TM213

Automation Runtime

PERFECTION IN AUTOMATION
A MEMBER OF THE ABB GROUP

Voraussetzungen		
Trainingsmodule	TM210 – Arbeiten mit Automation Studio	
Software	Automation Studio 4.6 Automation Runtime ab 4.61	
Hardware	X20 Steuerung und X20 I/O Module ETA210 oder ETAL210 + ETAL690 **www.br-automation.com/eta-system**	

Inhaltsverzeichnis

1 Einleitung

Das deterministische Echtzeit Betriebssystem – Automation Runtime – ist die Softwareplattform für die gesamte B&R Produktpalette. Automation Runtime bietet Dienste an, um das Zielsystem frei zu konfigurieren und diagnostizieren und Programme auszuführen.

Die Anforderungen an Automation Runtime sind ein modularer Aufbau, Konfigurierbarkeit und eine schnelle wiederholte Abarbeitung der Anwendung in einem zeitlich exakten Rahmen. Dadurch kann zur Laufzeit eine hohe Produktstückzahl in hoher Qualität und geforderter Genauigkeit erreicht werden.

Abbildung 1: Automation Runtime: eine Softwareplattform für die gesamte B&R Produktpalette

Dieses Trainingsmodul bietet einen Überblick über Automation Runtime, dessen Eigenschaften und Konfigurationsmöglichkeiten.

1.1 Lernziele

Mit Hilfe ausgesuchter Beispiele, welche typische Anwendungsfälle einer Applikation beschreiben, erlernen Sie, wie Automation Runtime für Ihre Anwendung im Automation Studio konfiguriert wird.

- Sie lernen die Anforderungen an ein Echtzeitbetriebssystem in der Automatisierung kennen.
- Sie lernen die Eigenschaften und Funktionsweise von Automation Runtime kennen.
- Sie erhalten einen Überblick über integrierte Server- und Clientfunktionen sowie Diagnosemöglichkeiten.
- Sie lernen die Vorteile eines durchgängigen Laufzeitsystems für die integrierte Automatisierung kennen.
- Sie lernen die Speicherverwaltung von Automation Runtime, Gültigkeitsbereiche und Speicherverhalten von Variablen kennen.
- Sie lernen das Hochlaufverhalten sowie das Laufzeitverhalten von B&R Steuerungen kennen.
- Sie lernen die Funktion von Automation Runtime I/O Management kennen.
- Sie lernen die Wechselwirkungen im Multitasking kennen.
- Sie lernen die Konfigurationsmöglichkeiten von Automation Runtime kennen.

1.2 Symbole und Sicherheitshinweise

Sofern dies hier nicht anders beschrieben wird, gelten die Symbolerklärungen und Sicherheitshinweise aus dem Trainingsmodul "TM210 – Arbeiten mit Automation Studio" und den verwendeten Handbüchern.

2 Automation Runtime

Durch den Grundgedanken der integrierten Automatisierung und der freien Skalierbarkeit von Automatisierungslösungen ergeben sich Anforderungen an das Projektierungswerkzeug sowie an das unterlagerte Laufzeitsystem.

Anforderungen an Automation Studio und Automation Runtime

Automation Studio ist die Projektierungsumgebung für die Automatisierungskomponenten von B&R, welche Steuerung, Antriebstechnik, Sicherheitstechnik und Visualisierung umfassen. Durch klare Strukturierungsmöglichkeiten für Projekte und die Verwaltung verschiedener Konfigurationen werden das Arbeiten im Team und die Abbildung von Varianten einer Maschine ermöglicht.

Dem Anwender stehen zahlreiche Programmiersprachen, Diagnosewerkzeuge und Editoren für die Projektierung zur Verfügung. Effizientes Arbeiten gelingt durch die Verwendung der, von B&R mitgelieferten, Standardbibliotheken und der im System integrierten IEC Programmiersprachen.Eine durchgängige Simulation ermöglicht das Projektieren und Testen von Anwendungen auch ohne Hardware.

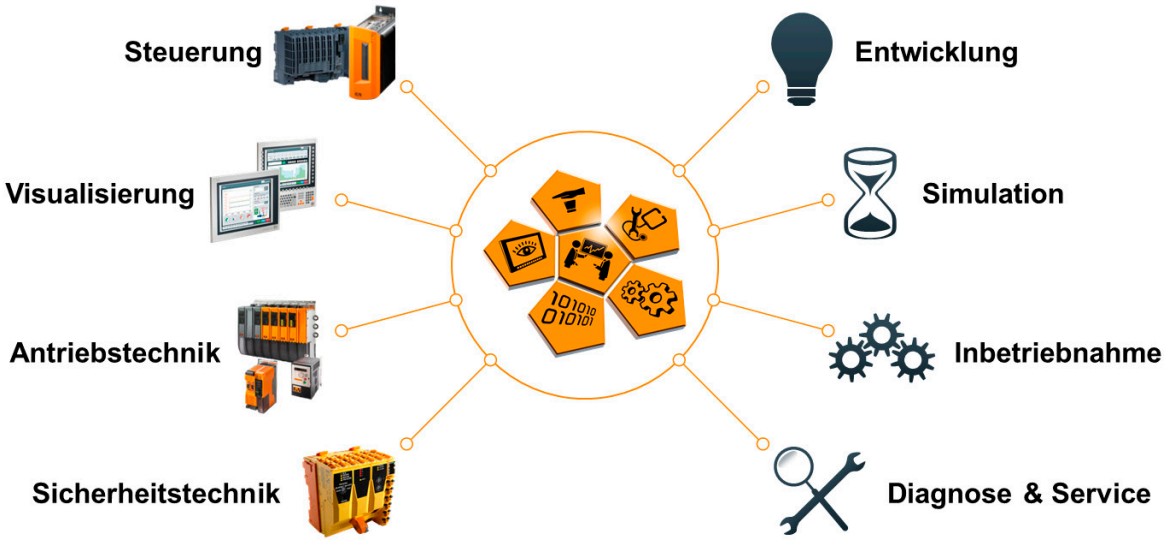


Abbildung 2: Automation Studio: ein Engineering-Tool für den Maschinenlebenszyklus

Eigenschaften von Automation Runtime

Durch die Anforderungen an das Projektierungswerkzeug und den erforderlichen Funktionen des Laufzeitsystems ergeben sich die Eigenschaften von Automation Runtime. Automation Runtime ist vollständig im entsprechenden Zielsystem eingebettet. Es unterstützt alle Hardwareplattformen und macht die Anwendungserstellung hardwareunabhängig.

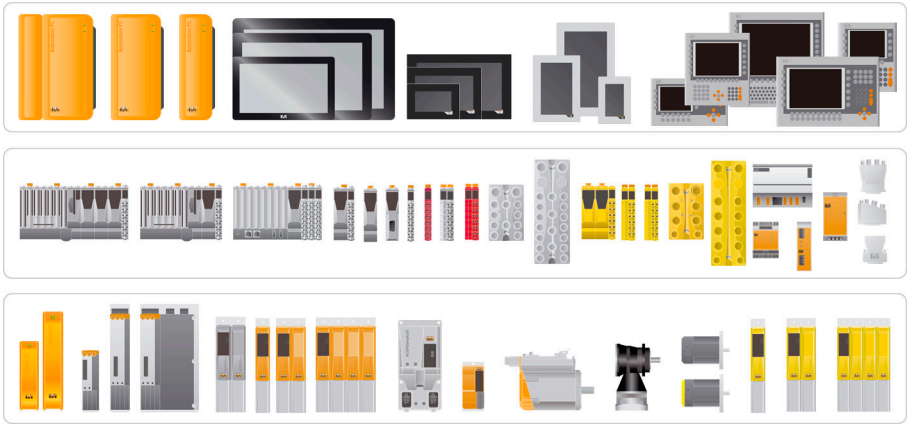


Abbildung 3: Automation Runtime - die Plattform für eine skalierbares System

Automation Runtime ermöglicht den Zugriff der Anwenderprogramme auf I/O Systeme, Schnittstellen, Feldbusse, Netzwerke und Speicher. Umfassende Konfigurationsmöglichkeiten für Zeitverhalten sowie Client- und Serveranwendungen bieten die geforderte Flexibilität für moderne Steuerungssyteme.

Das Automation Runtime bietet eine Reihe wichtiger Funktionen an:

- Es läuft auf allen B&R Zielsystemen.
- Es macht die Anwendung hardwareunabhängig.
- Es garantiert Determinismus durch ein zyklisches Laufzeitsystem.
- Es ermöglicht die Konfiguration von 8 verschiedenen Taskklassen und Zykluszeiten.
- Es garantiert die Reaktion auf Zeitverletzungen.
- Es gibt für alle Taskklassen konfigurierbare Toleranzen.
- Es bietet umfangreiche Funktionsbibliotheken nach IEC 61131-3.
- Es ermöglicht den Zugriff auf Netzwerke und Bussysteme.
- Es beinhaltet integrierte Client- und Serverschnittstellen.
- Es bietet eine OPC UA Client- und Serverschnittstelle sowie eine Benutzerverwaltung an.
- Es stellt umfassende Diagnosefunktionen bereit.

? Echtzeit Betriebssystem \ Arbeitsweise

2.1 Aufbau von Automation Runtime

Automation Runtime stellt dem Anwender eine hardwareunabhängige, multitaskingfähige und deterministische Basis für die Erstellung der Applikation zur Verfügung. Es verwaltet die Hardware- und Softwareressourcen und bietet eine durchgängige Diagnose an.

Die IEC Bibliotheksfunktionen von Automation Runtime erleichtern die Programmierung, verringern die Zeit zur Programmerstellung und helfen dabei Fehler zu vermeiden.

Automation Runtime erfüllt höchste Ansprüche an Determinismus und Geschwindigkeit.

Um diesen Leistungsvorteil bis in die Anwendung zu erhalten, liegt eine Abstraktionsschicht über dem Echtzeit Betriebssystem. Damit ist für den Anwender sichergestellt, dass bei einem Wechsel des Betriebssystems die Anwendung nicht angepasst werden muss. Die einheitliche Programmierschnittstelle bleibt somit immer erhalten.

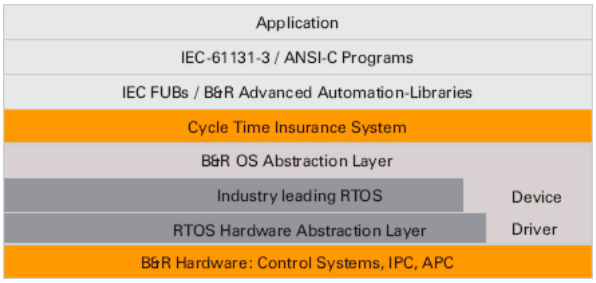


Abbildung 4: Automation Runtime Architektur

Programme, Tasks und Taskklassen

Ein Programm, welches einer Taskklasse zugeordnet und auf dem Zielsystem ausgeführt wird, wird auch Task genannt.

Tasks werden in einer Taskklasse zyklisch ausgeführt.

Über die Taskklasse werden Zykluszeit und Priorität festgelegt.

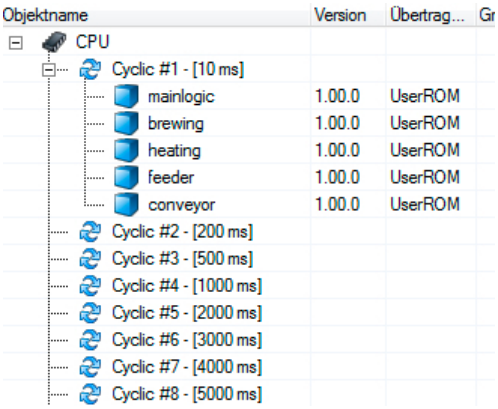


Abbildung 5: Taskklassensystem mit zyklischen Tasks in Taskklasse #1

? Echtzeit Betriebssystem

2.1.1 Automation Runtime Zielsysteme

Automation Runtime läuft auf unterschiedlichen Hardware Plattformen. Bei den X20 Steuerungen, Power Panels oder Automation PCs kommen PC basierende Hardwareplattformen zum Einsatz.

Automation Runtime Embedded

Mit Automation Runtime Embedded sind alle Zielsysteme eingeschlossen, auf denen Automation Runtime als eigenständiges Betriebssystem ausgeführt wird. Das X20 Steuerungsystem, Power Panels sowie PC basierende Hardwareplattformen sind für Automation Runtime Embedded geeignet.

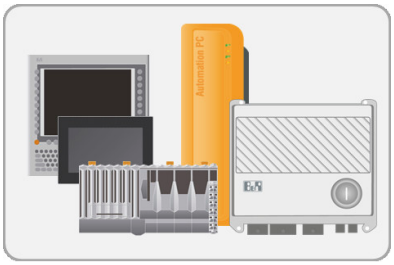


Abbildung 6: Automation Runtime auf Steuerungen aller Art

Automation Runtime Simulation - ARsim

Die ARsim ist eine auf Windows basierende Automation Runtime, welches keine Echtzeitfähigkeit besitzt und im Wesentlichen der Funktionalität aller anderen Zielsysteme entspricht. Mit der ARsim lassen sich alle Steuerungen, die Visualisierung und Antriebe vollständig simulieren.

Abbildung 7: Automation Runtime Simulation am PC

In Automation Studio kann zwischen realer Hardwarekonfiguration und Simulation gewechselt werden. Der Wechsel zur Simulation geschieht unter "Online" \ "Simulation aktivieren". Automatisch wird die Automation Runtime Simulation gestartet und eine Verbindung aufgebaut. Durch die nahtlose Integration der Simulation werden Tests am PC einfach und schnell ermöglicht.

Abbildung 8: Schaltfläche "Aktiviere Simulation"

Projekt Management \ Simulation \ ARsim

Automation Runtime Windows - ARwin

ARwin ist ein auf einem Windows Betriebssystem lauffähiges Zielsystem. Dem Windows-Betriebssystem wird ein Echtzeitbetriebssystem unterlagert, welches die völlige Kontrolle über die Ressourcen des PCs übernimmt. Das Hostbetriebssystem selbst wird als Task mit niedriger Priorität am Echtzeitbetriebssystems behandelt.

Abbildung 9: Automation Runtime - Echtzeitanwendungen unter Windows

B&R Hypervisor

Mit B&R Hypervisor kann auf einem Zielsystem Automation Runtime parallel zu Linux bzw. Windows betrieben werden. Mit B&R Hypervisor können die Hardware-Ressourcen eines Systems partitioniert werden. Im Anschluss können die CPUs, Speicherbereiche und Peripheriegeräte den laufenden Betriebssystemen zugeordnet werden. Während der Laufzeit wird jedes Betriebssystem von den anderen logisch getrennt.

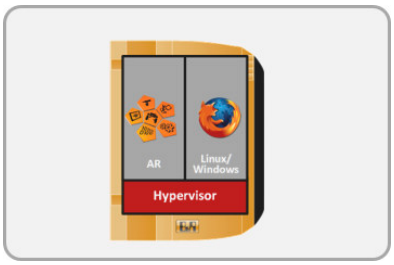


Abbildung 10: B&R Hypervisor - mehrere Betriebssysteme auf einem Automation PC

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4

- ARsim
- ARwin
- B&R Hypervisor

2.1.2 Server- und Clientfunktionen

Zahlreiche Client- und Serverfunktionen sind in Automation Runtime verfügbar. Es werden verschiedene Netzwerkdienste und -protokolle unterstützt.

In der Abbildung sind die einzelnen Protokolle farbig markiert.

Grün: Protokolle, die bereits bei einer neuen Automation Studio Konfiguration aktiviert sind.

Grau: Protokolle, die per Konfigurationseintrag zusätzlich aktiviert und konfiguriert werden können.

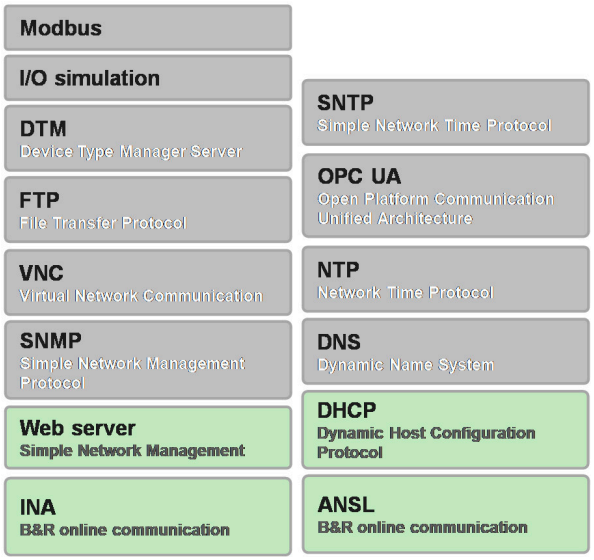


Abbildung 11: Konfigurierbare Automation Runtime Server- und Clientfunktionen

Die Konfigurationseinträge werden über das Kontextmenü der CPU und der Ethernet-Schnittstelle geöffnet. Für jedes Protokoll gibt es einen eigenständigen Konfigurationseintrag, über den das entsprechende Protokoll aktiviert und konfiguriert wird.

Zur Laufzeit können viele der angebotenen Funktionen mit Hilfe von Bibliotheken diagnostiziert und konfiguriert werden.

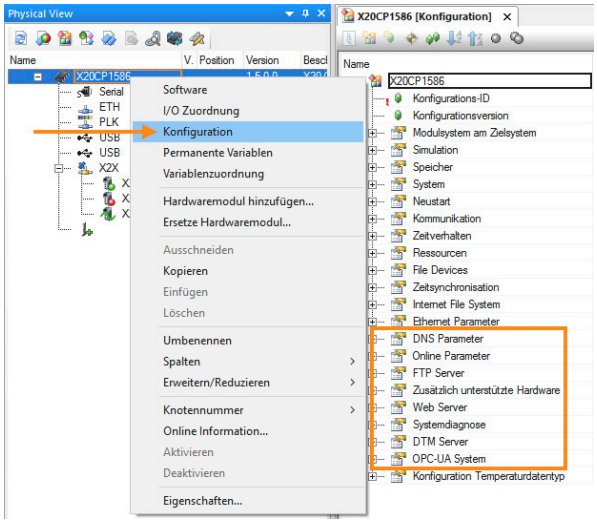


Abbildung 12: CPU und Ethernet-Konfiguration über Kontextmenü öffnen

Programmierung \ Editoren \ Konfigurationseditoren \ Hardwarekonfiguration \ CPU Konfiguration \ SG4

Kommunikation \ Ethernet \ AR Konfiguration \ Schnittstellenkonfiguration (SG4)

Programmierung \ Bibliotheken

- Konfiguration, Systeminfo, Laufzeitkontrolle
- Kommunikation

2.2 Diagnosefähigkeit von Automation Runtime

Unabhängig von der eingesetzten Hardwareplattform unterstützt Automation Runtime eine Vielzahl von Diagnosewerkzeugen, die Aufschluss über den aktuellen Systemzustand geben.

Die Verwendung der integrierten Diagnosewerkzeuge ist in der Automation Help dokumentiert. Eine Auflistung der Diagnosewerkzeuge ist in der Hilfebox unten. Weitere Ausführungen und Übungen sind im Trainingsmodul "TM223 - Automation Studio Diagnose" nachzulesen.

Abbildung 13: Automation Runtime - durchgängige Diagnose integriert

Diagnose und Service \ Diagnosewerkzeug

- Statusleiste
- Informationen zum Zielsystem
- System Diagnostics Manager
- Monitor Modus \ Online Vergleich
- Logger
- Profiler

Diagnose und Service \ I/O und Netzwerkdiagnose

Programmierung \ Bibliotheken \ Konfiguration, Systeminfo, Laufzeitkontrolle

2.3 Konnektivität, Benutzerverwaltung und Sicherheit

Die B&R Lösung ist vollkommen offen für eine **nahtlose Integration in bestehende Netzwerke** und ermöglicht die direkte Einbindung von Fremdgeräten. Wahlweise können B&R Komponenten als Feldbusmaster- oder Slave eingesetzt werden.

Ein OPC UA Server und Client ermöglicht die Veröffentlichung und den Austausch von Prozessdaten. Das Benutzer Rollen System hilft bei der Einschränkung von Schreib- und Lesezugriffen. Mit Hilfe von Transport Layer Security (TLS) wird eine verschlüsselte Verbindung ermöglicht.

Unterstützung von Fremdgeräten

Automation Studio ermöglicht die direkte Integration von Feldbussen und Fremdgeräten. Master- und Slavegeräte werden in Automation Studio in der Physical View oder im System Designer eingefügt.

