Bachelorarbeit

Sensoranbindung mit IO-Link

Stefanie Schmidiger

Dozent:  
Prof. Dr. Thierry Prud‘homme

Hochschule Luzern – Technik & Architektur  
Horw, Hochschule Luzern – Technik & Architektur, Juni 2016

Inhaltsverzeichnis

[1 Einleitung 2](#_Toc450917146)

[2 Aufgabenstellung 3](#_Toc450917147)

[3 IO-Link 4](#_Toc450917148)

[3.1 Physikalische IO-Link Schnittstelle 4](#_Toc450917149)

[3.2 Betriebsarten 5](#_Toc450917150)

[3.3 Datenübertragung 7](#_Toc450917151)

[3.4 IO-Link Protokoll Stack 14](#_Toc450917152)

[4 Sensoren 17](#_Toc450917153)

[4.1 I2C Schnittstelle 17](#_Toc450917154)

[4.2 SPI Schnittstelle 18](#_Toc450917155)

[5 Hardware 20](#_Toc450917156)

[5.1 Auswahl des IO-Link Demoboardes 20](#_Toc450917157)

[5.2 Programmer/Debugger 22](#_Toc450917158)

[5.3 Auswahl des IO-Link Stacks 22](#_Toc450917159)

[5.4 IO-Link Master 23](#_Toc450917160)

[6 Softwareimplementierung 23](#_Toc450917161)

[6.1 Demoapp 23](#_Toc450917162)

[6.2 Implementierung der Sensordaten in die Demoapp 25](#_Toc450917163)

[6.3 Finale IO-Link Applikation mit Sensordaten 32](#_Toc450917164)

[7 Fazit 35](#_Toc450917165)

[Abbildungsverzeichnis 36](#_Toc450917166)

[Tabellenverzeichnis 36](#_Toc450917167)

[Literaturverzeichnis 36](#_Toc450917168)

[Anhang 38](#_Toc450917169)

Abstract

The abstract provides an overview

based on information from other

sections of the report. The reader can

read the abstract and obtain enough

information to decide if they want to

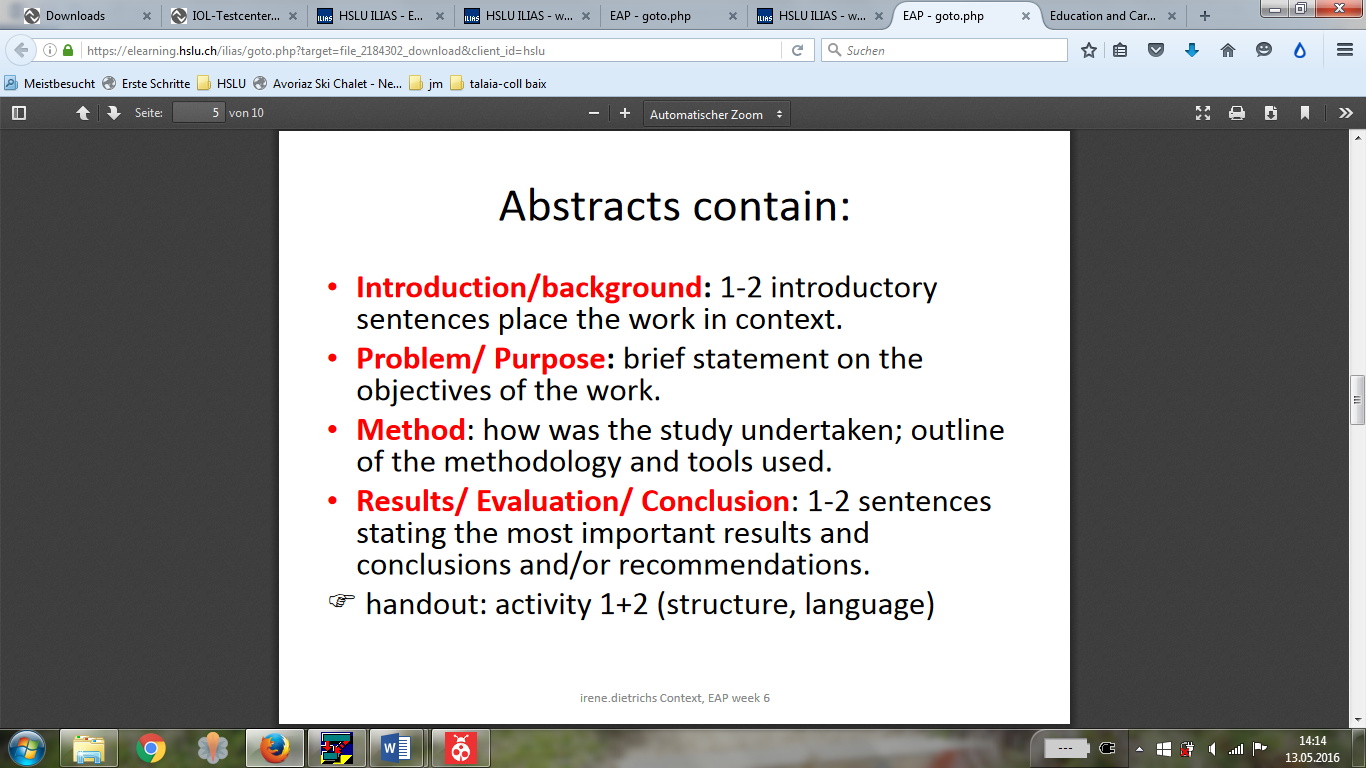
read the co

mplete report.

Length: 4

-

10 sentences



Start by giving some background

information to justify your study

•

Then describe main purpose

•

Summarize methodology used

•

Key results come next

•

Evaluate results

•

Briefly describe the conclusions,

which may inc

lude

recommendations

# Einleitung

Die Firma Leister möchte für ihre Laserschweissmaschine die Sensordaten zentral auslesen und verarbeiten. Für einen ersten Demonstrationsaufbau sollen deshalb die Daten eines Leistungssensores und ein Pyrometers ausgelesen, in ein IO-Link Protokoll verpackt und an den IO-Link Master weitergesendet werden. Diese beiden Sensoren haben eine SPI resp. I2C Schnittstelle und es gilt, die Daten mit einem Mikrocontroller auszulesen und in ein IO-Link Protokoll einzufügen. Der Aufbau des Demobordes sollte deshalb über eine SPI und eine I2C Schnittstelle verfügen und die Daten im IO-Link Protokoll mit 24V Spannungspegel weiterleiten können.

The introduction begins

with a general

statement about the topic, to engage

and interest the reader

, orient your

reader, and lead logically to your

purpose statement.

Approximate length

: 10 % of whole

text

Introduction

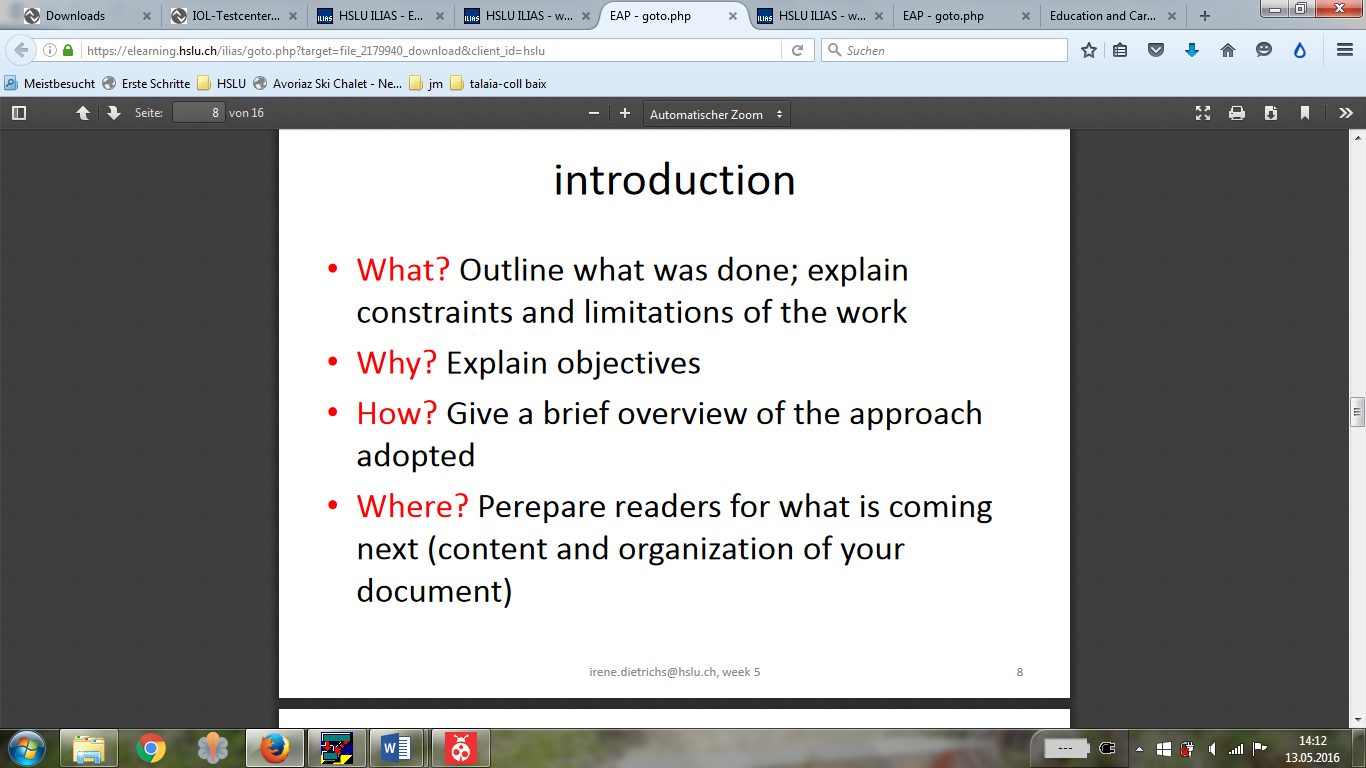
•Introduces the topic

•Gives limited background information to orientate the reader

•States the purpose of the text with controlling ideas to direct the text

•Moves from general to more specific

•A secondary purpose is to attract interest in the topic



# Aufgabenstellung

Eine Laserschweissmaschine der Firma Leister benötigt zahlreiche Sensoren zur Überwachung des Schweissprozesses. Zur Zeit werden diese Sensoren individuell angesteuert und die Daten auch individuell ausgelesen.

Im Rahmen der vorliegenden Arbeit sollen diese Sensoren mit Hilfe von IO-Link zentral angesteuert und ausgelesen werden können. Es soll ein Demonstrator-Setup erstellt werden, anhand wessen der Einsatz von IO-Link evaluiert wird. Für dieses Setup werden Daten von zwei Sensoren in das IO-Link Protokoll verpackt und via der IO-Link Schnittstelle an einen Beckhoff IO-Link Master weitergeleitet.

Für diese Aufgabe gilt es zuerst, ein geeignetes Demoboard zu finden. Mit diesem werden die Sensoren angesteuert und die Sensordaten ins IO-Link Protokoll verpackt.

Die Firma Leister möchte die Sensordaten seitens der SPS mit einer Rate von mindestens 2 kHz einlesen können. Es gilt zu evaluieren, ob dies mit IO-Link möglich ist und mit welcher maximalen Frequenz die Sensordaten eingelesen werden können.

Weiter muss evaluiert werden, ob beide Sensoren als einziges Device gelten dürfen oder ob pro Sensor ein IO-Link Anschluss benötigt wird. Berücksichtigt werden soll dabei auch das Einhalten der minimalen geforderten Datenupdaterate von 2 kHz.

Genauere Details und Eckdaten dieser Arbeit können der offiziellen Aufgabenstellung aus Anhang A entnommen werden.

# IO-Link

IO-Link ist ein herstellerunabhängiger Standard um mit Sensoren und Aktoren zu kommunizieren. Ein IO-Link System besteht aus einem oder mehreren IO-Link Master und IO-Link Slaves. Der IO-Link Master stellt die Verbindung zwischen den Slaves und dem System her. Der Master kommuniziert über Feldbusse oder produktspezifische Rückwandbusse. Ein IO-Link Master kann mehrere IO-Link Ports (Anschlüsse) besitzen. An jedem Port ist ein IO-Link Slave anschließbar (Punkt-zu-Punkt-Kommunikation). Somit ist IO-Link eine Punkt-zu-Punkt-Kommunikation und kein Feldbus. [1]

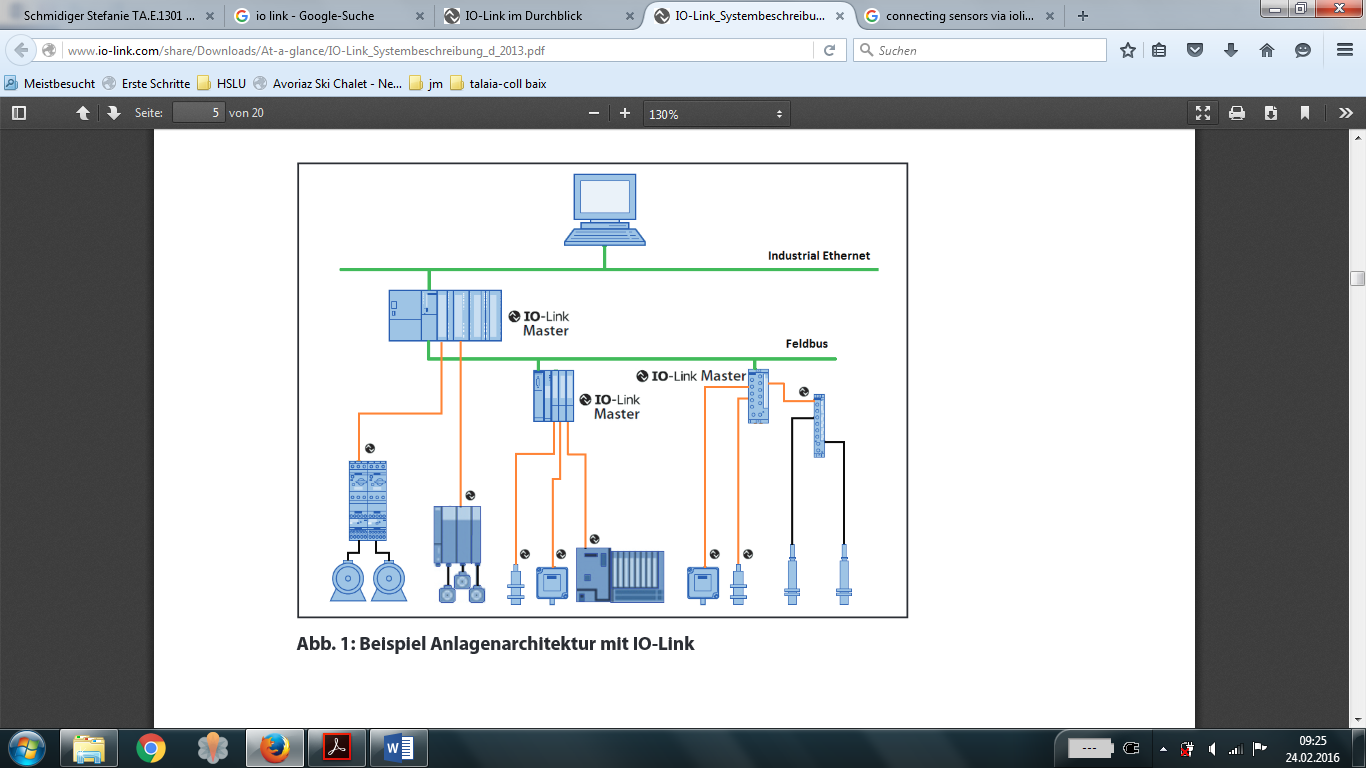


Abbildung 1 - Beispiel Anlagenarchitektur mit IO-Link [1]

## Physikalische IO-Link Schnittstelle

Bei IO-Link handelt es sich um eine serielle und bidirektionale Punkt-zu-Punkt Verbindung. Sensoren und Aktoren benötigen dabei eine 3-polige bis 5-polige Verbindung. Der Master verfügt über eine 5-polige Buchse , wobei die Anschlussbelegung nach IEC 60974-5-2 Standard wie folgt definiert ist:

* Pin 1: 24V
* Pin 3: 0V
* Pin 4: Schalt- und Kommunikationsleitung

Die Anschlussbelegung kann Abbildung 2 entnommen werden.

