



Universität Stuttgart

Institut für Steuerungstechnik
der Werkzeugmaschinen und
Fertigungseinrichtungen (ISW)



Studienarbeit

jbjhkbkj

eingereicht von

Lukas Schlotter

aus Stuttgart

Studiengang

Prüfer

Betreuer

Eingereicht am

M. Sc. Mechatronik

Prof. Dr.-Ing. Oliver Riedel

My supervisor, M.Sc.

7. November 2023

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	2
1.2	Anforderungsdefinition	2
1.3	Methodik	2
2	Grundlagen und Stand der Technik	3
2.1	Feldbusse	3
2.2	TCP/IP	4
2.3	Stäubli-Roboter	4
2.4	Anlagensteuerung	4
3	Konzeptionierung und Systementwurf	5
3.1	Feldbus-Verbindung	5
3.2	TCP/IP-Verbindung	5
3.3	Datenverwertung	5
4	Implementierung	9
4.1	Stäubli-Roboter in VAL3	9
4.1.1	EtherCAT	9
4.1.2	TCP/IP	9
4.2	.NET in C#	10
4.2.1	TCP/IP	10
4.2.2	Datenverwertung und Visualisierung	10
5	Validierung	11
6	Ausblick und Fazit	12
	Abbildungsverzeichnis	13
	Tabellenverzeichnis	14
	Literatur	15

1 Einleitung

Warum startet das hier mit ner 0? aTex allows you to manage citations within your document through the use of a separate bibtex file (filename.bib).

1 Einleitung

Bibtex files follow a standard syntax that allow you to easily reference the citations included in that file through the use of a bibliography management package. There are multiple bibliography management packages that you can use to manage citations. This guide will demonstrate how to use biblatex which allows for the most customization.

1.1 Motivation

Dieses Bild zeigt blabla bla von dem Buch [1] und auch [2]

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Tabelle 1.1: Table to test captions and labels.