Kommunikation \ Feldbusse

Projekt Management \ Hardware Management \ Fremdgeräte \ Fremdgerätemanager

Benutzerverwaltung und Sicherheit

Automation Studio unterstützt die Konfiguration von Benutzer- und Rollensystemen, einer Zertifikatsverwaltung sowie die Verwaltung von SSL/TLS Konfigurationen. Diese Konfigurationen werden im Paket "AccessAndSecurity" in der Configuration View verwaltet.

Zum Paket "AccessAndSecurity" kann über die Toolbox eine **Firewall** hinzugefügt werden. Sie dient dazu, den Netzwerkzugriff zu beschränken, basierend auf Absender oder Ziel und den genutzten Diensten.

Außerdem ist es möglich, Zugriffe über das ANSL-Protokoll auf authentifizierte Verbindungen einzuschränken und somit das Zielsystem gegen unerwünschte Zugriffe über ANSL abzusichern. Um ein System für ANSL-Authentifizierung zu konfigurieren, muss der ANSL-Verbindung eine definierte Rolle zugeordnet werden.

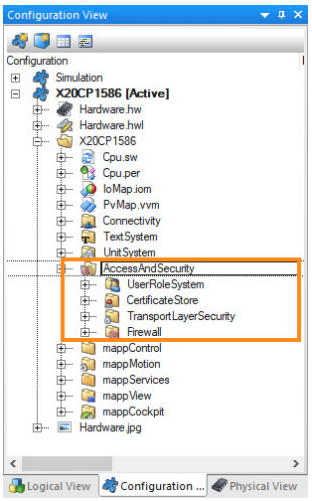


Abbildung 14: Benutzerverwaltung in der Configuration View

Programmierung \ Access & Security \

- Transport Layer Security (TLS)
- Benutzer-Rollen-System
- Firewall

Programmierung \ Editoren \ Konfigurationseditoren \ Hardwarekonfiguration \ CPU Konfiguration \ SG4 \ Eigenschaften der CPU - Online Parameter

OPC UA Server und Client

Automation Studio unterstützt die Konfiguration eines OPC UA Servers. Die Schreib- und Leserechte für die einzelnen OPC UA Codes werden mit Hilfe des Benutzer Rollen Systems verwaltet. OPC UA Client Funktionen werden über eine Bibliothek angeboten.

Kommunikation \ OPC UA

Programmierung \ Bibliotheken \ Kommunikation \ AsOpcUac

Automation Software \ Beispielprogramme \ Kommunikation mit OPC UA Client Funktionsblöcken

3 Installation und Inbetriebnahme

Jedes Zielsystem startet Automation Runtime und lädt das Automation Studio Projekt von einem Flash Speicher. Je nachdem, welches Zielsystem verwendet wird, wird als Speichermedium eine Compact Flash, CFast oder der integrierte Flash Speicher eingesetzt.

Es gibt 3 Möglichkeiten Automation Runtime zu installieren: Online-Installation, Offline-Installation oder USB-Installation.

Unabhängig von der ausgewählten Methode unterstützt Automation Studio den Installationsprozess.

Im Zuge der Erstinstallation wird der Flash Speicher der Steuerung, entsprechend den Anforderungen der Anwendung, partitioniert.

Beim Installieren von Automation Runtime werden Einstellungen wie die I/O Konfiguration und die Schnittstellenkonfiguration eingeschlossen.

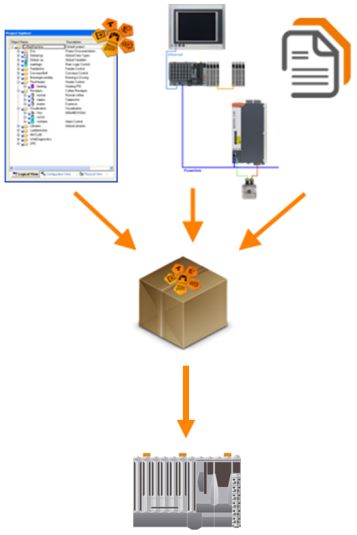


Abbildung 15: Installation und Inbetriebnahme

Wie wird ARwin installiert?

Ausnahme bei PC basierten Zielsystemen: Die erstmalige Installation von Automation Runtime für Windows (ARwin) erfolgt über ein eigenes Setup Tool.

Hardware \ X20 System \ X20 Module \ CPUs

- X20(c)CP1301, X20CP1381 und X20CP1382 \ Bedien- und Anschlusselemente \ System Flash programmieren
- X20(c)CP158x und X20(c)CP358x \ System Flash programmieren

Projekt Management

- Projektinstallation
- Simulation \ ARsim \ Erzeugen einer lauffähigen Projektstruktur

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ ARwin \ Installation / Konfiguration

3.1 Automation Runtime ändern und aktualisieren

Wenn neue Softwareversionen für die Laufzeitumgebung zur Verfügung stehen, so können diese Softwareversionen über den Upgrade-Dialog in Automation Studio hinzugefügt werden.
Der Upgrade-Dialog wird über das Menü "Extras" \ "Upgrades..." geöffnet.

Die Automation Runtime Version kann für die aktive Konfiguration des Automation Studio Projekts geändert werden.

Die einzelnen Softwareversionen für die Laufzeitumgebung werden über das Menü "Projekt" \ "Runtime Versionen ändern..." ausgewählt.

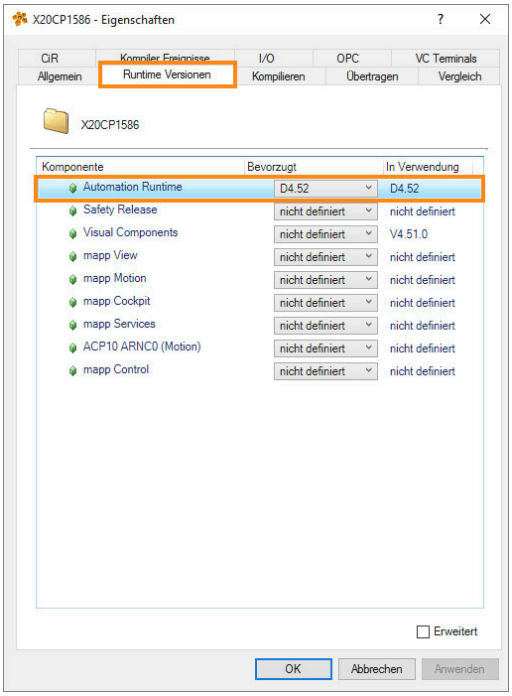


Abbildung 16: Ändern der Runtime Versionen

> **!** Eine Änderung der Version von Technologiepaketen (mapp Motion, mapp Services, etc.) wirkt sich auf alle Konfigurationen im Projekt aus, da die damit verbundenen Bibliotheken in der Logical View verwaltet werden.

> **?** Projekt Management
>
> - Arbeitsoberfläche \ Upgrades
> - Ändern der Runtime Version
>
> Echtzeit Betriebssystem \ Versionsübersicht

Upgrade mit Automation Studio

Wenn sich Automation Runtime sich bereits auf dem Zielsystem befindet, kann das Projekt inklusive der neuen Automation Runtime Version übertragen werden. Alternativ wird die Erstellung von Installationsmedien wie Compact Flash, CFast oder USB-Stick unterstützt.

Weiterführende Informationen:

- 3.3 "Projektinstallation" auf Seite 17

Bei der Übertragung ins Zielsystem wird ein Konflikt der Automation Runtime Versionen erkannt und im Transfer-Dialog angezeigt.

Abbildung 17: Darstellung von Versionsunterschieden im Transfer-Dialog; Neustart erforderlich

Upgrade ohne Automation Studio

Wenn ein Upgrade mit Automation Studio nicht möglich ist, so kann mit Runtime Utility Center ein vollständiges Installationsabbild erstellt werden. Dieses kann z. B. auf einem USB-Stick weitergegeben werden.

Projekt Management \ Projektinstallation \ Projektinstallation durchführen \ Export RUC

3.2 Konfigurations-ID und Partitionierung

Vor der Projektinstallation werden die Einstellungen für die Konfigurations-ID und die Partitionierung des Flash Speichers im Projekt kontrolliert. Zusätzlich wird noch eine Konfigurationsversion festgelegt. Die Einstellungen werden über das Kontextmenü der CPU aufgerufen.

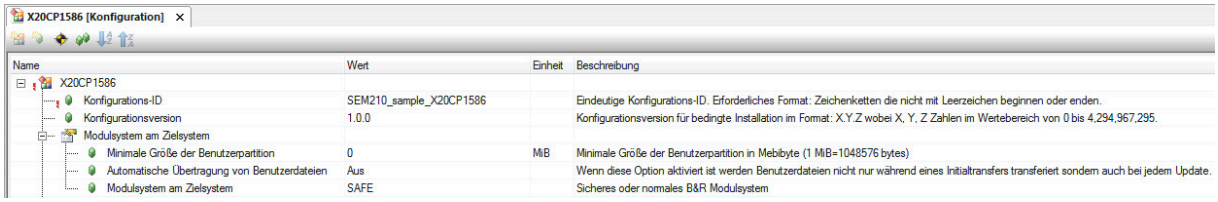


Abbildung 18: CPU Einstellungen: Einstellungen für Konfigurations-ID und Partitionierung

Konfigurations-ID

Ab Automation Runtime 4.25 wird jeder Konfiguration im Automation Studio Projekt eine eindeutige Konfigurations-ID zugewiesen. Die Konfigurations-ID dient zur eindeutigen Identifizierung des Projekts und wird mit "<AS Project name>_<Configuration name>" vorbelegt. Die Vergabe einer eindeutigen Konfigurations-ID im Projekt ist notwendig. Dadurch kann bei der Projektinstallation zwischen einer Aktualisierung (bei gleicher ID) und einem Initialtransfer (bei unterschiedlicher ID) unterschieden werden.

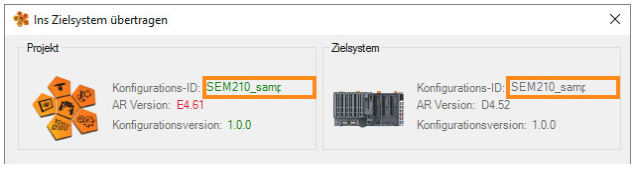


Abbildung 19: Vergleich der Konfigurations-ID im Transfer-Dialog

Partitionierung

Der Flash Speicher einer Steuerung ist als Dateisystem organisiert. Je nach gewählter Partitionierungsoption wird zwischen dem normalen und sicheren Dateisystem unterschieden.

Beim **normalen Dateisystem** wird eine Partition auf dem Flash erstellt, auf welcher Automation Runtime sowie die Anwendung gespeichert sind.

Beim **sicheren Dateisystem** hingegen sind Automation Runtime und die Applikation auf verschiedenen Partitionen abgelegt. Die Partitionsgrößen für Automation Runtime und die Applikation werden automatisch berechnet[1].

[1] Die Partitionsgrößen für Data1 und Data2 werden im Verhältnis 2:1 festgelegt. Dies ist erforderlich, damit während der Projektinstallation ausreichend Speicherplatz für das Transfermodul zur Verfügung steht.

Zusätzlich zum normalen und sicheren Dateisystem kann noch eine **Benutzerpartition** hinzugefügt werden. Auf dieser kann der Anwender zur Laufzeit z. B. Rezeptdaten speichern. Für die Benutzerpartition wird die Speichergröße manuell festgelegt.

Programmierung \ Editoren \ Konfigurationseditoren \ Hardwarekonfiguration \ CPU Konfiguration \ SG4

Projekt Management \ Projektinstallation

- Projektinstallation durchführen \ Einstellungen \ Systemkonfigurationseinstellungen
- FAQ

3.3 Projektinstallation

Automation Runtime kann über die Onlineverbindung übertragen werden. Voraussetzung dafür ist, dass die Onlineschnittstelle richtig konfiguriert wurde bzw. im entsprechenden Betriebszustand aktiv ist, siehe 5.1 "Automation Runtime starten" auf Seite 31. Alternativ stehen für die Übertragung des Automation Studio Projekts, inklusive Automation Runtime, die Offline-Installation und das Projektinstallationspaket zur Verfügung.

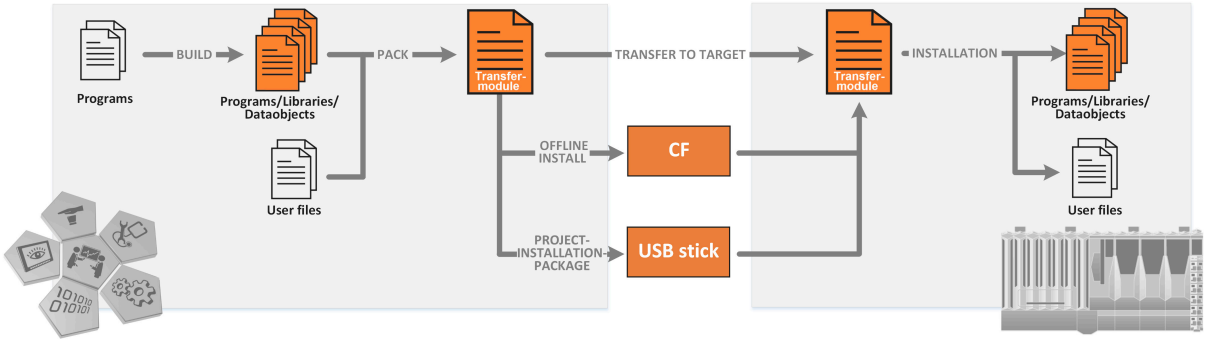


Abbildung 20: Schematische Darstellung der Varianten für die Projektinstallation

Verbindungsaufbau

Der Verbindungsaufbau zur Steuerung wird über die Zielsystemsuche des Automation Studio ermöglicht. Dabei wird das Netzwerk nach B&R Steuerungen durchsucht. Die Verbindungseinstellungen der Steuerungen können im Suchdialog temporär geändert werden.

Programmierung \ Build & Transfer \ Verbindungsaufbau zum Zielsystem \ Ethernet Verbindung \ Zielsystemsuche

Online-Installation

Nachdem der Verbindungsaufbau erfolgt ist, kann Automation Runtime übertragen bzw. eine Online-Installation durchgeführt werden.

Programmierung \ Build & Transfer \ Online Dienste \ Automation Runtime übertragen \ Übertragen auf SGx Zielsysteme \ Installation über Online Verbindung

Projekt Management \ Projektinstallation \ Szenarien \ Online-Inbetriebnahme

Offline-Installation

Bei der Offline-Installation wird mit Hilfe von Automation Studio ein Installationsmedium (Compact Flash bzw. CFast) erzeugt. Dieses wird anschließend in das Zielsystem gesteckt. Nach dem Einschalten der Versorgungsspannung werden Automation Runtime und die Applikation gestartet.

Projekt Management \ Projektinstallation \ Szenarien \ Offline-Inbetriebnahme

Projektinstallationspaket - USB-Installation / Netzwerkinstallation

Automation Runtime und die Applikationssoftware können durch die Erstellung eines Projektinstallationspakets mit Hilfe eines USB-Sticks, einer Compact Flash oder einem DHCP-Server auf das Zielsystem übertragen werden.

Wird ein USB-Stick mit Projektinstallationspaket an das Zielsystem angesteckt, so kann die Aktualisierung durch einen Neustart oder durch die Bibliothek "ArProject" ausgelöst werden.

Welcher Betriebsmodus für die Installation mit einem Projektinstallationspaket?

Bei der Installation mit einem Projektinstallationspaket kann es, je nach Gerätekonfiguration und Betriebszustand, erforderlich sein das Gerät in den Betriebsmodus "BOOT" zu versetzen. Im Modus "BOOT" ist der Mechanismus für die Installation eines Projektinstallationspaketes vom USB-Stick oder vom Netzwerk immer aktiv.

Projekt Management \ Projektinstallation

- Szenarien \ Offline-Inbetriebnahme
- Übersicht \ Übertragung \ Projektinstallationspaket

Programmierung \ Bibliotheken \ Konfiguration, Systeminfo, Laufzeitkontrolle \ ArProject

Gerätespezifische Installation

Es wird grob zwischen Geräten mit integriertem Flash Speicher und Geräten mit Compact Flash unterschieden. Die gerätespezifische Automation Runtime Installation, Erstinbetriebnahme sowie die Beschreibung für den Reset-Taster sind dem jeweiligen Anwenderhandbuch zu entnehmen.

Wie wird die Steuerung in den Modus "BOOT" versetzt?

Alle Zielsysteme verfügen die Möglichkeit, gezielt im Betriebsmodus "BOOT" zu starten. Abhängig vom Zielsystem geschieht dies mit dem Reset-Taster, Betriebsartenwahlschalter oder Knotenwahlschalter. Mit Hilfe des Reset-Tasters kann das System neu gestartet, sowie der Betriebsmodus gewechselt werden. Die mit einem Betriebsartenwahlschalter eingestellte Betriebsart wird nach einem Neustart übernommen. Weitere Informationen zu den möglichen Betriebsarten und Aktionen sind dem Datenblatt des jeweiligen Zielsystems zu entnehmen.

Programmierung \ Build & Transfer \ Online Dienste \ Automation Runtime übertragen

Hardware \ Power Panel \ Power Panel C70 \

- Inbetriebnahme \ Erstinbetriebnahme
- Gerätebeschreibung \ Bedien- und Anschlusselemente \ Reset-Taster / Betriebsmodi

Hardware \ X20 System \ X20 Module \ CPUs \ X20(c)CP1301, X20CP1381 und X20CP1382 \ Bedien- und Anschlusselemente \

- System Flash programmieren
- Taster für Reset und Betriebsmodus

Aufgabe: Automation Runtime Version kontrollieren und aktualisieren

Ziel dieser Übung ist es, sich mit den Möglichkeiten Automation Runtime am Zielsystem zu aktualisieren, auseinanderzusetzen. Es wird nach Automation Runtime Upgrades gesucht, die Einstellungen für die Runtime Versionen im Projekt kontrolliert, Konfigurations-ID und Partitionierung festgelegt und der geeignete Mechanismus für die Aktualisierung des Zielsystems ausgewählt.

1) Nach Automation Runtime Upgrades suchen

   Menü "Extras" \ "Upgrades"

2) Runtime Versionen kontrollieren bzw. anpassen

   Menü "Projekt" \ "Runtime Versionen ändern..."

3) Konfigurations-ID und Partitionierung festlegen

   CPU Konfiguration (Konfigurations-ID, Modulsystem am Zielsystem)

4) Online-, Offline-Installation oder Projektinstallationspaket für das Zielsystem passend auswählen

   Im Datenblatt werden dazu Hinweise im Abschnitt "System Flash programmieren" angeboten.

5) Installation durchführen

   Im Falle einer Online-Installation: Initialtransfer erzwingen

4 Speicherverwaltung

Der Speicher eines Automation Runtime Zielsystems ist in **RAM** und **ROM** unterteilt.

Teile dieser Speicher werden in der Laufzeit von Automation Runtime verwendet; der Rest steht der Applikation zur Verfügung.

4.1 Logische Unterteilung des Flash Speichers

Ein Automation Studio Projekt erzeugt beim Kompiliervorgang für das Automation Runtime ausführbare Module. Diese werden bei der Projektinstallation auf den Flash Speicher übertragen. Als Speichertypen kommen Compact Flash, CFast und integrierte Flash Speicher zum Einsatz.

Jedem Modul der Softwarekonfiguration wird automatisch ein Zielspeicher zugewiesen.
Module, die zu Automation Runtime gehören, werden in das **System ROM** übertragen.
Module, die zur Applikation gehören, werden in das **User ROM** übertragen.
Dies ist eine rein logische Unterteilung. Die Daten, die kommen, werden auf demselben Speichermedium abgelegt.

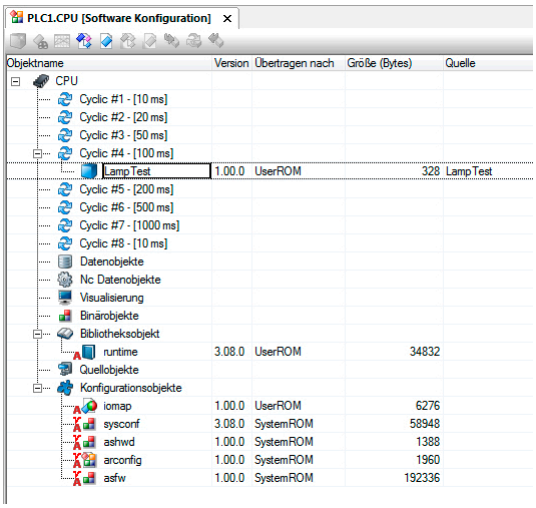


Abbildung 21: Zielspeicher für Softwareobjekte in der Softwarekonfiguration

Beim **normalen B&R Dateisystem** werden Module aus dem System ROM und User ROM in verschiedenen Ordnern auf der Partition "C" abgelegt.

