Beispiel Konfiguration - Erweitert

```
[ModBus]
    enableRTU = 0

    [ModBus.TCP]
    IP = "0"
    PORT = 502

//Konfiguration Holding Register
[ModBusReg]
//Beispiel Lesezugriff 16 Bit Variable
    [ModBusReg.Adr:1]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.gint16_ro"
        WritePermission = 0

//Beispiel Lesezugriff Gleitkomma-Variable
    [ModBusReg.Adr:2]
        Factor = 100
        Variablenname = "APPL.Application.GVL.greal_ro"
        WritePermission = 0

//Beispiel Lesezugriff 32 Bit Variable über Doppelregister
    [ModBusReg.Adr:3]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.gint32_ro_multiReg"
        WritePermission = 0
    [ModBusReg.Adr:4]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.gint32_ro_multiReg"
        WritePermission = 0

//Beispiel Lesezugriff Gleitkomma-Variable über Doppelregister
    [ModBusReg.Adr:5]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.greal_rw_multiReg"
        WritePermission = 1
    [ModBusReg.Adr:6]
        Factor = 1.0
        Variablenname = "APPL.Application.GVL.greal_rw_multiReg"
        WritePermission = 1
```

## Übertragung eines 32 Bit Gleitkomma-Wertes über Skalierung und Einzel-Register-Zuordnung

Variable `APPL.Application.GVL.greal_ro` ist im Konfigurationsbeispiel auf Einzel-Register 2 zugeordnet und mit Umrechungsfaktor 100 parametriert. Wird Register 2 über Modbus gelesen, wird der Variablenwert vor der Übertragung mit dem Umrechungsfaktor multipliziert. Der Empfänger muss die Skalierung durch Division mit dem Umrechungsfaktor wieder rückgängig machen.

### Beispiel:

`APPL.Application.GVL.greal_ro = 23.456`

Der Wert mit Faktor 100 multipliziert und in einen Ganzzahlwert konvertiert. 23.456 * 100 = 2345.6 --> Ganzzahl-Konvertierung (ohne Runden) --> 2345

Der Wert 2345 wird über Modbus übertragen.

### Übertragung eines 32bit Gleitkomma-Wertes über Multi-Register Zugriff

Variable `APPL.Application.GVL.glreal_rw_multiReg` ist im Konfigurationsbeispiel auf Register 5 und 6 aufgeteilt und kann über dieses Doppelregister als 32bit Wert übertragen werden.

Die Variable kann nur über Funktionscode 0x03 (Read Holding Registers) gelesen und über Funktionscode 0x10 (Write Multiple Registers) geschrieben werden. Die Anzahl der zu lesenden / schreibenden Register beträgt somit 2 oder mehr (falls weitere Register manipuliert werden).

Der Empfänger muss sich darum kümmern, die beiden Registerwerte bei Lesezugriffen wieder zu einem gültigen Gleitkomma-Wert zusammenzusetzen.

**Beispiel:**

```
APPL.Application.GVL.greal_rw_multiReg = 23.456
```

Darstellung als 32bit Zahl im Hex-Format: 0x41BBA752, Aufteilung in 2 16 Bit Register: 0x41BB, 0xA752

Somit enthält Register 5 das High Word 0x41BB (= 16827), Register 6 enthält das Low Word 0xA752 (= 42834). Diese Werte werden über Modbus übertragen.

Um die Binär-Darstellung der übertragenen Werte zu prüfen, können z. B. folgende Tools verwendet werden:

- HEX↔Dezimal-Konverter
  http://www.binaryconvert.com/convert_unsigned_short.html
- FLOAT↔HEX-Konverter
  http://www.binaryconvert.com/convert_float.html

## 14.2.4 Konfiguration von Arrays

Über eine spezielle Syntax können Daten-Arrays der Steuerung einfach für den Modbus Server konfiguriert werden. Dazu wird an die IEC-Syntax angelehnt der Start-Index und der End-Index des Arrays beim Sub-Knoten für die Adresszuweisung mit angegeben.

Modbus Register / Coil Adressen-Zuordnung:
Die Adresse wird beim entsprechenden Knoten als Index angegeben, gefolgt vom Start-Index und End-Index des Arrays, z. B.
`[ModBusReg.<Modbus Addr>:<array start index>:<array end index>]`

### Beispiel Konfiguration - Arrays

Definition des Arrays mit 2000 Elementen im Programm-Code der Steuerungs-Applikation:

```
VAR_GLOBAL
    dataArrayModbus : ARRAY [1..2000] OF REAL := [2000(0)];
END_VAR
```

Datenzuordnung auf die Modbus Holding-Register beginnend mit Register-Index 200:

```
[ModBusReg.Adr:200:1:2000] //test array configuration
    Factor = 1.0
    Variablenname = "APPL.Application.GVL.dataArrayModbus"
    WritePermission = 1
```

Das Array wird somit auf die Register 200 bis 2199 zugeordnet.

# 15 Datenrekorder

Über den Datenrekorder können Variablenwerte auf der Steuerung zu Laufzeit aufgezeichnet werden.

Über folgende Schnittstellen ist ein Zugriff auf den Datenrekorder möglich:

- Bibliothek "KREC" für IEC-Applikationen (Echtzeit-Zugriffe) im u-create studio
- Schnittstelle "DataRecApi " für C-Applikationen

Weiterführende Informationen siehe Onlinehilfe u-create studio.

Ein Datenrekorder muss zuerst von der Applikation angelegt und konfiguriert werden. Dies ist z.B. über die IEC-Applikation oder auch u-create scope möglich.

## 15.1 Konfiguration

Innerhalb einer IEC-Applikation kann ein (oder mehrere) Datenrekorder angelegt und konfiguriert werden. Die notwendigen Konfigurationen (Eigenschaften, Variablen, ...) werden in einem sogenannten "Profil" zusammengefaßt und abgelegt.

Die grundlegenden Eigenschaften eines Profils sind:

- Maximale Variablenanzahl
- Pufferkapazität
- Puffertyp
- Persistierung

### Maximale Variablenanzahl

Jede Variable, deren Wert aufgezeichnet werden soll, muss beim Datenrekorder angemeldet werden. Dabei wird der vollständigen Pfad der Variable angegeben. Die maximale Variablenanzahl definiert die Obergrenze für die Anzahl der Variablenanmeldungen. Variablen können während einer Aufzeichnung an- und abgemeldet werden, die Anzahl kann die angegebene maximale Variablenanzahl aber nicht überschreiten.

Nur Variablen mit folgenden Datentypen können beim Datenrekorder angemeldet werden:

| | | | |
|---|---|---|---|
| BOOL | REAL | LREAL | BYTE |
| ENUM | WORD | DWORD | LWORD |
| SINT8 | SINT16 | SINT32 | SINT64 |
| ENUM_SINT8 | ENUM_SINT16 | ENUM_SINT32 | ENUM_SINT64 |
| UINT8 | UINT16 | UINT32 | UINT64 |
| ENUM_UINT8 | ENUM_UINT16 | ENUM_UINT32 | ENUM_UINT64 |
| DATE | DATE64us | TIME | TIME64us |
| DT | DT64us | TOD | TOD64us |

## Pufferkapazität

Die Pufferkapazität definiert die maximale Anzahl von gespeicherten Variablenwerten mit Zeitstempel, die ein Profil speichern kann. Der Aufzeichnungsvorgang liest die Werte aller angemeldeten Variablen aus und schreibt diese zusammen mit einem Zeitstempel in den Puffer des Profils. Diesen Datensatz (Variablenwert mit Zeitstempel) nennt man "Sample".

Die Pufferkapazität multipliziert mit der maximalen Variablenanzahl bestimmt den Speicherbedarf eines Profils. Um Aufzeichnung in Echtzeit zu garantieren muss der Speicher vorab angelegt werden.

## Puffertyp

Folgende Arten eines Puffertyps können konfiguriert werden:

- **Continuous** (Endlosaufzeichnung): Aufzeichnung über die Pufferkapazität hinaus ist möglich, indem nach dem letzten Eintrag wieder von vorne begonnen wird. Datenverlust durch Überschreiben nicht gelesener Samples ist möglich, beim Auslesen sollten Daten auf Vollständigkeit geprüft werden.
- **SingleShot** (Einmalaufzeichnung): keine Aufzeichnung über die Pufferkapazität hinaus. Die Aufzeichnung endet spätestens beim Erreichen des Pufferendes. Kein Datenverlust, keine Vollständigkeitsprüfung gelesener Daten nötig.

## Persistierung

Standardmäßig bleiben die Konfigurationen und die gespeicherten Daten nach einem Neustart des Systems nicht erhalten. Dies kann jedoch durch eine Konfiguration des Datenrekorders geädert werden.

Folgende Konfigurationen sind möglich:

- Keine Persistierung: Das Profil wird bei jedem Hochlauf von der Applikation neu erzeugt und die Aufzeichnung startet mit leerem Puffer.
- Persistierung der Konfiguration: Die Profileinstellungen werden in einer Datei unter dem Pfad `{System.applPath}/application/control/config` abgepeichert. Im Zuge des Hochlaufs der Steuerung wird das Profil entsprechend der Konfiguration automatisch erzeugt. Die Aufzeichnung startet mit leerem Puffer. Es erfolgt keine dauerhafte Speicherung der aufgezeichneten Daten. Wählt man beim erstmaligen Erzeugen des Profils die Option `autoStart`, wird im Hochlauf die Aufzeichnung automatisch gestartet.
- Volle Persistierung: Alle Informationen (Konfiguration, sowie bereits aufgezeichnete Daten) werden dauerhaft gespeichert. Der Zustand und Daten werden binär im Verzeichnis `{System.applPath}/retain` abgelegt. Im Zuge des Hochlaufs wird das Profil entsprechend der Konfiguration automatisch erzeugt und in den Zustand zum Zeitpunkt der letzten Speicherung versetzt. Ist die Option `autoStart` ausgewählt, setzt die Aufzeichnung fort.

In Hinblick auf die Persistierung ist Folgendes zu beachten:

- Bei der Speicherung des Profilzustands können abhängig von der Profilgröße große Datenmengen anfallen. Speichert die Applikation den Zustand zyklisch in kurzen Abständen, kann dies die Lebensdauer des nicht flüchtigen Speichers deutlich reduzieren. Nachdem die Applikation die Speicherung der Zustände selbst steuert, ist hier ein sinnvoller Kompromiss zu suchen.
- Änderungen an der Applikation können dazu führen, dass persistierte Einstellungen eines Profils ungültig werden, wenn diese Bezug auf Variablen nehmen, die nicht mehr existieren. Ungültige Einstellungen, wie fehlende Trigger-Variablen, haben zur Folge, dass das Profil nicht mehr wie ursprünglich konfiguriert werden kann. Die Applikation erfährt dies durch entsprechende Fehlermeldungen. Fehlen jedoch nur einzelne Variablen, die zur Aufzeichnung angemeldet wurden, wird das Profil mit den verfügbaren Variablen betriebsfertig konfiguriert (und gegebenenfalls auch gestartet).
- Beim Löschen eines Profils werden auch dessen persistierte Daten (Konfiguration und Aufzeichnungszustand) gelöscht. Beim Löschen des Pufferinhalts wird auch der persistierte Aufzeichnungszustand gelöscht.

## 15.2 Datenaufzeichnung

An einem Profil angemeldete Variablen können aufgezeichnet werden. Eine Aufzeichnung besteht aus

- dem Erfassen der Werte aller angemeldeten Variablen und
- deren Speicherung (zusammen mit Metadaten) im Puffer.

Die Aufzeichnung der Variablen kann Zeit- oder Ereignisgesteuert sein. Die Daten werden gemeinsam mit einem Zeitstempel in einem Puffer gespeichert. Der Puffer liegt im RAM, kann aber zusätzlich auf einem Datenträger gespeichert werden. Der Puffer kann Bereichsweise - auch während einer laufenden Aufzeichnung - ausgelesen werden.

Eine Aufzeichnung kann manuell oder automatisch erfolgen. Bei automatische Aufzeichnungen kann über die Applikation folgendes konfiguriert werden:

- Zeitlicher Abstand der Einzelaufzeichnungen in µs
- Task-Priorität der Einzelaufzeichnungen (von 1 = hoch bis 31 = niedrig)
- CPU-Bindung des Tasks (ab 0, oder -1 für systemseitig gewählte Standard-Bindung, > 1 CPU für die Runtime verfügbar)

Mit diesen Einstellungen wird systemseitig ein Task angelegt, der sich bezüglich Periodizität und Priorität unter die übrigen Applikationstasks einreiht.

***Information***

*Das Zusammenwirken dieses Tasks mit übrigen Applikationstasks muss berücksichtigt werden. Lassen die Zeitstempel der Daten z.B. unerwünschten Jitter erkennen, ist die getroffene Konfiguration zu ändern. Weiters sollte die Priorität nicht zu hoch, der Abstand nicht zu kurz und die Datenmenge nicht zu groß sein, um eine Blockierung echtzeitkritischer Tasks zu vermeiden.*

Ein Profil kann folgende Zustände einnehmen:

| Zustand | Beschreibung |
|---|---|
| recording | Es wird eine Aufzeichnung durchgeführt. |
| stopped | Die Aufzeichnung ist gestoppt oder beendet. |
| waiting | Es wird auf den Start-Trigger bzw. Startverzug gewartet, bevor die Aufzeichnung durchgeführt wird. |

## Aufzeichnungsmodus

Folgende Aufzeichnungsarten sind möglich:

- Standard: pro Einzelaufzeichnung werden die Werte aller Variablen gespeichert
- Änderungsbasiert: Bei Änderung einer bestimmten Variable werden die Werte aller Variablen gespeichert

Eine änderungsbasierte Aufzeichnung überwacht nicht alle Variablen des Profils, sondern lediglich eine konkrete Variable. Ändert sich deren Wert, werden die Werte aller Profilvariablen aufgezeichnet. Optional lässt sich die Aufzeichnung der Werte gegenüber der Erkennung einer Änderung untersetzen.

### Beispiel

Änderungsbasierte Aufzeichnung mit einer überwachten Variablen `prodCnt` und Untersetzung der Speicherung um Faktor 2, nur die fett geschriebenen Werte werden gespeichert.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| prod-Cnt | **1** | 1 | 2 | 2 | **3** | 3 | 4 | 4 | **5** |
| weight | **14** | 15 | 16 | 14 | **15** | 13 | 14 | 15 | **16** |
| tmp | **60** | 60 | 60 | 61 | **61** | 60 | 60 | 60 | **60** |

***Information***

*Die erste Abtastung einer Variablen seit ihrer Anmeldung beim Profil bewirkt stets eine Speicherung, da der Variablenwert auf jeden Fall als geändert gilt.*

## Anzahlbegrenzungen

Für manche Anwendungen ist es relevant, dass Aufzeichnungen nicht endlos laufen, sondern nach einer definierten Anzahl an Aufzeichnungen automatisch gestoppt werden. Weiters kann es sinnvoll sein, die Aufzeichnung nicht sofort nach dem Starten zu beginnen, sondern erst nach einer bestimmten Zeitspanne (insbesondere, wenn ein Trigger den Start auslöst und relevante Daten erst eine Weile nach dem Trigger vorliegen).

Zu diesem Zweck können folgende Begrenzungen eingestellt werden:

- Startverzug: Definierte Anzahl von Aufrufe nach Aufzeichnungsstart, die keine Aufzeichnungen erzeugen. Erst danach werden die Variablenwerte aufgezeichnet.
- Anzahl der zu speichernden Aufzeichnungen: Anzahl der Aufrufe (ohne Startverzug) bis zum automatischen Stoppen der Aufzeichnung.