Die Energieversorgung ist mit Pin 1 und Pin 3 gewährleistet und liefert maximal 200 mA pro Slave. Die maximale Leiterlänge für ein ungeschirmtes Kabel zwischen Device und Master beträgt 20 m.

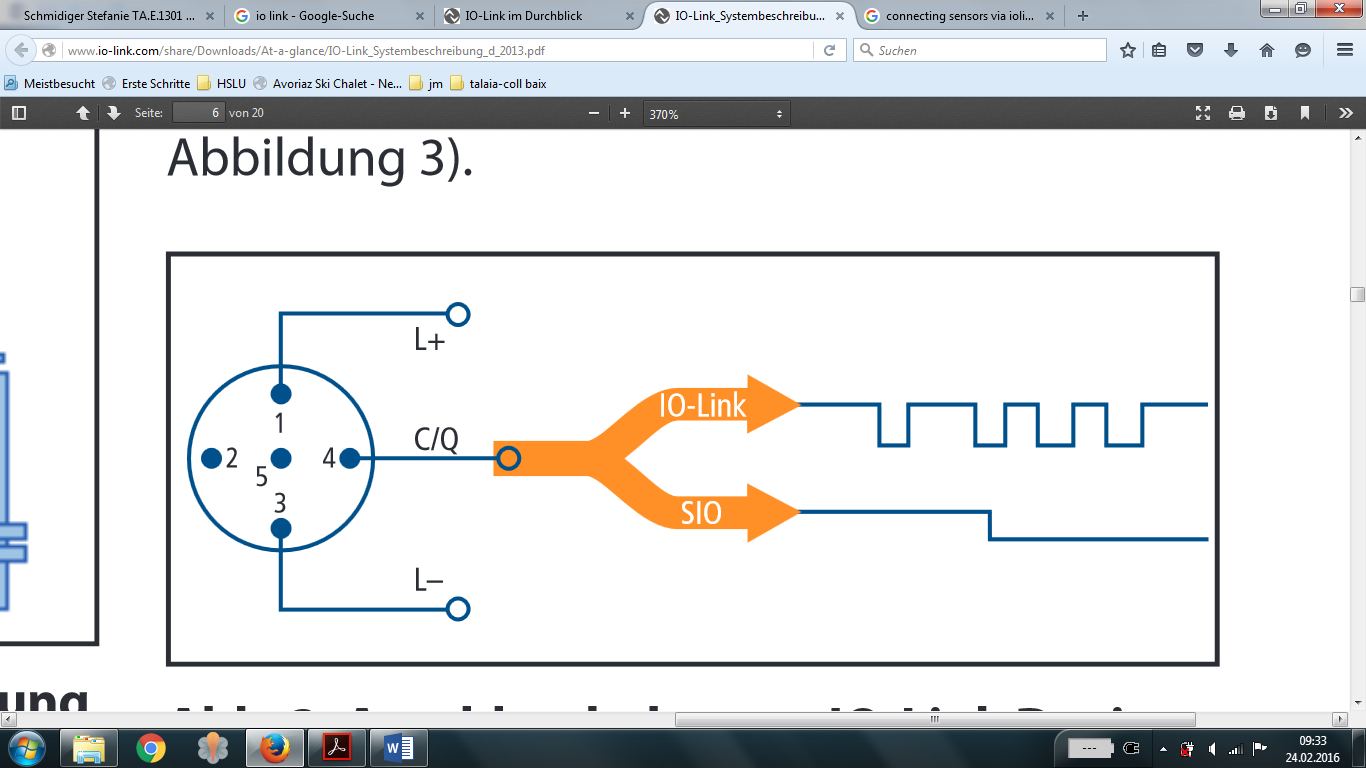


Abbildung 2 - Anschlussbelegung IO-Link Device [1]

Die Standardverbindung ist eine Port Klasse A Verbindung, die vier Pins umfasst. Grundsätzlich gibt es folgende zwei Portklassen zu unterscheiden:

### Port Class A

Die Verbindung umfasst vier Leitungen, wobei die Funktion von Pin 2 nicht vorgegeben. So kann dieser als zusätzlicher Digitalpin oder gar nicht verwendet werden. Die Buchsengrösse für das Anschlussgewinde variiert zwischen M5, M8 und M12.

### Port Class B

Dieser Porttyp ist für Geräte mit hohem Stromverbrauch geeignet, die Pins 2 und 5 stellen eine zusätzliche galvanisch getrennte Speisung zur Verfügung. Port Class B Steckverbindungen müssen eindeutig als solche gekennzeichnet werden und sind immer M12 Buchsen.

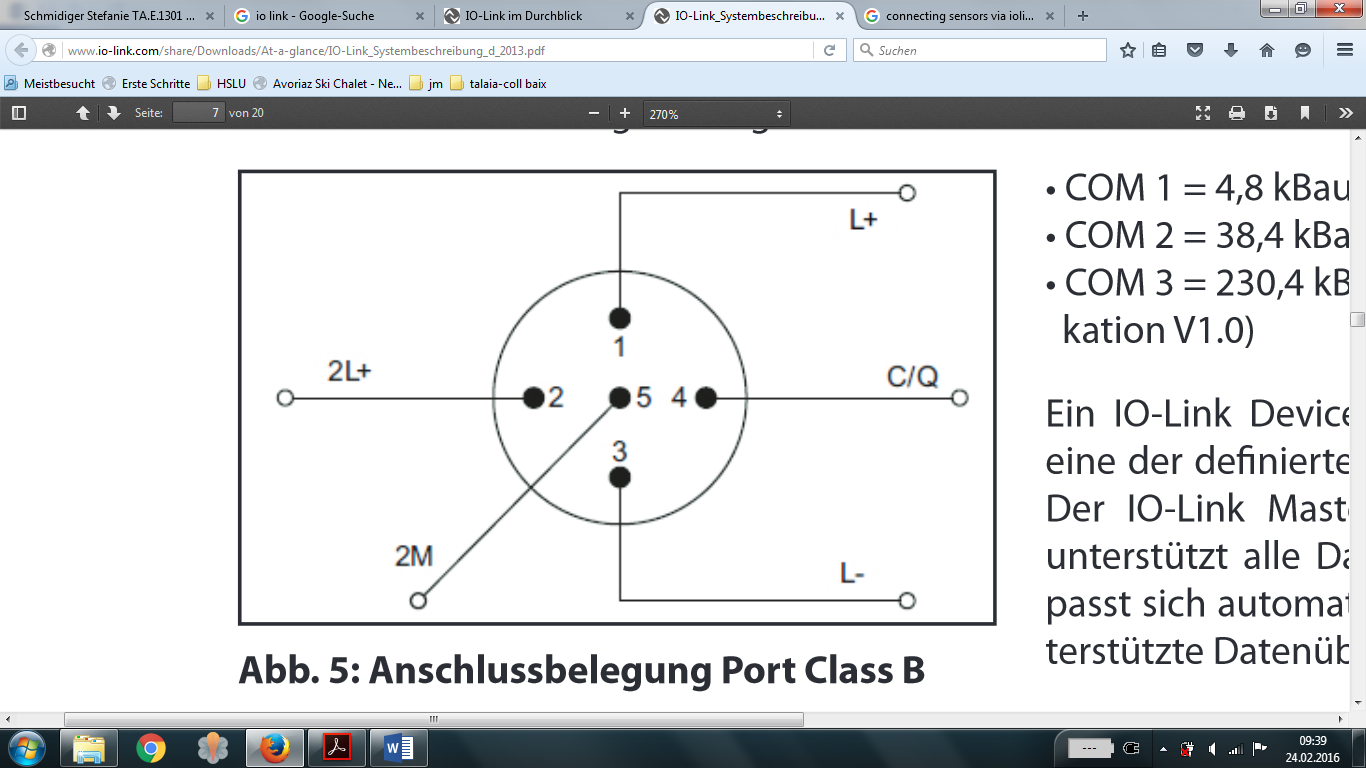


Abbildung 3 - Anschlussbelegung Port Class B

## Betriebsarten

Die IO-Link Ports des Masters lassen sich in drei verschiedenen Betriebsarten betreiben:

* IO-Link Der Port befindet sich in der IO-Kommunikation
* SIO Nach dem Einschalten befindet sich ein Port immer im SIO (Standard Input  
   Output) Modus und verhält sich wie ein digitaler Eingang. Der SIO Modus  
   ist ideal für Geräte, welche entweder High- und Low-Pegel erwarten oder  
   diese auf den Bus treiben.
* Deaktiviert Ein nicht benutzter Port ist deaktiviert.

Im IO-Link Betriebsmodus sind drei Übertragungsgeschwindigkeiten (für die IO-Link V1.1) spezifiziert:

* COM1 4.8 kBaud
* COM2 38.4 kBaud
* COM3 230.4 kBaud

Mit IO-Link V1.0 sind nur die Übertragungsgeschwindigkeiten 4.8 kBaud und 38.4 kBaud möglich. Beim neueren IO-Link V1.1 können Daten zusätzlich mit 230.4 kBaud übertragen werden.

Ein IO-Link Device unterstützt immer nur eine Übertragungsgeschwindigkeit, an welche sich der Master automatisch anpasst. Ein Master kann für jeden seiner Ports eine andere Übertragungsgeschwindigkeit verwenden.

### Erstverbindung von Device und Master

Ist der Port des Masters auf IO-Link eingestellt, so wird er versuchen, mit dem angeschlossenen Gerät zu kommunizieren. Dazu sendet der Master zuerst in der höchsten definierten Übertragungsgeschwindigkeit einen Wake Up Impuls und wartet auf die Antwort des Geräts. Bei einem Misserfolg versucht er es in der nächst niedrigen Übertragungsrate erneut.

Sobald der Master eine Antwort empfängt, beginnt die Kommunikation. Zuerst werden Geräteparameter ausgetauscht und allenfalls im System gespeicherte Parameter auf das Gerät übertragen. Anschliessend folgt die Übertragung der zyklischen Daten.

Das Device geht der Reihe nach folgende Operationsmodi durch:

* INACTIVE Kein IO-Link Master versucht mit dem Device zu kommunizieren
* STARTUP Device hat Wake Up Impuls erhalten vom Master
* PREOPERATE Master hat Geschwindigkeit vom Device getroffen (COMx), jetzt  
   werden Geräteparameter ausgetauscht
* OPERATE Es findet ein zyklischer Austausch an Prozessdaten statt

### IO-Link Device Description Datei

Die IO-Link Device Description (IODD) Datei ist eine Gerätebeschreibungsdatei und hält für die Systemintegration vielfältige Informationen bereit. Die Bereitstellung einer IODD wird von der IO-Link Community vorgeschrieben, wenn das Device verkauft wird. Die IODD wird benötigt, wenn ein Master offline aufgesetzt wird, also keine Verbindung zur Steuerung besteht. Dies ermöglicht es dem User, einen Port für ein bestimmtes Gerät offline aufzusetzen. Die IODD enthält alle Informationen, die für diese Schnittstelle benötigt werden (z.B. Anzahl Bytes die versendet oder empfangen werden, minimale benötigte Zykluszeit vom Device, usw.). So können auch Programme für die SPS im voraus geschrieben und Einstellungen vorgenommen werden, ohne dass eine direkte Verbindung zum laufenden System bestehen muss.

Der Aufbau der IODD ist strikt vorgeschrieben von der IO-Link Community. Ein IODD muss nach der Erstellung mit dem IODD Checker, der auf der Website von IO-Link gratis heruntergeladen werden kann, geprüft werden. So ist die korrekte Implementierung gewährleistet.

### Reaktionszeit und Zykluszeit

In der Gerätebeschreibungsdatei IODD des Slaves ist ein Wert für die minimale Zykluszeit angegeben, welcher bestimmt, mit welcher maximaler Frequenz der Master das Gerät ansprechen darf um zyklisch Daten zu erfragen bzw. zu übertragen. Zusätzlich benötigt der Master noch eine Bearbeitungszeit. Diese beiden Zeiten ergeben die Reaktionszeit des Systems. Jedes Gerät kann eine andere minimale Zykluszeit haben. Der Master hat aber nur eine Zykluszeit und so passt sich deshalb dem langsamsten verbundenen Gerät an. Beispielsweise kann an Port 1 ein Gerät mit einer minimalen Zykluszeit von 400 µs verbunden sein, an Port 2 ein Gerät mit einer minimalen Zykluszeit von 500 µs und an Port 3 eines mit einer minimalen Zykluszeit von 5 ms. Der Master wird also eine Zykluszeit von 5 ms wählen und auch die Devices, die eigentlich schneller operieren könnten, werden nur alle 5 ms abgefragt.

### Abwärtskompatibilität

An ein IO-Link Master der Version V1.0 können nur IO-Link Devices der Version V1.0 angeschlossen werden. Ein IO-Link Master der Version V1.1 kann jedoch gleichzeitig IO-Link Devices der Versionen V1.0 und V1.1 angeschlossen haben.

Eine Datenübertragungsrate von 230.4 kBaud ist nur mit der Version V1.1 möglich. Deshalb wird ein IO-Link Master der Version V1.1 mit verbundenem V1.1 Gerät benötigt um die geforderte Leserate zu erreichen. Ein V1.0 Device an einem V1.1 Master kommuniziert mit maximal 38.4 kBaud.

## Datenübertragung

Bei der Kommunikation zwischen Master und Slave gibt es vier verschiedene Datenarten. Die Datengrösse ist dabei immer in Oktetten angegeben. Ein Oktett entspricht einem Byte.

Prozessdaten  
Die Prozessdaten sind zyklische Daten wie beispielsweise Sensorwerte. Die Datenmenge einer Übertragung kann dabei vom Gerät einmalig beim Einschalten festgelegt werden und liegt zwischen 0 und 32 Oktette.

* Prozessdaten Output zyklische Daten vom Master an den Slave
* Prozessdaten Input zyklische Daten vom Slave an den Master

Wertstatus  
Der Wertestatus kann mit den Prozessdaten zyklisch übertragen werden und zeigt an, ob die Prozessdaten gültig oder ungültig sind.

Gerätedaten  
Ein Device muss vielfach konfiguriert werden können. Für diesen Zweck gibt es ein Adressbereich, auf den mit einem Index zugegriffen werden kann. Innerhalb dieses Adressbereiches gibt es die Direct Parameter Page 1, welche vorgegebene Daten enthalten muss, die der Master beim Aufsetzen der Kommunikation ausliest. Dazu gehören z.B. VendorID, DeviceID, Revisionsnummer oder die Baudrate.

Die Direct Parameter Page 2 kann beliebige Informationen enthalten; deren Interpretation und Inhalt werden vom Gerätehersteller festgelegt. Bei Bedarf kann der Master auf eine bestimmte Stelle im Adressbereich zugreifen und die darin enthaltenen Daten (Parameter, Identifikationsdaten oder Diagnoseinformationen) auslesen oder überschreiben. Somit kann azyklisch den Status des Devices verändert oder abfragt werden.

Ereignisse (Events)  
Wenn ein Ereignis auftritt, signalisiert das Gerät dies dem Master und dieser liest es aus. Ereignisse können Fehlermeldungen, Warnungen oder Wartungsdaten sein. Diese Ereignisse werden der Steuerung weitergeleitet. Auch können Ereignisse seitens des Masters weitergeleitet werden.