Beim **sicheren B&R Dateisystem** wird das Automation Runtime auf der Partition "C" abgelegt. Die Module des System ROM und des User ROM werden auf den Partitionen "D" und eine Kopie dieser Daten auf Partition "E" abgelegt.

Durch diesen Mechanismus werden Systemmodule von der Applikation getrennt.

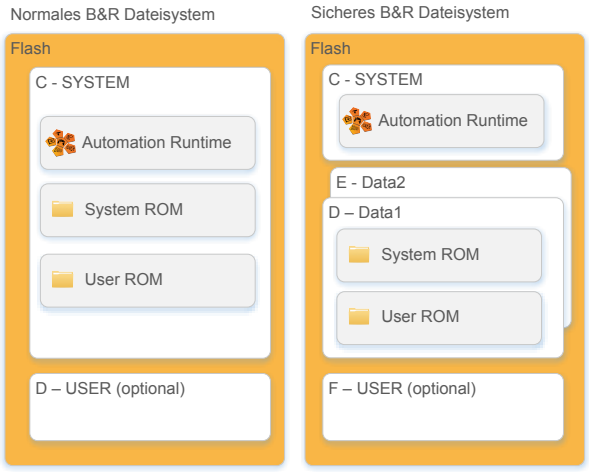


Abbildung 22: Vergleich vom normalen zum sicheren B&R Dateisystem

Zusätzlich kann noch eine **Benutzerpartition**, für die Verwaltung zur Laufzeit erzeugten Dateien, konfiguriert werden. Mit Hilfe der Transfereinstellungen bei der Projektinstallation werden Daten auf die Benutzerpartition übertragen.

Weiterführende Informationen:

- 5.5 "Einstellungen beim Übertragen von Programmen" auf Seite 47

Die Einstellungen für das normale oder sichere B&R Dateisystem werden in der CPU Konfiguration festgelegt.

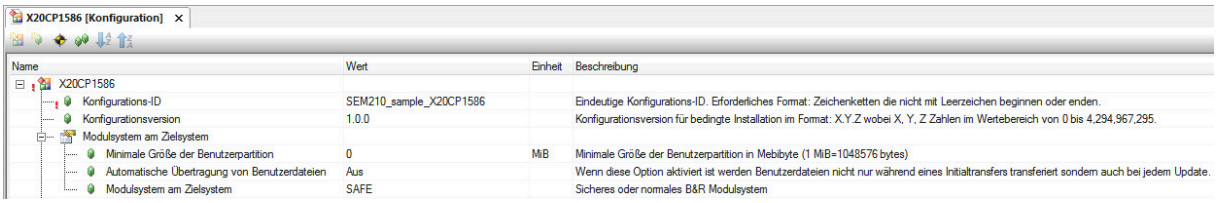


Abbildung 23: Konfiguration der Partitionierung für den Flash Speicher in der CPU Konfiguration

Echtzeit Betriebssytem \ Arbeitsweise \ Speicher \ Speichertypen

Projekt Management \ Configuration View \ Softwarekonfiguration

Projekt Management \ Projektinstallation \ Projektinstallation durchführen \ Einstellungen

- Systemkonfigurationseinstellungen
- Installationseinstellungen

4.2 Verwendete RAM Speicher

Für die Ausführung von Automation Runtime und der Applikation wird ein DRAM als schneller Schreib- und Lesespeicher verwendet.

Alternativ besitzen die Zielsysteme einen gepufferten RAM Speicher, welcher beispielsweise für den Datenerhalt über einen Neustart hinweg eingesetzt wird.

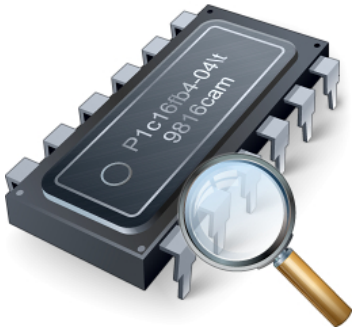


Abbildung 24: DRAM, SRAM und FRAM

Welchen Speicher hat die vorliegende Steuerung?

Die im Zielsystem eingesetzten Speichertypen und Speichergrößen sind im Datenblatt der ausgewählten CPU dokumentiert.

Weiterführende Informationen:

- 4.3 "Werkzeuge zum Ermitteln von Speicherinformationen" auf Seite 23

DRAM

Beim Hochlauf werden das Automation Runtime, alle Konfigurationen und die Tasks in das DRAM kopiert und dort ausgeführt. Dies ist erforderlich, da der Zugriff auf das DRAM schneller ist als auf einen Flash Speicher.

Ein Task benötigt im DRAM zusätzlich den projektierten lokalen- oder globalen Variablenspeicher. Die Initialisierung dieses Speichers wird automatisch vorgenommen.

SRAM und FRAM

Ein SRAM (statisches RAM) ist ein batteriegepufferter Speicher. Dadurch können Daten, auch über einen Spannungsausfall hinweg, erhalten werden. Dafür wird eine funktionsfähige Pufferbatterie vorausgesetzt.

Neuere Steuerungsmodelle benötigen keine Pufferbatterie mehr. Ein FRAM speichert im Gegensatz zum SRAM und DRAM die Daten auf einem nichtflüchtigen elektronischen Speichertyp.

Echtzeituhr

Die Pufferung der Echtzeituhr ist geräteabhängig unterschiedlich realisiert. Die Dauer der Nullspannungssicherheit sowie die Ganggenauigkeit in Abhängigkeit von der Umgebungstemperatur werden im jeweiligen Datenblatt der verwendeten CPU beschrieben.

Weiterführende Informationen:

- 4.4.3 "Initialisierung des Variablenspeichers" auf Seite 29
- 5.1.4 "Neustart und Spannungsausfall" auf Seite 34
- 5.1.5 "Verwendung von Retain Variablen" auf Seite 35

Was ist zu tun, wenn die Pufferbatterie ausgetauscht werden muss?

Informationen zum Wartungsintervall der Pufferbatterie und Anweisungen zum Austausch sind im Datenblatt der jeweiligen CPU dokumentiert.

Echtzeit Betriebssystem \ Arbeitsweise \ Speicher \ Speichertypen

4.3 Werkzeuge zum Ermitteln von Speicherinformationen

Automation Studio vereint eine Vielzahl von Diagnosewerkzeugen, welche unter anderem Aufschluss über den Systemzustand geben. Nachfolgend werden Diagnosewerkzeuge zur Ermittlung von Speicherinformationen vorgestellt.

Online Info

Über das Menü "Online" \ "Info..." wir der Online-Info-Dialog geöffnet. Der verfügbare Speicher und der Status der Pufferbatterie werden angezeigt. Zusätzlich können Datum und Uhrzeit der Steuerung ausgelesen und gesetzt werden. Die Änderung von Datum und Uhrzeit werden im Logger-Modul "System" protokolliert.

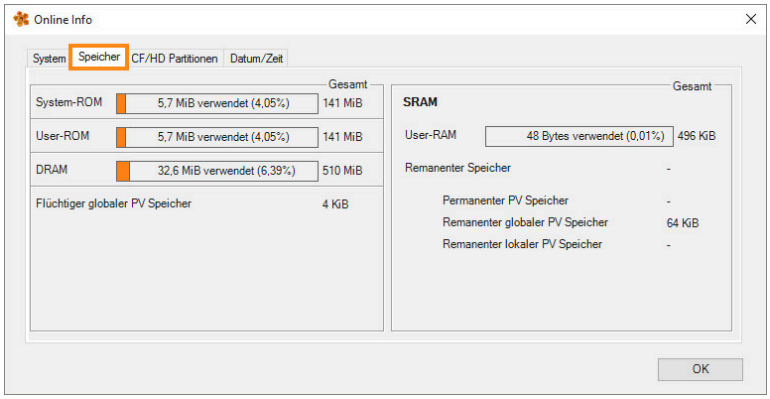


Abbildung 25: CPU Informationen auslesen

System Diagnostics Manager (SDM)

Der System Diagnostics Manager (SDM) ist integraler Bestandteil des Automation Runtime. Über das Menü "Extras" \ "Systemdiagnosemanager" wird ein Browserfenster geöffnet und es werden die Diagnoseseiten des SDM angezeigt.

In der Kategorie "System" \ "Memory" werden Informationen zur Speicherverwendung angezeigt. In der Kategorie "Software" erhält man Informationen, welche Module auf dem Zielsystem geladen wurden.

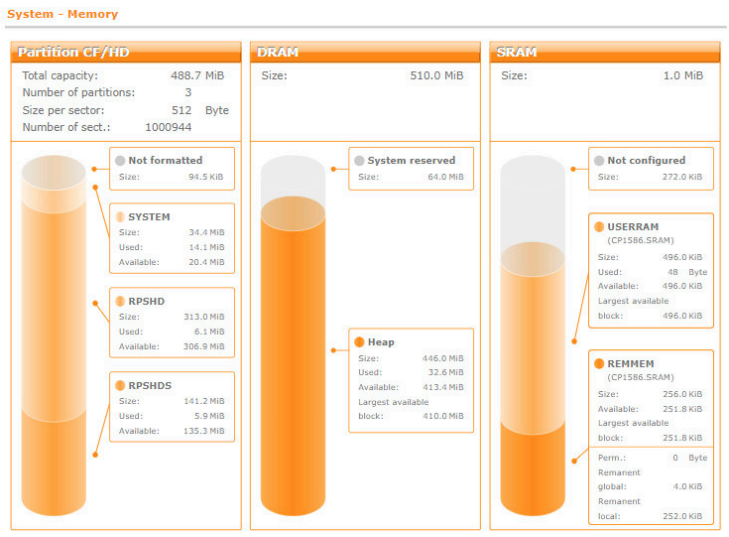


Abbildung 26: Speicherbelegung im SDM anzeigen

Online Softwarevergleich

Der Softwarevergleich vergleicht die in Automation Studio projektierten Module mit den auf dem Zielsystem installierten Module.

Der Softwarevergleich wird im Menü unter "Online" \ "Vergleich" \ Software" geöffnet.

In der Ansicht für den Softwarevergleich werden die Module des Projekts und der Steuerung gegenübergestellt. In der Spalte "Größe (Bytes)" wird der Zielspeicher des Moduls auf dem Zielsystem angezeigt. Es können die Modulversion, die Modulgröße, der Zielspeicher und das Kompilierdatum der Module verglichen werden.

Im Kontextmenü eines Moduls werden Optionen angezeigt. Module können einzeln gelöscht oder angehalten werden[2].

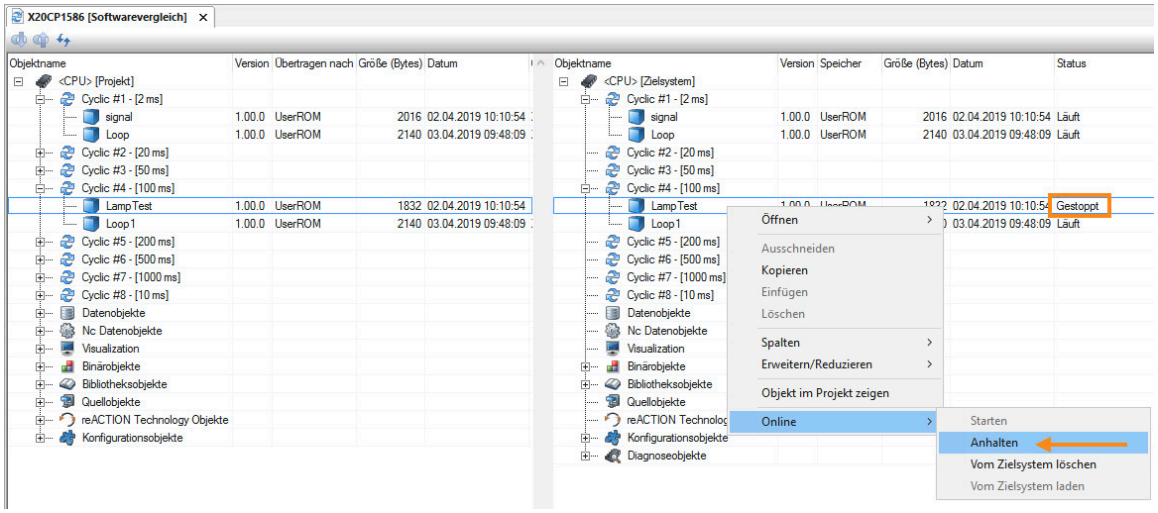


Abbildung 27: Öffnen des Online Softwarevergleich

Bibliotheksfunktionen

Mit Hilfe des Funktionsblocks MEMxInfo() aus der Bibliothek AsHW kann in der Applikation der freie Speicher am Zielsystem ermittelt werden.

Direkten Zugriff auf das Dateisystem am Flash Speicher erhält man beispielsweise mit den Bibliotheken FileIO und MpFile.

[2] Ab Automation Runtime B4.33 können jene Module, die über die Projektinstallation oder ein Projektupdate installiert wurden, ausschließlich über eine Projektinstallation oder ein Projektupdate gelöscht werden. Das Löschen von Modulen am Zielsystem ist unter anderem nur für zur Laufzeit generierte Datenobjekte möglich.

Diagnose und Service \ Diagnosewerkzeug

- Informationen zum Zielsystem
- System Diagnostics Manager
- Monitor Modus \ Online Softwarevergleich

Programmierung \ Bibliotheken

- Konfiguration, Systeminfo, Laufzeitkontrolle \ AsHW
- Datenzugriff und Datenablage \ FileIO

Services \ mapp Services \ MpFile: Dateiverwaltungssystem

Aufgabe: Datum und Uhrzeit setzen

Zur richtigen Interpretation von Systemereignissen ist es erforderlich, Datum und Uhrzeit auf der Steuerung richtig einzustellen. Dies ist im Online-Info-Dialog möglich. Im Logger-Modul "System" wird die Änderung protokolliert.

1) Datum und Uhrzeit unter "Online" \ "Info..." ändern
2) Logger öffnen, Menüeintrag "Öffnen" \ "Logger"
3) Ergebnis im Logger-Modul "System" kontrollieren

Im Logger-Modul "System" wird eine Warnung angezeigt.

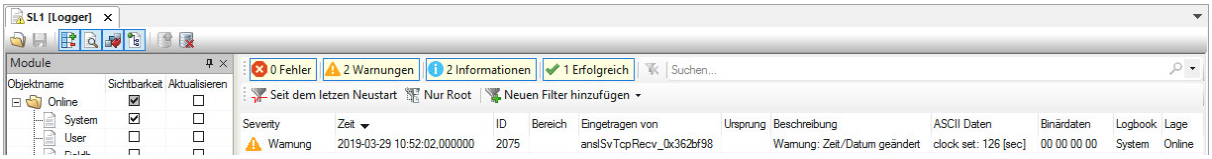


Abbildung 28: Im Logger-Modul "System"wird die Änderung von Datum und Uhrzeit angezeigt

Aufgabe: Freien Speicher am Zielsystem überprüfen

Das Projekt, welches während der Arbeit mit dem Trainingsmodul "TM210 – Arbeiten mit Automation Studio" erzeugt wurde, ist bereits auf dem Zielsystem installiert.

Mit einer der vorher beschriebenen Möglichkeiten ist die Speicherbelegung des Zielsystems zu ermitteln. Es ist festzustellen, welche Speichertypen auf dem eingesetzten Zielsystem existieren und wie diese verwendet werden. Des Weiteren ist zu kontrollieren, welche Module auf das Zielsystem geladen wurden.

1) Freien DRAM Speicher mit Online-Info-Dialog ermitteln
2) Freien Flash Speichers mit System Diagnostics Manager ermitteln
3) Mit System Diagnostics Manager kontrollieren, welche Programme auf dem Zielsystem sind
4) Mit Softwarevergleich kontrollieren, welche Programme auf dem Zielsystem sind

4.4 Globale und lokale Variablen

Variablen sind symbolische Elemente in der Programmierung, deren Aufbau und Größe durch ihren Datentyp bestimmt wird. Beim Kompilieren wird den Variablen vom Compiler eine Speicherstelle zugewiesen.

Der Gültigkeitsbereich (Scope) und die Eigenschaften einer Variable bestimmen das Verhalten beim Hochlauf und zur Laufzeit.

Programmierung \ Variablen und Datentypen

4.4.1 Lokale Variablen

Lokale Variablen werden im Gültigkeitsbereich eines Programms definiert. Der direkte Zugriff auf diese Variablen aus anderen Programmen ist nicht möglich.

Eine lokale Variable wird in einer .var Datei auf gleicher Ebene wie das Programm verwaltet.

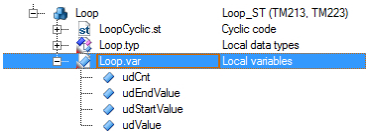


Abbildung 29: Lokale Variablen von Programm "Loop"

Müssen lokale Variablen von einem Programm auf lokale Variablen eines anderen Programms übertragen werden, so wird dies mit Hilfe einer Variablenzuordnungsdatei in der Configuration View realisiert.

Aufgabe: Programm "Loop" mit lokalen Variablen erstellen

Es wird ein neues Strukturierter Text Programm mit dem Namen "Loop" erstellt.
4 Variablen vom Datentyp UDINT werden deklariert: "udCnt", "udValue", "udStartValue" und "udEndValue".

Im zyklischen Programmteil wird eine Schleife erstellt, welche in weiteren Übungen erklärt und angewendet wird.

1) Neues ST Programm mit dem Namen "Loop" erstellen
2) Datei "Loop.var" öffnen und die 4 Variablen einfügen
3) Nach dem Speichern der Datei "Loop.var" können die Variablen im Programm "LoopCyclic.st" verwendet werden

```
PROGRAM _CYCLIC

    FOR udCnt := udStartValue TO udEndValue DO
        udValue := udValue + 1;
    END_FOR

END_PROGRAM
```

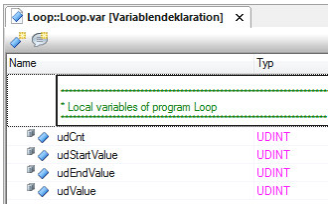


Abbildung 30: Variablendeklaration für das Programm "Loop"

Aufgabe: Mehrfachzuweisung eines Programms in die Software Konfiguration

Wenn die vorherige Aufgabe abgeschlossen wurde, kann mit der Mehrfachzuweisung des Programms begonnen werden.
Das Programm "Loop" wird zweimal per Drag & Drop aus der Logical View in die Software Konfiguration übernommen.

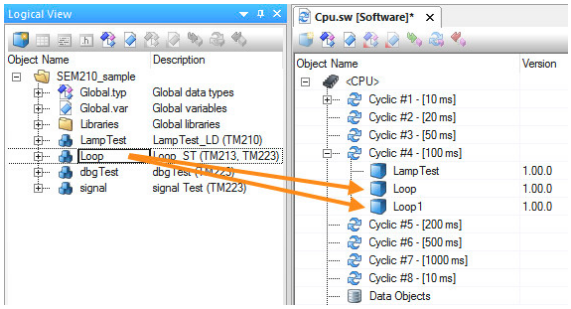


Abbildung 31: Mehrfachzuweisung von Programmen

1) Softwarekonfiguration öffnen
2) Logical View öffnen
3) Programm "Loop" per Drag & Drop zweimal in die Softwarekonfiguration einfügen

Wie wird der Variablenmonitor für "Loop" bzw. "Loop1" geöffnet?

Der Variablenmonitor für beide Instanzen des Programms "Loop" kann über das Kontextmenü in der Softwarekonfiguration oder der Logical View geöffnet werden. Beim Öffnen des Variablenmonitors über die Logical View erscheint ein Auswahldialog, für welche Instanz der Variablenmonitor geöffnet werden soll.
Alternativ wird der Variablenmonitor über die Tastenkombination **<STRG> + <w>** geöffnet.

Im Variablenmonitor von Task "Loop" und "Loop1" werden alle Variablen eingefügt. Beide Tasks verfügen nur über lokale Variablen. Eine Veränderung der Variablen im Variablenmonitor hat damit keine Auswirkung auf die jeweils andere Instanz des Programms "Loop".