1.2 Anforderungsdefinition

1.3 Methodik

2 Grundlagen und Stand der Technik

2.1 Feldbusse

Feldbusse: Elektrotechnik für Maschinenbauer ab S.485

Feldbusse: Profibus, CAN, Sercos

Ethernet basierte Feldbusse: Profinet, Ethernet/IP, EtherCAT, Sercos III

Ethernetbasierte Systeme sind bereit Feldbusse abzulösen

Ethernet: deutlich mehr Daten als klassisch

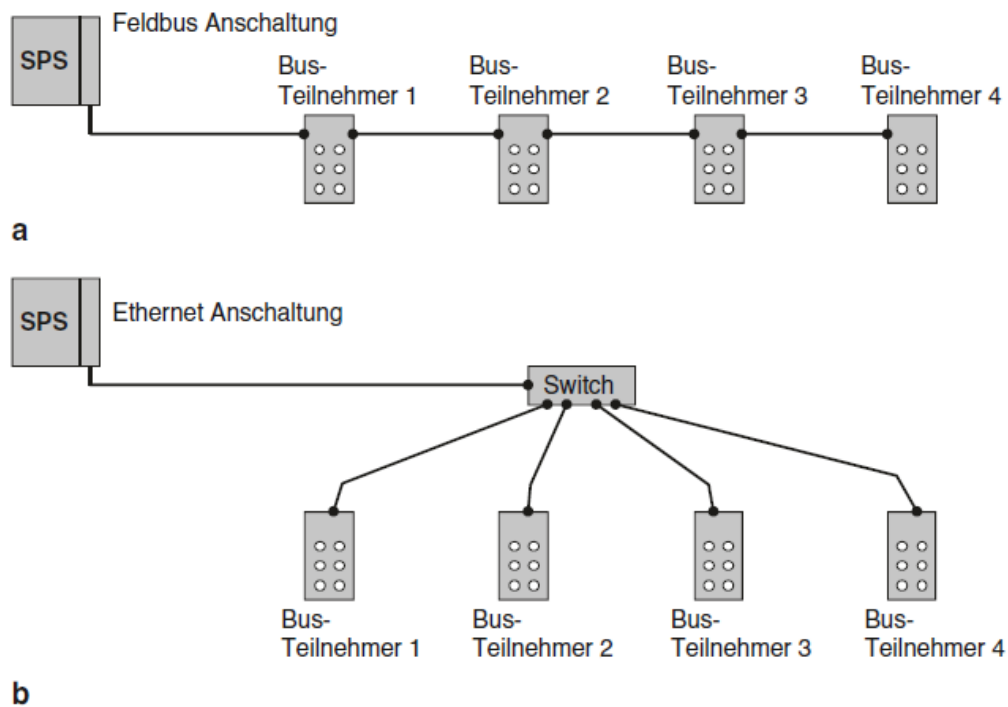


Bild H-19 Übergang von der Linienstruktur in die Sternstruktur bei Ethernet. (a) Linienstruktur bei Standard Feldbussystemen, (b) Sterntopologie bei Ethernet Feldbussystemen

Abbildung 2.1: Anschaltung Feldbus und Ethernet [3]

Multi-Master Bussen (z.B. CAN oder TCP/IP) vs. Mono-Master

2 Grundlagen und Stand der Technik

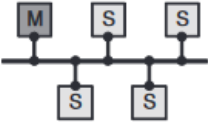
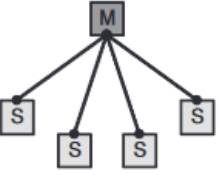
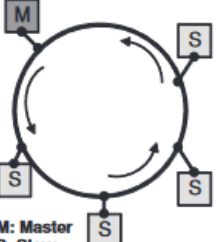
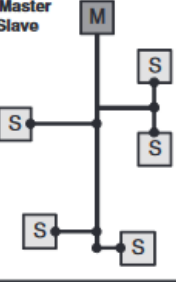
Topologie	Bus-/ Linien-Struktur	Stern-Struktur	Ring-Struktur	Baum-Struktur
Aufbau	 <p>M: Master S: Slave</p>	 <p>M: Master S: Slave</p>	 <p>M: Master S: Slave</p>	 <p>M: Master S: Slave</p>
Beispiele	Profibus CAN-Bus Bit-Bus P-Net	DIO-Bus Ethernet	Interbus S Ethernet Sercos	AS-Interface LON

Abbildung 2.2: Topologien

2.2 TCP/IP

MAC-Adresse eindeutig von Gerät. Für bessere Identifiuierung aber IP-Adresse

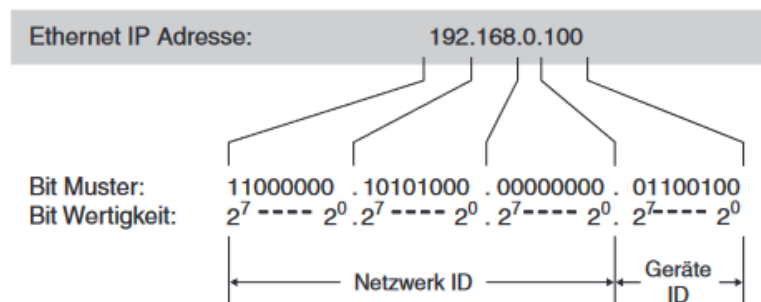


Abbildung 2.3: Topologien

Verschiedene Klassen an IP-Adressen. Meist Klasse C verwendet

Weil Multi-Master-Bus braucht man CSMA/CD-Verfahren -> nicht echtzeitfähig. Lösung: Echtzeitprotokolle