### Übertragungsart verschiedener Daten

Eine Übersicht über die Übertragung der oben beschriebenen Daten kann aus Abbildung 4 entnommen werden:

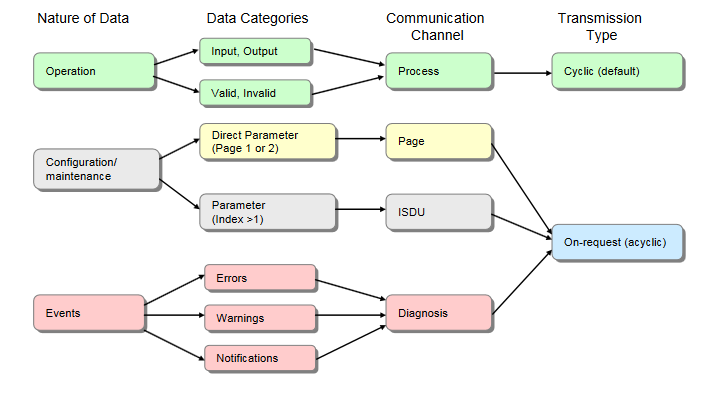


Abbildung 4 - Übertragungsmodi verschiedener Daten [2]

Die Prozessdaten und der Wertestatus bilden zusammen die Kategorie „Operation“ und werden zyklisch übertragen.

Ereignisse und Gerätedaten sind On-Request Daten und werden nur auf Anfrage oder bei Bedarf übertragen.

### Datenspeicherung

Aus der Abbildung 5 kann die Speicherverwaltung innerhalb eines Geräts entnommen werden. Dabei hat das Gerät einen Puffer für die vom Master empfangenen Prozessdaten und einen Puffer für die zu sendenden Daten. Die Puffergrösse wird dabei bei Kommunikationsbeginn einmalig festgelegt und liegt zwischen 0 und 32 Bytes.

Gerätedaten (Configuration / Maintainance Data) des IO-Link Devices werden einem festgelegten Adressbereich zugeordnet. Dabei ist der Direct Parameter Page 1 der Adressbereich 0x00 bis 0x0F zugeordnet und der Direct Parameter Page 2 der Adressbereich von 0x10 bis 0x1F.

Auch für Events ist ein 32 Byte grosser Puffer vorgesehen.

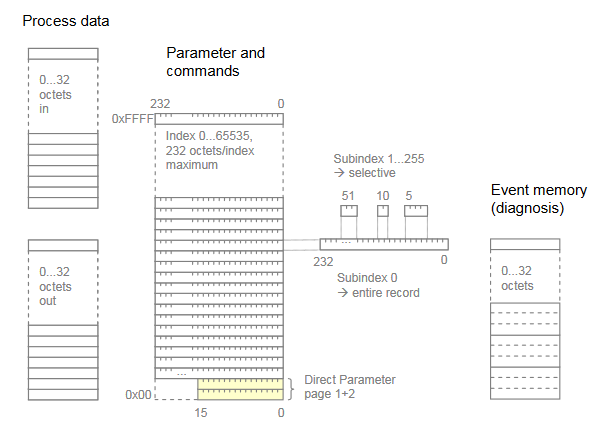


Abbildung 5 - Datenspeicherung bei IO-Link [2]

Der Inhalt jeder Speicherstellen der Direct Parameter Page 1 ist festgelegt. Beispielsweise ist an der Speicherstelle 0x04 die RevisionsID hinterlegt und an der Speicherstelle 0x06 die Anzahl vom Master versendeten Bytes etc. Eine detaillierte Auflistung der Inhalte der Direct Parameter Page 1 kann dem Anhang B entnommen werden.   
Der Inhalt der Speicherstellen der Direct Parameter Page 2 und allen anderen Speicherstellen bis 0xFFFF kann vom Hersteller der IO-Link Devices selber bestimmt werden. Beispielsweise kann ein Hersteller an die erste Speicherstelle der Direct Parameter Page 2 (Index 0x10) den Status des Geräts implementieren, liest aus dem Index 0x11 Befehle aus usw. Jedem Hersteller ist die Implementierung freigestellt, deshalb müssen bei der Verwendung eines Geräts die Informationen aus dem Datenblatt bezogen werden.

Die Speicherstellen ausserhalb der Direct Parameter Pages werden ISDU genannt und umfassen die Speicherstellen 0x20 bis 0xFFFF.

### M-Sequenz

Die Übertragung der zyklischen und azyklischen Daten werden mittels einer sogenannten M-Sequenz vorgenommen. Je nach Anzahl zu empfangener und übermittelnder Bytes an Prozessdaten wird ein anderer M-Sequenztyp für die Einbettung der Daten gewählt. Die Auswahl an M-Sequenz Typen ist aus Abbildung 6 ersichtlich.

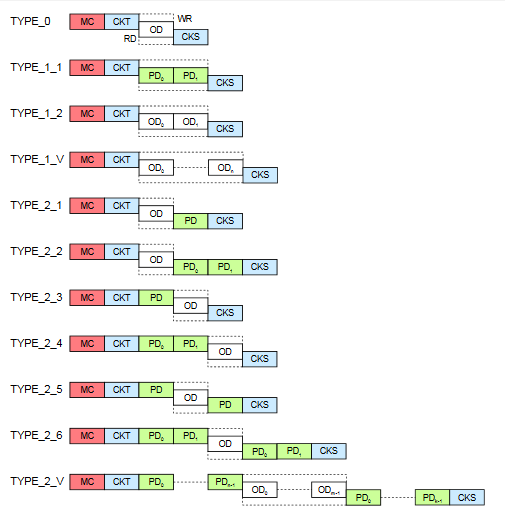


Abbildung 6 - M-Sequenz Typen [2]

In der Abbildung leicht nach oben verschoben sind die Oktette, die vom Master zum Slave gesendet werden. Leicht nach unten verschoben sind die Oktette, mit denen der Slave dem Master antwortet.

Die Kommunikation zwischen IO-Link Master und Slave besteht wie oben ersichtlich aus einem Oktett M-Sequenz-Control (MC), gefolgt von einem Oktett Check/Type (CKT) und der gewünschte Anzahl Prozessdaten (PD) und On-Request Daten (OD). Das letzte Oktett (CKS) repräsentiert den Wertestatus und enthält eine Checksumme.

Grundsätzlich gilt, dass bis zur M-Sequenz TYPE\_2\_6 der Versand von maximal zwei Oktette an Prozessdaten in die jeweilige Richtung möglich waren. Mit dem TPYE\_2\_V kann die Grösse der Prozessdaten frei gewählt werden (zwischen 0 und 32 Oktetten). Auch ist dabei die Anzahl Oktette an On-Request Daten frei wählbar.

Im folgenden Abschnitt wird auf den Inhalt der einzelnen Oktette eingegangen. Für eine detaillierte Entschlüsselung wird an dieser Stelle jedoch auf das Dokument „IO-Link Interface Spezifikationen V1.1.2“, Anhang A.1 verwiesen [2].

M-Sequenz Control  
Beim ersten vom Master versendeten Oktett handelt es sich um das Control Byte. Dieses beinhaltet die Offset-Adresse für einen ISDU-Zugriff, den Kommunikationskanal (Prozessdaten, Direct Parameter Page Zugriff, Diagnosedaten oder ISDU Zugriff) und die Information, ob es sich um einen Lese- oder einen Schreibezugriff handelt. Die Implementierung des Bytes kann aus Abbildung 7 entnommen werden.

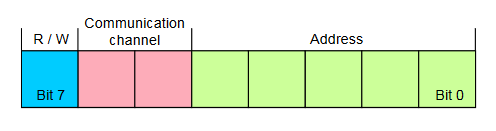


Abbildung 7 - M-Sequenz Control

M-Sequenz Checksumme / Typ  
Dieses Oktett besteht aus der Checksumme und der Information, welcher M-Sequenz Typ für die Daten verwendet wird. Dies ist wichtig, da der Slave wissen muss, wieviele Oktette an zyklischen Prozessdaten er bekommt und senden muss und wieviele Oktette für die On-Request Daten reserviert sind. Die Implementierung des Bytes kann aus Abbildung 8 entnommen werden.

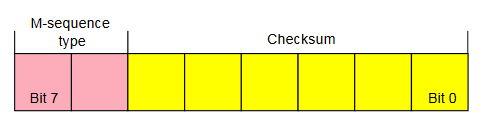


Abbildung 8 - M-Sequenz Checksumme / Typ

Prozessdaten  
Prozessdaten sind z.B. Messwerte von einem Sensor oder Ausgabewerte an einen Aktor.  
Eine bestimmte Menge an Oktetten ist bei jedem Zyklus für die Prozessdaten reserviert. Diese können entweder vom Typ „Prozess Daten Input“ sein, wenn sie an den Master gesendet werden, oder vom Typ „Prozess Daten Output“, wenn sie vom Master gesendet werden.

On-Request Daten  
Eine bestimmte Menge an Daten ist in jedem Zyklus für die On-Request Daten reserviert. On-Request Daten sind z.B. Zugriffe auf die Direct Parameter Pages oder Events. Obwohl diese Daten nur auf Anfrage (on request) versendet werden, muss trotzdem eine feste Anzahl Oktette im zyklischen Kommunikationsprotokoll dafür reserviert sein. Das IO-Link Protokoll muss echtzeitfähig sein, muss also deterministisch arbeiten. Das bedeutet, dass die Sensordatenübertragung immer gleichlang braucht, egal ob gleichzeitig noch nach On-Request Daten gefragt wird oder nur Prozessdaten übermittelt werden. Um dies zu gewährleisten, wird eine fixe Anzahl Bytes bei der zyklischen Übertragung immer für die On-Request Daten reserviert. Müssen keine On-Request Daten übertragen werden, so wird dieses Byte einfach leer gelassen (aber dennoch übertragen). Wenn On-Request Daten übertragen werden müssen, so wird es mit den gewünschten Informationen gefüllt.

Zwecks Performanceoptimierung kann die zyklisch zu übertragende Menge an On-Request Daten minimal gehalten werden (1 Oktett). Dies hat jedoch zur Folge, dass Events oder Zugriffe auf die Direct Parameter Pages über mehrere Zyklen übertragen werden müssen, da sie nicht in einem Oktett Platz haben.

In der Abbildung 6 sind die OD-Daten in der Mitte einer Sequenz zu finden, da sie weder immer vom Master gesendet noch immer vom Slave gesendet werden. Ist zum Beispiel 1 Oktett für On-Request Daten in einer M-Sequenz reserviert, so wird es vom Master gefüllt, wenn er eine Leseanfrage sendet für eine Speicherstelle der Direct Parameter Page. Bei der nächsten zyklischen Übertragung wird dieses Byte an OD-Daten vom Master leer gelassen werden und vom Slave mit dem gewünschten Inhalt der Direct Parameter Page Speicherstelle gefüllt.

M-Sequenz Wertestatus / Checksumme  
In diesem Oktett sind wiederum 6 Bit für die Checksumme der Antwort des Devices reserviert. Ein Bit wird zusätzlich benötigt für den Wertestatus (Prozessdaten gültig oder ungültig), ein weiteres Bit um ein Vorhandensein eines Events zu signalisieren. Die Implementierung des Oktetts kann aus Abbildung 9 entnommen werden.

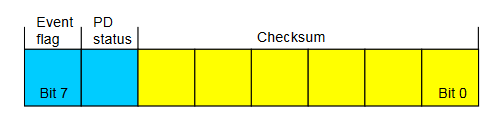


Abbildung 9 - M-Sequenz Wertestatus / Checksumme

### UART Frames

Die physikalische Datenübertragung für das IO-Link Protokoll erfolgt mittels UART Frames. Die UART Schnittstelle wird mit folgender Konfiguration betrieben:

* 1 Bit Start Bit
* 8 Bit Daten
* 1 Bit Gerades Parity Bit
* 1 Bit Stop Bit

Für 8 Bit Daten werden also total 11 Bit übertragen.

### Timing

Die minimale Zykluszeit, die vom Slave schlimmstenfalls bei Verwendung einer bestimmten M-Sequenz benötigt wird, kann berechnet werden. Sie setzt sich zusammen aus der Verzögerung zwischen zwei vom Master gesendeten UART Frames, aus der Reaktionszeit des Devices, aus der Verzögerung zwischen zwei vom Device gesendeten UART Frames und aus der Zeit für die Übertragung von einem UART Frame. Alle Timing Bedingungen hängen von der gewählten Baudrate ab (COM1, COM2 oder COM3):  
 

UART Transmission Delay vom Master  
Das Transmission Delay des Masters ist die Verzögerung zwischen dem Stopbit eines UART Frames und dem Startbit des nächsten UART Frames.  
 

UART Transmission Delay vom Device  
Das Transmission Delay des Devices ist die Verzögerung zwischen dem Stopbit eines UART Frames und dem Startbit des nächsten UART Frames.  
 

Reaktionszeit vom Device  
Die Reaktionszeit ist die Zeit zwischen dem Stopbit des Masters und dem Startbit des ersten UART Frames, das als Antwort übermittelt wird.



Übermittlung eines UART Frames  
Für die ganze Berechnung fliesst zusätzlich noch die für die Übermittlung eines UART Frames benötigte Zeit ein. Da dieses aus einem Startbit, 8 Datenbits, einem Paritybit und einem Stopbit besteht, werden total 11 Bit übertragen.



Die minimale Zykluszeit, die von einem Device benötigt wird bei Verwendung einer bestimmten M-Sequenz lässt sich also folgendermassen berechnen:

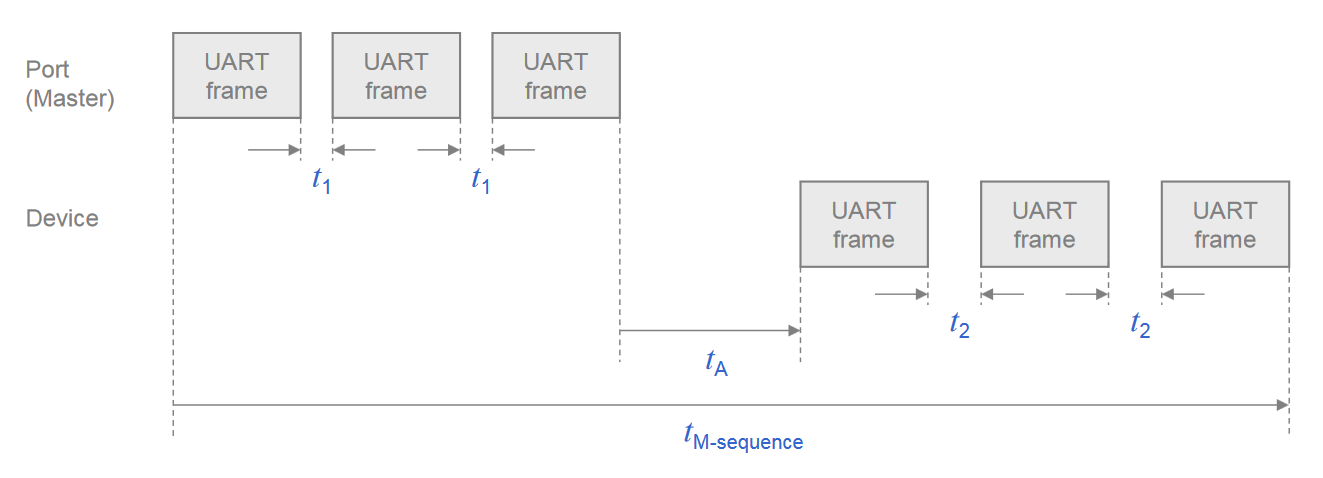


Abbildung 10 - Berechnung der minimalen Zykluszeit [2]

Um die maximal benötigte Zykluszeit eines Devices zu berechnen, wird davon ausgegangen, dass das OD-Byte vom Slave übermittelt wird, da der Slave mehr Zeit zwischen einzelnen UART Frames vergehen lassen kann.