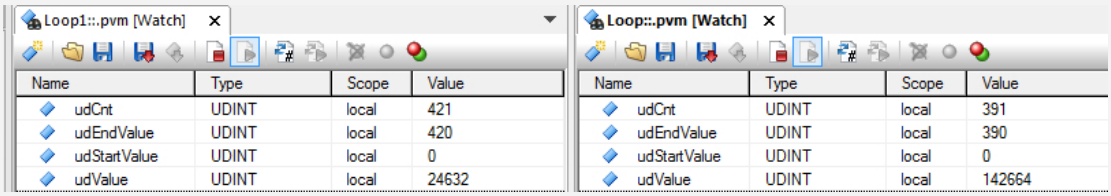


Abbildung 32: Variablenmonitor für die Tasks "Loop" und "Loop1"

Wieso können im Variablenmonitor nicht alle Variablen angezeigt werden?

Variablen, welche in der Variablendatei enthalten sind, jedoch im Programm nicht verwendet werden, sind auf dem Zielsystem nicht verfügbar.

Während des Build wird eine entsprechende Warnung ausgegeben.

Warnung 424: Variable <variable name> ist deklariert aber in der aktuellen Konfiguration nicht verwendet.

Programmierung \ Variablen und Datentypen \ Bezugsrahmen (Scope) von Deklarationen

Diagnose und Service \ Diagnosewerkzeug \ Variablenmonitor (Watch)

4.4.2 Globale und paketglobale Variablen

Die **globalen Variablen** werden in der Logical View auf oberster Ebene dargestellt. Sie sind innerhalb des ganzen Automation Studio Projektes sichtbar. Sie können in allen Programmen verwendet werden. Zur besseren Strukturierung des Projektes können weitere .var Dateien erstellt werden.

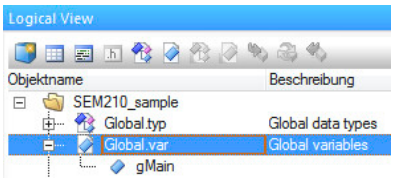


Abbildung 33: Globale Variablen in der Logcal View

Variablen, die innerhalb eines Pakets deklariert wurden, sind **paketglobale Variablen**. Diese sind nur in dem jeweiligen Paket und allen untergeordneten Paketen sichtbar.

Wann werden globale Variablen verwendet?

Globale Variablen sollen nur dann verwendet werden, wenn ein Datenaustausch zwischen mehreren Programmen gefordert ist. Alternativ dazu können lokale Variablen verschiedener Programme über eine Variablenzuordnung miteinander verbunden werden.

Aufgabe: Globale Variable "gMain" erstellen und in Programm "Loop" verwenden

In der globalen Variablendeklaration "Global.var" ist eine neue Variable mit dem Namen "gMain" vom Datentyp UDINT einzufügen.

Im Programm "Loop" ist der Wert dieser Variable zyklisch zu erhöhen.

```
gMain := gMain + 1;
```

1) Datei "Global.var" öffnen
2) Variable "gMain" vom Datentyp UDINT erstellen
3) Nach dem Speichern der Datei "Global.var" kann die Variable im Programm "Loop.st" im Programm verwendet werden.

Wird im Variablenmonitor für den Task "Loop" und "Loop1" die Variable "gMain" eingefügt, ist die Wertänderung in beiden Tasks sichtbar.

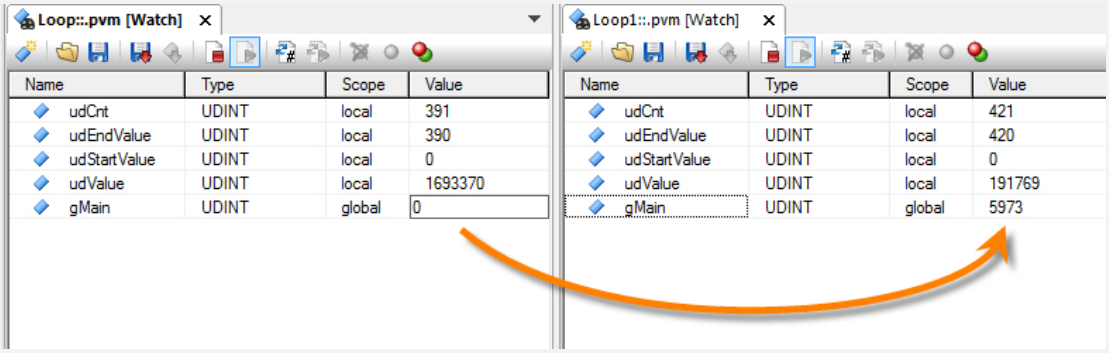


Abbildung 34: Beschreiben von globalen Variablen

Programmierung \ Variablen und Datentypen \ Bezugsrahmen (Scope) von Deklarationen

Programmierung \ Editoren \ Konfigurationseditoren \ Editor zur Variablen Zuordnung

4.4.3 Initialisierung des Variablenspeichers

Beim Hochlauf werden alle Variablen, für die kein Initialwert vergeben wurde, automatisch mit dem Wert "0" initialisiert. In der Variablendeklaration kann in der Spalte "Wert" ein Initialwert angegeben werden.

Die Initialisierung über die Variablendeklaration ersetzt beispielsweise folgende Programmzeilen im Init-Programm "**LoopInit.st**":

```
PROGRAM _INIT
    udEndValue := 1000;
END_PROGRAM
```

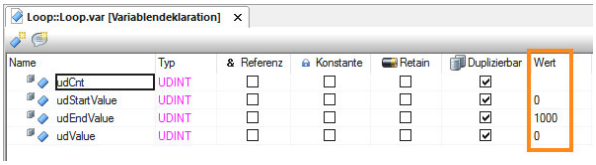


Abbildung 35: Variableninitialisierung in „Loop.var“

Aufgabe: Variable "udEndValue" initialisieren

Die Variable "udEndValue" des Programms "Loop" ist mit einem Intialwert von 1000 zu projektieren. Im Variablenmonitor ist der Wert zu beobachten.

1) Datei "Loop.var" öffnen
2) Der Variable "udEndValue" in der Spalte "Wert" den Initialwert 1000 geben

Im Variablenmonitor des Task "**Loop**" und "**Loop1**" wird die Variable "**udEndValue**" mit dem Wert 1000 initialisiert. Zur Laufzeit kann die Variable auf einen beliebigen anderen Wert verändert werden.

Programmierung \ Editoren \ Tabelleneditoren Allgemein \ Deklarationseditoren \ Tabelleneditor für Variablendeklaration

4.4.4 Verwendung von Konstanten

Konstanten sind Variablen, deren Werte während des Programmablaufes nicht veränderbar sind. Diese werden in der Programmierung als fixe Zahlenwerte verwendet. Dadurch wird die Lesbarkeit und Wartbarkeit von Programmen erhöht.

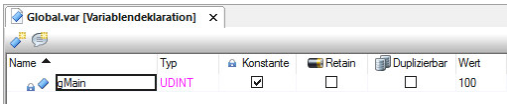


Abbildung 36: Deklaration von Konstanten

Aufgabe: Globale Variable "gMain" als Konstante

Die globale Variable "gMain" ist als Konstante mit einem Wert von 100 zu projektieren.

1) Datei "Global.var" öffnen
2) Der Variable "gMain" in der Spalte "Wert" den Wert 100 geben
3) Die Variable "gMain" als Konstante projektieren

Während des Build Vorgangs wird eine Fehlermeldung ausgegeben, da auf die Variable "**gMain**" ein schreibender Zugriff erfolgt.

Der schreibende Zugriff auf Konstanten in Programmen ist nicht zulässig. Zur fehlerfreien Ausführung des Programms darf die Variable "**gMain**" nicht als Konstante definiert werden.

LoopCyclic.st (Ln: 19, Col: 8) : Fehler 1138: Schreibzugriff auf Variable 'gMain' nicht zulässig.

Sinnvoller ist es die Variable "**udStartValue**" als Konstante mit dem Wert "0" zu initialisieren. Diese Variable wird im Programm nur lesend verwendet.

Wie und wo werden die Variablen im Projekt verwendet?

Um die lesende und schreibende Verwendung von Variablen in Programmen zu ermitteln wird die Querverweisliste verwendet. Eine Querverweisliste wird über das Hauptmenü "Projekt" \ "Querverweis erstellen" erzeugt.

Programmierung \ Variablen und Datentypen \ Variablen \ Konstanten

Projekt Management \ Arbeitsoberfläche \ Ausgabefenster \ Cross reference

5 Laufzeitverhalten

Dieses Kapitel beschreibt das Laufzeitverhalten der Anwendung auf dem Zielsystem.
Das Hochlaufverhalten, die Programminitialisierung und der zyklische Programmablauf werden erklärt.

5.1 Automation Runtime starten

Nach dem Einschalten des Zielsystems wird Automation Runtime gestartet. Vor dem Ausführen der zyklischen Programme werden noch einige Aufgaben erledigt.

Folgende Aufgaben werden vor dem zyklischen Programm ausgeführt:

- Überprüfen der Hardware
- Überprüfung ob Upgrade von Hardware-Firmware erforderlich ist
- Überprüfen der BR Module
- Kopieren der BR Module vom ROM ins DRAM
- Kopieren der Retain Variablen ins DRAM
- Variablenspeicher auf ihre Initialwerte setzen
- Abarbeitung der Programm-Initialisierung
- Aktivieren der zyklischen Programme

Tritt während der Hochlaufphase kein Fehler auf, startet das Zielsystem im RUN Modus.

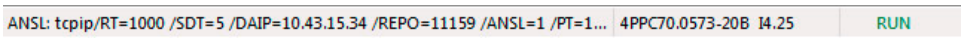


Abbildung 37: Power Panel C70 im RUN Modus

Echtzeit Betriebssystem \ Arbeitsweise \ Hochlaufverhalten

Echtzeit Betriebssystem \ Arbeitsweise \ Modul / Datensicherheit

5.1.1 Automation Runtime Betriebszustände

Nach dem Hochlauf der Steuerung sind folgende vier Betriebszustände möglich: BOOT, DIAGNOSE, SERVICE und RUN.

Im Betriebszustand "RUN" startet das Automation Runtime, die mit Automation Studio erstellte Anwendung und führt diese zyklisch aus.

Bestimmte Ereignisse führen dazu, dass das Zielsystem den Hochlauf abbricht und für Diagnosezwecke im entsprechenden Betriebszustand verweilt. Der aktuelle Betriebszustand wird anhand der Status LEDs der CPU und der Statusleiste im Automation Studio abgelesen. Details werden im Logger-Modul "System" protokolliert.

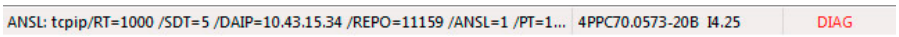
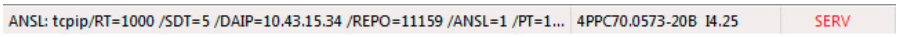
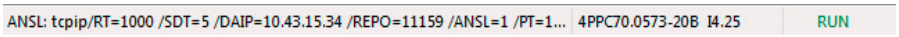
Betriebszustand	Bedingungen, welche den Systemzustand erzwingen
BOOT	• Keine Compact Flash gesteckt • Kein Betriebssystem auf Compact Flash bzw. CFast vorhanden[3] • Knotenschalter auf Stellung "00", Betriebsartenwahlschalter auf Stellung "BOOT", Reset-Taster [4]  Abbildung 38: Statusleiste - Modus "BOOT"
DIAGNOSE	• Fataler Systemfehler • Knotenschalter auf Stellung "FF", Betriebsartenwahlschalter auf Stellung "DIAG", Reset-Taster  Abbildung 39: Statusleiste - Modus "DIAGNOSTICS"
SERVICE	• Division durch Null[5] • PageFault • Zykluszeitverletzung • Fehlende Hardwaremodule • CPU Halt im Automation Studio ausgeführt  Abbildung 40: Statusleiste - Modus "SERVICE"
RUN	• Kein Fehler  Abbildung 41: Statusleiste - Modus "RUN"

Tabelle 1: Übersicht Automation Runtime Betriebszustände

Echtzeit Betriebssystem \ Arbeitsweise \ Betriebszustände

Hardware \ X20 System \ X20 Module \ CPUs

- X20(c)CP158x und X20(c)CP358x \ Betriebsmodusschalter
- X20CP1381-RT und X20CP1382-RT \ Bedien- und Anschlusselemente \ Taster für Reset und Betriebsmodus

[3] Voraussetzung hierfür ist, dass das eingesetzte System über ein Default Automation Runtime verfügt. PC basierende Systeme verfügen über kein Default Automation Runtime. In diesem Fall kann keine Verbindung zum Zielsystem aufgebaut werden. Es muss eine Offline-Installation durchgeführt werden.

[4] Die Einstellung der Betriebsart ist geräteabhängig. Je nach Gerät kann dafür Knotenwahl-, Betriebsartenwahlschalter oder Reset-Taster verwendet werden, siehe jeweiliges Anwenderhandbuch.

[5] Im Gegensatz zu Intel Zielsystemen führt bei ARM Zielsystemen eine Division durch 0 nicht zu einer Prozessor-Exception sondern zum Ergebnis 0.

5.1.2 Automation Runtime Hochlaufphasen

Während dem Hochlauf der Steuerung sind folgende Zwischenzustände möglich: STARTUP, FIRMWARE und INIT. Zu diesem Zeitpunkt kann über die Onlineverbindung nicht auf die Steuerung zugegriffen werden.

Diese Zustände werden auch in der Statusleiste von Automation Studio dargestellt. Diese Phasen sind meist sehr kurz.

Systemzustand	Beschreibung
STARTUP	Hier werden betriebssystemrelevante Initialisierungen durchgeführt.
FIRMWARE	Während dieser Phase werden Firmware-Aktualisierungen durchgeführt, siehe 5.1.3 "Hardwareversionen und Firmwareupdate" auf Seite 33.
INIT	Hier werden die Init-Unterprogramme der Applikation ausgeführt.

Tabelle 2: Übersicht Automation Runtime Hochlaufphasen

Woran erkennt man die Hochlaufphasen am X20 System?

Die Phasen vor dem Betriebsmodus "RUN" (STARTUP, FIRMWARE, INIT) werden beim X20 System zusätzlich durch die grün blinkende R/E LED signalisiert.

Echtzeit Betriebssystem \ Arbeitsweise \ Hochlaufphasen

Hardware \ X20 System \ X20 Module \ CPUs \ X20(c)CP158x und X20(c)CP358x \ Status-LEDs

5.1.3 Hardwareversionen und Firmwareupdate

In Automation Studio gibt es für jedes Hardwaremodul eine eigene Hardwarebeschreibungsdatei, ein dazu passendes Bild für den System Designer und optional eine Firmware. Beim Hochlauf von Automation Runtime wird die Firmware auf dem Hardwaremodul mit der am System installierten Firmware verglichen und im Bedarfsfall ein Firmware-Update durchgeführt.

Ausgeliefert wird Automation Studio mit jeweils nur einer Hardwareversion pro Hardwaremodul. Über den Menüpunkt "Extras" \ "Upgrades" können weitere **Hardwareversionen** installiert werden.

In der Physical View kann in der Spalte "Version" aus den installierten Hardwareversionen ausgewählt werden. Bei der Installation von neuen Hardwareversionen wird diese nicht automatisch aktualisiert, sondern muss manuell ausgewählt werden.

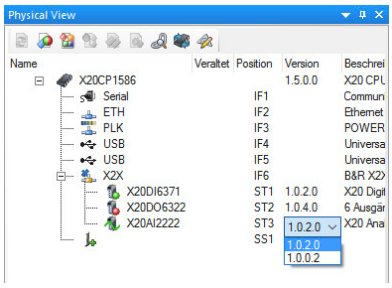


Abbildung 42: Auswahl einer Hardwareversion in der Physical View

Projekt Management \ Hardware Management \ Hardwareversionen

5.1.4 Neustart und Spannungsausfall

Das Neustartverhalten von Zielsystemen beim Wechsel von Betriebszuständen und nach Reset ist jeweils im Datenblatt der verwendeten Steuerung in den Abschnitten "Betriebsmodus- und Knotennummernschalter", "Betriebsmodusschalter" und "Reset" dokumentiert.

In Automation Studio kann das **Neustartverhalten** nach einem Reset oder Spannungsabfall konfiguriert werden. Diese Einstellung wird in der CPU Konfiguration festgelegt.

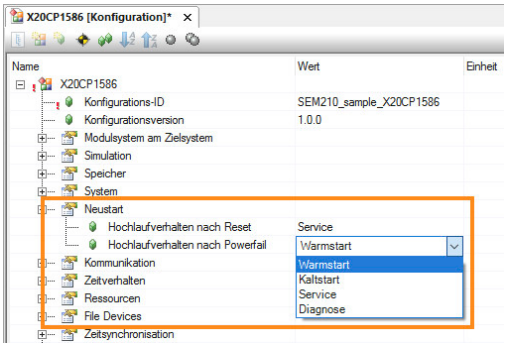


Abbildung 43: Konfiguration des Neustartverhaltens in den CPU Einstellungen

Warmstart

Ein Neustart mit Warmstart wird bei folgenden Aktionen ausgeführt:

- PowerON nach Spannungsausfall
- Betätigen des Reset-Tasters
- Ändern von Systemkonfigurationen und Übertragen auf das Zielsystem
- Ausführen eines Warmstarts mit Automation Studio
- Ausführen eines Warmstarts mit SYSreset() Funktion

Beim Warmstart behalten Retain Variablen ihren Wert.

Kaltstart

Ein Kaltstart wird bei folgenden Aktionen ausgeführt:

- Neustart nach einem Austausch einer Compact Flash, siehe 5.5 "Einstellungen beim Übertragen von Programmen" auf Seite 47 Beschreibung Offline-Installation und Initialtransfer
- Betätigen des Reset-Tasters
- Ausführen eines Kaltstarts mit Automation Studio
- Ausführen eines Kaltstarts mit SYSreset() Funktion
- Neustart, wenn die Pufferbatterie für das SRAM defekt ist

Beim Kaltstart werden Retain Variablen mit dem Wert "0" bzw. dem Initialwert aus der Variablendeklaration neu initialisiert. Permanente Variablen behalten ihren ursprünglichen Wert.

Programmierung \ Bibliotheken \ Konfiguration, Systeminfo, Laufzeitkontrolle \ SYS_Lib \ Funktionsbausteine und Funktionen \ System-Funktionen

PowerON Behandlung

Im spannungslosen Zustand liegen die Speicherbereiche SYSROM und USERROM auf der Compact Flash, CFast bzw. auf dem internen Speicher. Die remanenten[6] und permanenten Variablen und das USERRRAM liegen auf dem gepufferten RAM (SRAM). Beim Hochlauf der Steuerung wird Automation Runtime gestartet. SYSROM, USERROM, permanente und remanente Variablen werden von SRAM in DRAM kopiert.

Echtzeit Betriebssytem \ Arbeitsweise \ Modul / Datensicherheit \ PowerOn Behandlung

PowerOFF Behandlung

Viele B&R Zielsysteme sind mit einer Powerfail-Logik ausgestattet. Diese ermöglicht es, dass beim Auftreten von Spannungsausfällen das System in einen definierten Zustand gebracht wird. Bei einem Spannungsausfall werden folgende Aufgaben durchgeführt:

- Abschließen von Zugriffen auf Datenobjekte die im USERRAM liegen
- Umkopieren der remanenten Variablen (RETAIN) in das gepufferte SRAM

Echtzeit Betriebssytem \ Arbeitsweise \ Modul / Datensicherheit \ PowerOFF Behandlung

Aufgabe: Hochlauf und Betriebszustände

Zunächst ist die Konfiguration für das Neustartverhalten des eingesetzten Zielsystems zu überprüfen. Mit Hilfe von Betriebsartenwahlschalter, Reset-Taster und Automation Studio sollen verschiedene Betriebszustände erreicht werden. Im Datenblatt des verwendeten Zielsystems sind die Funktionen von Betriebsartenwahlschalter und Reset-Taster dokumentiert. Im Anschluss sind die Betriebszustände im Logger-Modul "System" zu kontrollieren.

5.1.5 Verwendung von Retain Variablen

Der Wert einer remanenten Variable (als Retain konfiguriert) bleibt nach einem Warmstart erhalten. Damit der Wert auch nach einem Kaltstart erhalten bleibt, muss die Variable als permanente Variable konfiguriert werden.