## Start- und Stopp-Trigger

Trigger sind Bedingungen, mit denen eine Aufzeichnung eines Profils ereignisgesteuert gestarten und/oder gestoppt werden kann. Eine Triggerbedingung bezieht sich immer auf eine bestimmte Variable. Diese muss nicht im Profil angemeldet sein, sondern wird über ihren vollständigen Pfad im Variablenbaum spezifiziert. Für den Datentyp gelten dieselben Einschränkungen wie bei angemeldeten Variablen. Folgende Arten von Bedingungen können mit dieser Variablen spezifiziert werden:

- Überschreitung eines konstanten Schwellwerts t: Die Bedingung tritt ein, wenn der Variablenwert t überschreitet, während dies beim letzten Aufruf noch nicht der Fall war.
- Unterschreitung eines konstanten Schwellwerts t: Die Bedingung tritt ein, wenn der Variablenwert t unterschreitet, während dies beim letzten Aufruf noch nicht der Fall war.
- Über- oder Unterschreitung eines konstanten Schwellwerts t: ODER-Kombination der beiden obigen Bedingungen
- Beliebige Änderung des Variablenwerts: Die Bedingung tritt ein, wenn sich der Variablenwert gegenüber dem letzten Aufruf verändert hat

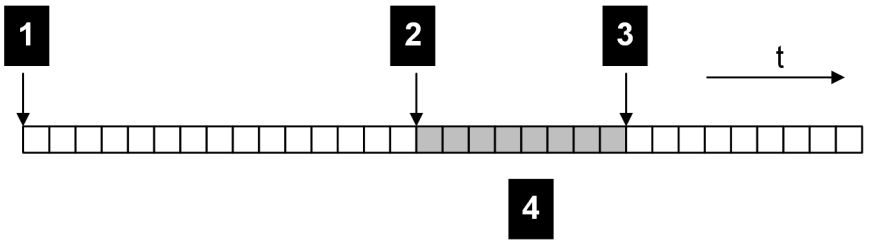
Beim Eintreten der Start-Trigger-Bedingung startet die Aufzeichnung, entsprechend der Konfiguration (d.h. entweder endlos oder bis zum Erreichen des Pufferendes). Falls eine Stopp-Trigger-Bedingung konfiguriert wurde, wird beim EIntreten dieser Bedingung die Aufzeichnung gestoppt.

Eine Start-Trigger Bedingung kann auch ohne dazugehörige Stopp-Trigger Bedingung eingestellt werden. Umgekehrt ist dies nicht möglich.

Es besteht weiters die Möglichkeit, eine Nachlaufbedingung (Post-Start-Trigger) zu definieren. Diese definiert eine Anzahl von Aufzeichnungen. Danach wird die Aufzeichnung automatisch gestoppt.

**Beispiel**

Gesamtkapazität: 32 Samples, Nachlaufbedingung: 25 % (= 8 Samples)

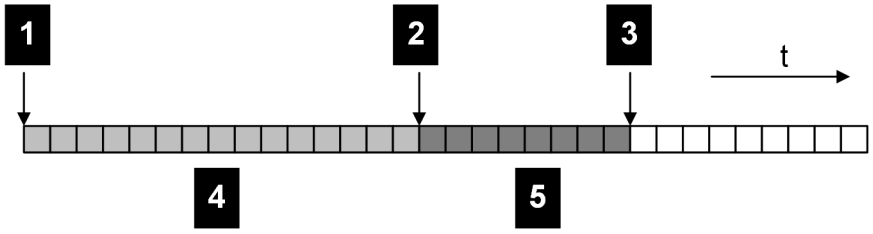



| | |
|---|---|
| 1 ... Aufzeichnungsstart | 2 ... Start-Trigger |
| 3 ... Aufzeichnungsstopp | 4 ... Nachlauf |

***Information***

*Für die Bemessung der Nachlaufkapazität in Prozent ist zu beachten, dass die Pufferkapazität im Falle des Puffertyps "Continuous" systemintern auf die nächste Zweierpotenz aufgerundet wird (z.B. 4000 wird auf 4096 aufgerundet). Die Prozent wirken auf den aufgerundeten Wert.*

Um Daten vor dem Eintreten des Start-Triggers aufzuzeichnen, muss in der Applikation eine Vorlauf (Pre-Start-Trigger-Aufzeichnung) konfiguriert werden. In diesem Fall wird der Puffer unmittelbar nach Aufzeichnungsstart gefüllt. Beim Eintreten des Start-Triggers setzt die Aufzeichnung im Rahmen der Nachlaufbedingung fort oder die Aufzeichnung stoppt, falls eine Nachlaufbedingung nicht konfiguriert wurde.




| | |
|---|---|
| 1 ... Aufzeichnungsstart | 2 ... Start-Trigger |
| 3 ... Aufzeichnungsstopp | 4 ... Vorlauf |
| 5 ... Nachlauf | |

| | |
|---|---|
| Gesamtkapazität: 32 Samples | Nachlauf: 25% = 8 Samples, max. Vorlauf: 75% = 24 Samples |

***Information***

*Bei Vorlauf- und Nachlaufbedingung wird die Schreibposition zum Zeitpunkt des Eintretens der Trigger-Bedingung im Profil abgespeichert. Diese Positionen blieben bis zur nächsten Aufzeichnung verfügbar.*

Die Vorlauf- bzw. Nachlaufbedingung überschreibt folgende Konfigurationen:

- Vorlauf setzt Startverzug außer Kraft
- Vorlauf oder Nachlauf setzt Stopp-Trigger außer Kraft

- Nachlauf setzt Anzahlbegrenzung außer Kraft

### Wirksamkeit von Einstellungen

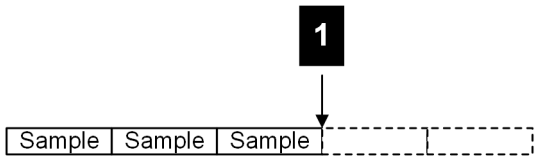
Alle mit Set- und Remove-Funktionen durchgeführten Einstellungen werden stets beim nachfolgenden Starten einer Aufzeichnung wirksam. Laufende Aufzeichnungen werden durch Set- und Remove-Aufrufe nicht unmittelbar beeinflusst.

Ausnahmen sind das An- und Abmelden von Variablen. Diese Funktionen wirken auch bei laufender Aufzeichnung und ermöglichen so eine dynamische Variablenkonfiguration ohne Unterbrechung von Wertverläufen.

## 15.3 Auslesen des Puffers

Die im Puffer eines Profils gespeicherten Samples lassen sich jederzeit auslesen, auch bei laufender Aufzeichnung. Jedes Profil führt eine Schreibposition mit, an welcher die nächste Einzelaufzeichnung gespeichert wird. Diese Position kann abgefragt werden.

**Beispiel**



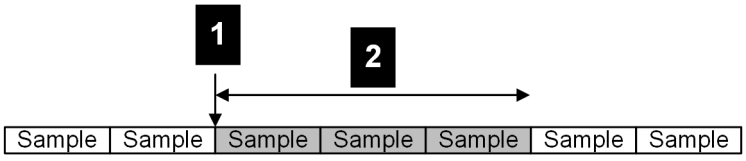

| | |
|---|---|
| 1 ... Schreibposition | |

Bei Verwendung von Triggern werden die Positionen, an welchen ein Aufruf das Eintreten der entsprechenden Trigger-Bedingung erkannt hat, geliefert.

Bei Verwendung eines Start- und Stopp-Triggers ist die Differenz zwischen den gelieferten Trigger-Positionen die Anzahl der aufgezeichneten Samples - 1, da die Stopp-Position die Position des letzten aufgezeichneten Samples bezeichnet.

Die Lesefunktionen ermöglichen das Auslesen beliebiger Pufferbereiche, wobei ein Bereich durch Startposition und Sample-Anzahl angegeben wird.

**Beispiel**



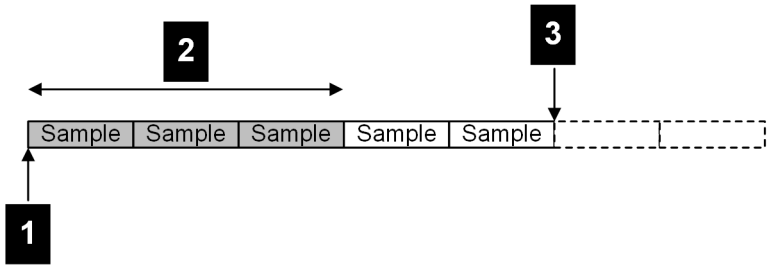

| | |
|---|---|
| 1 ... Startposition | 2 ... Anzahl |

Um schritthaltendes Lesen zu ermöglichen, wird neben Daten auch eine Position geliefert, welche beim nächsten Aufruf als Startposition verwendet werden kann.

Neben den gelieferten Lesepositionen werden für erstmaliges Lesen reservierte Positionen für die ältesten und neuesten Samples angeboten.

**Beispiel**

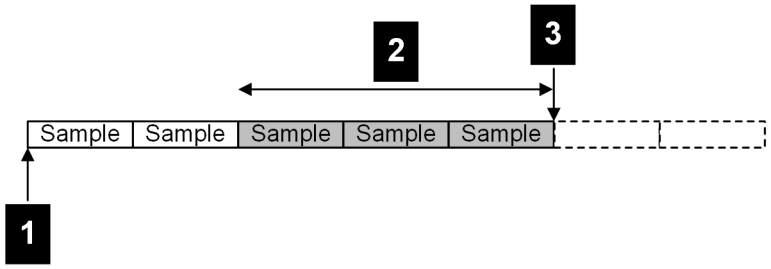
Startposition: Ältestes Sample




| | |
|---|---|
| 1 ... Pufferstart | 2 ... Anzahl |
| 3 ... Schreibposition | |

**Beispiel**

Startposition: Neuestes Sample




| | |
|---|---|
| 1 ... Pufferstart | 2 ... Anzahl |
| 3 ... Schreibposition | |

Im Zusammenhang mit dem Puffertyp "Continuous" ist die Gültigkeit einer Leseposition begrenzt. Nach Erreichen des Pufferendes bricht der Schreibindex um, wodurch bestehende Samples überschrieben werden. Die Lesefunktionen erkennen, wenn sich eine Startposition auf Samples bezieht, welche mittlerweile überschrieben wurden. In diesem Fall können die angeforderten Daten nicht geliefert werden. Der Ausgangsparameter für tatsächlich gelieferte Samples enthält dann entsprechend weniger Werte. Die gelieferte Leseposition für nachfolgende Aufrufe bezieht sich auf den aktuellen Pufferstand.

## 15.4 Aufzeichnungsrückruf aus Applikation

Für manche Anwendungen kann es sinnvoll sein, die Aufzeichnung von Prozessvariablen in Datenrekordern mit prozessnaher Prüfung und/oder Weiterverarbeitung zu kombinieren. Im Zuge jeder Einzelaufzeichnung wird eine Variable nur einmal gelesen, der Wert lässt sich aber für mehrere Aktionen verwenden.

Zu diesem Zweck kann ein Datenzeiger registriert werden, welche am Ende einer Einzelaufzeichnung aufgerufen wird und als Argumente das Ergebnis der Aufzeichnung sowie den Datenzeiger aus der Anmeldung erhält. Zur Aufzeichnung gehören folgende Informationen:

- Zeitstempel des Samples
- Aufzeichnungsposition
- Liste der gespeicherten Variablen-Samples bestehend aus Variablen-ID und –wert

Alle Informationen stehen nur lesend zur Verfügung. Sie werden bereits vor Aufruf der Applikationsfunktion im Profilpuffer gespeichert. Damit wird zum einen Datentreue gewährleistet, zum anderen ist die Periodizität der Aufzeichnung von Schwankungen unabhängig, die durch ungleichmäßige Laufzeit der Applikationsfunktion entstehen.

***Information***

*Die Applikationsfunktion sollte eine möglichst kurze und konstante Laufzeit haben. Falls für das Profil eine automatische Aufzeichnung konfiguriert wurde, darf sie keine blockierenden Aufrufe enthalten, da unbegrenzte Wartezeiten negative Auswirkungen auf die Aufzeichnung anderer Profile haben.*

## 15.5 Datenlokalität und Echtzeit

Das Variablenserver-Konzept, auf welchem die Datenaufzeichnung in Profilen basiert, ist prozessübergreifend definiert. Das bedeutet, dass Variablen physisch auf unterschiedliche Prozesse des Systems verteilbar sind. Der Determinismus von Datenaufzeichnungen hängt somit von der Datenlokalität ab: Zugriffe auf prozesslokale Variablen dauern kürzer, Zugriffe auf entfernte Variablen mittels Interprozesskommunikation dauern länger.

Um möglichst hohen Determinismus für echtzeitrelevante Aufzeichnungen zu erzielen, sollten prozesslokale Variablen mit Echtzeitqualität nicht zusammen mit entfernten Variablen im selben Profil angemeldet werden. Statt dessen empfiehlt es sich, Profile mit Echtzeitqualität, die möglicherweise auch in kürzeren Zyklen und auf höherer Priorität abtasten, von übrigen Profilen zu trennen.

## 15.6 Größenbeschränkungen

Bedingt durch die aktuelle technische Realisierung für 32-Bit-Systeme (max. 4 GB virtueller Adressraum, max. 32 Bit) gelten in Bezug auf die Datenmenge folgende theoretische Begrenzungen:

| max. Pufferkapazität | max. Variablenanzahl |
|---|---|
| 524.288 | 1 |
| 262.144 | 2 |

| max. Pufferkapazität | max. Variablenanzahl |
|---|---|
| 131.072 | 4 |
| 65.536 | 8 |
| 32.768 | 16 |
| 16.384 | 32 |
| 8.192 | 64 |
| 4.096 | 128 |
| 2.048 | 256 |

**Beispiel**

An Profilen mit zwischen 32.769 und 65.536 Samples lassen sich max. 8 Variablen zur Aufzeichnung anmelden.

Profile mit Puffertyp SingleShot

| max. Pufferkapazität | max. Variablenanzahl |
|---|---|
| 1.048.576 | 254 |
| 1.042.453 | 256 |

An Profilen mit max. Kapazität 1.048.576 lassen sich max. 254 Variablen zur Aufzeichnung anmelden. Umgekehrt lässt sich das Maximum von 256 Variablen nur an Profilen mit max. Kapazität 1.042.453 anmelden. Die Maxima beziehen sich auf 4 GB, für Linux mit z.B. 1 GB User-Adressraum sind die Angaben zu vierteln und je nach Speicherausbau nochmals zu reduzieren.

## 15.7 Anhang: C Schnittstelle

Folgendes Kapitel enthält Beispiele zur Verwendung der Programmierschnittstelle `DataRecApi` für C-Applikationen.

Alle Beispiele verwenden folgende Konfiguration:

```
[Xcrt]
    traceWord = -1

[Xcrt.Resource:0]
    name = "resource7"

[Xcrt.Resource:0.Task:0]
    name = "Task_1s"
    interval = 1000000
    priority = 5

[Xcrt.Module:0]
    codeFile = "libTestDataRecApp"
    moduleInitFunc="moduleInit"
    moduleStartFunc="moduleStart"
    moduleStopFunc="moduleStop"
    moduleExitFunc="moduleExit"

    [Xcrt.Module:0.TaskConnection:0]
        context = "resource7.Task_1s"
        prio = 1
        hookClientFunc = "moduleCallback"
```

Weiters haben alle Beispiele folgenden Kopfteil gemeinsam:

```
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include "TestDataRecApp.h"
#include "MemApi.h"
#include "LogApi.h"
#include "DataRecApi.h"
/
*****************************************************************************
* interface functions
*/
ModuleInitFunc moduleInit = &ModuleInit;
ModuleStartFunc moduleStart = &ModuleStart;
ModuleStopFunc moduleStop = &ModuleStop;
ModuleExitFunc moduleExit = &ModuleExit;
HookClientFunc moduleCallback = &ModuleCallback;
/
*****************************************************************************
* macros
*/
#define ReturnIfNot(cond) \
if (!(cond)) { LogApiTrace(traceid, "### ReturnIfNot@line %d ###",
__LINE__);return;}
/
*****************************************************************************
* constants
*/
static const char scProfileName[] = "TestDataRecAppP1";
static const char scVarNameTemp[] = "temperature";
static const char scVarNamePres[] = "pressure";
/
*****************************************************************************
* static variables
*/
static int32_t sTemp;
static float sPres;

static DataRecApiProfileId sP1Id;
static DataRecApiVarId sTempId;
static DataRecApiVarId sPresId;
```

## Allgemeine Hinweise

Automatische Aufzeichnungen sind in C-Applikationen mit der Funktion `DataRecApiSetRecContext` zu konfigurieren. Die Funktion definiert einen Task-Kontext für die Aufzeichnung anhand folgender Angaben:

- Zeitlicher Abstand der Einzelaufzeichnungen in µs
- Task-Priorität der Einzelaufzeichnungen (von 1 = hoch bis 31 = niedrig)
- CPU-Bindung des Tasks (ab 0, oder -1 für systemseitig gewählte Standard-Bindung)

Mit diesen Einstellungen wird systemseitig ein Task angelegt, der sich bezüglich Periodizität und Priorität unter die übrigen Applikationstasks einreiht. Zur Erfüllung bestimmter Anforderungen an den Determinismus der Aufzeichnung ist das Zusammenwirken dieses Tasks mit übrigen Applikationstasks zu berücksichtigen. Lassen die Zeitstempel der Daten z.B. unerwünschten Jitter erkennen, ist die Wahl der obigen Parameter zu überdenken. Weiters sollte die Priorität nicht zu hoch, der Abstand nicht zu kurz und die Datenmenge nicht zu groß sein, um eine Blockierung echtzeitkritischer Tasks zu vermeiden.