Wenn der Master also m UART Frames und der Slave n UART Frames zu übermitteln hat, so berechnet sich die maximal benötigte Zykluszeit folgendermassen:



Das Resultat stellt aber nur die Zeit dar, die zwischen dem Versand des Startbits des ersten UART Frames und dem Erhalten des Stopbits des letzten UART Frames verfliesst. Die Verarbeitungszeit des Masters ist noch nicht miteinberechnet. Daher sollte der oben berechnete Wert noch aufgerundet werden. Das Aufrunden der Zykluszeit ist sowieso notwendig, da die minimale Zykluszeit nur in 100 µs Schritten hinterlegt werden kann.

Der dann resultierende Wert hinterlegt der Device Hersteller in der Direct Parameter Page 1 beim Index 0x02 (MinCycleTime).

## IO-Link Protokoll Stack

In diesem Kapitel werden der Kommunikationsservice und das IO-Link Protokoll genauer betrachtet.

### ISO/OSI Modell

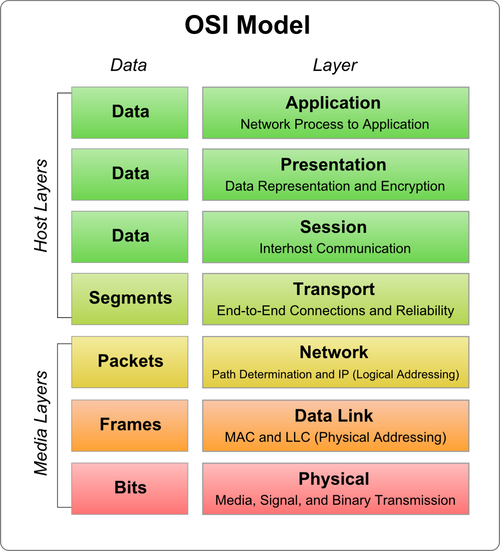
Das IO-Link Protokoll stützt sich ab auf das ISO/OSI Modell. Nach dem ISO/OSI Modell wird der Datenaustausch immer in sieben Layer unterteilt, die jeweils einen Teil der Kommunikation übernehmen.

Abbildung 11 - ISO/OSI Modell [5]

Das IO-Link Protokoll benötigt nicht alle diese Layer, sondern baut nur auf drei davon auf, dem Physical Layer, dem Data Link Layer und dem Application Layer. Im folgenden werden die Funktionen dieser drei Layer erläutert.

Physical Layer  
Der 1. Layer im ISO/OSI Modell ist nur für die Bitübertragung verantwortlich. Die physikalische Layer kümmert sich um den Spannungspegel, den Übertragungskanal (Kabel, wireless etc) und um das physikalische Übertragungsprotokoll (z.B. UART).

Data Link Layer  
Der 2. Layer im ISO/OSI Modell ist für Punkt-Punkt Übertragung verantwortlich. Er schaut, dass die angekommenen Bits stimmen indem er Paritybits oder Prüfsumme anschaut. Der Data Link Layer ist aber nicht nur für die fehlerlose Übertragung verantwortlich, sondern ist auch das Interface zwischen der physikalischen Datenübertragung und der übergeordneten Datenverarbeitung. Der Data Link Layer nimmt Pakete vom übergeordneten Layer an, verkleinert diese in Frames, fügt eine Prüfsumme an und reicht sie an den Physical Layer weiter zur Übertragung.

Application Layer  
Der 7. Und letzte Layer im ISO/OSI Modell verbindet das Protokoll mit der Usersoftware. Der Application Layer ist die Schnittstelle zwischen den Daten und dem Nutzer (bzw. der Applikation).

Abbildung 12 zeigt die Implementierung der Layer im IO-Link Protokoll. Auf die Aufgabe der einzelnen Funktionen wird hier nicht genauer eingegangen. Für detailliertere Informationen kann das Dokument „IO-Link Interface and System Specification V1.1.2“ [2] der IO-Link Community konsultiert werden.

Wie in der Abbildung ersichtlich, ist das IO-Link Protokoll ist sehr umfänglich und eine korrekte Implementierung aller Vorgaben deshalb zeitaufwändig. Dies bestätigt auch ein Ingenieur der Firma HMT, welcher bei der Implementierung eines IO-Link Stacks mitverantwortlich war. Das IO-Link Protokoll muss zuerst bis ins Detail verstanden werden und Timingvorgaben müssen nachgerechnet werden, bevor mit dem eigentlichen Codieren begonnen werden kann.

Da eine eigene korrekte Implementierung des IO-Link Stacks den Rahmen dieser Diplomarbeit sprengen würde, wird auf vorhandenen Code zurückgegriffen. Mehr dazu im Kapitel 5.3 Auswahl des IO-Link Stacks.

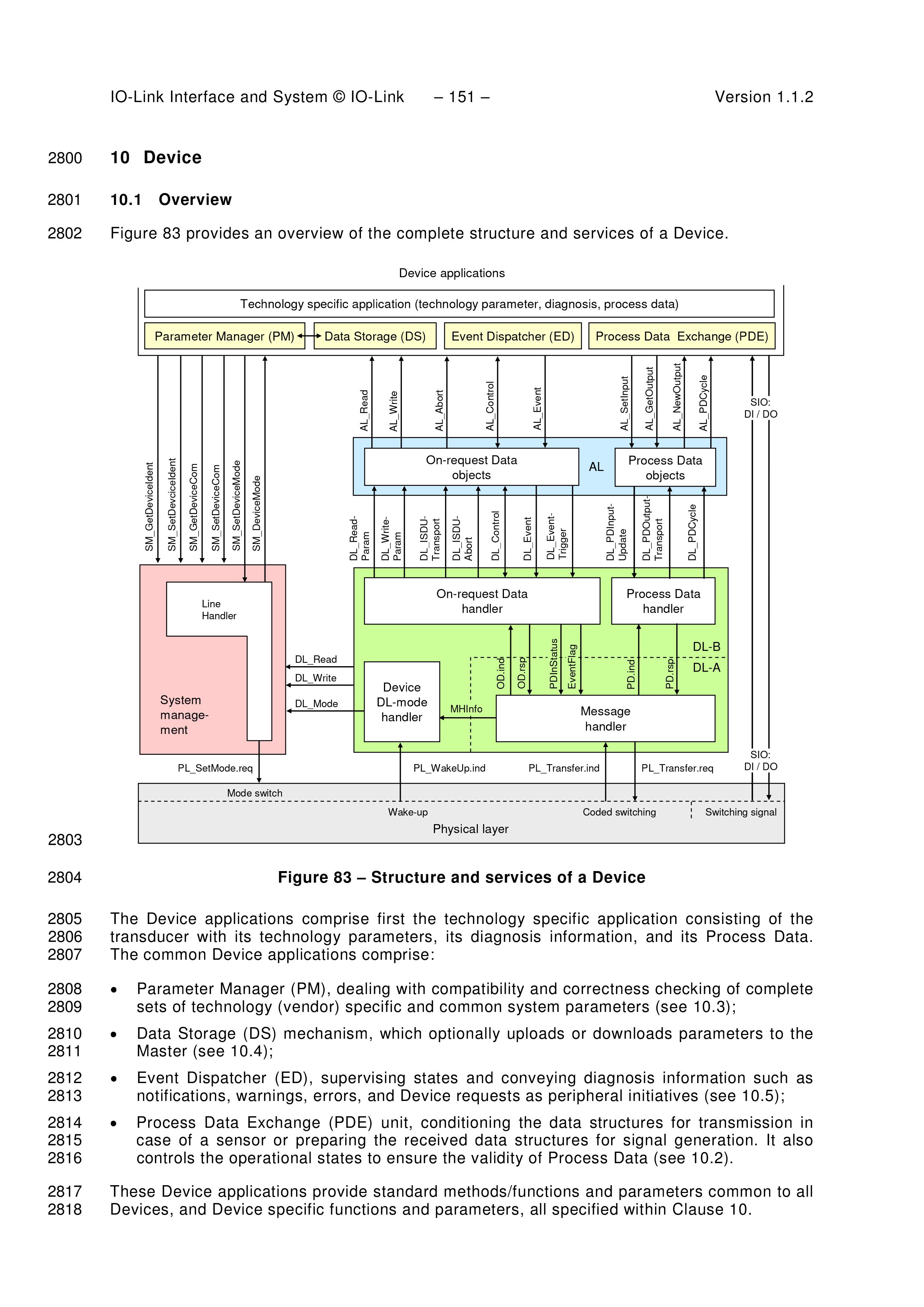


Abbildung 12 - Struktur und Service eines IO-Link Devices [2]

### ISO/OSI Modell im IO-Link Protokoll

Physical Layer  
Bei IO-Link ist der physikalische Layer sehr robust, er arbeitet mit einem 24V Pegel. Die Bits werden mit einer halbduplexen UART Verbindung zwischen Master und Slave übertragen. Das Konzept einer halbduplexen Kommunikation kann aus Abbildung 13 entnommen werden.

Transmitter

Receiver

**IO-Link Master**

Transmitter

Receiver

**IO-Link Slave**

Abbildung 13 - Halbduplexe Kommunikation

Halbduplex bedeutet, dass es zwar nur eine Verbindungsleitung zwischen zwei Komponenten gibt, diese aber in beide Richtungen Daten übermitteln kann. Die eine Komponente arbeitet dabei jeweils als Sender und die andere als Empfänger. Dabei können die Rollen jederzeit vertauscht werden.

Data Link Layer  
Der Data Link Layer unterscheidet zwischen zyklischen und azyklischen Daten. Der Message Handler ist dabei die Schnittstelle für einkommende Protokolle sofern die Kommunikation schon besteht. Zyklische Prozessdaten werden dann dem Process Data Handler weitergereicht und azyklische Daten werden dem On-Request Data Handler weitergereicht zur Bearbeitung. Bei einem Wake-Up Impuls übernimmt der Device DL-Mode Händler die Übermittlung der Initialisierungsdaten.

Application Layer  
Dem Anwender, welcher das IO-Link Modul steuert, steht der Application Layer und die darauf implementierten Funktionen zur Verfügung um Daten abzufragen oder zu übermitteln.

# Sensoren

Das Ziel dieser Arbeit ist es, die Prozessdaten zweier Sensoren via IO-Link an die übergeordnete SPS Steuerung weiterzuleiten. Bei den beiden Sensoren handelt es sich um einen Pyrometer und einen Leistungsmesser. Die Sensorwerte werden jeweils seriell übertragen mit SPI respektive I2C. Beide Sensoren arbeiten mit 3.3V.

## I2C Schnittstelle

Die Daten der Leistungsmessung werden mit I2C übertragen. Dabei wird die Leistung mit einer Photodiode gemessen, mit einem 10 Bit AD-Wandler abgetastet und im selben Baustein in ein I2C Protokoll verpackt. Dabei ist der Sensor als Slave implementiert und benötigt einen Strom von ca. 10mA. Die Implementation der Kommunikation kann aus Tabelle 1 entnommen werden.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Grösse** | **Wert** |
| I2C Slave Adresse | 7 Bit | 0b 1001101 |
| Geschwindigkeit | Fast mode | 300 kbit/s |
| Sensordaten | 10 Bit, verpackt in 2 Bytes | 1. Byte: 0 0 0 0 b9 b8 b7 b6 2. Byte: b5 b4 b3 b2 b1 b0 x x |
| Conversion Mode | Single conversion oder continuous conversion | Single conversion mode |

Tabelle 1 - I2C Protokoll Spezifikationen

Beim Single Conversion Mode wird der Sensorwert einmalig abgefragt und danach wird die Clockleitung vom Master wieder losgelassen. Der Sensor könnte auch im Continuous Mode betrieben werden indem der Slave ausgewählt wird mit der Adresse und danach kontinuierlich ein Clock generiert wird. Der AD-Wandler wird dann kontinuierlich wandeln und den Sensorwert mit I2C weiterleiten, ohne dass dabei die Slave Adresse erneut versendet werden muss oder die Kommunikation abgesetzt wird.

Weitere Informationen über die I2C Implementation beim Slave können aus dem Datenblatt der Komponente MCP3021 von Microchip entnommen werden. Der MCP3021 der verwendete AD-Wandler, der den digitalisierten Wert über I2C weiterversendet.

In der momentanen Betriebsart wird zuerst eine 7 Bit Adresse gefolgt von einem Readbit versendet und danach kommen zweimal 8 Bit Daten zurück. Jedes Byte wird bestätigt, es werden also insgesamt dreimal 9 Bit versendet, also total 27 Bit pro Sensorabfrage. Bei einer Geschwindigkeit von 300 kbit/s dauert diese Übertragung 90 µs.

## SPI Schnittstelle

Die Daten des Pyrometers werden über eine SPI Schnittstelle versendet. Dabei wird die Spannung über einer Photodiode gemessen und in einem Mikrocontroller verarbeitet. Dieser Mikrocontroller verfügt über eine SPI Schnittstelle, über welche die Daten abgefragt, kalibriert oder auch gesetzt werden können. SPI ist eine Vollduplexe serielle Schnittstelle. Da SPI nicht, anders als das I2C Interface, definierte Logikpegel hat, kann sie verschieden parametriert werden. Die Konfiguration der sensorseitigen SPI Konfiguration kann dem folgenden Abschnitt entnommen werden.  
Der Sensor zieht einen Strom von ca. 20 mA.

### SPI Konfiguration

Die Aufsetzung der SPI Schnittstelle kann aus Abbildung 14 entnommen werden.

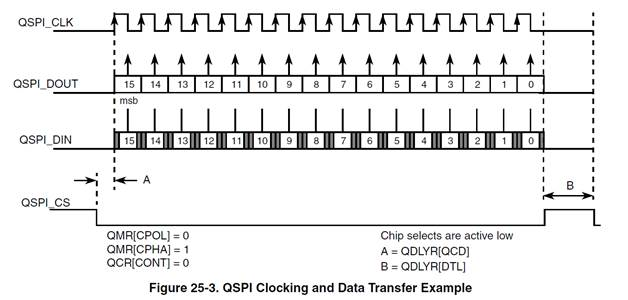


Abbildung 14 - SPI Konfiguration

Ein Datenpaket entspricht jeweils 16 Bit. Das MSB wird zuerst übermittelt. Die Leitung Slave Select ist low-aktiv und muss drei Systemclockzyklen vor der Datenübertragung auf 0V gezogen werden. Zwischen zwei Slave Selects müssen mindestens 600 Clockzyklen liegen um eine korrekte Datenverarbeitung im Mikrocontroller auf Sensorseite zu gewährleisten.



Der Slave unterstützt Baudraten bis 1MHz.

Datenleitungen und Clockleitung sind idle-low. Der Pegel der Datenleitung wird bei der steigenden Clockflanke gesetzt und bei fallender Clockflanke ausgelesen.

### SPI Kommandos

Das Protokoll, das auf Seiten des Sensors im Mikrocontroller implementiert ist, sieht folgendermassen aus:

|  |  |  |
| --- | --- | --- |
| **Kommandoname** | **Wert** | **Bedeutung** |
| COMMAND\_INQUIRY | 0x3100 | Dieses Kommando ist zur Überprüfung einer korrekten Aufsetzung SPI Schnittstelle. Die Antwort des SPI Slaves auf dieses Kommando muss dann lauten: 0xA005 |
|  |  |  |

Tabelle 2 - SPI Protokoll Spezifikationen

Auf Sensorseite sind noch viele weitere Kommandos implementiert, jedoch nur dürftig dokumentiert. Für Evaluationszwecke soll das oben erwähnte Kommando COMMAND\_INQUIRY verwendet werden, sodass wenigstens sichergestellt werden kann, dass die Kommunikation mit SPI funktioniert.

# Hardware

Um einen IO-Link Slave zu implementieren, werden verschiedene Komponenten benötigt. Zum einen braucht es ein Mikrocontroller um die Daten, welche von den beiden Sensoren via SPI resp. I2C weitergeleitet werden, abzufragen und ins IO-Link Protokoll einzubetten. Zum anderen wird ebenfalls ein Baustein benötigt, welcher die Schnittstelle zwischen dem mit 24V arbeitenden IO-Link Master und dem mit 3.3V arbeitenden Mikrocontroller herstellt. Der Aufbau der Hardware kann der Abbildung 15 entnommen werden.