2.3 Stäubli-Roboter

2.4 Anlagensteuerung

3 Konzeptionierung und Systementwurf

3.1 Feldbus-Verbindung

3.2 TCP/IP-Verbindung

3.3 Datenverwertung

Tabelle 3.1: Verfügbare Daten

Daten	Zugriff	Verwendung
CPU battery test	D (CpuIO)	-
CPU overcurrent	D (CpuIO)	-
Fast memory state	D (CpuIO)	-
CPU temperature	A (CpuIO)	Alarm, PM
CPU board temperature	A (CpuIO)	Alarm, PM
CPU fan speed	A (CpuIO)	Alarm, PM
Free RAM	A (CpuIO)	-
CFast memory remaining lifetime	A (CpuIO)	-
System management CPU usage	A (CpuUsage)	-
Robot control CPU usage	A (CpuUsage)	-
Synchronous VAL 3 CPU usage	A (CpuUsage)	-
Available CPU usage for VAL 3 processing	A (CpuUsage)	-
User Fieldbuses CPU usage	A (CpuUsage)	-
HMI CPU usage	A (CpuUsage)	-
SRS connection CPU usage	A (CpuUsage)	-
OPC-UA CPU usage	A (CpuUsage)	-
CPU Load Score (higher is better)	A (CpuUsage)	-
CPU Load Score (min)	A (CpuUsage)	-
VAL 3 instructions per sequencing	A (CpuUsage)	-
VAL 3 synr inst. per cyle (high priority)	A (CpuUsage)	-
VAL 3 synr inst. per cyle (low priority)	A (CpuUsage)	-
Valve feedback (1.1, 1.2, 2.1, 2.2)	D (DsiIO)	-
Axis brake feedback (1-6)	D (DsiIO)	-
Error on valves outputs	D (DsiIO)	Alarm

3 Konzeptionierung und Systementwurf

Tabelle 3.1: Verfügbare Daten

Daten	Zugriff	Verwendung
Error on brakes outputs	D (DsiIO)	Alarm
Error on safe digital inputs	D (DsiIO)	Alarm
DSI non-reduced brake supply voltage undershoot	D (DsiIO)	Alarm
DSI reduced brake supply voltage undershoot	D (DsiIO)	Alarm
DSI logic supply voltage undershoot	D (DsiIO)	Alarm
DSI overtemperature	D (DsiIO)	Alarm
DSI board temperature	A (DsiIO)	Alarm, PM
Axis motor temperature (1-6)	A (DsiIO)	Alarm, PM
Axis encoder temperature (1-6)	A (DsiIO)	Alarm, PM
DSI state	A (DsiIO)	-
DSI error code	A? (DsiIO)	? Grau? Alarm
Arm operation counter	A (DsiIO)	-
safe Input state (0-7)	D (DsiIoSafe)	-
Fast Input (1-2)	D (FastIO)	-
Fast Output (1-2)	D (FastIO)	-
Power unit identification (bit 1-3)	D (PSIO)	-
Main power state	D (PSIO)	-
Internal bus voltage state	D (PSIO)	-
Power unit internal temperature state	D (PSIO)	-
24V state	D (PSIO)	-
Buckfull mode feedback	D (PSIO)	-
Memorize a power dropout	D (PSIO)	-
Brake test warning	D (Rsi9IO)	Alarm
Brake test successful	D (Rsi9IO)	-
Temperature of RSI board	A (Rsi9IO)	Alarm, PM
Error Code of RSI board	A? (Rsi9IO)	?Grau? Alarm
STARC board temperature	A (StarCIO)	Alarm, PM
Axis drive case temperature (1-6)	A (StarCIO)	Alarm, PM
Motor Winding Temperature (1-6)	A (StarCIO)	Alarm, PM
Axis drive junction temperature (1-6)	A (StarCIO)	Alarm, PM
Geschwindigkeit Endmanipulator	getSpeed(...)	-
Schleppfehler Achsen 1-6	getPositionErr()	-
Drehmoment Achsen 1-6	getJointForce(...)	-
Bewegungsauftrag und Fortschritt	getMoveld()	-
Konfiguration des Roboters	getVersion(...)	-
Stromversorgung bei Stillstand abschalten	hibernateRobot()	-

Tabelle 3.1: Verfügbare Daten

Daten	Zugriff	Verwendung
Systemereignisse	getEvents(...)	-

Durch die Verfügbarkeit von Roboterdaten ergeben sich Nutzungspotentiale, wie z.B.:

- **Überwachung und Alarm:** Bei Überschreiten von Schwellwerten oder bei Fehlermeldungen Alarm auslösen
- **Predictive Maintenance:** Vorhersagen treffen, wann Wartung erfolgen soll, um Ausfälle vorbeugend zu verhindern.
- **Datenarchivierung und Compliance:** Datenspeicherung für Schadensfall oder gesetzliche Gewährleistung
- **Trendanalyse:** Muster und Tendenzen in den Daten erkennen

Prinzipiell lassen sich alle verfügbaren Daten sammeln, visualisieren und auswerten. Bei einigen dieser Daten wie z.B. des freien RAM-Speichers ist der daraus entstehende Nutzen beschränkt. Mittels Brainstorming wurden verschiedene Anwendungsszenarien ausgedacht, im nachfolgenden sollen jedoch nur die sinnvollsten hiervon vorgestellt werden.