Retain Variablen

Um Variablenwerte remanent speichern zu können, müssen folgende Voraussetzungen auf dem Zielsystem und bei der Variablendeklaration erfüllt sein:

- Zielsystem mit gepuffertem RAM - siehe Datenblatt
- Konfiguration der Variable durch Aktivieren der Option "Retain"

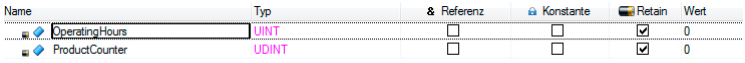


Abbildung 44: Konfiguration von Retain Variablen in der Variablendeklaration

6 Remanente Variablen sind in der Deklaration als "Retain" gekennzeichnet.

Beispiele für die Verwendung von Retain Variablen:

- Betriebsstundenzähler
- Stückzähler und Daten zur Anlageneffizienz
- Zustand und Status der Maschine zum Zeitpunkt eines Spannungsausfalles

Wie groß ist der remanente Speicher der Steuerung?

Je nach Zielsystem steht dem Anwender unterschiedlich viel remanenter Speicher zur Verfügung. Informationen über die Größe und Verfügbarkeit ist dem jeweiligen Datenblatt oder der Automation Help zu entnehmen.

Alternativ können Daten, welche nur einmalig beim Starten des Programms oder bei einer Änderung gelesen oder geschrieben werden, auf dem Flash Speicher abgelegt werden. Hierzu bieten sich die Funktionsblöcke der Bibliothek MpRecipe aus dem Technologiepaket mapp Services an.

Echtzeit Betriebssystem \ Arbeitsweise \ Speicher

Programmierung \ Editoren \ Tabelleneditoren Allgemein \ Deklarationseditoren \ Tabelleneditor für Variablendeklaration

Hardware \ X20 System \ X20 Module \ CPUs

- X20(c)CP158x und X20(c)CP358x \ Technische Daten
- X20(c)CP1301, X20CP1381 und X20CP1382 \ Technische Daten

Services \ mapp Services \ MpRecipe: Rezeptverwaltung

Permanente Variablen

Damit Retain Variablen auch nach dem Kaltstart erhalten bleiben, müssen diese zu den permanenten Variablen in der Configuration View (CPU.per) hinzugefügt werden.

Beispiele für Daten, welche permanent abgelegt werden:

- Betriebsstundenzähler
- Alle Daten die nicht auf einen Flash Speicher geschrieben werden, jedoch auch beim Kaltstart erhalten werden sollen

Es können nur **globale** retain Variablen als permanente Variablen konfiguriert werden.

Die Verwaltung der Variablenwerte im permanenten Speicher liegt in der Verantwortung des Anwenders.

Der Zustand von permanenten Variablen ohne korrekter Basis kann zu unerwünschtem Verhalten von Programmen führen. Dies ist besonders beim Austausch der CPU oder der Pufferbatterie zu beachten.

Gibt es Alternativen zum Speicherort der permanenten Variablen?

Als Alternative zur Verwendung permanenter Variablen, welche immer an eine bestimmte Steuerung gebunden sind, kann der Datenspeicher auf einem Technology Guard verwendet werden. Der Zugriff auf den Datenspeicher eines Technology Guards erfolgt über die Bibliothek AsGuard.

Programmierung \ Variablen und Datentypen \ Variablen\ Nichtflüchtige Variablen

Automation Software \ Technology Guarding

Programmierung \ Bibliotheken \ Konfiguration, Systeminfo, Laufzeitkontrolle \ AsGuard

Aufgabe: Anwendung von Retain Variablen

Im Programm "Loop" wird die Variable "udValue" mit jedem Schleifendurchlauf erhöht. Beim Warmstart soll der letzte Wert der Variable "udValue" erhalten werden.

1) Variable "udValue" als Retain deklarieren
2) Programm übertragen - Wert "udValue" im Variablenmonitor kontrollieren
3) Warmstart ausführen
4) Ergebnis kontrollieren

5.2 Initialisierung von Programmen

Für jedes Programm kann ein Init-Programm verwaltet werden. In einem Init-Programm können beispielsweise Variablen durch Berechnungen initialisiert werden.

Ausführen der Init-Programme

Bevor das erste zyklische Programm gestartet wird, werden alle Init-Programme einmalig, in der konfigurierten Reihenfolge, abgearbeitet. Solange die Init-Programme abgearbeitet werden, wird dies in der Statusleiste ("INIT") sowie durch die LEDs auf der CPU signalisiert.

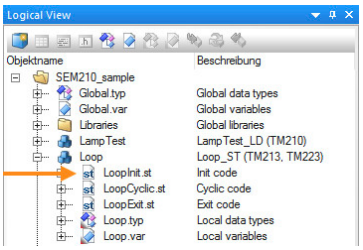


Abbildung 45: Initialisierung von Programmen

In diesem Beispiel würden die Init-Programme in folgender Reihenfolge abgearbeitet werden:

1) Init-Programm des Task "LampTest"
2) Init-Programm des Task "Loop"
3) Init-Programm des Task "Loop1"

Das Init-Programm ist nicht zykluszeitüberwacht und führt somit bei länger dauernden Initialisierungen zu keiner Zykluszeitverletzung.

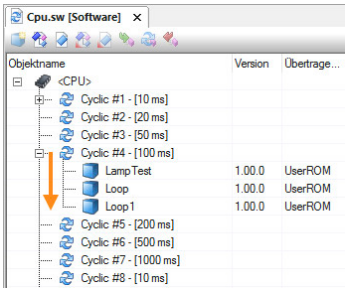


Abbildung 46: Abarbeitung der Init-Programme

Funktionsaufrufe im Init-Programm

Bei der Verwendung von Funktionsblöcken ist darauf zu achten, dass die Funktion bereits beim ersten Aufruf das Ergebnis liefert, weil das Init-Programm nicht zyklisch ausgeführt wird.

Funktionsblöcke, welche mehrere Aufrufe benötigen, müssen im zyklischen Programmablauf aufgerufen werden. Hinweise zur Verwendung von Funktionsblöcken im Init-Programm sind in der entsprechenden Dokumentation nachzulesen.

Echtzeit Betriebssystem \ Zielsysteme \ SG4 \ Laufzeitverhalten - SG4 \ Start der Init UPs - SG4

5.3 Ausführung von zyklischen Programmen

Ein Programm wird in der Softwarekonfiguration einer bestimmten Taskklasse zugewiesen.

Der Softwarekonfiguration zugewiesenen Programme werden als Tasks bezeichnet. Nur in der Softwarekonfiguration zugewiesene Programme werden nach dem Transfer von der Steuerung ausgeführt.

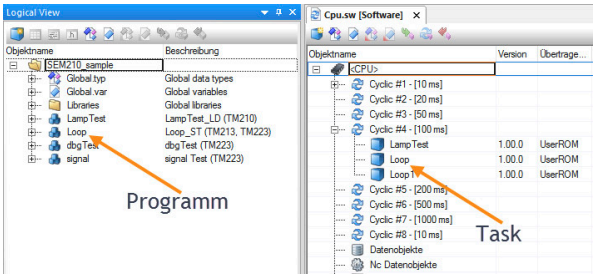


Abbildung 47: Programm und Task

Die Tasks werden konfigurierbaren Taskklassen zugeordnet und nacheinander ausgeführt. Ein Programm wird in jeder **Taskklasse (1 bis 8)** gleich "schnell" ausgeführt. Mit der Taskklasse ändert sich nur, in welchen zeitlichen Abständen das Programm ausgeführt wird. In Taskklasse 1 wird das Programm z. B. alle 10 ms ausgeführt, und damit öfter als ein Programm, das in Taskklasse 7 mit Zykluszeit 1000 ms läuft. Die Zeitpunkte für den Start der Tasks sowie der Zugriff auf das I/O System sind deterministisch.

5.3.1 Taskklasse und Zykluszeit

Ein Task wird zyklisch in der durch die Taskklasse vorgegebene Abarbeitungszeit, in weiterer Folge Zykluszeit genannt, abgearbeitet.

Um einen Task an seine Anforderungen optimal anpassen zu können, gibt es acht konfigurierbare Taskklassen. Jede Taskklasse beinhaltet Tasks mit gleicher Zykluszeit, Priorität und Toleranz.

In diesem Beispiel enthält die Taskklasse #4 drei Tasks. Es ist eine Zykluszeit von 100 Millisekunden konfiguriert.

In welche Taskklasse gehört das Programm?

Nicht alle Tasks müssen in der gleichen Taskklasse ablaufen. Steuerungsaufgaben, welche schnell abgearbeitet werden müssen, sind einer Taskklasse mit kleiner Zykluszeit zuzuordnen. Für langsamere Prozesse wird eine Taskklasse mit entsprechend höherer Zykluszeit ausgewählt.

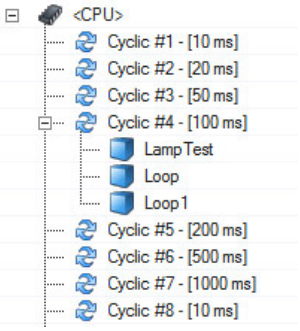


Abbildung 48: Taskklasse #4 mit drei Tasks

Ungeachtet der Ausführungsdauer der Tasks sieht der zeitliche Programmablauf so aus:

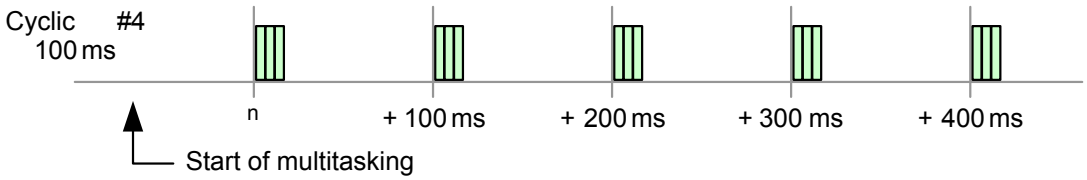


Abbildung 49: Tasks werden in jedem Zyklus erneut aufgerufen

Das bedeutet, dass alle 100 Millisekunden die Programme dieser Taskklasse ausgeführt werden. Voraussetzung dafür ist, dass die gesamte Abarbeitungszeit aller Tasks in dieser Taskklasse die Zykluszeit dieser Taskklasse nicht überschreiten.

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ Laufzeitverhalten - SG4 \ Start der zyklischen Tasks

5.3.2 Zykluszeit und Priorität

Die Priorität einer Taskklasse wird durch die Nummer der Taskklasse bestimmt.

Je kleiner die Nummer, desto höher ist die Priorität der Taskklasse.

Das Verschieben eines Tasks zwischen den Taskklassen verändert die Priorität und Zykluszeit dieses Tasks.

Die Taskklasse #8 hat im System die geringste Priorität, siehe "Taskklasse mit hoher Toleranz" auf Seite 46.

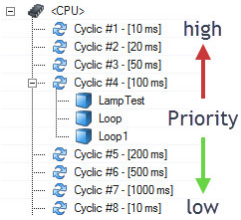


Abbildung 50: Taskklassen-Priorität

Wird der Task "Loop" von Taskklasse #4 in Taskklasse #1 verschoben, wird dieser Task alle 10 Millisekunden ausgeführt.

Die Laufzeit eines Tasks wird durch das Verschieben in eine andere Taskklasse nicht beeinflusst. Es ändert sich nur die Anzahl der Aufrufe im gleichen Zeitraum.

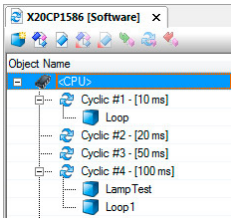


Abbildung 51: Unterschiedliche Taskklassen

Der Task "**Loop**" wird nun alle 10 ms ausgeführt. Die Tasks "**LampTest**" und "**Loop1**" in der Taskklasse #4 werden wie bisher alle 100 Millisekunden ausgeführt.

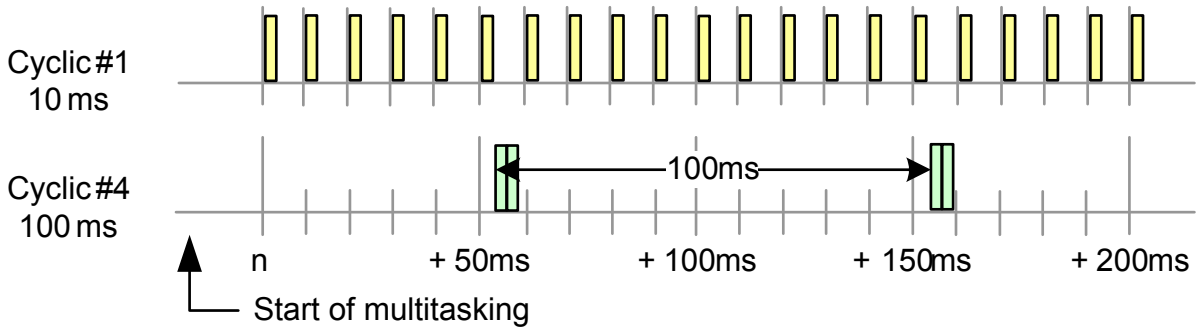


Abbildung 52: Ausführen von Tasks in verschiedenen Taskklassen

Aufgabe: Task "Loop" in die Taskklasse #1 verschieben

In der Softwarekonfiguration ist der Task "Loop" in die Taskklasse #1 zu verschieben.

Im Variablenmonitor sind die unterschiedliche Abarbeitungszeit der Task "Loop" und "Loop1", welche auf das gleiche Programm referenzieren, zu beobachten.

1) Softwarekonfiguration öffnen
2) Task "Loop" von Taskklassse #4 in Taskklassse #1 schieben.
3) Variablenmonitor für beide Tasks öffnen
4) Wert der Variable "udValue" beobachten

Das Programm "Loop" wird 10 mal öfter ausgeführt als das Programm "Loop1". Dadurch wird auch der Wert der Variable "udValue" um diesen Faktor öfter erhöht.

Loop::.pvm [Watch]

Name	Type	Scope	Value
udCnt	UDINT	local	2
udEndValue	UDINT	local	1
udStartValue	UDINT	local	0
udValue	UDINT	local	288
gMain	UDINT	global	158

Loop1::.pvm [Watch]

Name	Type	Scope	Value
udCnt	UDINT	local	2
udEndValue	UDINT	local	1
udStartValue	UDINT	local	0
udValue	UDINT	local	28
gMain	UDINT	global	158

Abbildung 53: Abarbeitungszeit in unterschiedlichen Taskklassen

Beispiel: Priorität Task CNC-Maschine

Im Programm einer CNC-Maschine ist es wichtig, den genauen Standort des Bohrers zu kennen. Die schnelle und genaue Bohrung - und somit der Position des Bohrers - hat eine sehr hohe Priorität. Die Positionsabfrage des Bohrers muss sehr oft passieren, damit die "Lücken der positionslosigkeit" so gering wie möglich sind.
Wäre diese Abfrage in einer hohen Taskklasse, würde in der Zwischenzeit keine aktuelle Position des Bohrers verfügbar sein, da das Programm nach Beendigung erst nach Ablauf der Zykluszeit wieder aufgerufen wird.

Dieser Task wäre in Taskklasse #1 angelegt.

5.3.3 Restzeit

Im laufenden Betrieb werden von Automation Runtime, neben der Abarbeitung der Tasks, **andere Systemaufgaben** durchgeführt. Diese stellen dem Anwender nützliche Funktionen zur Verfügung. Es wird beispielsweise eine ständige Checksummenprüfung für Softwaremodule durchgeführt und die Online Kommunikation bereitgestellt.
Je nach Prozessorleistung können diese Aufgaben mehr oder weniger schnell ausgeführt werden.

Die Zeit, in der Automation Runtime keine zyklischen Aufgaben ausführt, wird als **Restzeit (Idle)** bezeichnet.

Folgende Aufgaben werden in der Restzeit ausgeführt:

- Online Kommunikation
- Visual Components Visualisierung
- Zugriffe auf das Dateisystem

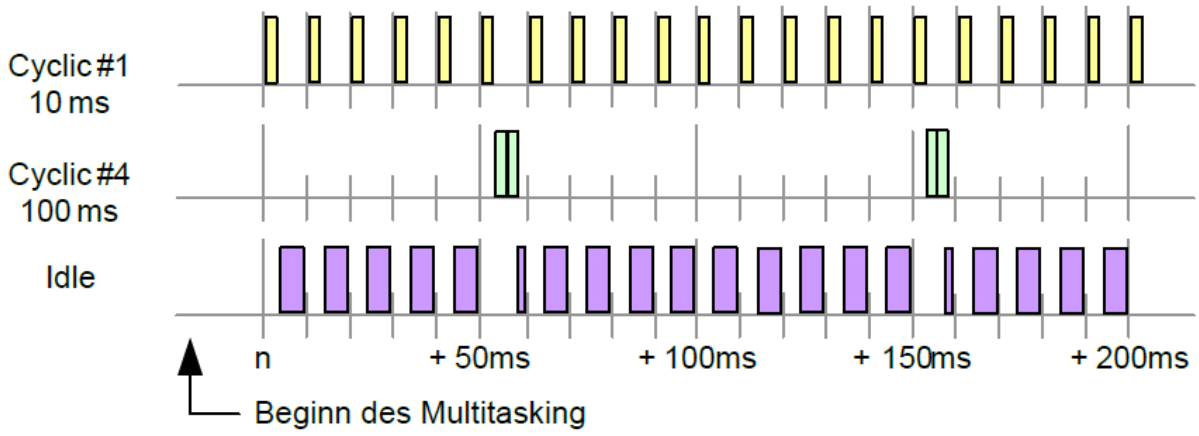


Abbildung 54: Restzeit (Idle) im zyklischen System

In der CPU Konfiguration wird die Basistaskklasse für die Restzeit festgelegt. Im Kontext dieser Taskklasse wird die konfigurierte Restzeit vom System bereitgestellt.

Die Restzeit kann mit Hilfe des **Profilers** ermittelt werden. Zudem zeigt der System Diagnostics Manager die durchschnittliche Systemauslastung an.

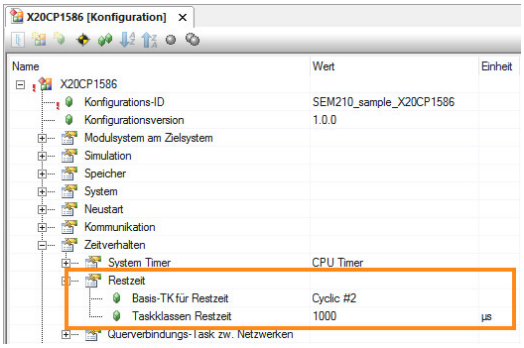


Abbildung 55: CPU Konfiguration: Basistaskklasse für die Restzeit

Was ist zu tun, wenn die Restzeit nicht ausreicht?

Wenn die Restzeit nicht ausreicht, kann durch das Verschieben von Programmen in höhere Taskklassen oder durch die Anpassung der Zykluszeit die Restzeit erhöht werden.

Echtzeit Betriebssystem \ Arbeitsweise \ Laufzeitverhalten \ Scheduling \ Restzeit

5.3.4 Start der Taskklassen

Nach dem Start des Multitasking Systems werden nicht alle Taskklassen gleichzeitig gestartet.

Der Startzeitpunkt ist immer die halbe Taskklassenzykluszeit.

Dies bedeutet, dass beispielsweise die 100ms Taskklasse nach 50ms gestartet wird. Durch die Verteilung des Startzeitpunktes aller Taskklassen kann die Prozessorleistung besser genützt und beispielsweise ein minimaler Output Jitter erreicht werden.

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ Laufzeitverhalten - SG4 \ Start der zyklischen Tasks

5.3.5 Task unterbricht Task

Durch die Priorität einer Taskklasse ist es möglich, dass ein Task mit hoher Priorität einen länger andauernden Task mit niedriger Priorität unterbricht.

Mit Hilfe des Variablenmonitors kann in den Tasks "Loop" und "Loop1" der Endwert der Variable "**udEndValue**" so verändert werden, dass der Task "**Loop1**" genau **zweimal** durch den Task "**Loop**" unterbrochen wird.

Die schematische Darstellung zeigt, wie das Zeitverhalten in diesem Fall aussehen kann. Für die Vollständigkeit und zum besseren Verständnis wurde die Abbildung um das Zeitverhalten der Taskklasse #3 ergänzt.