Pyrometer

Leistungs-messung

Mikrocontroller

PHY

IO-Link Master

Demoboard

3.3V

24V

3.3V

3.3V

Abbildung 15 - Hardware Aufbau

Als Baustein zwischen Mikrocontroller und IO-Link Master wird ein sogenanntes PHY eingesetzt. Dieses hat auf der einen Seite einen 24V und einen 0V Eingang und auf der anderen Seite einen 3.3V oder/und 5V Ausgang.

Anstelle eines PHYs kann auch ein einfacher Levelkonverter eingesetzt werden, der den 3.3V Spannungspegel auf 24V bringt. Aber ein PHY übernimmt meist noch zusätzliche Aufgaben wie der Schutz vor elektrischen Störungen im Spannungspegel, der Schutz vor Überstrom und stellt eine Clocksynchronisation mit dem Master her, da die Abweichung der Clockfrequenz bei COM3 zwischen Master und Slave nicht mehr als ±0.1 % betragen darf (siehe Tabelle 8 des Dokuments „IO-Link Interface and System Specification V1.1.2“ [2] ).

## Auswahl des IO-Link Demoboardes

Für die Auswahl des Demoboardes waren folgende Kriterien relevant:

* Formfaktor das Demoboard soll möglichst kompakt sein, um später auch in den   
   beschränkten Platz im Laserschweissgerät zu passen
* Headerpins Der I2C Sensor benötigt vier Anschlüsse: SCK, SDL, Vcc und GND  
   Der SPI Sensor benötigt sechs Anschlüsse: MOSI, MISO, nSS, CLK,  
   GND und Vcc. Insgesamt müssen also 6 GPIO Pins zur Verfü-  
   gung stehen und noch 4 Pins zur Energieversorgung.
* Schnittstellen Es sollen eine I2C und eine SPI Schnittstelle zur Verfügung stehen
* IO-Link Stack Um die Implementierungsarbeit zu erleichtern, soll ein bestehender  
   IO-Link Stack verwendet werden.
* Lieferfrist Das Demoboard sollte innerhalb von zwei Wochen lieferbar sein

Die Tabelle 3 listet Demoboards, die in Betracht gezogen wurden.

Da die Firma HMT ein Demoboard gratis zur Verfügung stellt und einen IO-Link Stack inkl. Demoprogramm dazuliefert, wird mit ihrem Demoboard gearbeitet. Es handelt sich dabei um das Board mit der Bezeichnung TM96.1 Genie Explorer I Var B.

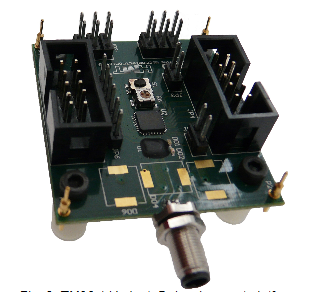


Abbildung 16 - TM96.1 Genie Explorer I Var B



Tabelle 3 - Hardwarevorschläge

Das Demoboard ist in Abbildung 16 ersichtlich. Es handelt sich um ein kleines und übersichtliches Evalboard von der Grösse 37mm x 37mm.

### PHY

Die Firma HMT hat auf dem Genie Explorer TM96.1 Var B ihr eigenes PHY eingesetzt. Andere Hersteller benutzen zum Teil einen einfachen Levelkonverter für diese Aufgabe, jedoch kann bei ressourcekritischen Anwendungen ein PHY nötig sein.

Das auf dem Demoboard eingesetzte PHY von HMT (HMT7742) übernimmt nämlich nicht nur den Schutz von elektrischen Störungen und Überstrom, sondern kümmert sich auch um die Kommunikation mit dem Master. IO-Link Master und IO-Link Slave kommunizieren über das UART Protokoll. Dies ist sehr ressourcenintensiv, da bei der Implementierung die Daten immer nach einem Byte entgegengenommen und das UART Empfangsregister ausgelesen werden muss. UART ist konzipiert für ein 8 Bit Datenpaket. SPI hingegen darf mehr als 8 Bit aufs mal übertragen, bevor das Empfangsregister ausgelesen werden muss. Eine SPI Schnittstelle kann für 8 Bit Daten Konfiguriert werden oder für 16 Bit oder noch mehr (sofern es der Mikrocontroller unterstützt).

Indem das PHY HMT7742 also eine SPI Schnittstelle zum Mikrocontroller zulässt, können Ressourcen gespart werden. Jetzt kann der Multi Byte Modus gewählt werden auf dem PHY, sodass die Daten vom Master zuerst vollkommen empfangen werden auf dem PHY und dann als Gesamtpaket an den Mikrocontroller übermittelt werden. Der Single Byte Modus ist aber trotzdem noch wählbar, wenn der Mikrocontroller jedes Byte empfangen möchte sofort nachdem es beim PHY angekommen ist und nicht warten möchte, bis die ganze Sequenz vom Master beim PHY angekommen ist.

HMT7742 übernimmt ebenfalls die Checksummenberechnung und die Clocksynronisation mit dem Master.

## Programmer/Debugger

Der AtMega328P Mikrocontroller, der auf dem Demoboard von HMT verbaut ist, kann mit einem Programmer von Atmel programmiert werden. Dabei gibt es eine Vielzahl von ICDs, welche die Aufgabe erfüllen können. Für Evaluationszwecke reicht eine einfache und kostengünstige Lösung, die trotzdem Single Wire Debugging unterstützt. Auf der Webseite von Atmel kann eine Auflistung an kompatiblen Programmern für den verwendeten Mikrocontroller AtMega328 gefunden werden.



Tabelle 4 - Debugger Auswahl

Aus Tabelle 4 können die von Atmel empfohlenen Programmer entnommen werden. Der Atmel-ICE wurde gekauft, da er schnell lieferbar und als Gesamtpaket mit Kabel und Gehäuse verfügbar ist. Der AVR Dragon wäre eine gute Alternative gewesen, doch nicht als Gesamtpaket mit Programmierkabel und Gehäuse schnell in die Schweiz lieferbar.

## Auswahl des IO-Link Stacks

Da es sich für die Firma Leister noch um eine Evaluation des IO-Link Interfaces handelt, soll noch ein kostenloser IO-Link Stack implementiert werden. Verschiedene Anbieter verkaufen Lizenzen für ihren IO-Link Stack, auf die auch bei einer Evaluation nicht verzichtet wird.

Ein kostenloser IO-Link Stack kann jedoch über folgende Quellen bezogen werden:

HMT Der Physical Layer des Stacks ist schon implementiert für das HMT Demoboard  
IQ2 Application Layer und Physical Layer müssen noch implementiert werden

Die meisten IO-Link Industrieanwendungen haben den Stack von TMG implementiert. Die Lizenz für diesen Stack kostet jedoch CHF 5000, wobei auf Anfrage die Antwort kam, dass man sicherlich eine günstige Lösung für Demozwecke finden könnte.

Da HMT kostenlos ein Development Board zur Verfügung stellt und darauf ein IO-Link Stack implementiert ist inkl. Demoprogramm, wird aber einfachheitshalber der Stack von HMT verwendet. Ausserdem muss bei HMT auch für Industrieimplementierungen keine Lizenz für ihren Stack gekauft werden, sofern ihr PHY verwendet wird.

## IO-Link Master

Die Firma Leister arbeitet mit einer SPS Steuerung von Beckhoff im Laserschweissgerät. Da bietet sich eine Verwendung eines IO-Link Masters von Beckhoff an. Die Klemme EL6224 wurde ausgewählt. Sie lässt sich an die bestehende Steuerung anfügen und kann im TwinCat (dem Programmiertool für Beckhoff SPS Steuerungen) konfiguriert und später auch programmiert werden.

# Softwareimplementierung

In diesem Kapitel wird zuerst die Inbetriebnahme der Demoapp besprochen und danach die Modifikationen am Demoprogramm, sodass dieses dann die Aufgabenstellung erfüllt.

## Demoapp

Für Genie Development Board von HMT ist eine Demoapplikation erhältlich. Diese Demoapplikation kann einen von drei Werten mit der IO-Link Schnittstelle übermitteln:

* Button Die Stellung des Buttons (0 oder 1) wird übermittelt
* Potentiometer Der Wert (0..255) des Potentiometers auf dem Board wird  
   übermittelt
* Sägezahn Der Wert eines intern generierten Sägezahns wird übermittelt

Im Kapitel 3.33.3.2 wurde auf den Unterschied zwischen zyklischen und azyklischen Daten eingegangen und im Zusammenhang damit auch auf die Direct Parameter Pages.  
In der Demoapp wurden auf der Direct Parameter Page 2 drei Einträge implementiert. Wenn an die Adresse 0x10 ein Wert geschrieben wird, so kann derselbe Wert an der Adresse 0x11 wieder ausgelesen werden (die Firma HMT nennt dies in ihrer Dokumentation „Mirror Output“ und „Mirror Input“). Der Wert an der Adresse 0x12 bestimmt, welcher der oben aufgelisteten drei Werte zyklisch übermittelt wird. Wird an die Adresse 0x12 eine 0 geschrieben, so wird zyklisch der Status des Buttons übertragen. Wird an die Adresse 0x12 eine 1 geschrieben, so wird zyklisch der Status des Potentiometers übertragen und wird eine 2 geschrieben, so übermittelt das IO-Link Device den Sägezahn. Eine Übersicht kann aus der Tabelle 5 entnommen werden, welche direkt aus dem Datenblatt der Demoapp kopiert wurde.

Beim Schreiben in die Tabelle handelt es sich um einen azyklischen Zugriff. Falls der Wert bei der Adresse 0x12 verändert wird, so ändert auch die Interpretation der zyklisch übertragenen Prozessdaten.

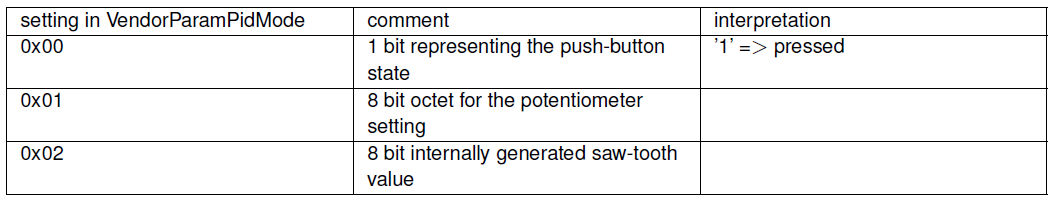
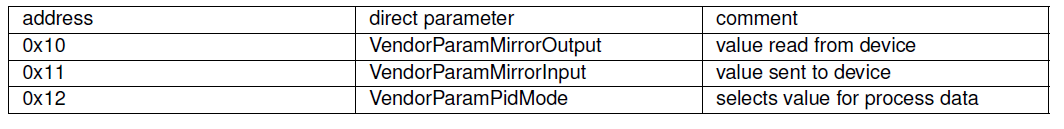


Tabelle 5 - Direct Parameter Page 2 der Demoapp

Die Implementation der Direct Parameter Page 1 ist vorgeschrieben und enthält Informationen zum IO-Link Device, z.B. VendorID, DeviceID, Revisionsnummer, Baudrate etc. Auf diese Daten kann wiederum azyklisch zugegriffen werden, jedoch sind die meisten davon nur lesbar, der Benutzer kann sie nicht überschreiben vom IO-Link Master aus. Welche Zugriffesrechte auf welche Daten möglich sind, kann aus der Tabelle im Anhang B entnommen werden.

Da die azyklischen Daten von einem separaten Handler im Data Link Layer verarbeitet werden, beeinflussen sie die Verarbeitung der zyklischen Prozessdaten (Sensorwerte) nicht. Dies ist ersichtlich in der Abbildung 12, wo es zwei separate Handler gibt für On-Request Daten und zyklische Daten. Auch bei der Übertragung macht das Vorhandensein oder Nichtvorhandensein von azyklischen Daten keinen Unterschied, da eine fixe Anzahl Bytes sowieso für die On-Request Daten vorgesehen ist und diese dann einfach gefüllt btw. Leergelassen werden kann (siehe Kapitel 3.3.3 zum Thema On-Request Daten).

### Inbetriebnahme der Hardware mit Demoapp

In der Demoapp müssen noch zwei Einstellungen vorgenommen werden, bevor sie auf den Mikrocontroller geladen werden kann.

Stackmode wählen  
Ein Merkmal des PHY Bausteines der Firma HMT ist, dass es einen Teil des IO-Link Protokolls selber übernehmen kann. So kann die Demoapp in drei verschiedenen Modi laufen:

* Single Byte Sobald ein ganzes Oktett beim PHY angekommen ist, wird dieses an   
   den Mikrocontroller weitergeleitet.
* Multi Byte Vom Master empfangene Daten werden erst vom PHY an den Mikro-   
   controller weitergeleitet, wenn alle zu empfangenen Daten ange-   
   kommen sind.
* Transparent Checksumme wird im Mikrocontroller berechnet und nicht im PHY,   
   PHY wird nur als Spannungslevelkonverter verwendet.

Um einen Modus auszuwählen, muss die entsprechende Headerdatei eingebunden werden in der Demoapp. Per Default ist der Multi Byte Mode implementiert. Die Einbindung der korrekten Headerdatei erfolgt im AtmelStudio, indem die Einstellungen des Projekts im Solution Explorer verändert werden. Dabei kann im Konfigurationsmanager die aktive Konfiguration auf Single Byte, Multi Byte oder Transparent Mode gesetzt werden.

Baudrate  
Die Baudrate kann dann im einzubindenden Header gewählt werden, indem der Wert einer vordefinierten Konstante auf die gewünschte Geschwindigkeit geändert wird.

Jetzt ist das Demoprogramm aufgesetzt und kann auf das Demoboard geladen werden.

### Prozessdatengrösse

In der Demoapplikation ist der M-Sequenztyp Type\_2\_5 implementiert. Es werden 1 Oktett Prozessdaten vom Master an den Slave übermittelt und ein Oktett Prozessdaten vom Slave an den Master. Zusätzlich ist noch ein Oktett für die On-Request Daten reserviert.

## Implementierung der Sensordaten in die Demoapp

Als nächster Schritt gilt es, die Übertragung der Sensordaten in einer eigenen Demoapp zu implementieren. Einfachheitshalber wird dafür nicht von vorne begonnen sondern die neuen Funktionalitäten werden der Demoapp nur angehängt.

### Prozessdatengrösse

Als erstes gilt es, die implementierte M-Sequenz so abzuändern, dass neu vier Oktette Prozessdaten an den Master übertragen werden innerhalb eines Zyklus und nicht nur ein Oktett. Um die Zykluszeit nicht unnötig zu verlängern, wird die Anzahl der Oktette, die zyklisch vom Master an das Device übermittelt werden (Process Data Output) auf null gesetzt.

PD\_IN\_SIZE Anzahl Oktette, die vom Slave zum Master übertragen werden

PD\_OUT\_SIZE Anzahl Oktette, die vom Master zum Slave übertragen werden

Diese beiden Variablen können im Header des gewählten Stacks (z.B. StackSingleByte.h) verändert werden. Wenn eine von 1 verschiedene Zahl verwendet wird, so muss die M-Sequenz Kompatibilität auch geändert werden. Diese M-Sequenz Kompatibilität ist angegeben in der Direct Parameter Page 1 im Adressfeld 0x03. Eine Übersicht von möglichen M-Sequenz Kompatibilitäten kann aus Tabelle 6 und Tabelle 7 gewonnen werden. Die M-Sequenz Kompatibilität lässt den Master wissen, mit welcher M-Sequenz der Slave kommuniziert. Ein Slave kann immer nur eine M-Sequenz unterstützen. Dank der M-Sequenz Kompatibilität weiss der Master, wie er mit dem Device kommunizieren muss.