Temperaturen

Neben der Temperatur von verschiedenen Computer-Chips und Platinen, wie z.B. CPU, CPU-Platine DSI-Platine, RSI-Platine und STARC-Platine sind verschiedene Temperaturwerte von den Antrieben abrufbar. Hier sind die Temperaturen der Motoren, Encoder, Antriebsgehäuse, Antriebswicklungen und Steuergeräte zu nennen. Diese Daten können sowohl zur Überwachung als auch zur Predictive Maintenance verwendet werden. Bei Überschreiten eines Grenzwertes kann ein Alarm bzw. eine Warnung ausgegeben werden. Da von Seiten des Herstellers keine zulässigen Grenzwerte vorgegeben sind, müssen die aufgezeichneten Messwerte unter Berücksichtigung von Schwankungen analysiert werden und Grenzwerte festgelegt werden, ab welchen Werten das Verhalten nicht mehr als "normal" angesehen werden kann. Die Dynamik von Temperaturen ist im Allgemeinen relativ gering, da sich die Materialien aufgrund ihrer Wärmekapazität erst aufheizen müssen. Aus diesem Grund ist das Übertragen von den Temperaturwerten in vergleichsweise großen Abständen zulässig z.B. alle 15 Sekunden. Im Falle eines technischen Defektes wäre eine möglichst frühe Warnung wünschenswert, jedoch ist die Zeit bis ein Techniker den Alarm wahrnimmt und entsprechende Aktionen starten kann vergleichsweise groß, weshalb dieser Anwendungsfall kein höhere Übertragungsrate rechtfertigt. Da die Genauigkeit von vielen Temperatursensoren nur im Bereich von 0,5K liegt und sehr genaue Temperaturwerte keinen zusätzlichen Mehrwert bieten, ist eine Übertragung der Temperaturen ohne Nachkommastellen ausreichend. Dadurch lässt sich ein Temperaturwert problemlos mit einem einzelnen Byte (Werte zwischen 0 und

255°C) übertragen.

Error-Meldungen

Von dem Stäubli-Roboter werden einige Fehlermeldungen erfasst, wie von den Ventil und Brems-Ausgängen, den digitalen Sicherheits-Eingängen, des DSI-Chip, RSI-Chip und Bremsentests. Ebenso werden "DSI non-reduced brake supply voltage undershoot", "DSI reduced brake supply voltage undershoot", "DSI logic supply voltage undershoot" und "DSI overtemperature" erfasst. Error-Meldungen wie diese sollen als Alarm bzw. Fehlermeldung ausgegeben werden. Ebenso ist ein Dokumentieren sinnvoll, um bei Ausfällen oder Defekten die Ursachensuche zu erleichtern. Jeder Fehler kann als Wert 1 oder 0 dargestellt werden, weshalb je Fehlermeldung ein Bit genügt. Alle Error-Meldungen lassen sich somit über 2 Bytes abbilden.

Schleppfehler

Der Schleppfehler entspricht der Abweichung zwischen Soll-Position und Ist-Position jeder einzelnen Achse. Ein herausstechend großer Schleppfehler kann auf Probleme mit der Steuerung, den Motoren, den Encoder oder den Achsen selbst hinweisen. Eine genaue Eingrenzung ist auf Basis der gegebenen Daten nicht möglich, jedoch ist eine Warnung sinnvoll. Zur Ermittlung der Grenze, bei deren Überschreiten eine Warnung ausgegeben werden soll, können aufgezeichnete Daten unter Berücksichtigung von Schwankungen herangezogen werden. Der Schleppfehler muss sehr regelmäßig erfasst werden, um die kontinuierliche Bewegung bestmöglichst abzubilden.