***Information***

*Wird für ein Profil manuelle Aufzeichnung gewählt, ist zu beachten, dass Aufrufe von `DataRecApiSampleValues` nicht nur zur Datenaufzeichnung notwendig sind, sondern auch für Zustandsüberführungen im Profil. Bei automatischer Aufzeichnung erfolgen die Aufrufe und damit verbundene Zustandsüberführungen durch den konfigurierten Task. Bei manueller Aufzeichnung muss dies in der Applikation realisiert werden.*

So umfasst ein Aufruf von `DataRecApiSampleValues` z.B. folgende Zusatzschritte, welche einen Zustandswechsel von `waiting` nach `recording` bewirken können:

- Prüfung einer konfigurierten Start-Trigger-Bedingung auf deren Eintreten
- Prüfung auf Ablauf eines mit `DataRecApiSetRecCount` eingestellten Startverzugs

Weiters bewirkt `DataRecApiSampleValues` im Falle des Puffertyps SingleShot beim Erreichen des Pufferendes einen Zustandswechsel von `recording` auf `stopped`.

Mittels der Funktion `DataRecApiSetRecMode` kann der Aufzeichnungsmodus (Standard oder Änderunsbasiert) eine Profils konfiguriert werden. Im Änderungsbasierten Modus kann nachträglich der Toleranzwert mit `DataRecApiSetVarTolerance` geändert oder mit `DataRecApiRemoveVarTolerance` aufgehoben werden.

***Information***

*In den änderungsbasierten Modi ist zu beachten, dass ein Profil Änderungen von Variablenwerten nur im Zuge des Aufrufs von `DataRecApiSampleValues` feststellt. Das Schreiben der Applikation auf die überwachte Variable alleine bewirkt noch keine Aufzeichnung!*

Über die Funktion `DataRecApiSetRecCount` können folgende Begrenzungen eingestellt werden:

- Startverzug: Anzahl der Aufrufe von `DataRecApiSampleValues` nach Aufzeichnungsstart, die keine Aufzeichnungen erzeugen, sondern einen Countdown für die Aufzeichnung realisieren.
- Anzahl der zu speichernden Aufzeichnungen: Anzahl der Aufrufe von `DataRecApiSampleValues` (ohne Startverzug) bis zum automatischen Aufzeichnungsstopp.

Sonderfall im Modus änderungsbasierte Aufzeichnung mit einer überwachten Variablen: Für die Anzahl der zu speichernden Aufzeichnungen zählt nicht die bloße Anzahl von Aufrufen von `DataRecApiSampleValues`, da nicht jeder Aufruf eine Speicherung zur Folge haben muss. Sondern es zählen nur die tatsächlich aufzeichnenden Aufrufe der Funktion. In den übrigen Modi zählen hingegen die reinen Aufrufe von `DataRecApiSampleValues`.

Trigger lassen sich mit `DataRecApiSetTrigger` einstellen und mit `DataRecApiRemoveTrigger` wieder aufheben.

Sonderfall im Modus änderungsbasierte Aufzeichnung mit einer überwachten Variablen: Für die Anzahl der zu speichernden Aufzeichnungen zählt nicht die bloße Anzahl von Aufrufen von `DataRecApiSampleValues`, da nicht jeder Aufruf eine Speicherung zur Folge haben muss. Sondern es zählen nur die tatsächlich aufzeichnenden Aufrufe der Funktion. In den übrigen Modi zählen hingegen die reinen Aufrufe von `DataRecApiSampleValues`.

Die Anmeldefunktion `DataRecApiAddVar` ermittelt im Zuge der Anmeldung die Lokalität einer Variablen. Diese Informationen wird über die Abfragefunktionen `DataRecApiGetFirstVar`, `DataRecApiGetNextVar` und `DataRecApiGetVars` dem Anwender zur Verfügung gestellt und lässt sich zu Prüfzwecken verwenden. Das gelieferte Flag `isLocal` gibt an, ob die Variable prozesslokal ist und somit echtzeitfähig abgetastet werden kann.

Mit `DataRecApiSetSampleHook` durchgeführte Einstellungen sind nicht dauerhaft (Code-Adressen sind hochlaufabhängig), sondern müssen im Zuge jedes Applikationshochlaufs (vor dem Aufzeichnungsstart) neu durchgeführt werden.

### 15.7.1 Profil mit manueller Aufzeichnung

Das folgende Beispiel zeigt die einfachste Anwendung: Die Applikation registriert über `MemApi` zwei Variablen beim Variablenserver und meldet deren Pfade bei einem Profil mit Puffertyp Continuous und einer Kapazität von 1000 Samples an.

In der Start-Funktion wird die Aufzeichnung gestartet, in der Stopp-Funktion wieder gestoppt. Die Applikationsfunktion `ModuleCallback` wird im Sekundentakt aufgerufen und zeichnet mittels Aufruf von `DataRecApiSampleValues` die Variablen auf.

Ein explizites Löschen des Profils in der Exit-Funktion kann entfallen, da nach Beenden aller Applikationen alle noch bestehenden Profile automatisch gelöscht werden. Persistierte Daten sind von diesem Löschen nicht betroffen.

```
void ModuleInit() {
    DataRecApiInfo info;
    DataRecApiResult rc;
    char varPath[80];
    LogApiAddId(traceid, "TestRec");
    LogApiTrace(traceid, "TestDataRecApp initialized");
    MemApiInit();
    MemApiAddVar(scVarNameTemp, MemApiSInt32, &sTemp, sizeof sTemp);
    MemApiAddVar(scVarNamePres, MemApiReal, &sPres, sizeof sPres);
    sTemp = 0;
    sPres = 0.0;
    strcpy(info.name, scProfileName);
    info.size = 1000; /* space for 1000 samples */
    info.maxVarCnt = 5; /* max. 5 variables per sample */
    info.bufType = DataRecApiBufTypeContinuous; /* ring buffer type */
    info.level = DataRecApiLevelApplRt; /* (unsupported yet) */
    info.autoStart = 0; /* application starts recording */
    rc = DataRecApiCreateProfile(&info, &sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk && sP1Id != DataRecApiNoId);
    snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNameTemp);
    rc = DataRecApiAddVar(sP1Id, varPath, 0, &sTempId);
    ReturnIfNot(rc == DataRecApiResultOk && sTempId != DataRecApiNoId);
```

```
    snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNamePres);
    rc = DataRecApiAddVar(sP1Id, varPath, 0, &sPresId);
    ReturnIfNot(rc == DataRecApiResultOk && sPresId != DataRecApiNoId);
}

/
*****************************************************************************/
void ModuleExit() {
    MemApiRemoveVar(scVarNamePres);
    MemApiRemoveVar(scVarNameTemp);
    MemApiExit();
    LogApiTrace(traceid, "TestDataRecApp unloaded");
    LogApiRemoveId(traceid);
}

/
*****************************************************************************/
void ModuleStart() {
    DataRecApiResult rc;
    LogApiTrace(traceid, "Starting TestDataRecApp");
    rc = DataRecApiStartRecording(sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk);
}
/
*****************************************************************************/
void ModuleStop() {
    DataRecApiResult rc;
    LogApiTrace(traceid, "Stopping TestDataRecApp");
    rc = DataRecApiStopRecording(sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk);
}
/
*****************************************************************************/
void ModuleCallback(UserFuncParam *arg, uint32_t provArg) {
    DataRecApiResult rc;
    rc = DataRecApiSampleValues(sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk);
    sTemp += 1;
    sPres += 0.1;
}
```

## 15.7.2 Profil mit automatischer Aufzeichnung

Das folgende Beispiel erweitert das Vorangegangene in zwei Punkten: Zum einen erfolgt die Aufzeichnung automatisch. Dazu wird mit `DataRecApiSetRecContext` eine Abtastung im Abstand von 10ms eingestellt, weiters mit `DataRecApiSetRecCount` ein Startverzug von 2 Sekunden und eine Aufzeichnungsdauer von 5 Sekunden. Zum anderen wird ein Rückruf aus der Aufzeichnung in die Applikation zwecks Minimum-Maximum-Ermittlung konfiguriert.

```
void ModuleInit() {
    DataRecApiInfo info;
    DataRecApiResult rc;
    char varPath[80];
    DataRecApiContext ctx;
    DataRecApiCount cnt;

    /* for comitted lines see example "manual recording" */

    snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNamePres);
    rc = DataRecApiAddVar(sP1Id, varPath, 0, &sPresId);
    ReturnIfNot(rc == DataRecApiResultOk && sPresId != DataRecApiNoId);

    ctx.intervalUs = 10000;
    ctx.priority = 16;
```

```
    ctx.affinity = -1;
    rc = DataRecApiSetRecContext(sP1Id, &ctx);
    ReturnIfNot(rc == DataRecApiResultOk);

    cnt.startDelayCnt = 200;
    cnt.durationCnt = 500;
    rc = DataRecApiSetRecCount(sP1Id, &cnt);
    ReturnIfNot(rc == DataRecApiResultOk);

    rc = DataRecApiSetSampleHook(sP1Id, UpdateMinMaxTemp, 0);
    ReturnIfNot(rc == DataRecApiResultOk);
}
```

Rückruffunktion:

```
static int32_t sMinTemp = INT32_MAX, sMaxTemp = INT32_MIN;
/
*****************************************************************************
*/
static void UpdateMinMaxTemp(
    uint64_t timeStampUs, /**< [in] time stamp (µs since the epoch) */
    DataRecApiPos pos, /**< [in] sample position */
    DataRecApiVarSample *pSamples, /**< [in] variable samples */
    int32_t sampleCnt, /**< [in] length of \em pSamples */
    void *arg /**< [in] user argument */
    ) {
    int i;

    for (i = 0; i < sampleCnt; ++i) {
        if (pSamples[i].var == sTempId) {
            int32_t curTemp = pSamples[i].value.valSINT32;
            if (curTemp < sMinTemp) {
                sMinTemp = curTemp;
            }
            if (curTemp > sMaxTemp) {
                sMaxTemp = curTemp;
            }
        }
        return;
    }
}
```

## 15.7.3 Profil mit getriggerter Aufzeichnung

Das folgende Beispiel verwendet manuelle Aufzeichnung, wobei die Aufzeichnungsspanne durch einen Start- und einen Stopp-Trigger begrenzt wird. Beide Trigger-Bedingungen beziehen sich auf die von der Applikation angelegten booleschen Variable `active`. Der Start der Aufzeichnung soll durch einen Übergang von FALSE auf TRUE ausgelöst werden, der Stopp durch einen Übergang von TRUE auf FALSE. Diese Bedingungen sind mit Hilfe der Über- und Unterschreitungsoperatoren spezifizierbar. Die Variable `active` muss dazu anfangs den Wert FALSE haben.

Um den Pfad der Variablen im Variablenbaum angegeben zu können, muss sie wie die aufgezeichneten Variablen mittels `MemApi` registriert werden. Im Gegensatz zur überwachten Variablen einer änderungsbasierten Aufzeichnung muss sie jedoch nicht beim Profil angemeldet sein.

Setzt man die Variable `active` auf TRUE und einige Sekunden später wieder auf FALSE (im Beispiel nicht Bestandteil des Applikationscodes), werden innerhalb dieser Spanne die Variablen `temperature` und `pressure` auf-

gezeichnet. Nach gestoppter Aufzeichnung ergibt sich die Anzahl der Samples aus der Differenz der Trigger-Positionen + 1, da die Stopp- Position die Position des letzten Samples angibt.

```
/
*****************************************************************************
* constants
*/
enum { cFALSE = 0, cTRUE = 1 };
static const char scProfileName[] = "TestDataRecAppP1";
static const char scVarNameTemp[] = "temperature";
static const char scVarNamePres[] = "pressure";
static const char scVarNameAct[] = "active";

/
*****************************************************************************
* static variables
*/
static int32_t sTemp;
static float sPres;
static int8_t sActive;
static DataRecApiProfileId sP1Id;
static DataRecApiVarId sTempId;
static DataRecApiVarId sPresId;

/
*****************************************************************************/
void ModuleInit() {
    DataRecApiInfo info;
    DataRecApiTrigger trig;
    DataRecApiResult rc;
    char varPath[80];

    LogApiAddId(traceid, "TestRec");
    LogApiTrace(traceid, "TestDataRecApp initialized");

    MemApiInit();
    MemApiAddVar(scVarNameTemp, MemApiSInt32, &sTemp, sizeof sTemp);
    MemApiAddVar(scVarNamePres, MemApiReal, &sPres, sizeof sPres);
    MemApiAddVar(scVarNameAct, MemApiBool, &sActive, sizeof sActive);

    sTemp = 0;
    sPres = 0.0;
    sActive = cFALSE;

    strcpy(info.name, scProfileName);
    info.size = 1000; /* space for 1000 samples */
    info.maxVarCnt = 5; /* max. 5 variables per sample */
    info.bufType = DataRecApiBufTypeContinuous; /* ring buffer type */
    info.level = DataRecApiLevelApplRt; /* (unsupported yet) */
    info.autoStart = 0; /* application start recording */

    rc = DataRecApiCreateProfile(&info, &sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk && sP1Id != DataRecApiNoId);

    snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNameTemp);
    rc = DataRecApiAddVar(sP1Id, varPath, 0, &sTempId);
    ReturnIfNot(rc == DataRecApiResultOk && sTempId != DataRecApiNoId);

    snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNamePres);
    rc = DataRecApiAddVar(sP1Id, varPath, 0, &sPresId);
    ReturnIfNot(rc == DataRecApiResultOk && sPresId != DataRecApiNoId);

    memset(&trig, 0, sizeof trig);
    snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNameAct);
    trig.startCond.op = DataRecApiTriggerOpPosSlope;
    trig.startCond.val.valBOOL = cFALSE;
    strncpy(trig.startCond.var, varPath, sizeof trig.startCond.var - 1);
    trig.stopCond.op = DataRecApiTriggerOpNegSlope;
    trig.stopCond.val.valBOOL = cTRUE;
    strncpy(trig.stopCond.var, varPath, sizeof trig.stopCond.var - 1);
```

```
    rc = DataRecApiSetTrigger(sP1Id, &trig);
    ReturnIfNot(rc == DataRecApiResultOk);

}

/
*****************************************************************************/
void ModuleExit() {
    MemApiRemoveVar(scVarNameAct);
    MemApiRemoveVar(scVarNamePres);
    MemApiRemoveVar(scVarNameTemp);
    MemApiExit();

    LogApiTrace(traceid, "TestDataRecApp unloaded");
    LogApiRemoveId(traceid);
}

/
*****************************************************************************/
void ModuleStart() {
    DataRecApiResult rc;
    LogApiTrace(traceid, "Starting TestDataRecApp");
    rc = DataRecApiStartRecording(sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk);
}

/
*****************************************************************************/
void ModuleStop() {
    DataRecApiResult rc;
    LogApiTrace(traceid, "Stopping TestDataRecApp");
    rc = DataRecApiStopRecording(sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk);
}

/
*****************************************************************************/
void ModuleCallback(UserFuncParam *arg, uint32_t provArg) {
    DataRecApiSampleValues(sP1Id);
    sTemp += 1;
sPres += 0.1;
}
```

### 15.7.4 Profil mit Persistierung

Das folgende Beispiel unterscheidet sich stärker von den Vorangegangenen. Es verwendet ein Profil mit Persistierung und muss somit mehrere Fälle behandeln: Zum einen, dass das Profil noch nicht existiert und neu angelegt werden muss, zum anderen, dass das Profil aufgrund der bestehenden Persistierung bereits systemseitig angelegt und somit nur mehr neu gestartet werden muss.

Die Wiederherstellung von Profilen erfolgt zwischen den Init- und Start-Aufrufen der Applikation, da auch die Pfade der konfigurierten Variablen persistiert werden und diese erst nach Initialisierung aller Laufzeitsysteme gültig sind. Dies erfordert eine Änderung des Applikationsaufbaus: Das Ermitteln oder erstmalige Erzeugen des Profils erfolgt somit im Zuge des Start-Aufrufs. Dieser Schritt wird in einer Extrafunktion `InitRecording` integriert. Auch die übrigen Schritte werden in separate Funktionen herausgehoben, um die Aufzeichnung besser vom übrigen Applikationscode zu entkoppeln.