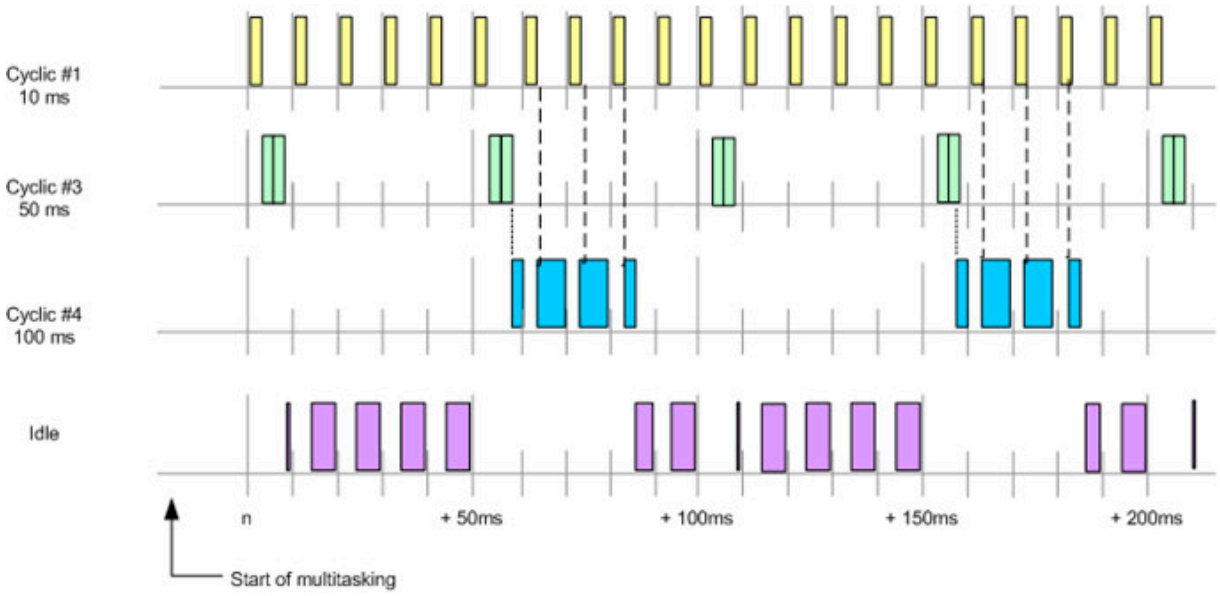


Abbildung 56: Multitasking mit Taskklasse #1, #3 und #4; Taskklasse #4 wird von Taskklasse #1 unterbrochen

Für die Taskklasse #4 steht über die gesamte Abarbeitungszeit ein konsistentes I/O Eingangsabbild zur Verfügung. Daher werden die Tasks in dieser Taskklasse nicht durch die Unterbrechung beeinflusst.

5.4 Zykluszeit und Toleranz

Jede Taskklasse hat eine definierte Zykluszeit. In dieser Zeit müssen alle Tasks der Taskklasse abgearbeitet werden.

Die Zykluszeiten der Taskklassen sind bei der Erstellung eines Projekts bereits festgelegt. Der Benutzer kann die **Zykluszeit** jeder Taskklasse **individuell** verändern.

Wenn die gesamte Laufzeit aller Tasks einer Taskklasse die Zykluszeit dieser Taskklasse überschreitet, kommt es zu einer Zykluszeitverletzung. Eine konfigurierbare Toleranz zögert die Zykluszeitverletzung hinaus. Ein Neustart wird durchgeführt.

Die Zykluszeit und die Toleranz wird in den Eigenschaften der Taskklasse konfiguriert. Der Eigenschaftsdialog wird über das Kontextmenü der Taskklasse geöffnet.

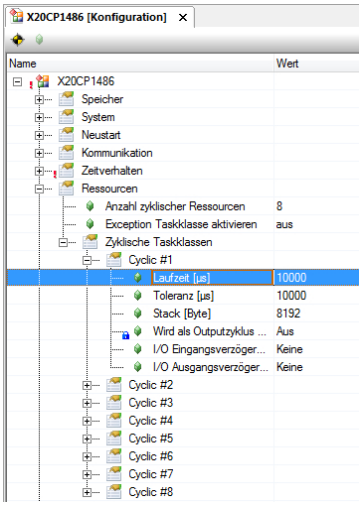


Abbildung 57: Konfiguration der Zykluszeit und Toleranz

Wird die konfigurierte Zykluszeit einer Taskklasse überschritten, verschiebt sich der Start des nächsten Zyklus um die eingestellte Toleranz nach hinten. Dies kann zu Problemen im Zeitverhalten der Applikation führen. Durch Setzen der Toleranz auf den Wert "0" wird dieses Verhalten vermieden.

Programmierung \ Editoren \ Konfigurationseditoren \ Hardwarekonfiguration \ CPU Konfiguration

Aufgabe: Kontrolle der Systemauslastung durch Programm "Loop"

Im Task "Loop", welcher in der Taskklasse #1 läuft, ist durch Verändern des Wertes der Variable "udEndValue" eine erhöhte Laufzeit zu verursachen.

Mit Hilfe des System Diagnostics Manager wird die durch die Tasks verursachte Systemauslastung und die zur Verfügung stehende Restzeit überwacht.

1) Der Variable "udEndValue" den Wert 50000 geben
2) Systemauslastung mit Hilfe des System Diagnostics Manager ermitteln
3) Den Wert der Variable "udEndValue" stufenweise erhöhen

Zwischen den jeweiligen Schritten sind die Ergebnisse im System Diagnostics Manager zu analysieren.

Im System Diagnostics Manager wird die mittlere Systemauslastung über einen auswählbaren Zeitraum visualisiert. In der Grafik kann der Mittelwert und das Maximum abgelesen werden.

Details zum Laufzeitverhalten eines bestimmten Task können nur mit dem Profiler ermittelt werden. Dazu sind im Trainingsmodul "TM223 - Automation Studio Diagnose" weitere Übungen vorgesehen.

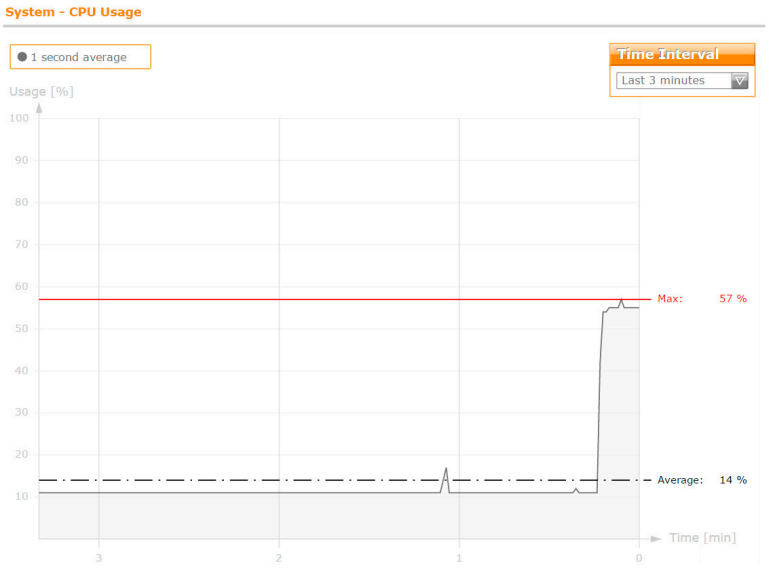


Abbildung 58: Darstellung der CPU Auslastung im System Diagnostics Manager

Diagnose und Service \ Diagnosewerkzeug

- System Diagnostics Manager (SDM)
- Profiler

5.4.1 Zykluszeitverletzung

Automation Runtime überwacht bei der Ausführung der Taskklassen das Zeitverhalten. Überschreitet die Laufzeit der Tasks die konfigurierte Zykluszeit, so wird eine Zykluszeitverletzung ausgelöst. Wird eine Zykluszeitverletzung erkannt, startet das Zielsystem im Betriebsmodus "SERVICE".

Abbildung 59: CPU im Betriebsmodus "SERVICE"

In diesem Beispiel wird die Zykluszeitverletzung durch Wertänderungen im Variablenmonitor erreicht. Im Variablenmonitor werden beim Auftreten der Zykluszeitverletzung alle Werte „eingefroren“ und nach dem Neustart im Modus "SERVICE" mit dem Wert „0“ dargestellt.

Der Logger hilft bei der Ursachenanalyse, wenn das Zielsystem im Betriebsmodus "SERVICE" hochläuft.

Der Logger wird über das Hauptmenü "Öffnen" \ "Logger" oder die Tastenkombination **<STRG> + <I>** geöffnet.

In der Abbildung wurde als Fehlerursache eine Zykluszeitverletzung in der Taskklasse #1 eingetragen.

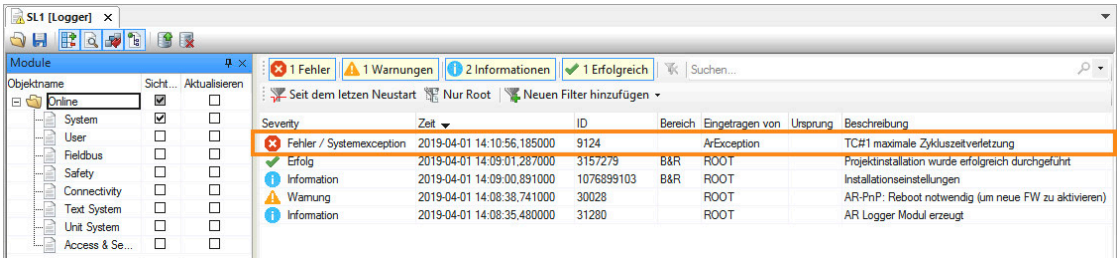


Abbildung 60: Fehlerursache im Logger analysieren: Zykluszeitverletzung in Taskklasse #1

Alternativ zum Logger hilft bei der Ursachenanalyse der Profiler.

Weiterführende Informationen:

- TM223 – Automation Studio Diagnose

Diagnose und Service \ Diagnosewerkzeug \ Logger

Aufgabe: Variable „udEndValue“ bis zum Erreichen einer Zykluszeitverletzung erhöhen

Ziel dieser Aufgabe ist es, die Variable "udEndValue" im Programm "Loop" schrittweise zu erhöhen, bis eine Zykluszeitverletzung auftritt. Am Zielsystem wird ein Neustart durchgeführt. Die Steuerung startet im Betriebsmodus "SERVICE". Anschließend ist eine Analyse mittels Logger durchzuführen.

1) Den Wert der Variable "udEndValue" stufenweise, bis zur Zykluszeitverletzung, erhöhen
2) Logger-Eintrag der Zykluszeitverletzung analysieren

5.4.2 Auf Zykluszeitverletzung im Anwenderprogramm reagieren

Im Produktivsystem ist der Wechsel in den Modus "SERVICE" ohne Reaktionsmöglichkeit nicht erwünscht. Auf Ereignisse wie eine Zykluszeitverletzung kann in einem Exception Programm reagiert werden.

Die Exception Taskklasse wird in der CPU Konfiguration in der Kategorie Ressourcen aktiviert.

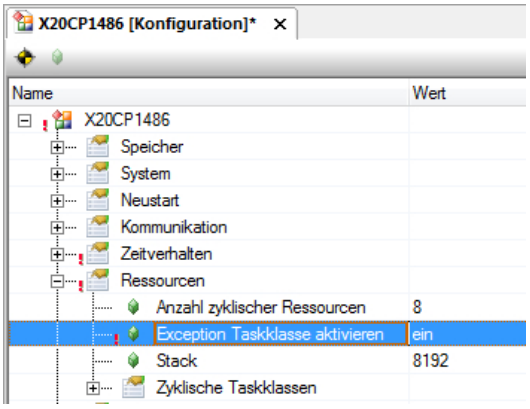


Abbildung 61: Aktivieren der Exception Taskklasse

Ein Programm wird aus Logical View in der Software Konfiguration in die Exception Taskklasse eingefügt.

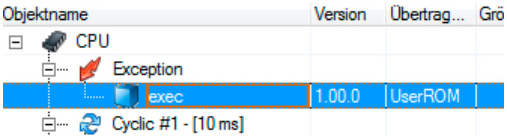


Abbildung 62: Programm in Exception Taskklasse zuordnen

In den Eigenschaften des Exception Tasks wird eine **Exception-Nummer** eingegeben. Die Exception-Nummer ist vom System vorgegeben und kann in der Automation Help nachgelesen werden.

Tritt zur Laufzeit eine Zykluszeitverletzung auf (Exception-Nr. 145), wird dieser Task aufgerufen. Im Exception-Task kann je nach Anforderung auf die Zykluszeitverletzung reagiert werden.

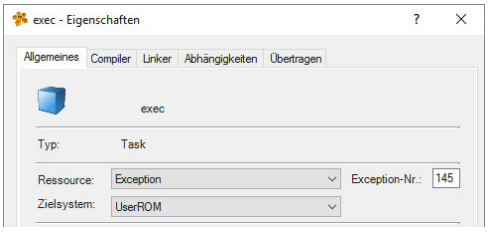


Abbildung 63: Konfiguration des Exception Tasks

Programmierung \ Editoren \ Konfigurationseditoren \ Hardwarekonfiguration \ CPU Konfiguration \ SGx \ Eigenschaften der CPU - Resources

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ Laufzeitverhalten - SG4 \ Exception Taskklasse - SG4

5.4.3 Taskklasse mit hoher Toleranz

Aufgaben, welche zwar so schnell wie möglich, jedoch mit einer niedrigen Priorität abgearbeitet werden, sind in der Taskklasse #8 auszuführen.

Diese wird wenn möglich alle 10 ms aufgerufen, kann aber durch jede höher priore Taskklasse unterbrochen werden. Durch die konfigurierte Toleranz ist festgelegt, dass die Taskklasse #8 mindestens ein mal alle 30 Sekunden fertig abgearbeitet werden muss.

Aufgaben, welche in Taskklasse #8 ausgeführt werden sollten:

- Dateizugriff vom Anwenderprogramm
- Komplexe, nicht zeitkritische Berechnungen

5.5 Einstellungen beim Übertragen von Programmen

Online-Installation

Programme und Konfigurationen werden nach einer Änderung auf das Zielsystem übertragen. Bei der Änderung einer Systemkonfiguration ist ein Neustart erforderlich.

Bei der Übertragung von Programmänderungen muss unterschieden werden, ob dies zur Entwicklungszeit oder am Produktivsystem geschieht.

Je nach Anwendungsfall können die **Transfereinstellungen** bei der Projektinstallation angepasst werden. In der Abbildung werden die Standardwerte bei der Projektinstallation angezeigt.

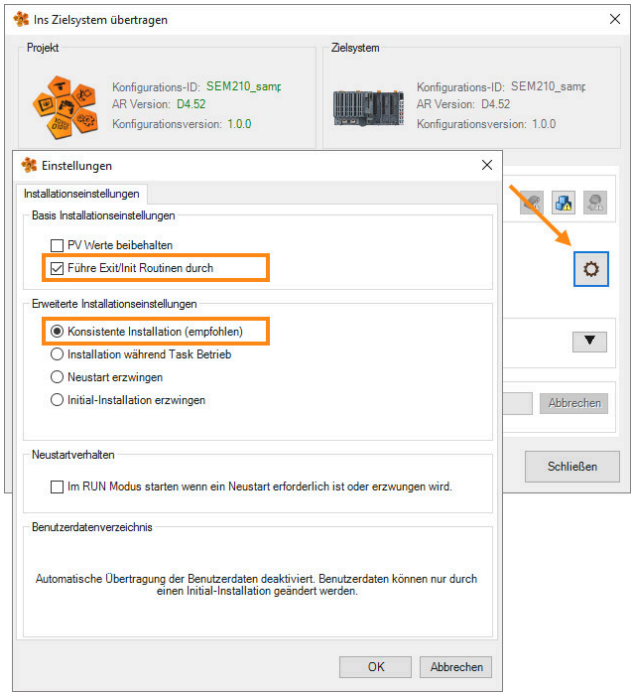


Abbildung 64: Konfiguration der Transfereinstellungen bei der Projektinstallation

PV Werte beibehalten

Diese Einstellung definiert, ob für bestehende und durch den Transfer aktualisierte Tasks, die Werte der Prozessvariablen über den Installationsvorgang hinweg erhalten bleiben.

Exit- / Init-Programme ausführen

Wurden im Init-Programm oder im zyklischen Programm dieses Tasks Ressourcen (Speicher, Schnittstellen) angefordert, so können diese Ressourcen mit Hilfe des Exit-Programms wieder korrekt freigegeben werden.

Konsistente Installation

Während der Installation werden alle Taskklassen angehalten. Damit ist die Konsistenz aller am Zielsystem laufenden Tasks sichergestellt.

Installation während der Taskausführung

Während der Installation werden nur jene Tasks angehalten, welche installiert werden. Die andere Tasks werden, von der Installation unbeeinflusst, weiterhin zyklisch ausgeführt.

Neustart bzw. Initialtransfer erzwingen

Mit der Option "Neustart erzwingen" wird das Zielsystem nach dem Projekttransfer neu gestartet.
Mit der Option "Initialtransfer erzwingen" wird ein Initialtransfer durchgeführt.

Benutzerverzeichnis

Wenn eine Benutzerpartition erstellt wurde, können mit dieser Option die Benutzerdaten auf diese Partition kopiert werden.

Weiterführende Informationen:

- 3.2 "Konfigurations-ID und Partitionierung" auf Seite 16

Projekt Management \ Projektinstallation

- Szenarien \ Online-Update
- Projektinstallation durchführen \ Einstellungen \ Installationseinstellungen

Offline-Installation

Bei der Offline-Installation gibt es in den Installationseinstellungen die Möglichkeit, den nichtflüchtigen Speicher zu erhalten. Das bedeutet, dass die Werte von Retain Variablen und permanenten Variablen erhalten bleiben, unabhängig davon, ob die CPU einen Warm- oder Kaltstart ausführt.

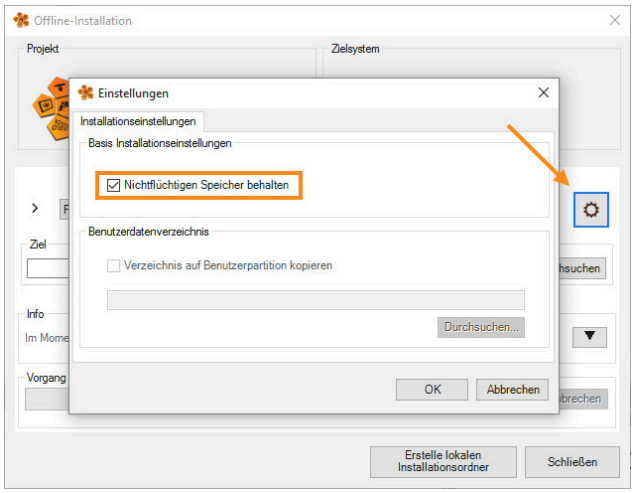


Abbildung 65: Installationseinstellung der Offline-Installation

Aufgabe: Transfereinstellungen anpassen

Ziel dieser Übung ist es, die Auswirkungen der Transfer-Einstellungen zu erfahren.
Im Task "Loop", im Variablenmonitor, wird der Wert von "udEndValue" auf 1000 gesetzt.
Im Anschluss wird im Quelltext von "Loop" etwas geändert; dies kann ein Kommentar sein. Die Änderungen werden kompiliert und das Übertragungsfenster geöffnet. Mit geänderten Transfereinstellungen bleibt der zuvor im Variablenmonitor eingestellte Wert für "udEndValue" erhalten.

1) Den Wert der Variable "udEndValue" auf 1000 setzen
2) Quelltext im Programm "Loop" ändern
3) Kompilieren und Übertragungsfenster öffnen

4) Transfereinstellungen anpassen

5) Kontrolle, ob Wert von "udEndValue" erhalten bleibt

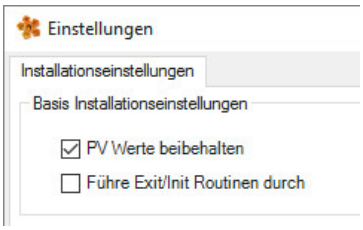


Abbildung 66: Einstellungen zum Erhalten der Werte von Prozessvariablen; Ausführung von Init- /Exit-Programmen wird verhindert

6 I/O Behandlung

Eine Hauptfunktion der Steuerung ist der schnelle und deterministische Transport von Informationen. Die Ein- und Ausgangszustände der I/O Klemmen kommunizieren ständig mit der Applikation der Steuerung.