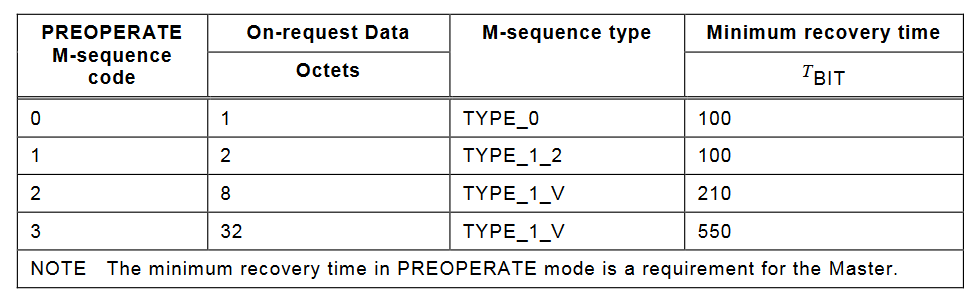
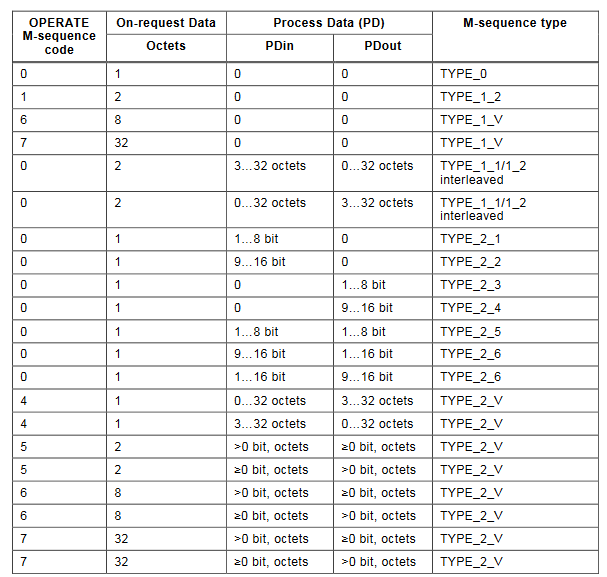


Tabelle 6 - M-Sequenz Preoperate

Tabelle 7 - M-Sequenz Operate



Die Tabelle 6 listet die möglichen M-Sequenzen auf, die im Preoperate Modus (siehe Kapitel 3.2.1 für eine Übersicht der verschiedenen Betriebsmodi) eines Ports zur Verfügung stehen. Da in diesem Modus nur On-Request Daten ausgetauscht werden und noch kein zyklischer Datenaustausch stattfindet, sind nur On-Request Daten implementiert in den verfügbaren M-Sequenzen im Preoperate Modus.

Sobald sich ein Port im Operate Modus befindet, können auch zyklische Prozessdaten ausgetauscht werden. In diesem Falle gibt es viele Möglichkeiten, wie die Prozessdaten gestaltet werden können. Je nach Anzahl Oktette, die übermittelt werden sollen, muss die passende M-Sequenz gewählt werden (eine Übersicht an möglichen M-Sequenzen im Operate Modus kann aus Tabelle 7 entnommen werden).

In der mitgelieferten Demoapp werden ein Oktett Prozessdaten vom Master an den Slave gesendet und ein Oktett Prozessdaten vom Slave an den Master. Für den Preoperate Modus ist dabei die M-Sequenz TYPE\_0 implementiert und für den Operate Modus die M-Sequenz TYPE\_2\_5. Dies entspricht dem M-Sequenz Code 0 im Preoperate sowie im Operate Modus.

Um die Übertragung von der Leistungsmessung und dem Pyrometer in einer Sequenz zu gewährleisten, werden aber mehr als ein Oktett zyklische Prozessdaten benötigt. Beide Sensoren liefern je 2 Byte Sensorwerte und so werden vier Oktette Prozessdaten benötigt, um beide Messwerte zusammen zu überliefern. Aus Tabelle 7 kann entnommen werden, dass für vier Oktette PD die M-Sequenz TYPE\_2\_V benötigt wird (alle Typen die mit „interleaved“ gekennzeichnet sind, werden nicht unterstützt vom IO-Link Stack der Firma HMT). Die Variable „M-Sequenz Capability“, die in der Direct Parameter Page 1 enthalten ist, muss also auf den M-Sequenz Code 4 gesetzt werden im Operate Modus und kann auf M-Sequenz Code 0 verbleiben im Preoperate Modus. Diese Variablen können im Header des verwendeten Stacks verändert werden. So muss jetzt im singleStackMode.h die Konstante MSEQ\_CAPABILITY auf MSEQCAP\_OP\_CODE\_4 gesetzt werden.

### Übertragungsgeschwindigkeit

Die minimal benötigte Sensordatenrate von 2 kHz auf dem Master ist nur erreichbar, wenn die IO-Link Schnittstelle mit der schnellsten Übertragungsgeschwindigkeit arbeitet. Die Applikation wird also für 230.4 kBaud (dies entspricht COM3) implementiert

### Single Byte Mode

Unter dem Punkt Stackmodus im Kapitel 6.1.1 wurden die verschiedenen Stack Modi beschrieben: Single Byte Modus, Multi Byte Modus und Transparent Modus. Wenn viele Oktette vom Master an den Slave gesendet werden sollen, so ist es notwendig, den Single Byte Modus zu verwenden. Das Device hat nämlich ein beschränktes Zeitfenster, bis es eine Antwort an den Master zurück senden muss. Diese Zeit hängt von der gewählten Baudrate ab und entspricht der Zeit tA aus dem Kapitel 3.3.5 Timing. Bei 230.4 kBaud ist die Verarbeitungszeit tA maximal 43.4 µs.  
Wenn jetzt das PHY wartet, bis alle Oktette vom Master angekommen sind und diese erst dann an den Slave überträgt (Multi Byte Mode), so hat der Slave noch weniger Zeit für die Bearbeitung der Anfrage als wenn jedes Oktett einzeln nach Erhalten an den Mikrocontroller gesendet wird (Single Byte Mode). Beim Single Byte Mode muss das PHY nur noch ein Oktett an den Mikrocontroller übergeben sobald das letzte Oktett vom Master angekommen ist, bevor der Mikrocontroller mit der Verarbeitung beginnen kann. Beim Multi Byte Mode geht nach Erhalten des letzten Oktetts vom Master viel Zeit verloren für die vollständige Übertragung aller Oktetts an den Mikrocontroller.

Schon im Single Byte Mode muss darauf geachtet werden, dass die Verarbeitungszeit im Mikrocontroller kurz gehalten wird und nicht viele rechenintensive Operationen ausgeführt werden müssen. Um das Timing nicht zu gefährden, wird das PHY für diese Aufgabe im Single Byte Mode betrieben.

### Leistungsmessung mit I2C

Der Genie Explorer TM96.1 Var B ist mit einem AtMega328P bestückt. Dieser Mikrocontroller hat ein I2C Baustein integriert, der vom IO-Link Stack noch nicht verwendet wird. Die I2C Datenleitung SDA ist dabei auf Port C Pin 4 verfügbar und die I2C Clockleitung SCL ist auf Port C Pin 5 verfügbar. Diese beiden Mikrocontrollerpins sind laut Schema des Demoboards auf den Header JP4 herausgeführt. Der Sensor kann also auf einfachste Weise verdrahtet werden.

Speisung  
Die Speisung sollte nicht mit einem GPIO Pin realisiert werden, da dieser nicht genügend Strom liefern kann, ohne dass die Spannung einbricht. Diese Information kann aus den Datenblatt des AtMega328P gewonnen werden, die relevante Grafik sieht folgendermassen aus:

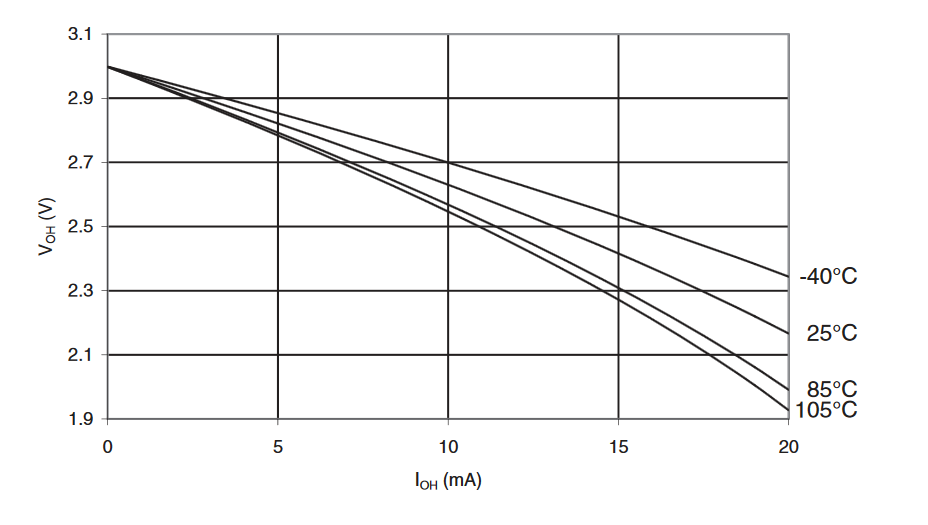


Abbildung 17 - GPIO Source Current AtMega328P

Diese Grafik ist bei einer Versorgungsspannung des Mikrocontrollers von 3.3V. Auf der X-Achse ist dabei der Strom aufgetragen, der von einem GPIO Pin bezogen wird und auf der Y-Achse die Spannung, die dieser Pin dann noch haben wird.

Der Sensor zur Leistungsmessung bezieht 10 mA, was bei einer Speisung mit GPIO Pin dazu führen würde, dass die Speisung dieses Sensors nur noch ca. 2.65 V betragen würde. Die Speisung muss also direkt vom PHY bezogen werden und darf nicht über einen Mikrocontrollerpin laufen. Diese Spannung ist vielerorts abgreifbar auf dem Demoboard und die jeweiligen Orte können dem Schema in Verbindung mit dem Layout entnommen werden.

Sensorabfrage  
Eine Abfrage des Sensors sieht in der implementierten Konfiguration aus wie in Abbildung 18 ersichtlich. Ausgelesen werden die Werte jeweils bei steigender Clockflanke. Mit der implementierten I2C Geschwindigkeit von 300 bit/s dauert die gesamte Abfrage 90 µs.



Abbildung 18 - I2C Kommunikation

Das erste Byte, das vom Mikrocontroller versendet wird, besteht aus der I2C Adresse und dem Read-Bit. Dieses Byte hat den binären Wert 1001 1010. Die Adresse ist dabei binär 1001101 und angehängt folgt das Readbit. Das 9. Bit ist das Acknowledge.

Das zweite und dritte Byte, das die Antwort des Sensors ist, enthält 10 Bit Sensordaten. In Kapitel 4.1 wurde die genaue Implementierung der I2C Schnittstelle beschrieben. Aus Abbildung 18 kann entnommen werden, dass der Sensor antwortet mit:  
 0 0 0 0 0 0 1 1 ACK 1 0 1 0 1 1 1 0 NACK  
Der 10 Bit Sensorwert beträgt hier binär:  
 0 0 1 1 1 0 1 0 1 1

### Pyrometer mit SPI

Der Genie Explorer TM96.1 Var B ist mit einem AtMega328P bestückt. Dieser Mikrocontroller hat ein SPI Baustein integriert, der vom IO-Link Stack aber schon verwendet wird für die Kommunikation mit dem PHY. Mit SPI können zwar mehrere Slaves an einer Leitung hängen, aber trotzdem wird hier nicht der integrierte SPI Baustein verwendet für die Sensordatenabfrage. Da das PHY knappe Zeitangaben hat vom Master, die es einzuhalten hat, ist eine störungsfreie Kommunikation nur garantiert, wenn kein anderes Device an derselben Kommunikationsleitung hängt. Darum wird die SPI Kommunikation mit dem Pyrometer mit Bitbanging realisiert.

Speisung  
Wie im Kapitel 6.2.5 unter der Überschrift „Speisung“ aufgeführt, kann ein GPIO Pin nicht genügend Strom liefern, um einen Sensor mit Strom zu versorgen. Da der Pyrometer rund 20mA Strom benötigt für den normalen Gebrauch, wird er wiederum direkt an die 3.3V Spannungsregelung des PHYs gehängt.

Sensorabfrage  
Das SPI Interface wird mit Pintoggling implementiert. Delays sind nicht notwendig, da der Sensor Geschwindigkeiten von bis zu 1 MHz unterstützt. Der Mikrocontroller arbeitet aber nur mit einem 8 MHz Oszillator und hat mehr als acht Instruktionen auszuführen zwischen den einzelnen Pintoggles. Darum wird der Mikrocontroller eine Togglingrate von 1 MHz gar nie erreichen können. Die implementierte Kommunikation sieht folgendermassen aus:

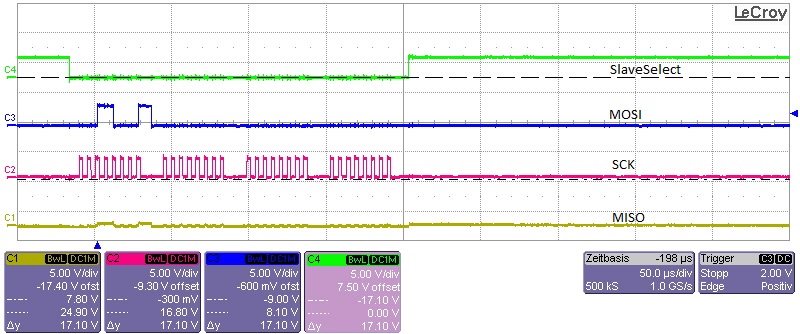


Abbildung 19 - SPI Kommunikation

Beim gesendeten Kommando handelt es sich um COMMAND\_INQUIRY, bei dem die Antwort 0xA005 lauten sollte. Wie in Abbildung 19 ersichtlich, bleibt die MISO Leitung aber low, der Sensor gibt keine Antwort. Dies liegt möglicherweise an einem zu langsamen Clocksignal. Aus Zeitgründen konnte nicht weiter nach dem Grund einer fehlenden Antwort gesucht werden. Die Verwendung des mikrocontrollerinternen SPI Interfaces wurde aber evaluiert doch Tests haben ergeben, dass das PHY diese Leitung nicht loslässt und anstelle des Pyrometers antwortet, auch wenn es nicht aktiv ist (SlaveSelect des PHYs wurde auf idle belassen, trotzdem antwortet das PHY). Deshalb wurde die Kommunikation mit dem Pyrometer dann durch Bitbanging realisiert.

Da Bitbanging wie oben erwähnt bedeutend langsamer ist als 1MHz, dauert die Kommunikation auch entsprechend länger. Rund 220 µs lang wird der Slave mittels SlaveSelect ausgewält und bleibt für die Kommunikation in Bereitschaft. Bei einer Clockfrequenz von 1 MHz würde die ganze Kommunikation nur etwa 40 µs dauern.

### Kleinste mögliche Zykluszeit

Für die Berechnung der kleinsten möglichen Zykluszeit fliessen mit ein:

* Die Zeit benötigt für die Übertragung von 2 bzw. 4 Oktetten Prozessdaten
* Die Zeit benötigt für die Sensorabfrage
* Die Verarbeitungszeit im Master

In den folgenden Abschnitten wird auf die verschiedenen Punkte eingegangen.

Übertragungszeit bei 4 Oktetten Prozessdaten  
Im Kapitel 3.2.4 wurde die Berechnung der vom Device benötigten Zykluszeit besprochen. Konkret heisst das für den TYPE\_2\_V folgendes, vorausgesetzt es wird nur ein Oktett an On-Request Daten übermittelt und das Device arbeitet mit 230.4 kBaud:



Die beiden vom Master versendeten Oktette bestehen aus dem M-Sequenz-Control (MC) Oktett und dem Oktett Check/Type (CKT).  
Die sechs vom Slave versendeten Oktette sind das On-Request Oktett, vier PD Oktette und ein CKS Oktett.

Die Übertragungszeit, die aus den oberen Berechnungen hervorgeht, ist 434 µs. Wenn also Sensordatenbereitstellung und Verarbeitungszeit des Masters in 66 µs Platz haben, so ist eine Zykluszeit von 500 µs möglich.