Ebenfalls im Zuge der Initialisierung erfolgt das Anlegen eines Applikationstasks, welcher das Profil alle 10 Sekunden komplett persistiert. Die Funktion `StartRecording` veranlasst den Start des Task, die Funktion `StopRecording` lässt den Task terminieren.

```
static void InitRecording() {
    DataRecApiInfo info;
    DataRecApiResult rc;
    char varPath[80];

    /*
     * create new profile or get restored profile
     */
    sP1Id = DataRecApiGetProfile(scProfileName, &info);
    if (sP1Id == DataRecApiNoId) { /* create profile */
        strcpy(info.name, scProfileName);
        info.size = 1000;
        info.maxVarCnt = 5;
        info.bufType = DataRecApiBufTypeContinuous;
        info.level = DataRecApiLevelApplRt;
        info.autoStart = 0;
        rc = DataRecApiCreateProfile(&info, &sP1Id);
        ReturnIfNot(rc == DataRecApiResultOk && sP1Id != DataRecApiNoId);

        snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNameTemp);
        rc = DataRecApiAddVar(sP1Id, varPath, 0, &sTempId);
        ReturnIfNot(rc == DataRecApiResultOk && sTempId != DataRecApiNoId);

        snprintf(varPath, sizeof varPath, "APPL.MEM.%s", scVarNamePres);
        rc = DataRecApiAddVar(sP1Id, varPath, 0, &sPresId);
        ReturnIfNot(rc == DataRecApiResultOk && sPresId != DataRecApiNoId);

        {
            DataRecApiContext ctx;
            ctx.intervalUs = 2000000;
            ctx.priority = 16;
            ctx.affinity = -1;
            rc = DataRecApiSetRecContext(sP1Id, &ctx);
            ReturnIfNot(rc == DataRecApiResultOk);
        }
    } else { /* profile restored */
        rc = DataRecApiStopRecording(sP1Id);
    }
    sRun = 1;
    sStarted = 0;
    sTerminated = 0;
    sTaskHdl = CreateTask("LowPriorTask", LowPriorTask, 0, 24, 4096);
    ReturnIfNot(sTaskHdl != 0);
}

/
*****************************************************************************/
static void StartRecording() {
    DataRecApiResult rc;
    int ok;

    rc = DataRecApiStartRecording(sP1Id);
    ReturnIfNot(rc == DataRecApiResultOk);

    ok = ResumeTask(sTaskHdl); ReturnIfNot(ok);
    sStarted = 1;
}

/
*****************************************************************************/
static void StopRecording() {
    DataRecApiResult rc;
    DataRecApiPos pos;
    sRun = 0; /* tell LowPriorTask to terminate */

    rc = DataRecApiStopRecording(sP1Id);
```

```
    ReturnIfNot(rc == DataRecApiResultOk);
}

/
*****************************************************************************/
static void ExitRecording() {
    if (sStarted) {
        while (!sTerminated) {
            LogApiTrace(traceid, "waiting for LowPriorTask to terminate...");
            sleep(1/*secs*/);
        }
    }
}

/
*****************************************************************************/
void ModuleInit() {
    /*
     * create variables to be recorded
    */
    MemApiInit();
    MemApiAddVar(scVarNameTemp, MemApiSInt32, &sTemp, sizeof sTemp);
    MemApiAddVar(scVarNamePres, MemApiReal, &sPres, sizeof sPres);
}

/
*****************************************************************************/
void ModuleExit() {
    ExitRecording();
    MemApiRemoveVar(scVarNamePres);
    MemApiRemoveVar(scVarNameTemp);
    MemApiExit();
}

/
*****************************************************************************/
void ModuleStart() {
    /*
     * now all variables are available such that profiles can be restored and
     * variables can be added to profiles
     */
    InitRecording();
    sTemp = 0;
    sPres = 0.0;
    StartRecording();
}

/
*****************************************************************************/
void ModuleStop() {
    StopRecording();
}
```

## Implementierung der Persistierung im Abstand von 10 Sekunden (mit zusätzlicher Terminierungsprüfung pro Sekunde):

```
/* libk2ctrl exports for advanced applications */
extern int CreateTask(const char *name, void (*fctAddr)(void *), void *param,
        int prior, int stackSize);
extern int ResumeTask(int hdl);

/
*****************************************************************************
* static variables
*/
static int sTaskHdl;
static int sRun, sStarted, sTerminated;

/
*****************************************************************************/
void LowPriorTask(void *arg) {
```

```
    enum { cSaveStatePeriodSecs = 10 };
    int cnt = cSaveStatePeriodSecs;
    DataRecApiResult rc;
    while (sRun) {
        sleep(1/*secs*/);
        if (--cnt == 0) {
            rc = DataRecApiSaveState(sP1Id);
            cnt = cSaveStatePeriodSecs;
        }
    }
    sTerminated = 1;
}
```

***Information***

*Mit der Erzeugung freilaufender Tasks mittels `CreateTask` übernimmt die Applikation Mitverantwortung für die Systemstabilität.*

*Es ist darauf zu achten, dass nach der Rückkehr aus der Exit-Funktion (Beenden der Applikation) alle erzeugten Tasks terminiert wurden bzw. diese **keine** Aufrufe der angebotenen APIs (u.a. `DataRecApi`) mehr ausführen!*

### 15.7.5 Auswertung von Aufzeichnungen

Der folgende Code ergänzt die vorangehenden Beispiele um die Auswertung durchgeführter Aufzeichnungen. Die Parameter der Beispielfunktion `PrintSamples` spezifizieren den zu lesenden Pufferbereich in vertikaler (`pos, cnt`) und horizontaler Richtung (`maxVarCnt`).

```
static void PrintSamples(int maxVarCnt, DataRecApiPos pos, int cnt) {
    char *pBuf;
    DataRecApiSample *pSample;
    DataRecApiVarSample *p;
    int actVarCnt, i, k, rc;

    pBuf = malloc(DataRecApiUtilGetSampleMemSize(maxVarCnt, cnt));
    ReturnIfNot(pBuf != 0);

    pSample = (DataRecApiSample *)pBuf;
    rc = DataRecApiReadSamples(sP1Id, maxVarCnt, &pos, pSample, &cnt);
    ReturnIfNot(rc == DataRecApiResultOk);

    for (i = 0; i < cnt; ++i) {
        actVarCnt = pSample->header.varSampleCnt;
        printf("[%d] recNo=%d, varSampleCnt=%d, timeStampUs=[%s]:\n",
        i,
        pSample->header.sampleRecNo,
        actVarCnt,
        TimeStamp2DateStr(pSample->timeStampUs));

    for (k = 0, p = pSample->varSamples; k < actVarCnt; ++k, ++p) {
        switch (p->type) {
            case DataRecApiVarType_SINT32:
             printf(" name=%s, value=%d (size=%d)\n", VarIdToName(p->var), p-
>value.valSINT32, p->size);
             break;
            case DataRecApiVarType_REAL:
             printf(" name=%s, value=%f (size=%d)\n", VarIdToName(p->var), p-
>value.valREAL, p->size);
             break;
            default:
             printf(" name=%s, value=%" PRIx64 " (type=%d, size=%d)\n", Va-
rIdToName(p->var), p->value.valNONE, p->type, p->size);
             break;
```

```
        }
    }
    pSample = (DataRecApiSample *)&pSample->varSamples[maxVarCnt];
    }
    free(pBuf);
}
```

## Beispielaufruf

```
PrintSamples(5, DataRecApiPosNewest, 1000);
```

## Hilfsfunktionen zur Formatierung von Zeitstempeln und einfachen Ermittlung von Variablennamen:

```
static const char *TimeStamp2DateStr(uint64_t timeStamp) {
static char sDateStr[80];
    time_t secs, usecs;
    struct tm *p;

    /* seconds and µs since 1970-01-01 */
    secs = (int)(timeStamp / 1000000);
    usecs = (int)(timeStamp % 1000000);

    p = localtime(&secs);
    if (p != 0) {
        snprintf(sDateStr, sizeof sDateStr, "%04u-%02u-%02u %02u:%02u:%02u.
%03u", p->tm_year+1900, p->tm_mon+1, p->tm_mday, p->tm_hour, p->tm_min, p-
>tm_sec, (unsigned)usecs);
    }
    else {
        snprintf(sDateStr, sizeof sDateStr, "xxxx-xx-xx xx:xx:xx.xxx");
    }
    return sDateStr;
}

/
*****************************************************************************/
static const char *VarIdToName(DataRecApiVarId varId) {
    if (varId == sTempId) {
        return scVarNameTemp;
    }
    if (varId == sPresId) {
        return scVarNamePres;
    }
    return "unknown";
}
```

# 16 Diagnose

In diesem Kapitel werden die Diagnosemöglichkeiten von u-create beschrieben. Je nach Lieferumfang können nicht alle Diagnosemöglichkeiten zur Verfügung stehen.

## 16.1 Steuerungsdiagnose

Die Anzeige von Fehlern im Betrieb oder von Betriebszuständen erfolgt über die LEDs auf der CPU-Baugruppe.

**PWR (Power)**

| Anzeige | Bedeutung |
|---|---|
| Dunkel | Keine Versorgungsspannung |
| Grün | Gerät läuft |

**SF (System Fault)**

| Anzeige | Bedeutung |
|---|---|
| Dunkel | Keine Versorgungsspannung oder kein Fehler |
| Rot | Schwerer Systemfehler (z.B. Fatal Error) |

**RUN**

| Anzeige | Bedeutung |
|---|---|
| Dunkel | Keine Versorgungsspannung |
| Grün | Applikation wird abgearbeitet oder Setup beendet |
| Grün blinkend | Applikation ist gestoppt |
| Gelb | Betriebsbereit |
| Gelb blinkend | Setup wird ausgeführt |
| Rot blinkend | Setup fehlgeschlagen |

**Link/Activity-LED**

| Anzeige | Bedeutung |
|---|---|
| Dunkel | Keine Verbindung |
| Grün/Gelb blinkend | Daten werden übertragen |
| Grün | EtherCAT-Verbindung besteht (100 MBit/s, Full Duplex) |

Über die Service App "DevAdmin" können Informationen der Steuerung eingesehen werden und ein Statusreport, sowie ein Crashreport ausgelöst werden (siehe Device Administration (DevAdmin)).

## 16.2 Diagnosedaten für Weidmüller

Wenn Sie ein Problem mit dem u-create System haben, bei dem Sie Unterstützung von Weidmüller benötigen, sammeln Sie bitte von ihrem System folgende Informationen und schicken Sie diese an Weidmüller.

### 16.2.1 Statusreport

Um die Zustandsdaten der Steuerung zu sichern, benutzen Sie bitte die Funktion des Statusreports. Dieser kann folgendermaßen ausgelöst werden.

- über die Service-App (DevAdmin)

#### Statusreport über Service-App (DevAdmin) auslösen

Siehe "Device Administration (DevAdmin)".

### 16.2.2 Crashreport

Tritt ein Systemfehler (z.B. "unhandled exception") auf, wird automatisch eine Crashreport-Datei erstellt und auf dem Speichermedium der Steuerung abgelegt. Beim nächsten Steuerungshochlauf wird geprüft, ob eine neue Datei vorliegt. Falls ja, werden entsprechende Meldungen im Meldesystem abgesetzt. Generell ist darauf zu achten, dass die Meldungen zu quittieren sind. Die Crashreport-Datei kann via Service-App (siehe Device Administration (DevAdmin)) auf einen USB-Stick gespeichert werden. Nähere Informationen siehe entsprechendes Projektierungshandbuch der verwendeten Steuerung.

### 16.2.3 Zugriff auf Flash-Speichermedium

Der Zugriff auf das Flash-Speichermedium kann nur über eine SSH-verschlüsselte SFTP- oder SCP-Verbindung erfolgen. Dazu wird ein SFTP-Programm, das auf dem PC installiert ist, benötigt. Im Internet werden diverse Programme zum Download angeboten.

Um mit dem SFTP-Programm eine Verbindung zur Steuerung herzustellen, muss die IP-Adresse der Steuerung mit der Portnummer im SFTP-Programm eingegeben werden. Zusätzlich dazu werden für einen Verbindungsaufbau folgende Daten zur Authentifizierung benötigt, die ebenfalls im SFTP-Programm eingegeben werden müssen:

- Anmeldedaten
- SSH-Schlüssel

Eine detaillierte Beschreibung entnehmen Sie der Hilfe des verwendeten SFTP-Programms.

***Information***

*Das gewählte SFTP-Programm muss eine Authentifizierung mittels SSH-Verschlüsselung unterstützen.*

### Anmeldedaten

Auf der Steuerung sind folgende Benutzer standardmäßig angelegt und können für den Verbindungsaufbau verwendet werden:

- service
- admin

***Information***

*Die Benutzerdaten können nicht geändert werden.*

### SSH-Schlüssel

Aus Sicherheitsgründen ist es notwendig, das SSH-Schlüsselpaar (Private- und Public-Schlüssel) zu ändern.

## Ändern des SSH-Schlüsselpaares

Zum Generieren eines neuen SSH-Schlüsselpaares wird ein spezielles Programm benötigt. Dieses kann im Internet heruntergeladen werden, wobei das Programm "PuTTYgen" (in einer Version < 0.75) empfohlen wird.

### Benutzer "service"

Um einen neuen SSH-Schlüssel unter Verwendung von "PuTTYgen" zu ändern, gehen Sie wie folgt vor:

1) Starten von "PuTTYgen" am PC.

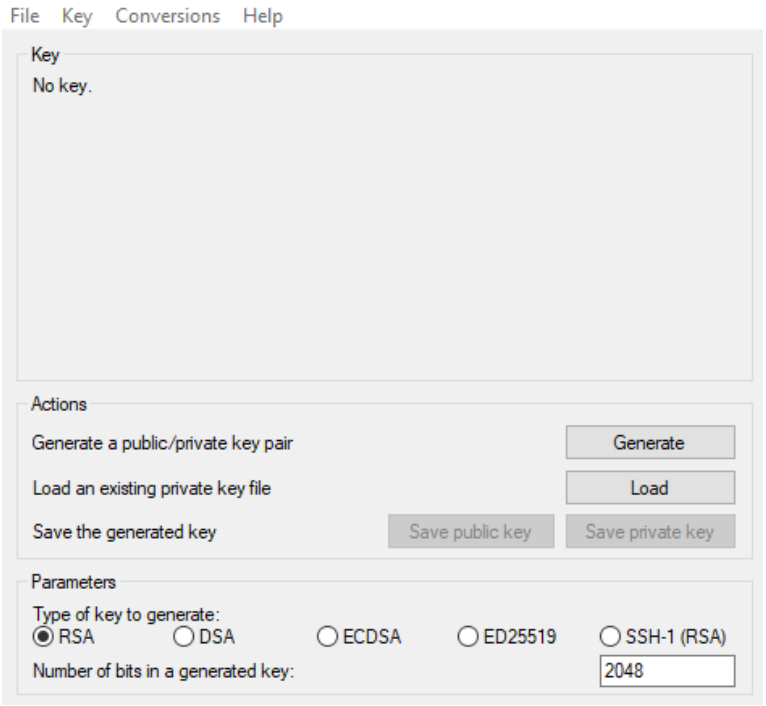
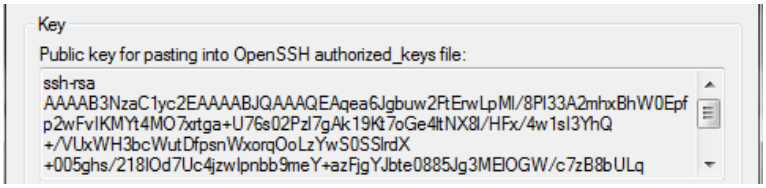



Abb. 16-19: PuTTYgen

2) Auswahl des Typs "RSA" (standardmäßig ausgewählt) im unteren Bereich des Dialogs.
3) Generieren der Schlüssel mittels "Generate".
4) Speichern des Private-Schlüssel mittels "Save private key" unter `[Dateiname]_key.ppk` in ein Verzeichnis am PC.
5) Anlegen einer neuen Textdatei `authorized_key.ppk` in einem Verzeichnis am PC.
6) Selektieren und Kopieren des gesamten Textes in dem Textfeld im oberen Bereich des Dialogs unter "Public key for pasting into OpenSSH authorized_keys file:".