Die I/O Verwaltung des Automation Runtime ist so ausgelegt, dass höchste Ansprüche an **Reaktionszeiten** und **Konfigurierbarkeit** erfüllt werden.

Der I/O Scheduler überträgt das Eingangsabbild von der I/O Klemme vor dem Start der Taskklasse und das Ausgangsabbild am Ende der Tasks einer Taskklasse.

Die Abbildung zeigt den grundlegenden Ablauf der I/O Behandlung in Automation Runtime. Der I/O Scheduler stellt der Taskklasse ein konsistentes Eingangsabbild zur Verfügung. Dies wird über die grünen Pfeile symbolisiert. Mit Ende des letzten Tasks in der Taskklasse werden die Ausgangsdaten dem I/O Scheduler übergeben. Dies ist durch die roten Pfeile angedeutet.

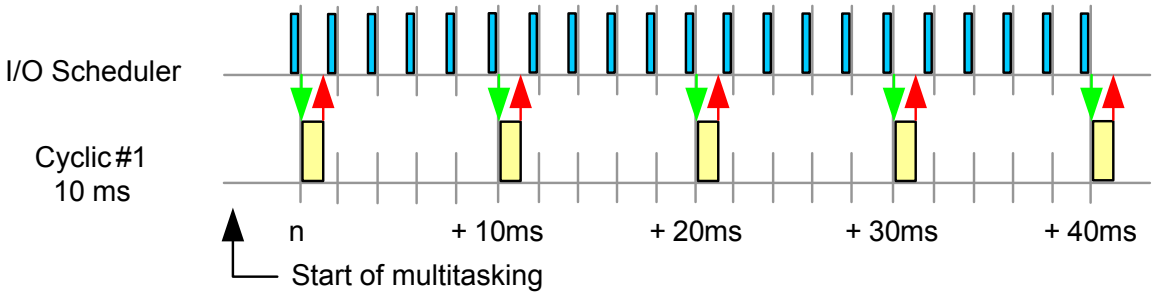


Abbildung 67: I/O Scheduler: Eingangsabbild bereitstellen; Schreiben des Ausgangsabbildes am Ende der Laufzeit der Tasks

In der I/O Zuordnung werden Prozessvariablen mit Kanälen von I/O Modulen verbunden. Der I/O Scheduler wird durch die I/O Zuordnung gesteuert.

Standardmäßig ist die Taskklasse #1 mit 10 Millisekunden konfiguriert.

Die Standardkonfiguration für die POWERLINK und X2X Schnittstelle ist eine Zykluszeit von 2 Millisekunden.

Die Zykluszeit der Taskklasse kann an die I/O Zykluszeit angepasst werden. Diese Einstellung ist in der CPU Konfiguration zu finden

Programmierung \ Editoren \ Konfigurationseditoren \ Hardwarekonfiguration \ CPU Konfiguration

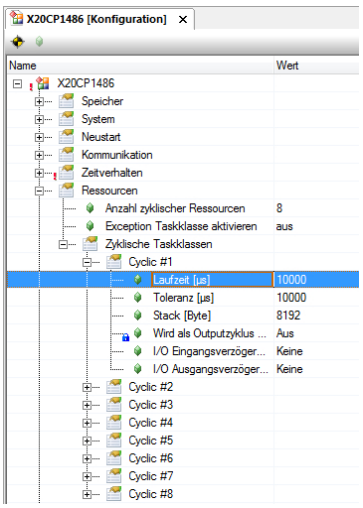


Abbildung 68: Konfiguration der Zykluszeit für Taskklasse #1

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ I/O Management - SG4

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ I/O Management \ Arbeitsweise

6.1 I/O Konfiguration und I/O Zuordnung

6.1.1 I/O Konfiguration

In der I/O Konfiguration werden modulspezifische Eigenschaften konfiguriert.
Durch die angebotene Funktionalität der I/O Module ergeben sich viele Einsatzmöglichkeiten und Betriebsarten, bei denen die Module eingesetzt werden können. In der I/O Konfiguration werden unter anderem das Verhalten für die Modulüberwachung, Einzelkanalkonfigurationen und Funktionsmodelle eingestellt.

Die I/O Konfiguration wird über das Kontextmenü des gewünschten I/O Modul geöffnet. Die Konfigurationsmöglichkeiten eines I/O Moduls sind im jeweiligen Datenblatt im Abschnitt "Registerbeschreibung" dokumentiert.

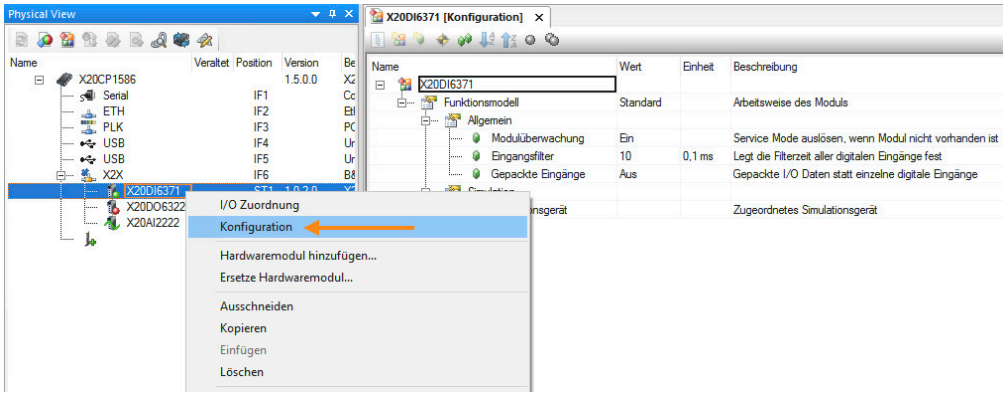


Abbildung 69: I/O Konfiguration eines digitalen Eingangsmodul

Programmierung \ Editoren \ Konfigurationseditoren \ Hardwarekonfiguration \ I/O Konfiguration

6.1.2 I/O Zuordnung

Die I/O Zuordnung legt fest, welche Daten aus dem I/O Abbild einer Prozessvariable zugewiesen werden. Jede Variable in einer .var Datei, welche in einem Programm verwendet wird, kann unabhängig vom Gültigkeitsbereich einem Kanal eines I/O Moduls zugeordnet werden.
Die I/O Zuordnung wird über das Kontextmenü eines I/O Moduls geöffnet.

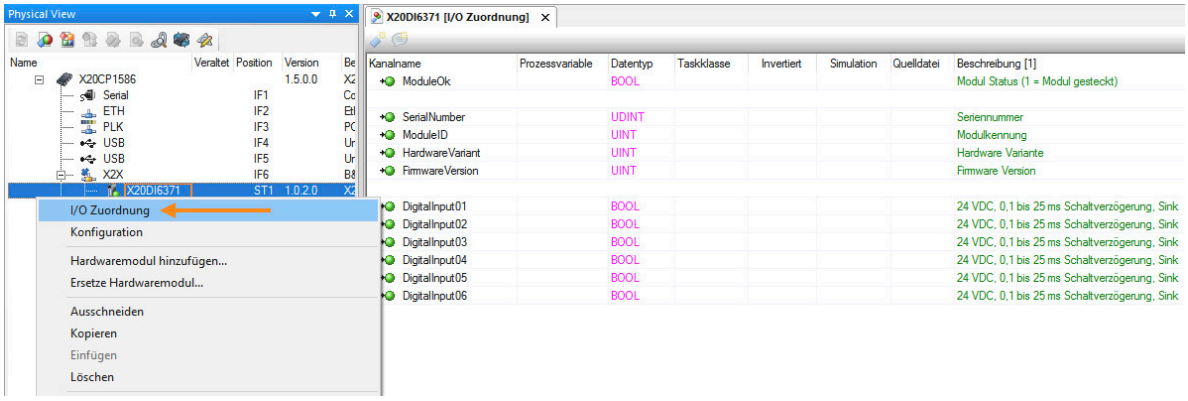


Abbildung 70: I/O Zuordnung einer digitalen Eingangskarte

Welche Taskklasse wird bei der I/O Zuordnung benötigt?

Bei globalen Variablen kann dem Kanal eine Taskklasse zugewiesen werden, in welcher die Datenübertragung des I/O Abbildes erfolgen soll. Bei der Einstellung "Automatic" wird beim Kompilieren des Projektes automatisch die schnellste Taskklasse ermittelt, in welcher die Variable verwendet wird.

Programmierung \ Editoren \ Konfigurationseditoren \ I/O Zuordnung

6.2 Behandlung von I/O Abbildern

Der I/O Scheduler stellt ein konsistentes I/O Abbild für die Abarbeitung aller Tasks einer Taskklasse zur Verfügung und startet die Taskklasse exakt zum konfigurierten Zeitpunkt. Damit ist sichergestellt, dass am Beginn der Taskklasse alle Eingänge, und am Ende der Tasks einer Taskklasse alle Ausgänge übertragen werden.

Nach welcher Taskklasse richtet sich der I/O Scheduler?

Jede Taskklasse arbeitet mit einem eigenen I/O Abbild. Während der gesamten Task-Laufzeit bleiben die Eingangszustände konsistent.

Zyklus der I/O Daten

Die Bereitstellung der I/O Daten erfolgt in der eingestellten I/O Zykluszeit des verwendeten Bussystems. Die Konfiguration wird über das Kontextmenü des jeweiligen Feldbus geöffnet.

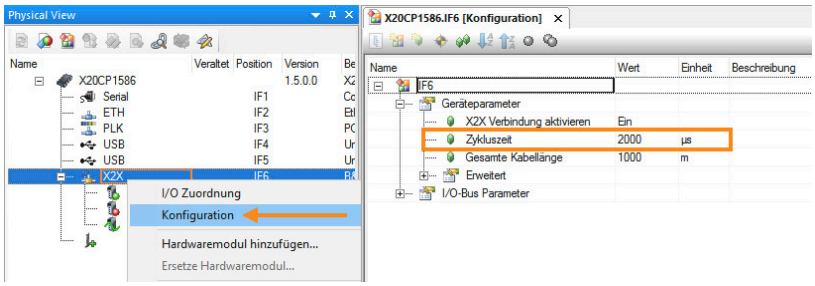


Abbildung 71: Eigenschaften der X2X Link Schnittstelle Öffnen

Die in den Eigenschaften konfigurierte I/O Zykluszeit ist die Basis für das Kopieren der I/O Daten in das I/O Abbild. Dies bedeutet, dass in der eingestellten X2X Zykluszeit die Eingänge als I/O Abbild bereitgestellt und das Ausgangsabbild geschrieben werden.

Ist die Zykluszeit der Schnittstelle ausreichend?

Ob die eingestellte X2X bzw. POWERLINK Zykluszeit für die Anzahl der konfigurierten I/O Module zulässig ist, kann mit dem Network Analyzer festgestellt werden. Dieser wird über das Hauptmenü "Öffnen" \ "Network Analyzer" geöffnet.

Diagnose und Service \ Diagnosewerkzeug \ Network Analyzer

Die Grafik zeigt, dass alle 2 Millisekunden ein Eingangsabbild bereitgestellt wird. Für die Anwendung bedeutet dies, dass die Eingangsdaten ab dem Start der Taskklasse nicht älter als die eingestellte Zykluszeit sind, und die Ausgänge spätestens nach dieser Zeit geschrieben werden.

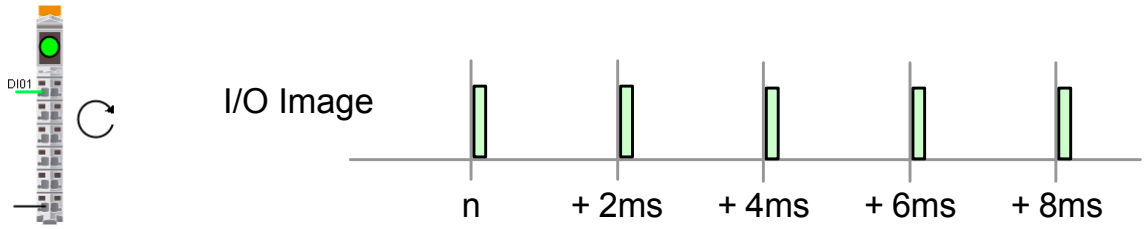


Abbildung 72: Lesen des Eingangsabbildes

I/O Daten in der Taskklasse

Der I/O Scheduler steuert den Zeitpunkt, zu dem das I/O Abbild der Taskklasse bereitgestellt und die Taskklasse gestartet wird.

Standardmäßig wird der I/O Scheduler mit dem System Tick von 1000 µs aufgerufen.

Der System Tick wird in der CPU Konfiguration im Bereich Zeitverhalten eingestellt.

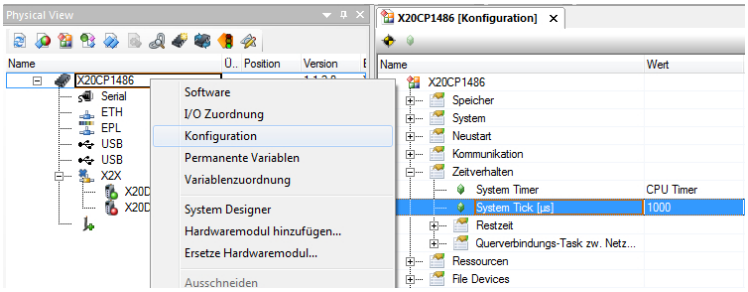


Abbildung 73: Konfigurierbarer System Tick

Die Grafik zeigt, dass der I/O Scheduler doppelt so oft wie das Abbild aktualisiert wird, aufgerufen wird.

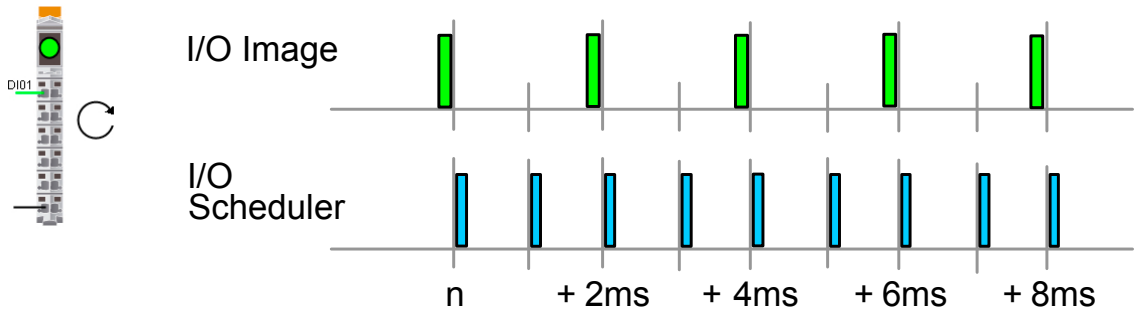


Abbildung 74: Lesen des Eingangsabbildes; Aufruf des I/O Schedulers

In der Konfiguration für das Zeitverhalten kann die Zykluszeit des Feldbus als System Timer verwendet werden. Als Standard wird die Feldbuszykluszeit im Verhältnis 1:1 für den I/O Scheduler übernommen. I/O Zyklus und I/O Scheduler arbeiten nun synchron im eingestellten Verhältnis.

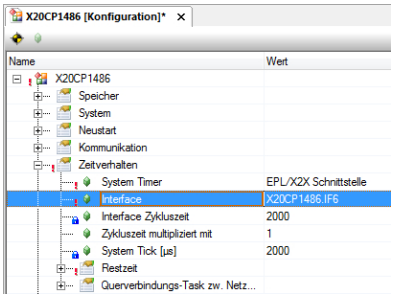


Abbildung 75: Synchronisierung des System Ticks

Durch die Verwendung des Feldbus Timers als System Timer und die Multiplikation der Zykluszeit mit dem Faktor 1, wird der I/O Scheduler mit der konfigurierten I/O Zykluszeit aufgerufen.

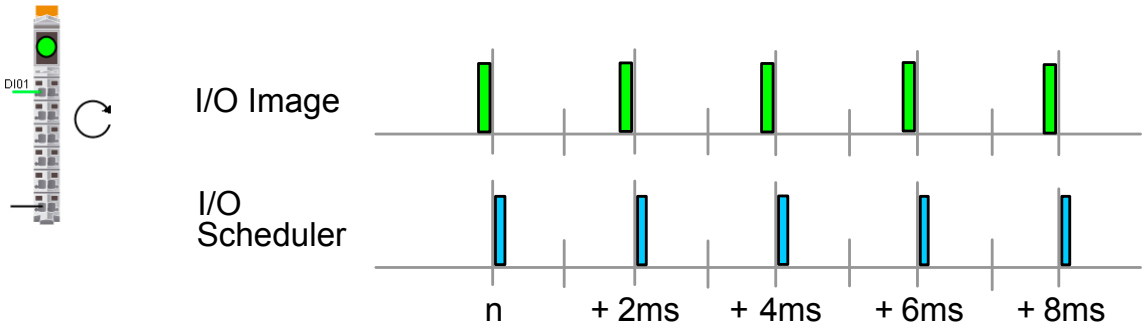


Abbildung 76: Synchronisierung von I/O Abbild und I/O Scheduler

So wie in der I/O Zuordnung festgelegt, werden die I/O Daten aus dem Eingangsabbild an die konfigurierten Prozessvariablen zugewiesen.

Für die Tasks einer Taskklasse bedeutet dies, beim Aufruf des I/O Schedulers mit einer Zykluszeit von 2 Millisekunden folgendes:

Taskklasse #1

Der I/O Scheduler startet alle 10 ms die Taskklasse #1 und stellt dieser das Eingangsabbild für die zugeordneten Variablen (grüner Pfeil) zur Verfügung.

Am Ende der Tasks von Taskklasse #1 werden die Variablen der zugeordneten Ausgänge in das Ausgangsabbild geschrieben (roter Pfeil).

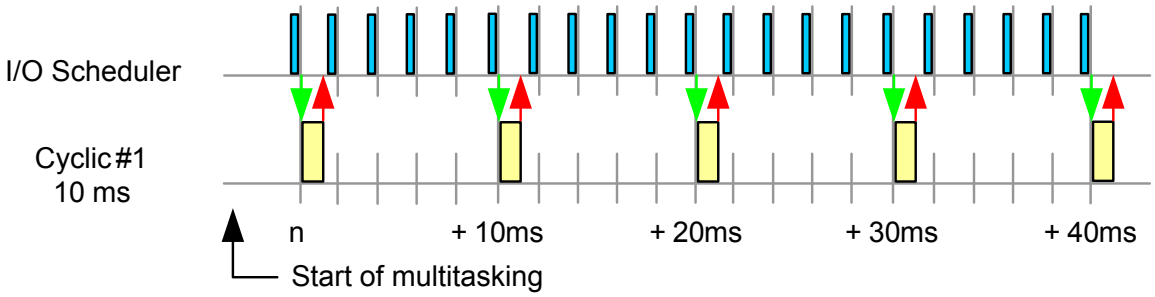


Abbildung 77: I/O Behandlung für Taskklasse #1

Taskklasse #4

Der I/O Scheduler startet zum Zeitpunkt , n+50 ms, n+150 ms die Taskklasse #4 und stellt dieser das Eingangsabbild für die zugeordneten Variablen (grüner Pfeil) zur Verfügung.

Am Ende der Tasks von Taskklasse #4 werden die Variablen der zugeordneten Ausgänge in das Ausgangsabbild geschrieben (roter Pfeil).

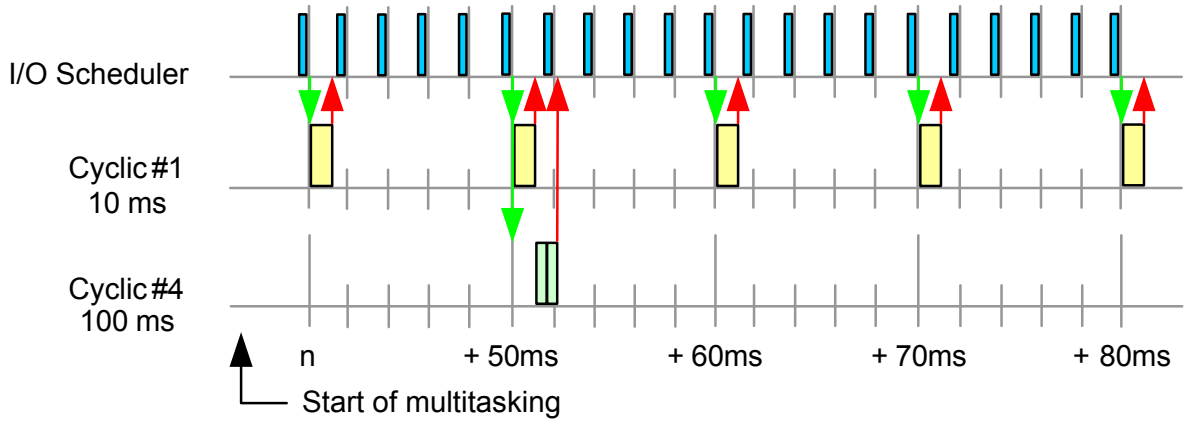


Abbildung 78: I/O Behandlung für Taskklasse #4

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ I/O Management - SG4

6.2.1 Hochlauf

Beim Hochlauf der Steuerung wird die I/O Konfiguration auf die I/O Module übertragen, Schnittstellen und Feldbusgeräte werden initialisiert. Dies geschieht noch vor der Initialisierung der Programme. Damit ist sichergestellt, dass bereits im Init-Programm auf gültige I/O Daten zugegriffen werden kann.