Übertragungszeit bei 2 Oktetten Prozessdaten  
Wird eine schnellere Übertragungszeit benötigt, so ist dies nur umsetzbar, indem jeder der beiden Sensoren ein eigener Port bekommt, also ein eigenes Device darstellt. Falls dies der Fall ist, so können die Anzahl Oktette, die für die Prozessdaten benötigt werden, von vier auf zwei gekürzt werden. Die minimale Zykluszeit berechnet sich dann folgendermassen:



Die beiden vom Master versendeten Oktette bestehen aus dem M-Sequenz-Control (MC) Oktett und dem Oktett Check/Type (CKT).  
Die vier vom Slave versendeten Oktette sind das On-Request Oktett, zwei PD Oktette und ein CKS Oktett.

Mit dem IO-Link Protokoll und maximaler unterstützter Baudrate beträgt die Übertragungszeit 312.5 µs. Diese Zeit muss noch etwas aufgerundet werden, da die Verarbeitungszeit des Masters noch nicht in die Berechnung mit einbezogen wurde. Die reine Übertragungszeit vom Senden des ersten Startbits durch den Master bis zum Empfang des letzten Stopbits vom Device wird bei zwei Oktett Sensordaten 312.5 µs benötigt. Die IO-Link Community empfiehlt in ihren IO-Link Spezifikationen eine minimale Zykluszeit von mindestens 400 µs (oder 2.5 kHz) in diesem Fall (zwei Oktette Prozessdaten, ein Oktett On-Request Daten). Die bedeutet, dass der Master innerhalb von 87.5 µs alle Berechnungen auszuführen hat und der Slave während dieser Zeit die Sensoren abfragen muss.

Bearbeitungszeit im Master  
Der IO-Link Master benötigt noch Zeit, um die erhaltenen Prozessdaten der Steuerung weiterzuleiten und mit allen verbundenen Sensoren zu kommunizieren. Je mehr Ports ein Master besetzt hat, desto grösser wird seine Bearbeitungszeit ausfallen.

Sensorabfrage  
Für die Sensorabfrage wird mindestens 90 µs benötigt, bei aktuellem Stand des Projektes sogar noch mehr. Momentan frägt die Applikation die Sensoren nacheinander ab. Zuerst verstreichen also 90 µs für die Abfrage der Leistungsmessung und dann verstreichen 220 µs für die Abfrage des Pyrometers. Insgesamt dauert eine Aktualisierung beider Sensorwerte also 310 µs. Die Abfrage des Pyrometers könnte aber noch kürzer gehalten werden indem die Clockfrequenz auf 1 MHz erhöht wird. Wenn dann die beiden Sensorabfrage immer noch nacheinander stattfinden, so dauern sie total 90 µs + 40µs = 130 µs.  
Die schnellste Lösung ist eine parallele Abfrage mit Interrupts realisiert. So dauert die gesamte Sensordatenabfrage nur so lange wie die längere Sensorabfrage der beiden, also 90 µs wenn SPI mit 1 MHz läuft, ansonsten 220 µs bei Verwendung der momentan implementierten SPI Schnittstelle.

Kleinste mögliche Zykluszeit bei 4 Oktetten an Prozessdaten  
Die kleinste mögliche Zykluszeit berechnet sich aus der Übertragungszeit und der benötigten Zeit für die Sensordatenabfrage bzw. der Verarbeitungszeit im Master.

Bei vier Oktetten an Prozessdaten beträgt die Übertragungszeit 434 µs. Die Zeit für die Sensordatenabfrage beträgt momentan 310 µs. Die Zykluszeit muss also mindestens 744 µs betragen. Da sie nur in 100 µs Schritten aufgelöst werden kann, muss beim Slave in der Direct Parameter Page 1 ein Wert von 800 µs hinterlegt werden beim Eintrag der MinCycleTime.

Diese Zeit könnte aber verkleinert werden, indem der Clock der SPI mit 1 MHz arbeitet und die Sensoren parallel abgefragt werden. So käme zur Übertragungszeit von 434 µs nur noch 90 µs für die Sensorabfrage dazu. Die Zykluszeit müsste mindestens 524 µs betragen, würde also als 600 µs implementiert werden.

Bei 4 Oktetten Prozessdaten, d.h. bei Implementierung der beiden Sensoren in einem IO-Link Slave sind 2 kHz Datenrate auf Seiten des Masters nicht möglich.

Kleinste mögliche Zykluszeit bei 2 Oktetten an Prozessdaten  
Die kleinste mögliche Zykluszeit berechnet sich wiederum aus Übertragungszeit und benötige Zeit für die Sensordatenabfrage. Dieses mal wird aber nur ein Sensor abgefragt pro Slave.

Bei zwei Oktetten an Prozessdaten beträgt die Übertragungszeit 312.5 µs. Die Zeit für die Sensordatenabfrage beträgt momentan 220 µs bzw. 90 µs. Die Zykluszeit muss also mindestens 532.5 µs bzw. 402.5 µs betragen. Da sie nur in 100 µs Schritten aufgelöst werden kann, muss beim Slave in der Direct Parameter Page 1 ein Wert von 600 µs bzw. 500 µs hinterlegt werden. Wenn nun beide Slaves je einen Port desselben Masters besetzen, so übernimmt der IO-Link Master die Zykluszeit des langsamsten verbundenen Geräts, also 600 µs.

Wiederum ist eine Optimierung möglich, indem die SPI Kommunikation mit 1 MHz arbeitet und somit nur noch 40 µs in Anspruch nimmt. Die Zeit für die Sensorabfrage würde dann 40 µs bzw. 90 µs betragen. Die Zykluszeit des jeweiligen Slaves wäre also minimal 312.5 µs + 40 µs = 352.5 µs bzw. 312.5 µs + 90 µs = 402.5 µs. Wiederum würden diese Werte in 100 µs Schritte aufgelöst und würden dann 400 µs bzw. 500 µs betragen. Wenn nun beide Slaves je einen Port desselben Masters besetzen, so übernimmt der IO-Link Master die Zykluszeit des langsamsten verbundenen Geräts, also 500 µs.  
Eine Datenrate von 2 kHz ist also hier möglich, sofern beide Sensoren ein eigenes IO-Link Gerät darstellen und die SPI Kommunikation mit 1 MHz arbeitet.

## Finale IO-Link Applikation mit Sensordaten

Das IO-Link Protokoll ist ein sehr zeitkritisches Protokoll. Da die Sensordatenabfrage ebenfalls Zeit benötigt, kann die Abfrage via SPI resp. I2C nicht erst nach Erhalt des zyklischen Datenpakets vom Master stattfinden, da sie nicht ins knappe Zeitfenster passt. Die Sensordatenabfrage muss unmittelbar nach Übermittlung der Prozessdaten an den Master (oder in diesem Fall ans PHY) beginnen. Sie darf also nur den Zeitraum zwischen den einzelnen Prozessdatenübertragungen in Anspruch nehmen. Sensordaten müssen zur Verfügung stehen, sobald der Master wieder nach Prozessdaten fragt.

Die Endlosschleife im main sieht in der Demoapp folgendermassen aus:

// enter infinite loop: processing is interrupt controlled from now on

for (;;)

{

// enter sleep until interrupt wakes us up

sleep\_mode();

// check if it's time to run user code

const Stack::Parameter\* paramWrite;

if (Stack::instance.canRunUserCode(paramWrite))

{

Stack::instance.stopInterrupt();

theApp.run(paramWrite);

Stack::instance.restartInterrupt();

}

}

In der Dokumentation der Demoapp von HMT findet man folgenden Satz über die Methode canRunUserCode: “When the stack is communicating, this function will run between two sequences, just after the completion of the device response. This function will return true once per IO-Link communication cycle, even in the absence of IO-Link communication […]”.  
Dies bedeutet, dass die Funktion canRunUserCode einmal pro Zyklus wahr ist, nämlich sofort nach erfolgreicher Übermittlung der Prozessdaten ans PHY. Innerhalb des if-Blocks werden also die zyklischen Prozessdaten aktualisiert und deshalb ist auch dies der Ort, um die Sensordaten in den dafür vorgesehenen Puffer zu schreiben. Da die Methode run(paramWrite) schon die Prozessdaten in die jeweilige Speicherstelle schreibt, müssen die Sensoren vor deren Ausführung abgefragt werden. Der ideale Ort ist also die Abfrage an erster Stelle gleich zu Beginn des if-Block einzufügen. Sie darf nicht zwischen die stopInterrupt- und restartInterrupt-Methoden kommen, da die Sensorabfrage Interruptflags benötigt und der Interrupt des PHYs nicht länger als 10 µs ausgeschaltet bleiben darf.

Im Kapitel 6.1 wurde auf die Funktionalitäten der Demoapp und die Implementierung der Direct Parameter Page 2 eingegangen. Für die im Rahmen dieser Diplomarbeit erstellten Applikation mit Sensordaten wurde ein Eintrag in der Direct Parameter Page hinzugefügt am Index 0x12. Zusätzlich zu den vorhandenen einstellbaren Prozessdaten können jetzt noch die Sensorwerte übertragen werden. Dies geschieht folgendermassen:  
Wenn an die Speicherstelle 0x12 eine 3 geschrieben wird, so werden neu Sensordaten zyklisch in den Prozessdaten übertragen. Die neue Applikation hat also folgende Einträge in der Direct Parameter Page 2 implementiert:

Tabelle 8 - Implementierte Direct Parameter Page 2



Mit dem Wert an der Speicherstelle 0x12 kann also die Interpretation des Inhalts der zyklischen Prozessdaten verändert werden. Bis anhin waren die Einträge 0..2 implementiert und neu hat auch der Wert 3 eine Bedeutung.

Die Einträge 0..2 sind für nur ein Oktett Prozessdaten konzipiert und werden auch weiterhin nur eines von vier Oktetten füllen, nämlich das Low Byte, das erste übermittelte Byte. Die anderen Oktette werden leer gelassen (aber trotzdem übertragen).

Bei der Übertragung der Sensorwerte vom Pyrometer und vom Leistungsmesser werden alle vier Oktette mit Informationen gefüllt. Der Aufbau sieht dabei folgendermassen aus:

Tabelle 9 - Aufbau der Prozessdaten bei Sensorwertübertragung



Die ersten beiden Oktette der Prozessdaten Input sind die Sensordaten der Leistungsmessung, welche über I2C abgefragt wurden. Da der Sensor einen 10 Bit Wert zurückliefert, werden Bit 11 bis Bit 16 mit Null gefüllt.

Der Sensorwert des Pyrometers ist 16 Bit lang und wird im gleichen Format als Prozessdaten weitergeleitet, wie er eintrifft. Momentan beträgt dieser Wert immer 0, da die Kommunikation noch nicht erfolgreich aufgesetzt werden konnte.

Die verwendete M-Sequenz TYPE\_2\_V hat folgenden Aufbau:

MC

CKT

OD

PDI0

PDI1

PDI2

PDI3

CKS

Abbildung 20 - Verwendete M-Sequenz

Leicht erhöht dargestellt sind die Oktette, die vom Master übermittelt werden, die anderen werden vom Slave versendet.

Die konkrete Implementierung ist aus Abbildung 21 ersichtlich.

Für eine einfachere Darstellung wurde die UART bereits decodiert. Beim untersten (gelben) Signal handelt es sich um die bidirektionale CQ-Leitung des IO-Link Protokolls. Es wird mit 24V UART gearbeitet. Wie in Abbildung 20 ersichtlich, werden die ersten beiden Oktette vom Master versendet. Diese haben hier den Wert 0xF1 und 0x94. Beim nächsten Oktett handelt es sich um die On-Reqeust Daten. Diese sind hier leer gelassen, haben also den Wert 0x00. Darauf folgen 4 Oktette Prozessdaten, die Sensorwerte beinhalten. Der Wert der Leistungsmessung ist hier 0x0081, derjenige des Pyrometers 0x0000. Der Pyrometer liefert noch keine validen Sensordaten, doch wurde die Schnittstelle und der Platzhalter im IO-Link Protokoll trotzdem implementiert um eventuelle spätere Arbeit zu erleichtern. Das letzte Byte ist wieder ein Kontrollbyte vom Slave und hat den Wert 0x14.  
Um auch die Zeit für die Sensordatenabfrage darstellen zu können, wurden auch die Clockleitung der Leistungsmessung (grün) und die Clockleitung des Pyrometers (blau) aufgezeichnet. Dank dieser Information kann die berechnete Zeit von Kapitel 6.2.6 verifiziert werden. Dort wurde eine theoretische Zykluszeit Δt von 744 µs berechnet, die dann auf 800 µs aufgerundet werden muss. Die Realität zeigt, dass noch Zeit zwischen den einzelnen Abfragen verstreicht und so total 820 µs für Übertragung und Sensorabfrage benötigt werden. Dies wurde gemessen zwischen dem ersten Bit der UART (gelb) bis zum letzten Bit der SPI (blau) und wurde rot eingezeichnet.

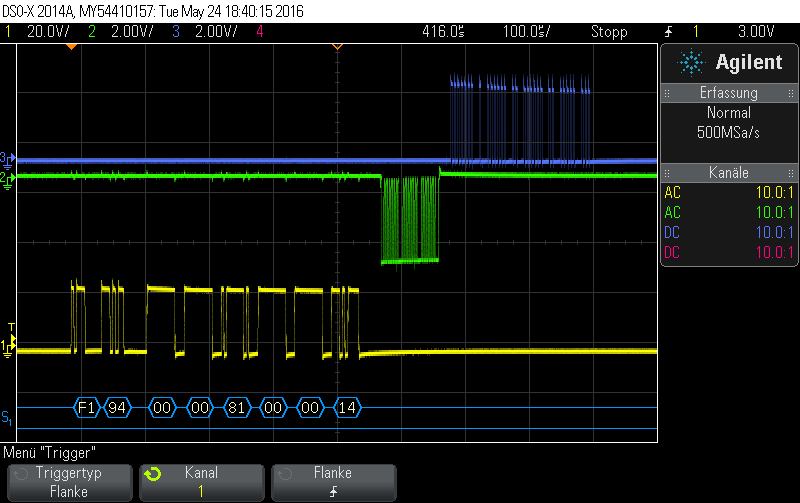


Abbildung 21 - M-Sequenz Implementiert

IO-Link Protokoll

Leistungsmessung mit I2C

Pyrometer mit SPI

Δt

Implementierung des Codes im Anhang -> nur Demoapp.cpp, Demoapp.h (dort wo PidMode verändert wurde), SingleStack.h für Processdatengrösse, main für Abfrage, i2c und spi files

### Test mit TwinCat

Das Evalboard wurde mit der neuen Applikation mit dem IO-Link Master verbunden, welcher an einer SPS Steuerung angehängt ist. Der Aufbau sieht dabei folgendermassen aus:

Bei der SPS handelt es sich um die CX5010-0111 Steuerung von Beckhoff. An diese Steuerung wird die IO-Link Masterklemme EL6224 von Beckhoff gehängt. Die Kommunikation zwischen den beiden Komponenten erfolgt über den Feldbus „EtherCat“.  
Der IO-Link Master besitzt vier Ports, an welche je ein IO-Link Device angehängt werden kann. Am Port 1 hängt dann das Demoboard und überträgt zyklisch die Sensorwerte im IO-Link Protokoll. Das Demoboard selber ist mit einem PHY und einem Mikrocontroller bestückt. Das PHY übernimmt unter anderem die Spannungslevelkonvertierung während der Mikrocontroller die Sensorwerte abfragt und jene in das IO-Link Protokoll integriert.