7) Kopieren des Textes in die Datei `authorized_key.ppk`.
8) Speichern und schließen der Datei.

Das SSH-Schlüsselpaar wurde generiert. Der Public-Schlüssel muss danach auf die Steuerung übertragen werden.

Zum Übertragen des Public-Schlüssels für den Benutzer "service" gehen Sie wie folgt vor:

1) Starten des SFTP-Programms und Verbinden auf die Steuerung.
2) Wechseln in das Verzeichnis `ssh-key` und Download der Datei `authorized_key.ppk` vom PC auf die Steuerung (bestehende Datei wird dabei überschrieben).
3) Neustart der Steuerung.

Der SSH-Schlüssel wurde geändert. Bei einer erneuten SFTP-Verbindung zur Steuerung muss der neu erzeugte Private-Schlüssel angegeben werden.

**Benutzer "admin"**

Um einen neuen SSH-Schlüssel unter Verwendung von "PuTTYgen" zu ändern, gehen Sie wie folgt vor:

1) Starten von "PuTTYgen" am PC.

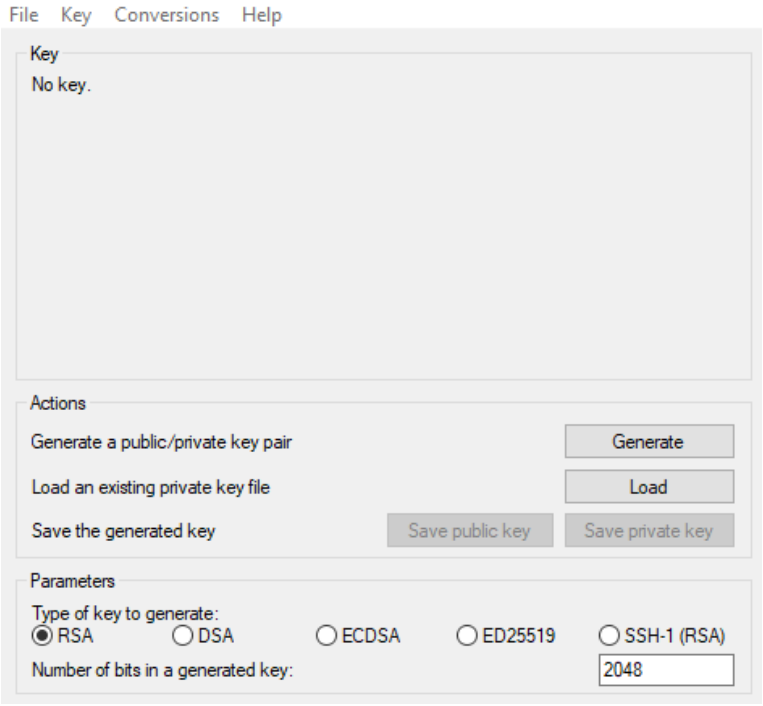
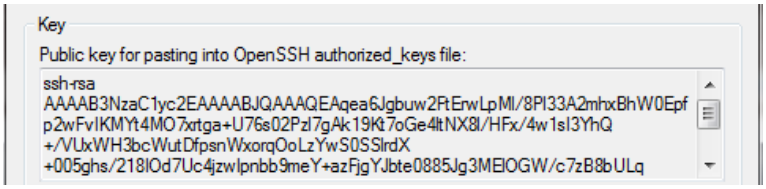



Abb. 16-20: PuTTYgen

2) Auswahl des Typs "RSA" (standardmäßig ausgewählt) im unteren Bereich des Dialogs.
3) Generieren der Schlüssel mittels "Generate".
4) Speichern des Private-Schlüssel mittels "Save private key" unter `[Dateiname]_key.ppk` in ein Verzeichnis am PC.
5) Anlegen einer neuen Textdatei `admin` in einem Verzeichnis am PC.
6) Selektieren und Kopieren des gesamten Textes in dem Textfeld im oberen Bereich des Dialogs unter "Public key for pasting into OpenSSH authorized_keys file:".




7) Kopieren des Textes in die Datei `admin`.
8) Speichern und schließen der Datei.

Das SSH-Schlüsselpaar wurde generiert. Der Public-Schlüssel muss danach auf die Steuerung übertragen werden.

Zum Übertragen des Public-Schlüssels für den Benutzer "Admin" gehen Sie wie folgt vor:

1) Rechter Mausklick auf das Symbol "Software Service" in der Taskleiste am PC.
2) Mittels "Open Software Service Path" das Verzeichnis öffnen.

3) Kopieren der Datei `admin` in folgendes Verzeichnis (eine bereits bestehende Datei kann überschrieben werden):

```
SystemSoftware\u-create studio_[version]\dsarmxilinxkebwm-[version]_arm\config\key
```

Der neue Public-Schlüssel ist nun abgelegt und wird automatisch beim nächsten Ausführen von "Create Target" (über u-create studio) mit übertragen. Damit wurde der SSH-Schlüssel geändert. Bei einer erneuten SFTP-Verbindung zur Steuerung muss der neu erzeugte Private-Schlüssel angegeben werden.

## 16.3 USB über Ethernet-Adapter

Den notwendigen Treiber, um Ethernet über USB nutzen zu können, finden Sie auf der Weidmüller Website unter: www.weidmueller.com.

***Information***

*Der USB-Schnittstelle wird automatisch die IP-Adresse 192.168.227.1 zugewiesen.*

# 17 Instandhaltung

Unter Instandhaltung des Systems fallen folgende Punkte:

- Durchführen eines Systembackups bzw. -restores.
- Durchführen eines Firmware-Update
- Informationen zu möglichen Fehlern und deren Behebung

Die Sicherung der Systemdaten umfasst folgende Informationen:

- Firmware, die auf dem Speichermedium gespeichert ist
- Applikationsprogramme und -Dateien inklusive Konfigurationsdateien, Meldungstexten
- Betriebsdaten, die von den Applikationsprogrammen gespeichert wurden
- Daten auf SRAM und EEPROM (z.B. Retaindaten, Benutzerdaten)

***Information***

*Es wird empfohlen, bereits an einem neuen System, sowie vor Änderungen am System (z.B. Firmwareupdate) ein Backup zu erstellen und dieses für den Fall einer eventuell später erforderlichen Wiederherstellung (Restore) sicher aufzubewahren.*

Die Durchführung der Maßnahmen zur Instandhaltung ist in den jeweiligen Projektierungshandbüchern beschrieben.

## 17.1 Durchführung eines Firmwareupdates der Systemkomponenten

Das u-create System ermöglicht eine zentrale, automatische Aktualisierung der auf den Hardwaremodulen laufenden Software.

Es gibt folgende Varianten des Updates:

- Update automatisch beim Steuerungs-Hochlauf ausführen

Bevor das Firmwareupdate durchgeführt werden kann, muss ein Wechseldatenträger (z.B. SD-Karte) mit den notwendigen Daten erstellt werden.

### Vorbereitung

Für die Vorbereitung des Firmware-Updates wird Folgendes benötigt:

- Ein leerer Wechseldatenträger mit mind. 2 GB
- Lauffähiges System auf der Steuerung (muss nicht den aktuellsten Versionsstand besitzen).
- Das gewünschte Speichermedium muss am PC angesteckt sein.
- Firmware-Update-Tool am PC installiert.

Das Firmware-Update-Tool stellt folgende Möglichkeiten zur Verfügung:

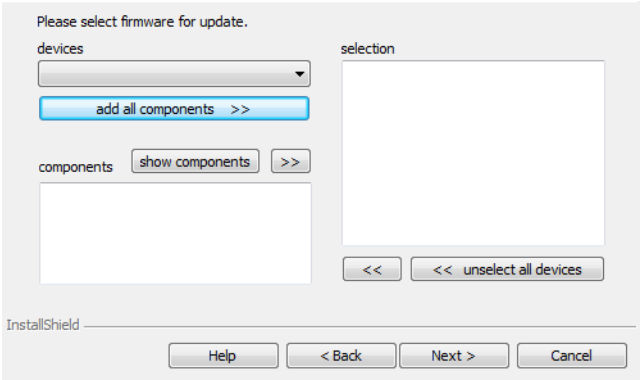
- Firmware-Update für Steuerung und Buskoppler

### 17.1.1 Firmware-Update für Geräte

Dieses Kapitel beschreibt die Erstellung eines Speichermediums zum Durchführen eines Firmware-Updates der Steuerung.

Um ein Firmware-Update auf einem Speichermedium zu erstellen, gehen Sie wie folgt vor:

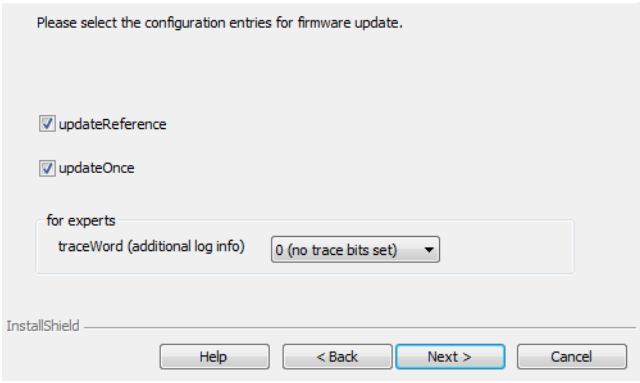
1) Starten des Firmware-Update-Tools durch Doppelklick auf "u-create FirmwareUpdate.exe".
2) Nachfolgenden Dialog mit "Next" bestätigen.
3) Auswahl des gewünschten Geräts aus der Liste "devices".
   Drücken von "show components", um alle Softwarekomponenten für die eine Aktualisierung möglich ist, in der Liste "components" anzuzeigen.
4) Selektieren der gewünschten Aktualisierung und mit >> Übernahme in die Liste "selection".




5) Durch "<<" oder "<< unselect all devices" können einzelne oder alle Aktualisierungen der Liste "selection" entfernt werden.
6) Um Aktualisierungen für mehrere Geräte zu erstellen, Schritt 4 bis 5 wiederholen.

Bestätigen mit "Next".

7) Auswahl des Root-Verzeichnisses des angesteckten Speichermediums und Bestätigen mit "Next".
8) Folgender Konfigurationsdialog öffnet sich:




| | |
|---|---|
| updateReference | ... Die Update-Dateien werden zusätzlich auf dem Gerät abgespeichert. |

| | |
|---|---|
| updateOnce | ... Das Update kann nur einmal ausgeführt werden. Eine Verwendung des Wechselmediums an einem weiteren Gerät ist nicht möglich. |
| traceWord | ... 0: keine Protokollierung der Durchführung des Firmware-Updates. -1: Protokollierung der Durchführung des Firmware-Updates im Systemtrace. |

9) Drücken von "Next".
10) Mittels "Install" wird das Speichermedium erstellt.

***Information***

*Das Speichermedium darf nie während des Schreibvorgangs entfernt werden!*

11) Beenden des Firmware-Update-Tools mit "Finish".

***Information***

*Es wird empfohlen, vor dem Entfernen des Speichermediums unter Windows immer "Gerät auswerfen" durchzuführen.*

Das Speichermedium zum Durchführen eines Firmware-Updates wurde erstellt und kann nun vom PC abgesteckt werden.

### 17.1.2 Automatisches Update beim Steuerungs-Hochlauf

Gehen Sie wie folgt vor:

1) Ausschalten der Steuerung.
2) Einsetzen der SD-Karte in die Steuerung.
3) Einschalten der Steuerung.
4) Während des Updates befindet sich die Steuerung im Zustand INIT. Nach einer Weile wechselt die RUN-LED von statischem Orange zu blinkendem Grün.

***Information***

*Während des Updates darf die Spannungsversorgung der Steuerung nicht getrennt werden.*

5) Griff öffnen und die SD-Karte herausnehmen.
6) Neustart der Steuerung durchführen.

Die Firmware und/oder der Bootloader ist aktualisiert.

## 17.2 Installieren eines Hotfix

Ein Hotfix dient zum nachträglichen, kurzfristigem Beheben lokaler Fehler in einer Systemkomponente außerhalb des regulären Release-Zyklus. Ein Hotfix wird in Form einer zusätzlichen Software Unit zur Verfügung gestellt und

installiert sich zu einer definierten, kompatiblen Systemversion. Der Hotfix wird im u-create studio unter Software Service als eigene Systemversion dargestellt. Sobald eine Software Unit, die einen Hotfix enthält, installiert wurde, wird sie beim Erstellen eines Targets oder bei der Online-Übertragung einer Applikation auf die Steuerung berücksichtigt.

## Hotfix installieren

1) Ausführen der `Setup.exe` Datei mit Administrator-Rechten.
2) Klicken auf **Next**. Folgende Informationen werden angezeigt:

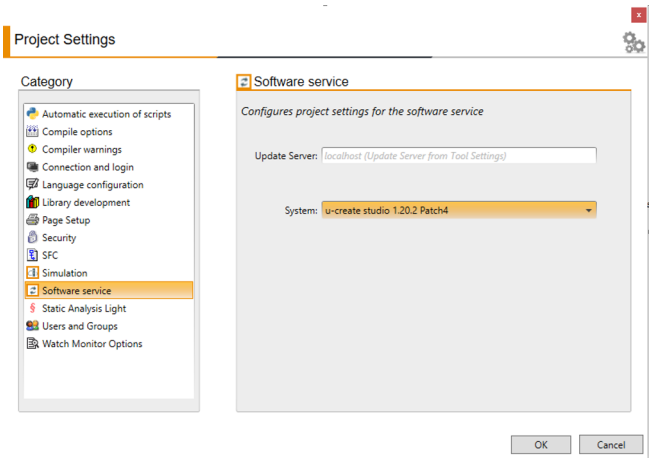
| | |
|---|---|
| Software Unit Name | Name der zu installierenden Software Unit |
| System Version Name | Name der daraus resultierenden Systemversion |
| System Base Version | Systemversion-Basis, auf die der Hotfix aufgesetzt wird |

3) Klicken auf **Next**, um die Installation zu starten.
4) Klicken auf **Finish**.

Der Hotfix wurde installiert.

## Hotfix verwenden

1) Starten von u-create studio.
2) Einstellen des installierten Hotfix unter **Tools ► Options ► Software service**.




3) Auswählen des installierten Hotfix (System Version Name aus Setup Dialog) unter **System**.
4) Klicken auf **OK**.

Der installierte Hotfix wird verwendet.

Sobald der Hotfix in u-create studio eingestellt ist, kann dieser über "Erstelle Target" (offline) oder "Software Update" (online) auf die Maschine übertragen werden.

***Information***

- Der Software Versionsstand der Maschine wird durch ein Hotfix verändert. Wird eine Applikation auf die Maschine übertragen, bleibt der veränderte Software Versionsstand durch Auswählen von "Nur Applikation" erhalten.
- Ein Hotfix kann jederzeit deinstalliert werden, ohne die ursprüngliche Systemversion zu beeinflussen.

# 18 Technische Daten

Detaillierte Angaben zu den technischen Daten entnehmen Sie bitte den jeweiligen Handbüchern.

# 19 Richtlinien, Normen und Verordnungen

Die Angaben zu den eingehaltenen Richtlinien, Normen und Verordnungen finden Sie in den jeweiligen Projektierungshandbüchern.

# 20 Anhang: Tutorial - IEC-Projekt erstellen

In diesem Tutorial wird Schritt für Schritt beschrieben, wie ein u-create studio-Projekt erzeugt und programmiert wird. Anschließend wird gezeigt, wie das Projekt auf eine Steuerung geladen und dort ausgeführt wird.

## 20.1 Neues Projekt anlegen

Starten Sie u-create studio und rufen Sie aus dem Hauptmenü den Befehl **Datei > Neues Projekt** auf.