4 Implementierung

4.1 Stäubli-Roboter in VAL3

4.1.1 EtherCAT

4.1.2 TCP/IP

Für die Implementierung der TCP/IP-Verbindung auf dem Controller des Stäubli-Roboters muss in der SRS eine Socket-Verbindung angelegt werden. Hierzu wird in der E/A-Verwaltung ein Client angelegt, welcher die IP-Adresse und den Port des Servers zugewiesen bekommt. Darüber hinaus wird ein sogenannter Timeout von 0 s gesetzt. Bei einem Timeout von 0 wird auf den Vorgang, welcher ein Lesen oder Schreiben sein kann gewartet. Bei einem Timeout kleiner 0 wird hingegen nicht bis zur Ausführung des Vorgangs gewartet. Bei einem Timeout größer 0 wird hingegen eine gewisse Zeit gewährt, bis zu dieser der Timeout durchgeführt werden kann. Die Nachricht soll in diesem Fall jedoch direkt gelesen oder geschrieben werden, weshalb kein Spielraum im Rahmen des Timeouts gewährt wird. [4] Die Socket-Verbindung wird als E/A-Verbindung in VAL3 betrachtet, weshalb eine globale Variable mit dem Namen des Clients angelegt werden kann und hierüber auch gelesen und beschrieben werden kann. Die Socket-Verbindung wird nur dann erstellt, wenn sie im Rahmen des Programmablaufs z.B. durch die Befehle `sioSet` und `sioGet` benötigt wird. Der Client versucht dann eine Verbindung zum Server aufzubauen. `usepackageffcode`

```
num sioGet(sio siInput, num& nData[])
```

Diese Funktion schreibt ein gelesenes Zeichen oder einen gelesenen Array von Zeichen von `siInput` in das Array `nData`. Als Rückgabewert dient die Anzahl der gelesenen Zeichen.

```
num sioSet(sio siOutput, num& nData[])
```

Mit dieser Funktion kann in VAL3 die zu übermittelnde Nachricht `nData` versendet werden, indem der E/A-Verbindung `siOutput` die Nachricht zugewiesen wird. Zurückgegeben wird die Anzahl der geschriebenen Zeichen oder 1 im Falle des Timeouts.

Das Versenden von Nachrichten erfolgt über einen Byte-Array, das heißt durch die Aneinanderreihung mehrerer Bytes. Folglich muss die zu versendete Nachricht in einen Byte-Array umgewandelt werden und beim Empfangen muss der Byte-Array interpretiert werden.

```
num toBinary(num nValue[], num nValueSize, string sDataFormat, num& nDataByte[])
```

Diese Funktion wandelt einen numerischen Wert, welcher das Datenformat `sDataFormat` besitzt in einen Byte-Strom und speichert diesen im Array `nDataByte`. Über das Datenformat wird beispielsweise angegeben ob es sich um einen Gleitkommawert handelt, ob ein

4 Implementierung

Vorzeichen vorliegt und ob das Little-Endian oder das Big-Endian-Format angewandt wird. Mit `nDataSize` kann die Anzahl der zu kodierenden Zeichen beschränkt werden. `num fromBinary(num nDataByte[], num nDataSize, string sDataFormat, num& nValue[])`

Umgekehrt ermöglicht diese Funktion, einen empfangen Byte-Array in numerische Werte zu konvertieren. Das Ergebnis im Datenformat `nDataFormat` wird in `nValue` gespeichert. Die Anzahl der zu decodierenden Bytes wird festgelegt durch `nDataSize`, wenn nicht alle Bytes des Eingangs-Array `nDataByte` decodiert werden sollen.

4.2 .NET in C#

4.2.1 TCP/IP

4.2.2 Datenverwertung und Visualisierung

5 Validierung

6 Ausblick und Fazit

Abbildungsverzeichnis

2.1	Anschaltung Feldbus und Ethernet [3]	3
2.2	Topologien	4
2.3	Topologien	4

Tabellenverzeichnis

1.1	Table to test captions and labels.	2
3.1	Verfügbare Daten	5
3.1	Verfügbare Daten	6
3.1	Verfügbare Daten	7

Literatur

- [1] T. Tantau, *Tikz & pgf*, 2013.
- [2] M. Kohm und J.-U. Morawski, *KOMA-Script – ein wandelbares LaTeX-2-Paket*, 2013.
- [3] E. Hering, R. Martin, J. Gutekunst und J. Kempkes, *Elektrotechnik und Elektronik für Maschinenbauer*. Springer, 2017.
- [4] Staubli, *VAL 3-Handbuch*. Staubli International A, 2022.