6.2.2 Abbildung von I/O Daten

Die nachfolgende Abbildung zeigt den Zusammenhang zwischen der I/O Konfiguration und der I/O Zuordnung.

Die I/O Konfiguration wird beim Hochlauf der Steuerung oder beim Erkennen eines konfigurierten I/O Moduls auf das Modul übertragen.

Im zyklischen Betrieb werden die Eingangsdaten als Speicherblock bereitgestellt und durch die I/O Zuordnung auf die konfigurierten Prozessvariablen verteilt. Zum Schreiben der Ausgangsdaten werden die Prozessvariablen gesammelt und als Speicherblock dem I/O System übergeben.

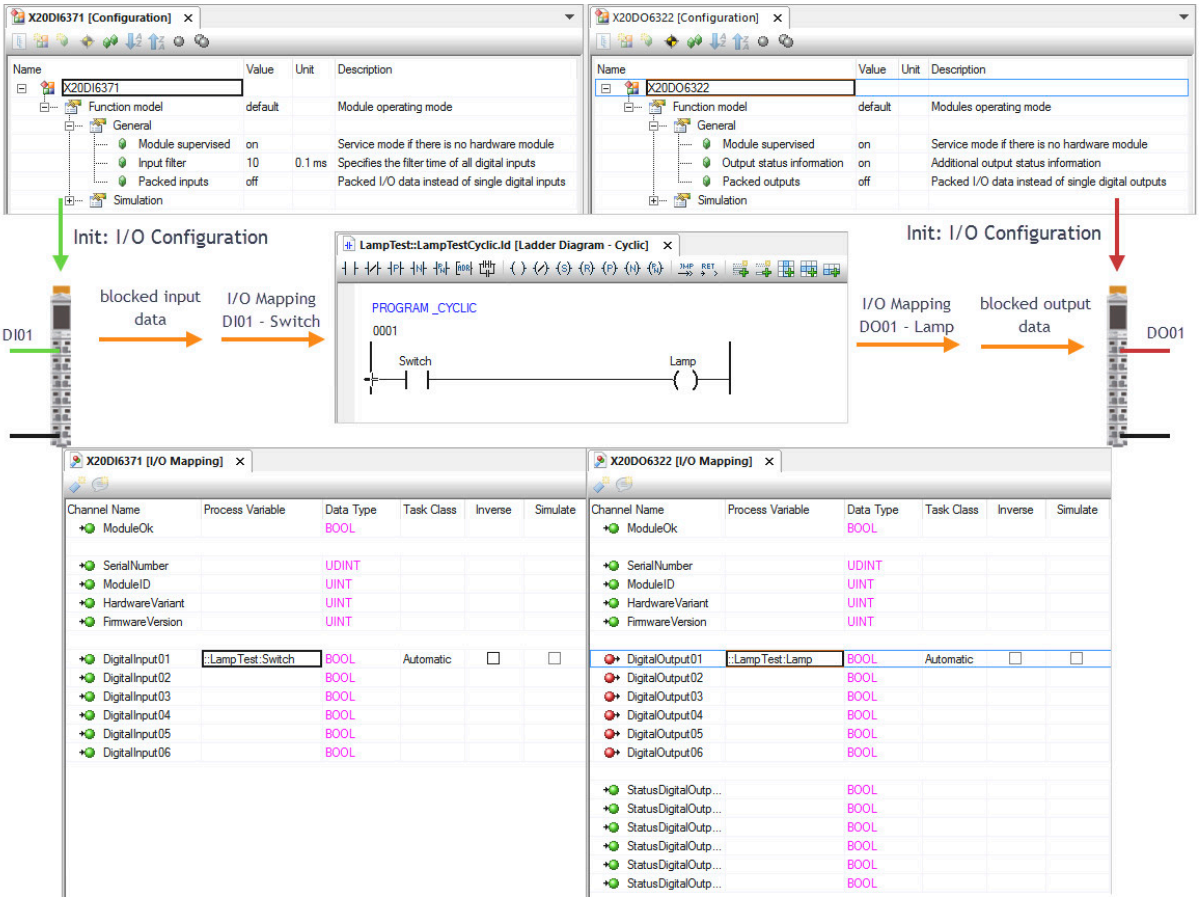


Abbildung 79: Grundlegende Funktion der I/O Behandlung

6.2.3 Einstellungen für das Zeitverhalten von I/O Abbildern

Für die Taskklassen stehen Konfigurationsmöglichkeiten für die zeitliche Behandlung von I/O Abbildern zur Verfügung. Der Standardfall ist das unmittelbare Schreiben der Ausgänge beim Beenden des letzten Task in der Taskklasse. Dies führt dazu, dass bei schwankender Tasklaufzeit ein Jitter bei den Ausgängen resultiert.

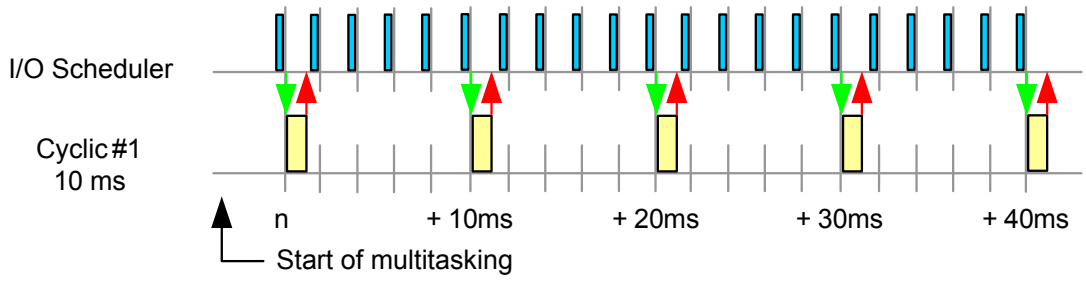


Abbildung 80: Ausgänge werden unmittelbar nach Taskende dem I/O System übergeben

Alternativ kann über die CPU Einstellungen für die Taskklasse der Zeitpunkt für das Einlesen der Eingangsabbilder und das Schreiben der Ausgangsabbilder verzögert werden.

Wird für die Taskklasse #1 der Parameter "I/O Ausgangsverzögerung" auf den Wert "Auf Ende des Zyklus" konfiguriert. So werden die Ausgangsdaten erst am Ende der Taskklasse dem I/O Scheduler übergeben. Dies bewirkt, dass selbst bei schwankender Tasklaufzeit die Ausgänge einer Taskklasse jitterfrei geschrieben werden.

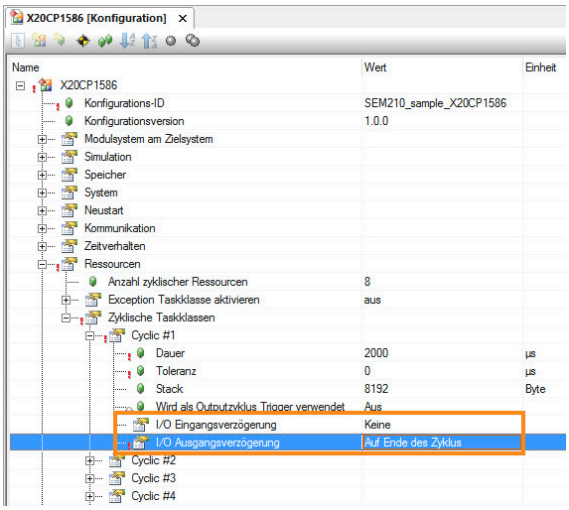


Abbildung 81: Konfiguration für die Verzögerung der Ausgangsdaten an das Ende des Zyklus

Die Grafik zeigt, wie sich die Verzögerung des Ausgangsabbildes an das Ende des Zyklus auswirkt.

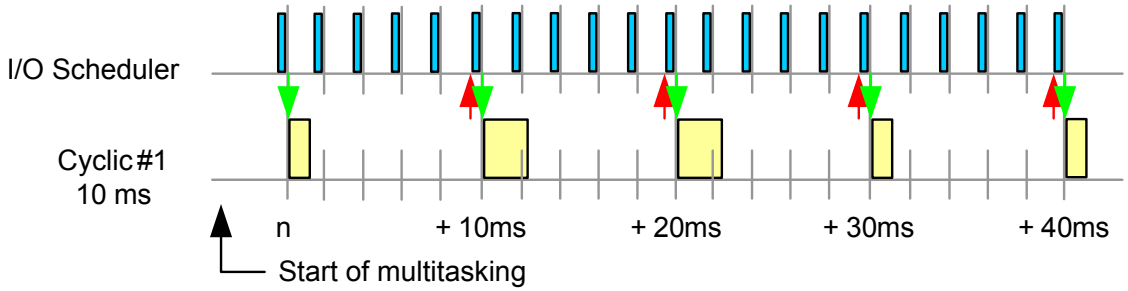


Abbildung 82: Jitterfreies Schreiben des Ausgangsabbildes am Ende des Zyklus

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ I/O Management - SG4

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ I/O Management - SG4 \ Arbeitsweise

6.3 Fehlerbehandlung bei I/O Modulen

Jedes konfigurierte I/O Modul wird vom I/O System überwacht. Der Anwender hat die Möglichkeit, zu konfigurieren, wie das System auf Fehlersituationen reagieren soll.

Die Überwachung des I/O Moduls wird in der I/O Konfiguration über die Eigenschaft "**Modulüberwachung**" aktiviert oder deaktiviert.

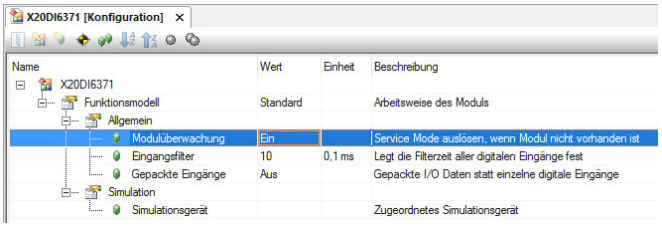


Abbildung 83: Konfiguration der Modulüberwachung

Modulüberwachung aktiviert

Wenn die Modulüberwachung aktiviert ist, führen die nachfolgenden Zustände zu einem Neustart im Betriebsmodus "SERVICE":

- Das auf einem Steckplatz definierte Modul ist nicht vorhanden oder nicht gesteckt.
- Das auf dem Steckplatz physikalisch gesteckte Modul stimmt nicht mit dem für diesen Steckplatz konfigurierte Modul überein.
- Das Modul kann im laufenden Betrieb nicht mehr vom I/O System angesprochen werden.

Wird beim Hochlauf oder während der Laufzeit ein fehlendes, falsch konfiguriertes oder falsch gestecktes I/O Modul erkannt, so wird dies in der Logger-Datei "System" protokolliert.

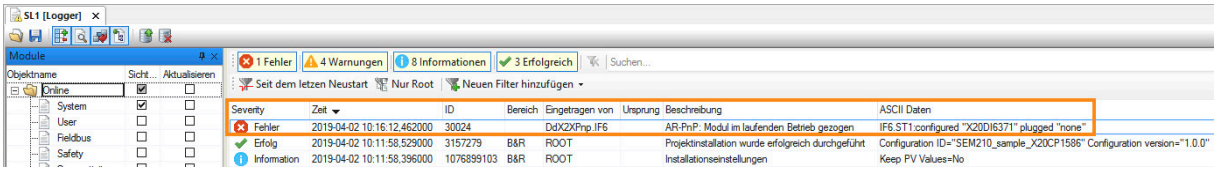


Abbildung 84: Logger-Eintrag "Module removed while running"

Welches Modul hat den Logger-Eintrag verursacht?

Mit Hilfe der ASCII Daten im Logger-Eintrag kann der Steckplatz des Moduls identifiziert werden. "IF6.ST3" bedeutet, dass auf der Schnittstelle IF6, was in diesem Fall die X2X Schnittstelle ist, das Modul "ST3", dies ist der dritte Steckplatz auf diesem Bus, gezogen wurde. Die Schnittstellenbezeichnungen der CPU sind in der Physical View sowie im Datenblatt der verwendeten CPU ersichtlich.

Modulüberwachung deaktiviert

Bei deaktivierter Modulüberwachung kann durch eine Variablenzuordnung auf den Kanal "**ModuleOK**" das I/O Modul von der Applikation aus überwacht werden. Fehlende oder falsch gesteckte Module verursachen keinen Neustart im Betriebsmodus "SERVICE". Nur der Wert vom Kanal "ModuleOk" wird "FALSE". Die Überwachung der Module liegt daher in der Verantwortung der Applikation.

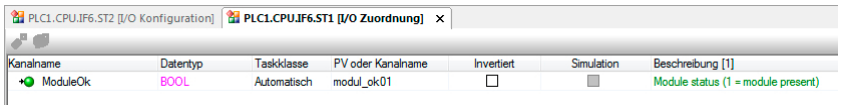


Abbildung 85: Modulüberwachung durch die Applikation

Echtzeit Betriebssystem \ Zielsysteme \ Zielsysteme - SG4 \ I/O Management - SG4 \ Arbeitsweise \ Fehlerbehandlung

Aufgabe: Busverbindung oder I/O Modul im laufenden Betrieb entfernen

Im laufenden Betrieb der Steuerung wird entweder die Verbindung zum X20 Buscontroller entfernt oder I/O Modul entfernt. Die Steuerung bricht die Applikation ab und führt einen Neustart im Betriebsmodus "SERVICE" aus. Die Logger-Datei wird ausgelesen und die Einträge analysiert.

1) Busteilnehmer oder I/O Modul im laufenden Betrieb entfernen
2) Neustart und Hochlauf im Betriebsmodus "SERVICE" abwarten
3) Logger-Datei "System" auslesen und analysieren

6.4 reACTION Technology

Ein wichtiges Thema sind die mit dem Steuerungssystem erreichbaren Reaktionszeiten. Durch den Einsatz von Hochleistungssteuerungen, einem schnellen Echtzeitnetzwerk und intelligenter Netzwerkarchitektur können, mit teilweise erheblichem Aufwand, Reaktionszeiten von 100 µs oder darunter erreicht werden. Vom Markt werden aber immer noch kürzere Reaktionszeiten gefordert.

Durch den Einsatz von reACTION Technology wird die Ausführung von kleinen Logikprogrammen von der Steuerung direkt auf ein I/O Modul ausgelagert. reACTION Technology Programme laufen am reACTION I/O Modul in einem sehr schnellen Zyklus, wodurch sich Reaktionszeiten im µs-Bereich realisieren lassen.

Neben den geringen Reaktionszeiten im reACTION I/O Modul, werden zusätzlich Steuerung und Feldbus entlastet.

reACTION Technology Programme werden in der Logical View projektiert. Beim Übertragen des Projektes, wird das reACTION Technology Programm direkt auf dem I/O Modul installiert und dort ausgeführt.

Zur Konfiguration und Diagnose von reACTION Technology stehen auf den reACTION I/O Modulen in der I/O Zuordnung Datenpunkte zur Verfügung.

Programmierung \ Programme \ reACTION Technology

- Technologiebeschreibung
- Beispiele für den Einsatz der reACTION-Technology
- reACTION Diagramm

7 Zusammenfassung

Das Betriebssystem dient als Schnittstelle zwischen der Applikation und der Hardware. Die Ressourcen der Zielsysteme werden einheitlich vom Betriebssystem verwaltet.

Das Multitasking von Automation Runtime ermöglicht einen modularen Aufbau der Applikation. Dadurch, dass die Anwendung in individuelle Task gegliedert ist und diese Tasks in verschieden Taskklassen laufen, werden die Ressourcen optimal ausgenutzt. Die eigenen Anforderungen können durch die Anpassung des Multitasking Systems an das Zeitverhalten der Anwendung zur Gänze erfüllt werden.

Es entsteht eine optimal konfigurierte, leistungsstarke Anwendung, die die vorhandenen Ressourcen gezielt nutzt.

Abbildung 86: Automation Runtime: eine Softwareplattform für die gesamte B&R Produktpalette

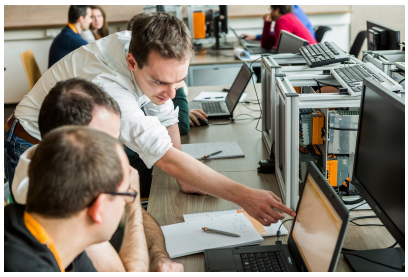
Die Plattformunabhängigkeit, konfigurierbare Schnittstellen sowie Client- und Serverprotokolle machen Automation Runtime zu einem leistungsfähigen Betriebssystem. Die Offenheit des Systems ermöglicht die direkte Integration von Fremdgeräten, Anbindungen zu OPC UA Clients und Servern und implementiert die erforderlichen IT Sicherheitsmechanismen.

Durch zahlreiche Diagnosewerkzeuge und Softwarebibliotheken ist der Zustand von Automation Runtime immer transparent. Dies hilft bei der Inbetriebnahme und bei der Umsetzung individueller Applikationsanforderungen.

Angebote der Automation Academy

Die Automation Academy ist für die zielgerichtete Weiterbildung unserer Kunden und eigenen Mitarbeiter verantwortlich.
Mit der Automation Academy erreichen Sie Handlungskompetenz in kurzer Zeit!
Unsere Seminare ermöglichen Ihnen, Ihre Kompetenz auf dem Gebiet der Automatisierungstechnik zu erweitern.
Sie werden danach in der Lage sein, selbständig effiziente Automatisierungslösungen mit B&R Systemen zu realisieren. Sichern Sie sich entscheidende Wettbewerbsvorteile, um auf die ständig wachsenden Anforderungen des Marktes schneller reagieren zu können.

Seminare



Qualität und Aktualität sind bei Seminaren unverzichtbar. Das Lerntempo im Seminar wird durch das Vorwissen der Teilnehmer und deren Anforderungen bestimmt. Unsere Seminare bieten hohe Flexibilität beim Aufbau des Wissens durch die ideale Kombination von Gruppenarbeit und individuellem Lernen.
Die effizienten Seminarblöcke werden von didaktisch und fachlich bestens ausgebildeten Technikern geleitet.

Trainingsmodule

Die Basis für das Arbeiten im Seminar und beim Selbststudium bilden unsere Trainingsmodule. Die kompakten Module basieren auf einem einheitlichen didaktischen Konzept. Strukturierte und aufeinander aufbauende Inhalte erleichtern das Verständnis für komplexe Zusammenhänge. Sie stellen das ideale Begleitmaterial zum umfangreichen Hilfesystem dar. Die Trainingsmodule sind als Download verfügbar und sind als Druckversion bestellbar.

Unterteilung in Themengebiete:

- ⇨ Steuerungstechnik
- ⇨ Antriebstechnik
- ⇨ Sicherheitstechnik
- ⇨ Visualisieren und Bedienen
- ⇨ Prozessleittechnik
- ⇨ Diagnose und Service
- ⇨ Konnektivität

ETA System

Mit dem ETA System stehen realitätsnahe Aufbauten für Ausbildung, Lehre und Labor zur Verfügung. Zwei verschiedene mechanische Grundkonstruktionen stehen zur Auswahl. Das ETA light System ermöglicht eine hohe Mobilität, ist platzsparend und eignet sich gut fürs Labor. Das ETA Standard System bietet eine robuste mechanische Konstruktion und beinhaltet vollständig verdrahtete Sensoren und Aktoren.

Finden Sie mehr heraus!

Sie haben ein aktuelles Weiterbildungsanliegen? Sie interessieren sich für die Angebote der B&R Automation Academy? Hier sind sie genau richtig.
Folgen Sie dem Link um weitere Informationen zu erhalten:
www.br-automation.com/academy