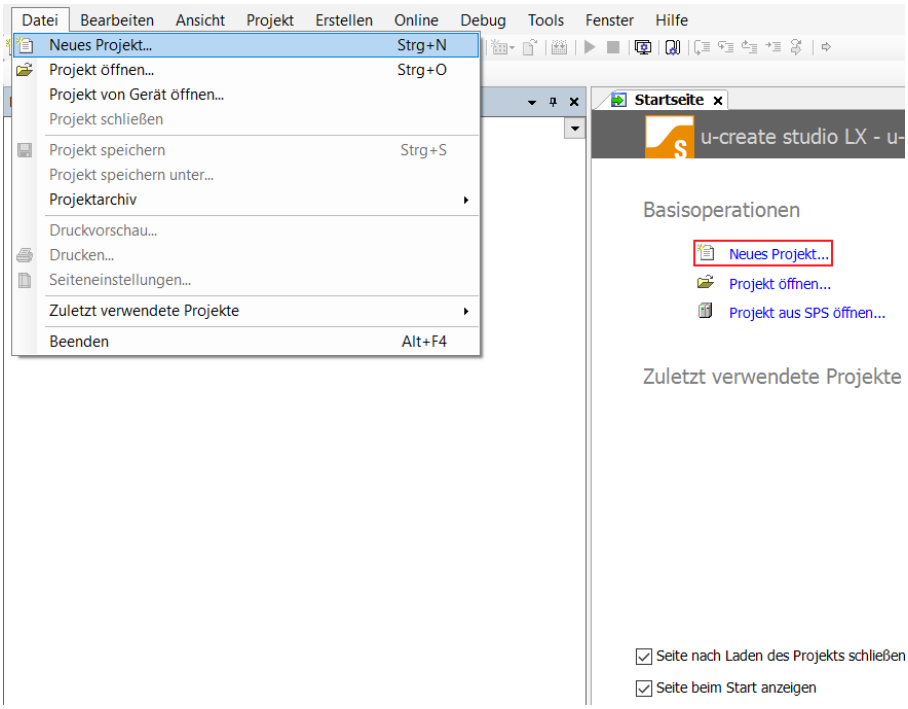



Abb. 20-21: u-create studio - Neues Projekt

Wählen Sie im Dialog "Neues Projekt" ein leeres Projekt aus und geben Sie einen Namen und einen Speicherort für das Projekt an:

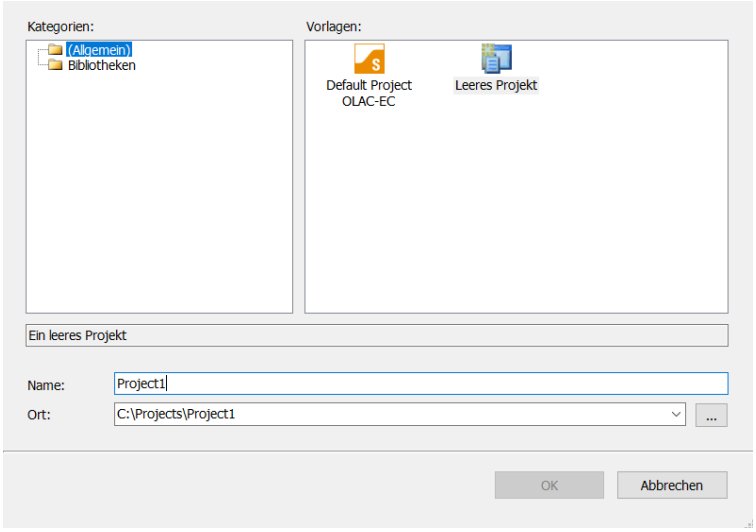



Abb. 20-22: Neues Projekt anlegen

u-create studio legt automatisch das Projekt in dem angegebenen Verzeichnis an und vergibt als Datei-Erweiterung den Zusatz "`.project`".

Das leere Projekt wird geöffnet und es kann ein Zielsystem in den Projektbaum eingefügt werden. Dazu wird das Kontextmenü des Projekts im Projektbaum geöffnet und auf "Gerät anhängen..." geklickt. Es öffnet sich ein Dialog zum Auswählen einer Steuerung. In diesem Dialog wird die gewünschte Steuerung markiert und über "Gerät anhängen" oder per Doppelklick dem Projekt hinzugefügt. Danach kann der Dialog geschlossen werden.

Danach kann im Projektbaum unter dem Knoten **<Projektname> > SPS-Logik > Application** ein Programmbaustein angelegt werden. Dazu wird das Kontextmenü des Knotens **Application** geöffnet und **Objekt hinzufügen > POU...** gewählt.

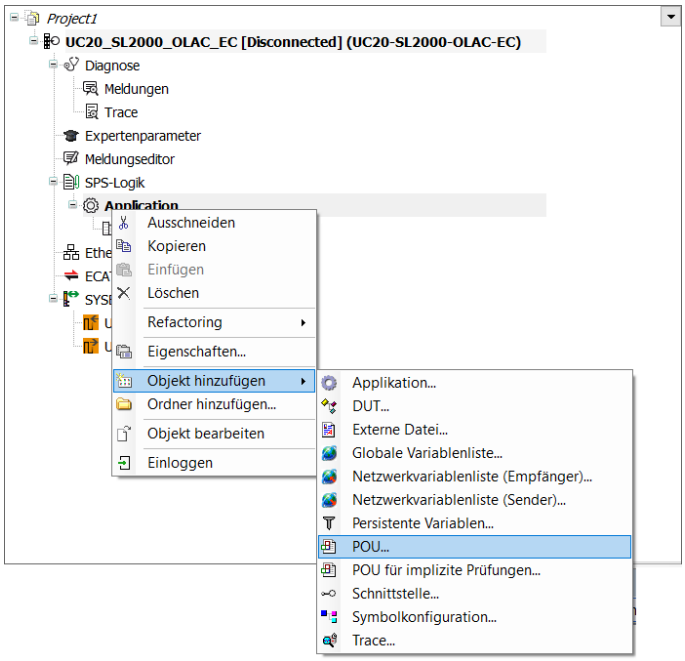



Abb. 20-23: POU hinzufügen

Es öffnet sich ein Dialog, in dem der Name, der Typ und die Implementationssprache des POUs eingestellt werden können. In diesem Beispiel wird für den Baustein der Name "myProg", der Typ "Programm" und die Sprache ST (Structured Text) verwendet:

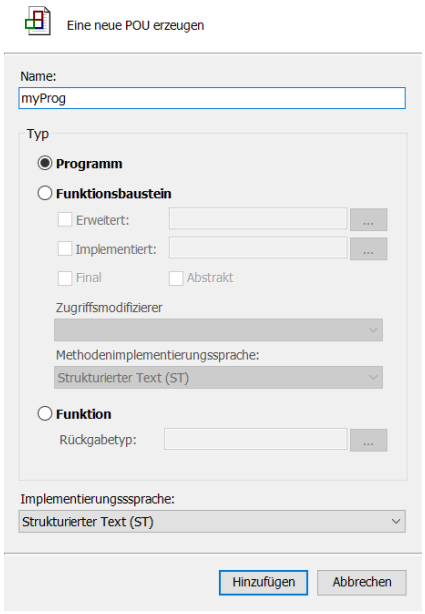



Abb. 20-24: POU Einstellungen

Nachdem die Einstellungen getroffen wurden, wird der POU mittels "Hinzufügen" unter dem Knoten **Application** eingefügt und der Programmiereditor automatisch im Arbeitsbereich geöffnet und ist bereit für weitere Programmierung.

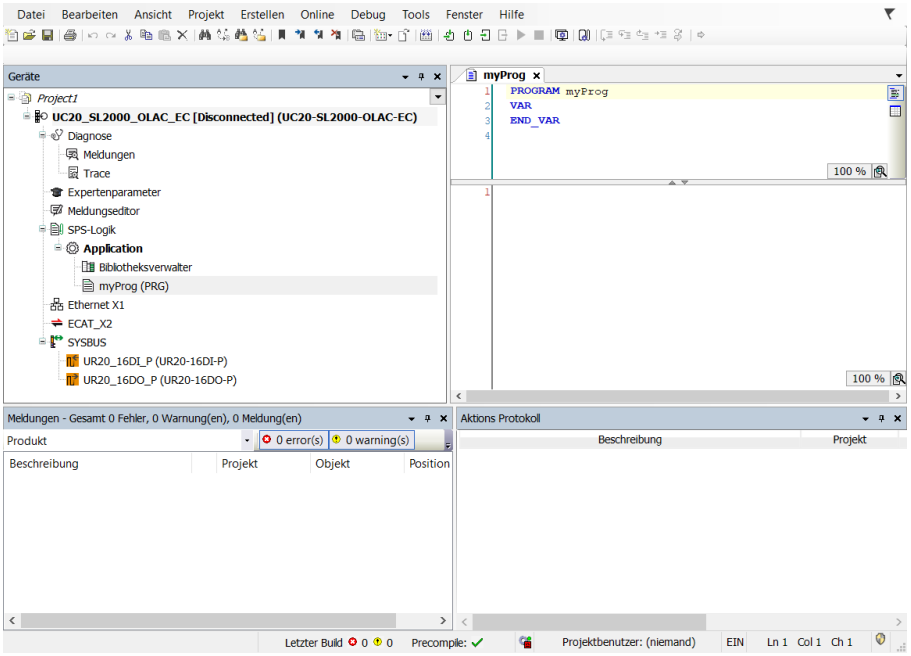



Abb. 20-25: Programmiereditor des POUs

## 20.2 Einfaches Programm erstellen

Der neu angelegte POU wird nun durch Programmierung erweitert: Es soll eine numerische Variable angelegt werden, die ihren Wert zyklisch verändert.

In der beim Erzeugen des Bausteins gewählten Programmiersprache ST (Structured Text) können die erforderlichen Eingaben (Deklaration einer numerischen Variable, Wertveränderung der Variable) im angezeigten Editor durch Tastatureingaben vorgenommen werden. Dabei ist der obere Editier-Bereich für Variablendeklarationen und der darunter liegende (noch leere) Editier-Bereich für Programmaktionen vorgesehen.

Es gibt aber auch Werkzeuge in u-create studio, die helfen, die richtigen Programmbefehle zu erzeugen. Über den Hauptmenübefehl **Bearbeiten > Variable Deklarieren...** können Variablen dialoggestützt angelegt werden:

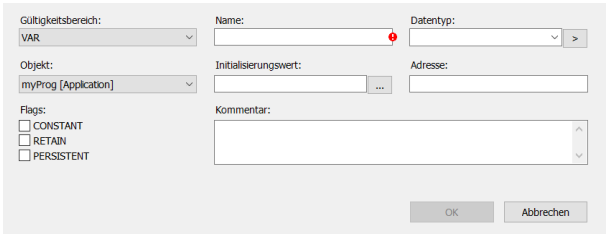



Abb. 20-26: Der Dialog "Variablendeklaration" hilft beim Anlegen einer Variablen

Geben Sie den Namen und den Datentyp der Variable ein. Falls Ihnen die in u-create studio verfügbaren Datentypen noch nicht bekannt sind, können Sie mit der an das Eingabefeld "Typ" angrenzenden Schaltfläche ">" und dann über "Eingabehilfe" einen Eingabe-Assistenten öffnen und dort den gewünschten Datentyp auswählen:

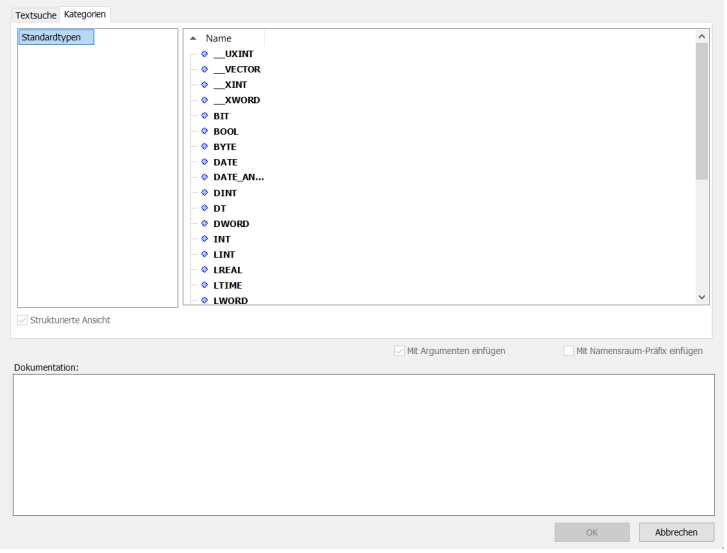



Abb. 20-27: Eingabehilfe für die Auswahl eines Datentyps

In diesem Tutorial soll eine Variable x vom Typ DINT deklariert werden:

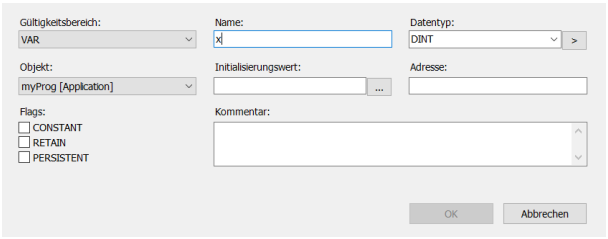



Abb. 20-28: Dialog "Variablendeklaration" mit eingegebenen Variablennamen und Datentyp

Der erzeugte Befehl (Deklaration einer Variable) wird im oberen Teil des Editors eingefügt. Im darunter liegenden Programmier-Bereich des Editors muss das Kommando zur zyklischen Wertänderung dieser Variable händisch eingefügt werden:

```
x := x + 1;
```

Der fertige Code sieht im u-create studio-Editor wie folgt aus:

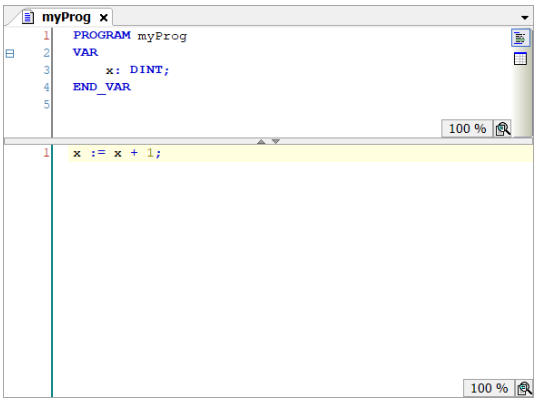



Abb. 20-29: Einfaches Programm in u-create studio

Nach dem Erstellen des Programmes muss ein Task angelegt werden, in dem der Programmbaustein abgearbeitet werden soll. Dazu wird mit der rechten Maustaste auf **Application** im Projektbaum geklickt und **Objekt hinzufügen > Taskkonfiguration...** gewählt. Es öffnet sich ein Dialog durch den ein Knoten für mehrere Tasks im Projektbaum angelegt werden kann. Durch **Hinzufügen** wird der Knoten im Projektbaum unter **Application > Taskkonfiguration** angelegt, unter dem sich bereits ein Task befindet. Dieser wird automatisch im Arbeitsbereich geöffnet.

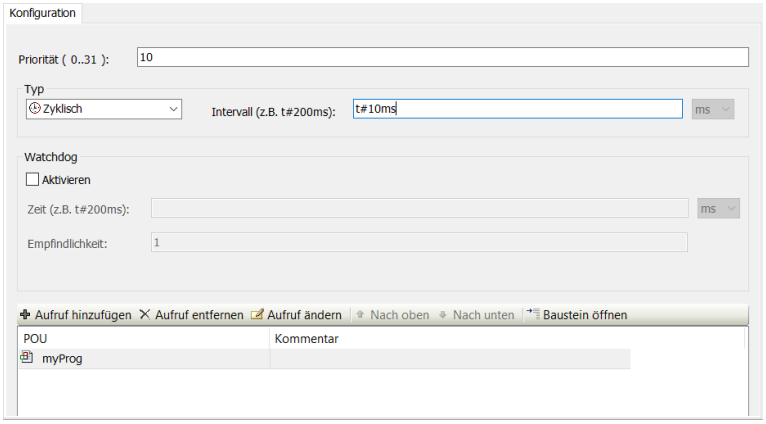



Abb. 20-30: Taskkonfiguration

In diesem Fenster können Priorität und Intervall des Tasks eingestellt und die gewünschten Programmbausteine angehängt werden. Das Zuteilen der POUs erfolgt mittels **Aufruf hinzufügen**. Es wird ein Dialog geöffnet, in dem der gewünschte POU ausgewählt und an den Task angehängt werden kann.

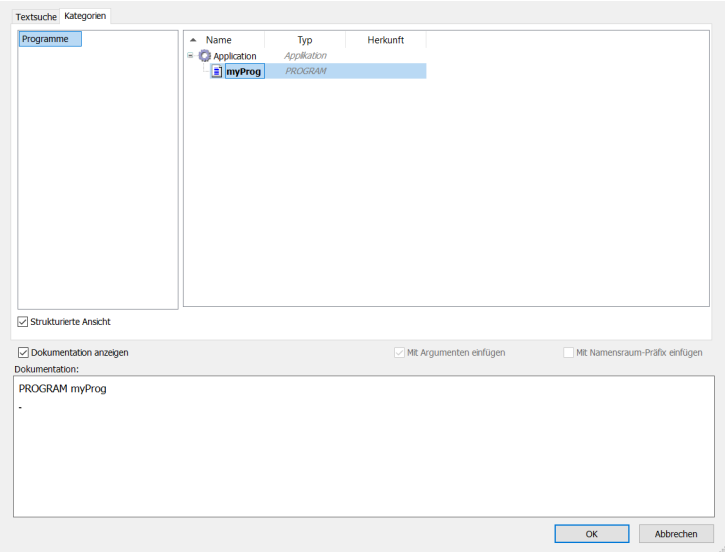



Abb. 20-31: POU an Task hängen

Der an den Task angehängte POU wird im Projektbaum unter dem Knoten **Application > Taskkonfiguration > Task** hinzugefügt.

Außerdem kann für den Task ein Watchdog konfiguriert werden, nach dessen Ablauf der Task beendet werden soll. Zusätzlich kann die Empfindlichkeit des Watchdogs eingestellt werden.

## 20.3 u-create studio-Projekt speichern

Das gesamte Projekt wird mit dem Menübefehl **Datei > Projekt Speichern** oder über das Speichern-Symbol in der Symbolleiste in dem beim Erstellen des Projektes angegebenen Verzeichnis gespeichert. Beim Speichern wird das Projekt automatisch kompiliert.

## 20.4 Firmware auf Gerät laden

Es besteht die Möglichkeit, über einen Wechseldatenträger (z.B. SD-Karte) die Firmware auf ein Gerät (z.B. Steuerung) zu installieren. Dabei wird das gesamte Steuerungsbetriebssystem, das Laufzeitsystem und die Applikation auf dem Datenträger gespeichert. Dieser kann für beliebig viele Geräte verwendet werden.

Dazu müssen im u-create im Menü **Tools ► Optionen...** unter **Software Service** zuerst folgende Konfigurationseinstellungen eingetragen werden:

| Name | Wert |
|---|---|
| Update Server | localhost |
| System | Systemversion, welche auf der Steuerung bzw. dem aktiven stationären Bediengerät installiert werden soll |

Um die Firmware für den Wechseldatenträger zu erstellen, muss dieser zuerst am PC angesteckt werden. Danach wird im Kontextmenü des Geräts im Projektbaum **Erstelle Target** gewählt. Folgender Dialog öffnet sich:

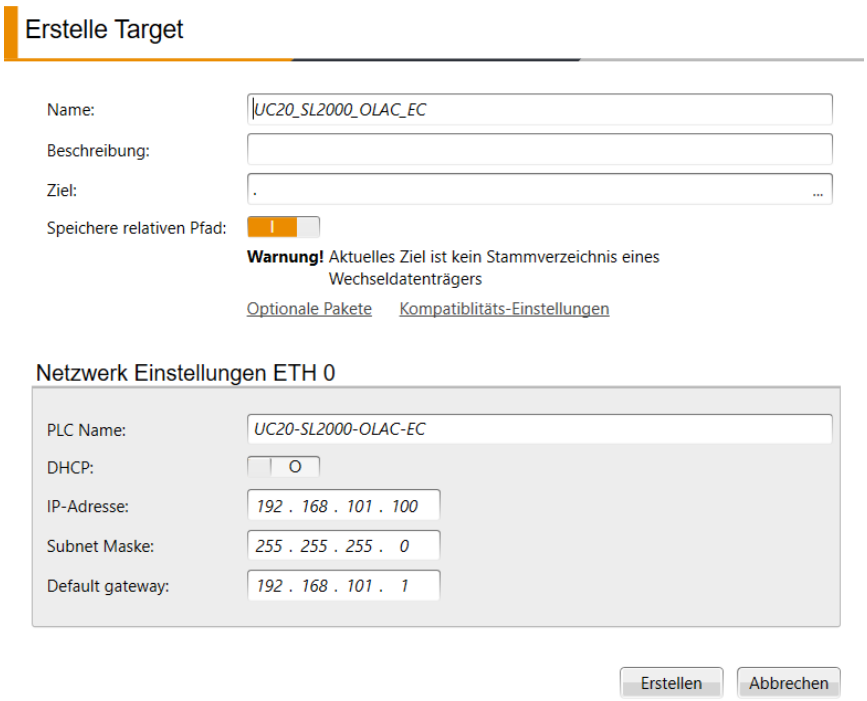



Abb. 20-32: Dialog - Erstelle Target

| Name | Beschreibung |
|---|---|
| Name: | Name der Steuerung bzw. des aktiven stationären Bediengeräts |
| Beschreibung: | Individuelle Beschreibung |
| Ziel: | Auswahl des Wechseldatenträgers oder eines beliebigen Verzeichnisses.<br>Wenn kein Wechseldatenträger ausgewählt wird, so wird eine Warnung angezeigt und der Inhalt wird in dem ausgewählten Verzeichnis angelegt.<br>Wenn ein Wechseldatenträger ausgewählt wird, so wird geprüft, ob der Wechseldatenträger ein gültiges Service Medium ist. Falls er kein gültiges Medium ist, wird eine Warnung angezeigt und der Wechseldatenträger kann mittels Klicken auf "Vorbereiten Service Medium (erfordert Administrator Rechte)" gültig bzw. bootfähig gemacht werden. |
| Speichere relativen Pfad: | Wenn diese Option gewählt wird so wird der angegebene Pfad relativ zum Projekt gespeichert |
| PLC Name: | Name der Steuerung bzw. des aktiven stationären Bediengeräts |
| DHCP: | Verwendung eines DHCP-Servers |

| Name | Beschreibung |
|---|---|
| IP-Adresse: | IP-Adresse der Steuerung bzw. des aktiven stationären Bediengeräts |
| Subnet-Maske: | Subnetzmaske |
| Default gateway: | Gateway |

Mittels **Optionale Pakete** können zusätzliche Softwarekomponenten ausgewählt werden (z. B. OPC-UA). Durch Setzen des Häckchens beim gewünschten Paket und OK wird dieses auf dem Wechseldatenträger gespeichert und steht zur Installation zur Verfügung.

Über **Kompatibilitäts-Einstellungen** wird ein weiterer Dialog geöffnet, in dem mittels "Hinzufügen" ein Bereich von Seriennummern eingestellt werden kann. Damit kann die Firmware am Wechseldatenträger nur auf den Geräten installiert werden, deren Seriennummer in diesem Bereich liegen. Somit kann eine unbeabsichtigte Installation auf einem falschen Gerät verhindert werden.

Nachdem alle Einstellungen durchgeführt wurden, kann mittels "Erstellen" (Dialog "Erstelle Target") die Software auf dem Wechseldatenträger gespeichert werden.

***Information***

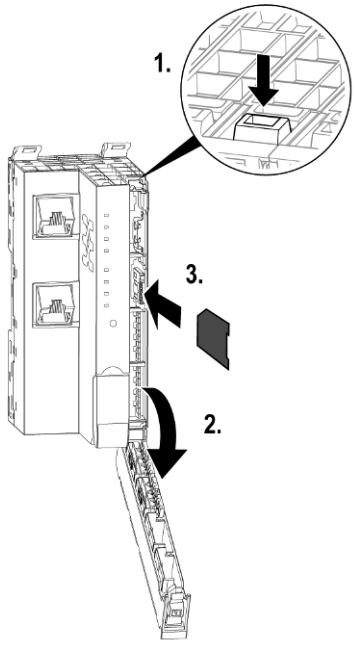
*Der Wechseldatenträger darf nie während eines Speichervorgangs abgezogen werden! Sonst kann es zu Datenverlust kommen.*

Nach der Meldung "Erstellen Target erfolgreich abgeschlossen" kann der Wechseldatenträger vom PC entfernt und der Dialog geschlossen werden. Das auf dem Wechseldatenträger geladene Softwarepaket kann nun auf gewünschten Geräten installiert werden.

## Firmware auf Gerät installieren

### Achtung

Während eines Update-Vorgangs darf weder der Wechseldatenträger gezogen werden, noch darf die Spannungsversorgung des Geräts unterbrochen werden. Dies kann zu einer Zerstörung der Firmware auf den Komponenten führen und somit einen weiteren Betrieb unmöglich machen.




1) Gerät stromlos setzen
2) Frontklappe öffnen
3) SD Karte einstecken

Um die vorbereitete Firmware auf dem Wechseldatenträger auf einem Gerät ohne Display zu installieren gehen Sie wie folgt vor:

1) Das Gerät stromlos setzen
2) Anstecken des vorbereiteten Wechseldatenträger ins Gerät
3) Durchführen eines Neustarts des Geräts.
4) Der Updatevorgang wird durchgeführt und durch eine orange blinkende LED angegezeigt
5) Der Updatevorgang ist vollständig beendet, sobald die Run/Stop LED grün leuchtet.
6) Das Gerät stromlos setzen und den Wechseldatenträger aus dem Gerät entfernen
7) Durchführen eines Neustarts des Geräts.

Die Firmware wurde installiert.

## 20.5 Konfiguration der Steuerung

Als nächster Schritt muss die Steuerung, auf der das Projekt später ausgeführt werden soll, festgelegt und konfiguriert werden. Dazu wird die Steuerungskonfiguration in u-create studio mittels Doppelklick auf die Steuerung im Projektbaum geöffnet und auf den Karteireiter "Kommunikationseinstellungen" gewechselt:

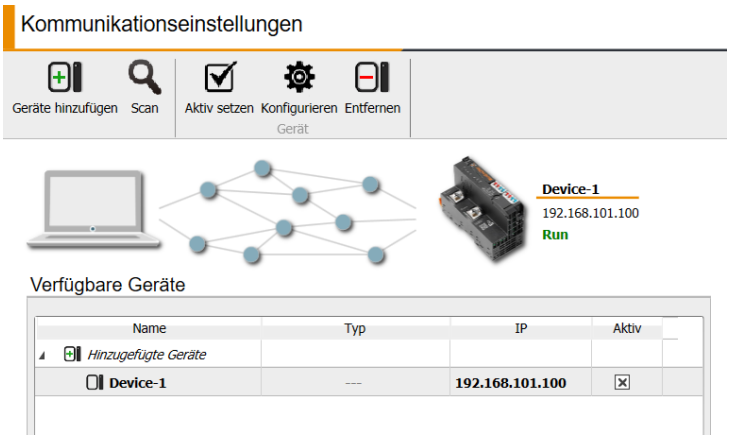



Abb. 20-33: Kommunikationseinstellungen

Durch Klicken auf **Scan** im oberen Bereich des Fensters werden alle im Netzwerk verfügbaren Steuerungen angezeigt.

Wählen Sie die für Ihr Projekt vorgesehene Steuerung aus und drücken Sie auf die Schaltfläche "Aktiv setzen". Damit wird der Status der Steuerung im Projektbaum auf "RUN" gesetzt und die Steuerung wird grün hinterlegt.

Der nachfolgend beschriebene Download des übersetzten Projekts wird auf diese Steuerung durchgeführt.

## 20.6 Projekt kompilieren und auf Steuerung laden

Mit dem Menübefehl **Online > Selektiver Download** oder durch Klicken auf das Symbol der Symbolleiste öffnet sich der Einlog-Dialog.

In diesem Dialog müssen Benutzername und Passwort eingegeben werden (Standardwert: Administrator/tobechanged):

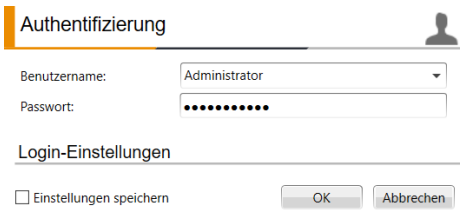



Abb. 20-34: Einloggdialog

Danach öffnet sich der Dialog für den selektiven Download.

Es kann ausgewählt werden welche Teile des Projektes auf das Gerät geladen werden sollen.

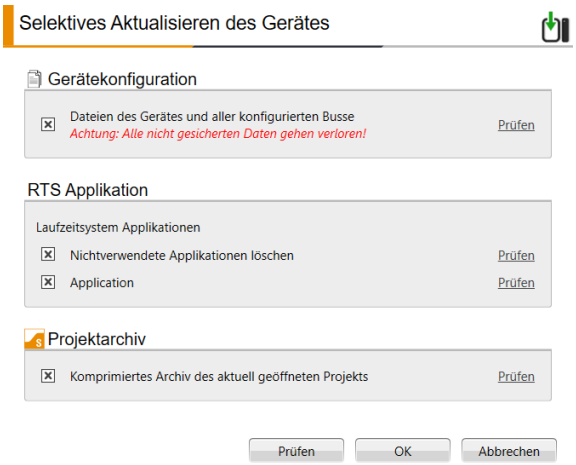



Abb. 20-35: Auswahl der Daten, die auf die Steuerung geladen werden

Wird eine neue Konfiguration auf die Steuerung geladen, muss die Steuerung neu gestartet werden. Dazu erscheint nach dem Download der Konfiguration ein Dialog. Wird dieser bestätigt, wurde nur die Konfiguration auf die Steuerung geladen und die Steuerung führt einen Neustart aus. Danach kann der Download der weiteren Softwarekomponenten erfolgen. Wird der Dialog nicht bestätigt, werden alle gewünschten Softwarekomponenten auf die Steuerung geladen. Danach muss die Steuerung neu gestartet werden.

## 20.7 Projekt starten

Das hinuntergeladene Projekt wurde noch nicht gestartet. Öffnen Sie dafür in u-create studio den Baustein myProg mittels Doppelklick und rufen Sie den Menübefehl **Debug > Start** (bzw. Funktionstaste **F5** oder in der Symbolleiste) auf.

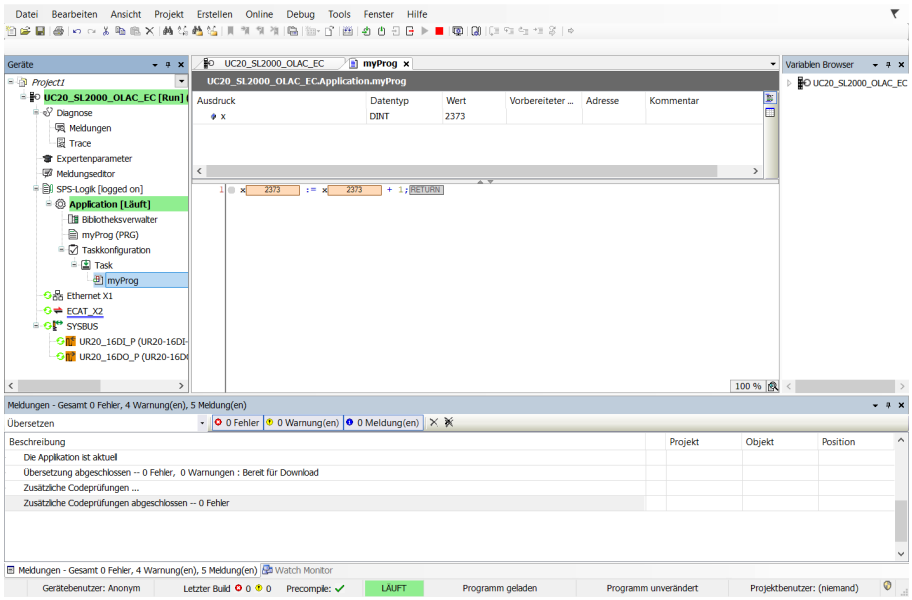



Abb. 20-36: Gestartetes Beispiel-Projekt

# 21 Anhang: Adressierung im Ethernet (Grundlagen)

Durch die Trennung in physikalische und logische Protokollschichten (Ethernet und TCP/IP) existieren in einem Netzwerk zwei Adresstypen:

- eine feste Ethernet-Adresse (MAC-ID) für jedes Gerät und
- eine IP-Adresse, die jedem Gerät im Netzwerk vergeben wird.

Von der Anwendung werden Daten immer an eine IP-Adresse gesendet oder von dort empfangen. Damit sie letztendlich beim Empfänger ankommen, muss ein Zusammenhang zwischen logischer IP-Adresse und physikalischer Ethernet-Adresse hergestellt werden. Dazu dient das Adress Resolution Protocol ARP. In jedem Netzwerk-PC ist eine ARP-Tabelle abgelegt, die zu IP-Adressen des Netzwerks die entsprechende physikalische Ethernet-Adresse angibt. Ist eine Ethernet-Adresse nicht in der ARP-Tabelle aufgeführt, kann der IP-Treiber sie i. d. R. mittels eines ARP-Requests ermitteln.

## Ethernet-Adresse (MAC-ID)

MAC-ID (Media Access Control) ist die fest vergebene Adresse, die ein Ethernet-Gerät eindeutig kennzeichnet.

## IP-Adresse

Eine IP-Adresse nach dem Standard IPv4 wird üblicherweise durch 4 mit Punkten getrennte Dezimalzahlen (je 1 Byte) angegeben. Beispiel für eine IP-Adresse: 192.168.181.1

Mit einer IP-Adresse wird sowohl ein Netzwerk als auch ein einzelner Teilnehmer im Netzwerk adressiert. Dazu enthält die IP-Adresse:

- die Net-ID (gibt die Adresse eines Netzwerkes an) und
- die Host-ID (gibt die Adresse eines einzelnen Teilnehmers in diesem Netzwerk an). Sie muss innerhalb eines Netzwerkes eindeutig sein, d.h. es können nicht zwei Endgeräten mit der selben Host-ID im Netzwerk betrieben werden.

Welche der Zahlen einer IP-Adresse nun die Net-ID und die Host-ID darstellen, wird durch die Angabe einer so genannten Netzmaske (Subnet Mask) bestimmt.

## Netzklassen

Die Netzmaske für IP-Adressen definiert durch eine ”0” als Platzhalter, welche Bits zur Adressierung der Teilnehmer (Host-ID) verwendet werden. Eine "1" als Platzhalter definiert, welche Bits die Netzwerkadresse (Net-ID) enthalten. Je nach Anzahl dieser Bits gehören Netzwerke verschiedenen Netzklassen an:

| Netzklasse | Netzmaske | Beschreibung |
|---|---|---|
| A | 255.0.0.0 | Großes Netzwerk |
| B | 255.255.0.0 | Mittleres Netzwerk |

| Netzklasse | Netzmaske | Beschreibung |
|---|---|---|
| C | 255.255.255.0 | Kleines Netzwerk mit maximal 254 Teilnehmern |

**Beispiel:** In einem kleinen Netzwerk mit der Netzmaske 255.255.255.0 ist bei der IP-Adresse 192.168.181.1 die Net-ID 192.168.181 und die Host-ID ist 1. Wird (in einem anderen, mittelgroßen Netzwerk) die Netzmaske 255.255.0.0 eingestellt, dann ist bei der IP-Adresse 192.169.100.1 die Net-ID 192.169 und die Host-ID ist 100.1.

## Gateway

Netzwerke mit verschiedenen Net-IDs werden über Router oder Gateways miteinander verbunden. Soll ein Teilnehmer eines Netzwerks Daten an Teilnehmer in anderen Netzwerken senden, so muss dafür noch die IP-Adresse des Gateways angegeben werden. Für die Adressierung im Internet-Protokoll IP sind somit drei Angaben nötig:

- IP-Adresse
- IP-Netzmaske
- IP-Adresse des Gateways

Informationen zu den Adressen Ihres Hausnetzes erhalten Sie von Ihrem Netzwerkadministrator.

## Einstellen der IP-Adresse

Die Einstellungen für die IP-Adressierung können bei jedem Endgerät manuell konfiguriert werden. In großen Netzwerken wird dies meist zentral und automatisch durch DHCP (Dynamic Host Configuration Protocol) erledigt. Hier verwaltet ein DHCP-Server einen Bereich von IP-Adressen und teilt sie den DHCP-fähigen Endgeräten zu. Die Weidmüller Steuerungen sind DHCP-fähig. Hierzu muss bei der **Steuerungskonfiguration** in u-create studio der Befehl **Enable DHCP** aktiv sein. Die IP-Adresse wird dann vom DHCP-Server im Netzwerk zugeteilt. Beim Booten erfragt die Steuerung über das Netzwerk beim DHCP-Server ihre IP-Adresse.

# 22 Anhang: Tutorial FoE

FoE (Datentransfer über EtherCAT) muss über die SPS-Anwendung erfolgen. Dieses Kapitel zeigt ein Beispiel für das Aktualisieren der Firmware eines UR20-FBC-EC-ECO Feldbuskopplers. Die Informationen werden im Meldungsfenster angezeigt.

```
PROGRAM PRGECatFileTransfer
```

## // Declaration

```
VAR
    state: IoApi_State;
    err: UDINT;
    pass: UDINT;
    slavedir: STRING:= '';
    masterdir: STRING:= '/home/admin/FB-EC-FULL-0007674-01_08_00-9.bsc';
    dir: IoEcat_FileDirection:= IoEcat_FileDirection.download;
    mDevHandle: IoApi_Hdl;
    ReqBootStrap: BOOL;
    LeaveBootstrap: BOOL;
    EcatCoupler: K_Io.IO_DEVICE_REF;
    GetState: BOOL;
    SlaveState: IoEcat_State;
    miState: DINT; msState: STRING;
    msFileName: STRING;
    msFileDir: STRING:='/home/admin/";
    mbLoadNewFW: BOOL;
END_VAR
```

## // Program blocks

```
(* Requirements:
    Included libraries: K_IoApi / K_SystemCallLibrary / Standard
Description:
    1. Copy the firmware files to "/home/admin" on the
    UC20-SL2000 (e.g. via WinSCP).
    2. Configure device name and firmware file.
    3. Set mbLoadNewFW to TRUE in order to execute firmware update.
*)

//Configure device name and desired firmware file
EcatCoupler := UR20_FBC_EC_ECO;    //Device name at ECAT_X2
msFileName:= 'FB-EC-ECO-0007674-01_00_01-6.bsc';

//Don't change the following code
CASE miState OF

0: //Ready for FW Update
IF mbLoadNewFW THEN
    mDevHandle:= EcatCoupler.DeviceHandle
    mbLoadNewFW:= FALSE;
    miState:= 10;
END_IF

10: //Enter Bootstrap
state:= IoApi_EcatEnterBootstrap(devHdl:= mDevHandle);
IoApi_EcatGetState(devHdl:= mDevHandle,pState:= SlaveState);
IF state = IoApi_State.IoApiStateOk AND
SlaveState = IoEcat_State.Bootstrap THEN
    msState:= CONCAT(EcatCoupler.DeviceName,':Bootstrap entered');
    KSys_TraceUOS(ADR(msState));
    miState:= 11;
END_IF

11: //Download FW-File
dir:= IoEcat_FileDirection.download;
slavedir:= DELETE(msFileDir,msFileName);
```

```
masterdir:= CONCAT(msFileDir,msFileName);
state := K_IoApi.IoApi_EcatFileTransfer(devHdl:= mDevHandle,
direction:= dir,
slaveFileName:= slavedir,
masterFileName:= masterdir,
password:= pass,
errorCode:= err);
IF state = IoApi_State.IoEcatFoEResultSuccess AND err = 0 THEN
    miState:= 20;
    msState:= CONCAT(EcatCoupler.DeviceName,':File downloaded successful');
    KSys_TraceUOS(ADR(msState));
ELSIF err <> 0 THEN
    msState:= CONCAT(EcatCoupler.DeviceName,':Error downloading File');
    KSys_TraceUOS(ADR(msState));
    miState:= 21;
END_IF

20: //Leave Bootstrap
state:= IoApi_EcatLeaveBootstrap(devHdl:= mDevHandle);
IoApi_EcatGetState(devHdl:= mDevHandle,pState:= SlaveState);
IF SlaveState = IoEcat_State.Operational THEN
    msState:= CONCAT(EcatCoupler.DeviceName,':Leaving Bootstrap successful');
    KSys_TraceUOS(ADR(msState));
    miState:= 21;
END_IF

21: //Wait for device
IoApi_EcatGetState(devHdl:= mDevHandle,pState:= SlaveState);
IF SlaveState = IoEcat_State.PreOperational THEN
    miState:= 22;
END_IF

22: //Device OK
IoApi_EcatGetState(devHdl:= mDevHandle,pState:= SlaveState);
IF SlaveState = IoEcat_State.Operational AND NOT EcatCoupler.HasError THEN
    msState:= CONCAT(EcatCoupler.DeviceName,':FW Update successful');
    KSys_TraceUOS(ADR(msState));
    miState:= 0;
END_IF
END_CASE
```

# 23 Anhang: Tutorial - C-Funktionen aus der IEC aufrufen

Das u-create System bietet die Möglichkeit, im u-create studio C++ erstellte C-Funktionen in der IEC aufzurufen.

***Information***

*Die nachfolgend beschriebene Funktion ist nur auf Geräten mit einer "OpenLinux automation control" Lizenz möglich.*

Dazu existiert im u-create studio C++ ein Template mit einer vordefinierten Schnittstelle zum Datenaustausch. Es wird empfohlen, in der IEC eine eigene Funktion zu erstellen, die sich um einen Aufruf der C-Funktion mit der korrekten Datenübergabe kümmert. Der Datenaustausch selbst erfolgt über Zugriffe auf einen gemeinsam definierten Speicherbereich. Aus diesem Grund werden lediglich Adressen auf diesen Speicherbereich übergeben. Da die Datentypen in der IEC und in C nicht vergleichbar sind, muss bei der Übergabe der Parameter auf eine korrekte Typkonvertierung geachtet werden.

***Information***

*Es gibt keine automatische Typüberprüfung. Der Anwender ist selbst für eine richtige Zuordnung verantwortlich. Stimmt z.B. die Größe der Datentypen nicht überein, wird auf einen falschen Speicherbereich zugegriffen.*

Das Beispiel in diesem Kapitel beschreibt das Erstellen einer C-Funktion im u-create studio C++ und den Aufruf dieser in einer im u-create studio erstellten IEC-Applikation.

## 23.1 Voraussetzungen und benötigte Komponenten

Folgende Komponenten werden benötigt:

- u-create studio C++
- u-create studio
- u-create studio-Projekt mit einer lauffähigen IEC-Applikation
- Dazu passende Hardware: UC20-SL2000-OLAC-EC, UC20-SL2000-OLAC-EC-CAN

Die Erstellung dieses Beispiels setzt Grundkenntnisse im Umgang mit u-create studio und u-create studio C++ voraus.

## 23.2 Aufgabenstellung

In dieser Beispielapplikation soll eine selbst erstellte C-Funktion über eine IEC-Funktion in der IEC-Applikation aufgerufen werden. Die IEC-Funktion stellt der C-Funktion 3 lokale Variablen mit verschiedenen Datentypen

(`DWORD`, `STRING[106]` und `REAL`) und einer globalen Variable des Typs `DWORD` zur Vefügung. Die Werte der Variablen sollen in der C-Funktion folgendermaßen verändert werden:

- Die globale Variable des Typs `DWORD` soll mit der Variable des Typs `DWORD` addiert werden
- Die Variable des Typs `REAL` soll mit 3 multipliziert werden
- Der Variable des Typs `STRING[106]` sollen Zeichen hinzugefügt werden
- Der Rückgabewert der C-Funktion soll das doppelte der Variable des Typs `DWORD` betragen

Die geänderten Werte der Variablen sollen danach wieder in der IEC-Applikation zur Verfügung stehen.

Dazu sind folgende Schritte notwendig:

- Erstellen der IEC-Funktion
- Erstellen der C-Funktion und Download auf die Steuerung
- Aufruf der IEC-Funktion in der IEC-Applikation und Download

## 23.3 Erstellen der C-Funktion und Download

Nach dem Starten vom u-create studio C++ wird ein neues Projekt über **File ► New ► C/C++ Target Application** in der Menüleiste angelegt.

In dem sich öffnenden Dialog werden folgende Parameter eingestellt:

| Parameter | Wert |
|---|---|
| Project name | myProject |
| Location | C:\myProjects |
| Target | • UC20-SL2000: Control PLC V<VersionNr> ARMHF Linux<br>• KEBA-Steuerung: Control PLC V<VersionNr> X86 Linux |
| Project Type | CoDeSys IEC Functions in C |

Mittels **Finish** wird das Projekt angelegt und im Projektbaum angezeigt.

In diesem Projekt sind bereits eine C-Funktion und die vordefinierte Schnittstelle angelegt.

Im Projektbaum wird unter **myIEC_C_Call ► include** die Datei `MyIEC_C_Call.h` mittels Doppelklick geöffnet. In dieser Datei werden die Strukturen für die Eingangs- und Ausgangsvariablen entsprechend deklariert und somit die Schnittstelle an die IEC-Schnittstelle angepasst.

In der Struktur `struct func1_IN` werden die Eingangsparameter der IEC-Funktion entsprechend abgebildet. Die Anzahl muss übereinstimmen und es muss ein korrektes Typmapping erfolgen. Dazu wird in diesem Beispiel der darin existierende Code durch Folgenden ersetzt:

```
struct func1_IN {
    int32_t *in_dword;
    const char *in_string;
    float *in_real; };
```

In der Struktur `struct func1_OUT` werden die Ausgangsparameter der IEC-Funktion entsprechend abgebildet. Die Anzahl muss übereinstimmen und es muss ein korrektes Typmapping erfolgen. Dazu wird in diesem Beispiel der darin existierende Code durch Folgenden ersetzt:

```
struct func1_OUT {
    float *out_dword;
    char *out_string; };
```

In der Struktur `struct func1_Instance` wird die globale Variable der IEC-Funktion abgebildet. Dazu wird der darin existierende Code durch Folgenden ersetzt:

```
struct func1_Instance {
    int32_t *inst_global;
};
```

Zusätzlich müssen noch die beiden Dateien `stdio.h` und `string.h` inkludiert werden.

Um die C-Funktion zu erstellen wird im Projektbaum unter **myIEC_C_Call ► src** die Datei `MyIEC_C_Call.c` im Arbeitsbereich geöffnet. Der Programmcode in der Funktion `int func1(...)` wird entfernt und durch die nachfolgenden Codeteile ersetzt.

In der ersten Codezeile der Funktion wird die Variable `char msg [106]` definiert. Diese Variable wird später zum Zwischenspeichern eines Textes genutzt.

In der nächsten Codezeile werden die Werte der Eingangsparameter (Adressen auf Speicherbereich), welche von der IEC-Applikation beim Aufruf der C-Funktion übergeben werden, im u-create studio im Trace-Monitor ausgegeben.

```
printf("func1 (called from IEC) with params: in_dword:%i, in_string:%s,
in_real:%f \n", *(in->in_dword), in->in_string, *(in->in_real));
```

In den nächsten Zeilen werden die Berechnungen durchgeführt.

Zuerst wird die Variable `inst_global` mit der Variable `in_dword` addiert:

```
*(inst->inst_global) += *(in->in_dword);
```

Danach wird die Variable `in_real` mit 3 multipliziert und das Ergebnis der Variable `out_dword` zugewiesen:

```
*(out->out_dword) = *(in->in_real) * 3;
```

Als Nächstes wird in die Variable `msg` ein Text bestehend aus der Variable `in_string` und einer zusätzlichen Zeichenkette geschrieben. Anschließend wird der Inhalt der Variable `msg` in die Variable `out_string` kopiert.

```
sprintf(msg,"C-Function answering to \"%s\" .", in->in_string);
strcpy(out->out_string , msg);
```

Zum Schluss wird die Variable `in_dword` mit 2 multipliziert und das Ergebnis als Rückgabewert der Funktion zurückgegeben.

```
return *(in->in_dword) * 2;
```

Die C-Funktion wurde erstellt.

Nun wird das Projekt mittels **Project ► Build All** in der Menüleiste gebaut und das Generat (`libMyIEC_C_Call.so`) mittels **Download** auf die Steuerung in das Verzeichnis `/appldisk/application/control/ccontrol/` geladen.

## 23.4 Aufruf der IEC-Funktion in der IEC-Applikation und Download

In der IEC-Applikation kann nun die erstellte IEC-Funktion verwendet werden. Dazu wird im u-create studio ein Programm (z.B. `PLC_PRG`) um folgenden Programmcode erweitert:

### Lokale Variablen

```
VAR
    rv1: DINT;

    res1 : REAL;
    res2 : STRING [106];
END_VAR
```

Im Codeteil wird die erstellte IEC-Funktion `MyCFunction` wie folgt aufgerufen:

```
rv1 := MyCFunction( in_dword := 17, in_string := 'Hello c-Function', in_real := 25.58, out_real => res1, out_string => res2);
```

Die Variablen `res1` und `res2` enthalten die in der C-Funktion berechneten Variablen, welche in weiterer Folge in der Applikation verwendet werden können.

Danach kann das Projekt auf die Steuerung geladen (siehe Onlinehilfe u-create studio) und ausgeführt werden.

# Index