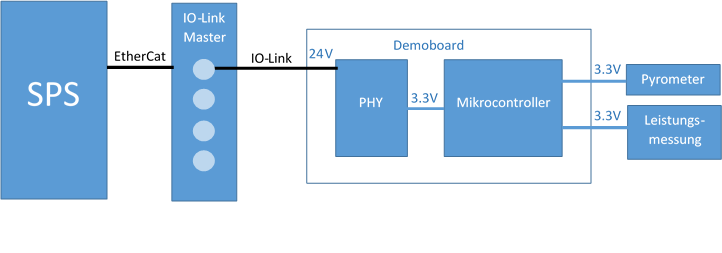


Abbildung 22 - Testaufbau mit SPS

Da die Kommunikation mit dem Pyrometer noch nicht läuft, werden die 2 Oktette, die für dessen Sensorwerte vorgesehen sind, einfach leer gelassen. Nur die 2 Oktette der Leistungsmessung werden mit den erhaltenen Sensordaten gefüllt. Diese gesamte M-Sequenz mit den 4 Oktetten Prozessdaten sieht auf dem Oszilloskop aus wie in Abbildung 21 ersichtlich. Im TwinCat können die Prozessdaten dann ebenfalls ausgelesen werden. Ersichtlich ist dies aus Abbildung 23.

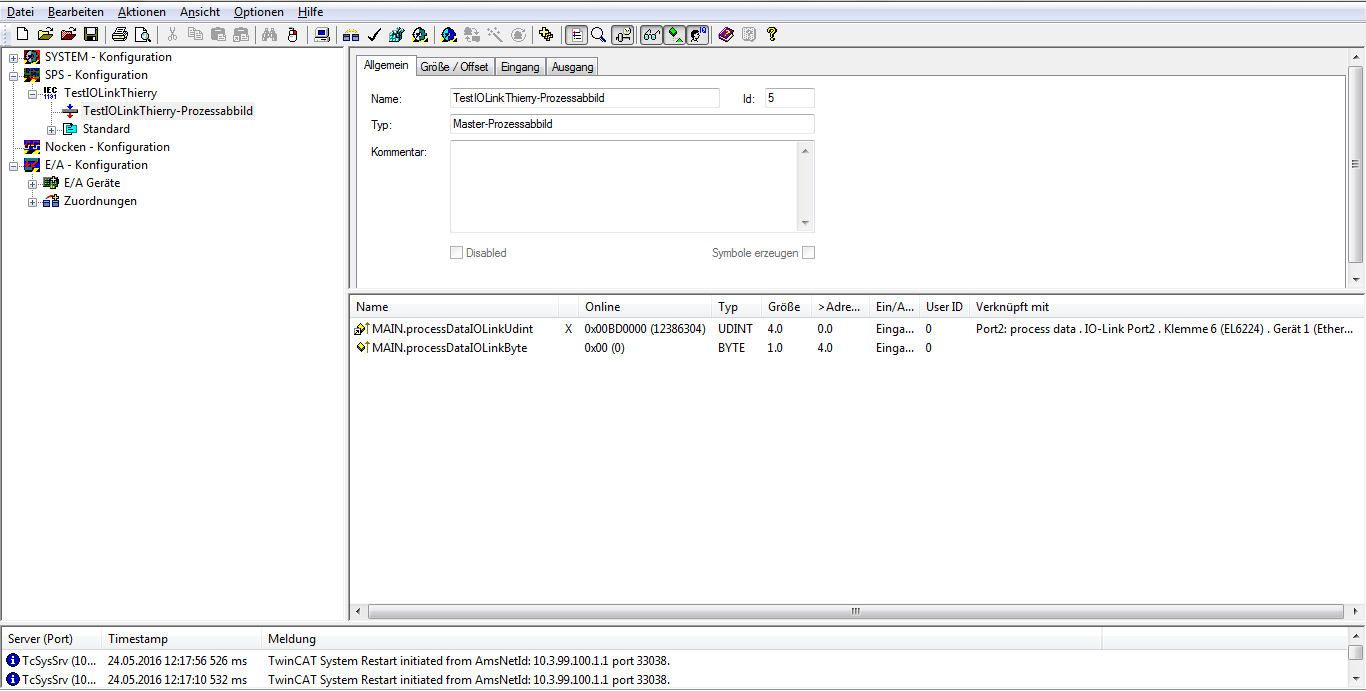


Abbildung 23 - Prozessdaten im TwinCat

Dabei befindet sich die Steuerung im Run-Modus und die vier Byte Prozessdaten sind mit einer Variable verknüpft. Der Momentanwert der Variable kann der Abbildung 23 entnommen werden und beträgt 0x00BD0000 (rot eingerahmt). Der Sensorwert der Leistungsmessung entspricht also 0x00BD, derjenige des Pyrometers 0x0000 (noch keine Kommunikation mit dem Pyrometer möglich).

Zykluszeit  
Die im Kapitel 6.2.6 berechneten Zykluszeiten wurden aufgrund Zeitmangel noch nicht überprüft. Die Steuerung arbeitet immer noch mit derselben Zykluszeit, die in der ursprünglichen Demoapplikation implementiert wurde (3 ms).

Stabilität der Applikation  
Das erstellte Programm arbeitet noch nicht stabil, es fällt etwa alle 2 bis 3 Sekunden aus noch ungeklärten Gründen aus dem Operationsmodus. Dies ist ersichtlich am kurzen Aufblitzen der roten LED auf dem Demoboard, das einen Übergang der Applikation in den Preop Modus signalisiert. Auch erkennbar ist die Instabilität, wenn die CQ-Leitung der IO-Link Kommunikation aufgezeichnet wird und aperiodische Sequenzen zeigt.   
In einer späteren Projektphase müsste noch nach einer Erklärung gesucht werden für dieses instabile Verhalten. Vermutlich können nicht alle Timingvorgaben eingehalten werden, was zu einem Übergang zurück in den Preop Modus führt. Grundsätzlich handelt es sich beim im Rahmen dieser Arbeit erstellten Demonstrationsaufbau um ein nicht serienreifes Produkt. Ein serienreifes Produkt wurde auch nie angestrebt, denn der Einsatz von IO-Link soll hier nur evaluiert und anhand eines Demonstrationsaufbaus aufgezeigt werden und die maximale Sensordatenrate berechnet werden.

### IODD

Aufgrund von Zeitmangel wurde keine IO-Link Device Description erstellt für diese Applikation. Dies behindert die Funktionalität der Applikation jedoch nicht, denn eine IODD ist nur notwendig für die Offlinekonfiguration der SPS oder bei kommerziellem Vertrieb der Applikation.

# Fazit

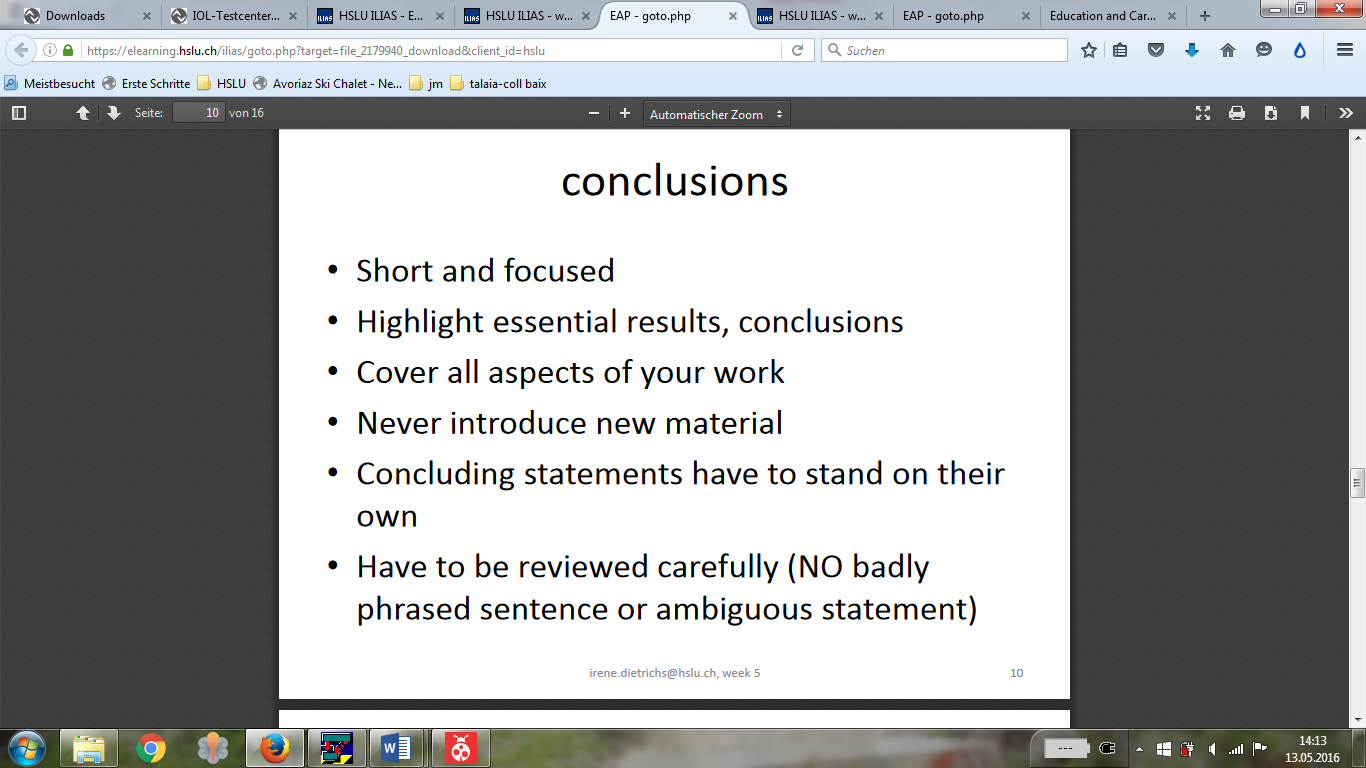
blabla

Conclusion

/ Discussion

•Brief summary of main ideas

•recommendation, prediction and/or solution



## Projektmanagement

Hier auf Projektplan verweisen, Probleme wegen HW von Leister

## Lessons Learned

* Hardware nur mit Dokumentation annehmen

+ Dokumentation immer nachgeführt

* CQ Leitung früher anschauen, vielleicht ist der Fehler beim TwinCat

Abbildungsverzeichnis

[Abbildung 1 - Beispiel Anlagenarchitektur mit IO-Link [1] 4](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860429)

[Abbildung 2 - Anschlussbelegung IO-Link Device [1] 5](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860430)

[Abbildung 3 - Anschlussbelegung Port Class B 5](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860431)

[Abbildung 4 - Übertragungsmodi verschiedener Daten [2] 8](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860432)

[Abbildung 5 - Datenspeicherung bei IO-Link [2] 9](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860433)

[Abbildung 6 - M-Sequenz Typen [2] 10](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860434)

[Abbildung 7 - M-Sequenz Control 11](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860435)

[Abbildung 8 - M-Sequenz Checksumme / Typ 11](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860436)

[Abbildung 9 - M-Sequenz Wertestatus / Checksumme 12](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860437)

[Abbildung 10 - Berechnung der minimalen Zykluszeit [2] 13](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860438)

[Abbildung 11 - ISO/OSI Modell [5] 14](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860439)

[Abbildung 12 - Struktur und Service eines IO-Link Devices [2] 16](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860440)

[Abbildung 13 - Halbduplexe Kommunikation 16](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860441)

[Abbildung 14 - SPI Konfiguration 18](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860442)

[Abbildung 15 - Hardware Aufbau 20](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860443)

[Abbildung 16 - TM96.1 Genie Explorer I Var B 21](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860444)

[Abbildung 17 - GPIO Source Current AtMega328P 28](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860445)

[Abbildung 18 - I2C Kommunikation 29](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860446)

[Abbildung 19 - SPI Kommunikation 30](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860447)

[Abbildung 20 - Verwendete M-Sequenz 35](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860448)

[Abbildung 21 - M-Sequenz Implementiert 36](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860449)

[Abbildung 22 - Testaufbau mit SPS 37](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860450)

[Abbildung 23 - Prozessdaten im TwinCat 37](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860451)

Tabellenverzeichnis

[Tabelle 1 - I2C Protokoll Spezifikationen 17](#_Toc451860452)

[Tabelle 2 - SPI Protokoll Spezifikationen 19](#_Toc451860453)

[Tabelle 3 - Hardwarevorschläge 21](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860454)

[Tabelle 4 - Debugger Auswahl 22](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860455)

[Tabelle 5 - Direct Parameter Page 2 der Demoapp 24](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860456)

[Tabelle 6 - M-Sequenz Preoperate 26](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860457)

[Tabelle 7 - M-Sequenz Operate 26](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860458)

[Tabelle 8 - Implementierte Direct Parameter Page 2 34](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860459)

[Tabelle 9 - Aufbau der Prozessdaten bei Sensorwertübertragung 35](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860460)

[Tabelle 10 - Direct Parameter Page 1 [2] 46](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc451860461)

Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | PROFIBUS Nutzerorganisation e.V. (PNO), «IO-Link Systembeschreibung,» IO-Link, Juli 2013. [Online]. Available: http://www.io-link.com/share/Downloads/At-a-glance/IO-Link\_Systembeschreibung\_d\_2013.pdf. [Zugriff am Februar 2016]. |
| [2] | I.-L. Community, „IO-Link Interface and System Specification,“ Juli 2013. [Online]. Available: http://www.io-link.com/share/Downloads/Spec-Interface/IOL-Interface-Spec\_10002\_V112\_Jul13.pdf. |
| [3] | I.-L. Community, „IO-Link,“ Juli 2013. [Online]. Available: http://www.io-link.com/share/Downloads/Spec-Interface/IOL-Interface-Spec\_10002\_V112\_Jul13.pdf. |
| [4] | S. Feuz, HTI Burgdorf, 2007. [Online]. Available: https://prof.hti.bfh.ch/uploads/media/Bericht\_IO-Link.pdf. [Zugriff am April 2016]. |
| [5] | „Tech Faq,“ [Online]. Available: http://www.tech-faq.com/osi-model.html. [Zugriff am April 2016]. |

Anhang

[A Aufgabenstellung 43](#_Toc451864997)

[B Direct Parameter Page 1 46](#_Toc451864998)

[C Software 47](#_Toc451864999)

[D Projektplan 48](#_Toc451865000)

1. Aufgabenstellung

Horw, 22. Februar 2016

Seite 1/3

**Diplomarbeit im Fachbereich Elektrotechnik**

hslu_d

**Aufgabe für Frau Stefanie Schmidiger**

**Sensoranbindung via IO-Link**

**Fachliche Schwerpunkte**

Automation & Embedded Systems

**Einleitung**

Laserschweisssysteme verfügen über eine Vielzahl von externen Sensoren zur Prozessüberwachung. Verwendet werden z.B. Pyrometer oder Kraft- und Wegmesssensoren. Diese Sensoren werden von einer zentralen Steuereinheit angesteuert und deren Daten ausgelesen. Bis heute werden die Sensoren über individuelle Schnittstellen angebunden. Im Rahmen der Arbeit soll ein Demonstrator-Setup er-stellt werden, mit dem die Einsetzbarkeit des IO-Link-Standards für diesen Zweck gezeigt werden soll. Die Evaluation entsprechender Komponenten und deren Einbindung in das bestehende Software-Framework sind hierbei wichtige Aspekte.

**Aufgabenstellung**

In dieser Arbeit wird ein IO-Link Slave entwickelt. Dieser Slave wird die Daten von 2 unterschiedli-chen Sensoren empfangen, bearbeiten und der IO-Link Schnittstelle zur Verfügung stellen. Er wird mit einem IO-Link Demoboard realisiert und programmiert. In einem zweiten Schritt wird dieser IO-Link Slave über einen Beckhoff IO-Link Master eingelesen.

**Projekt- und Terminplan**

Erstellen Sie einen Projekt- und Terminplans mit Meilensteinen. Stellen Sie einen Plan auf, wie Sie die Arbeit aufteilen.

**Pflichtenheft**

Ein detailliertes Pflichtenheft muss in Absprache mit Leister geschrieben werden. Es muss abgeklärt werden, mit welcher Zykluszeit und mit welcher Auflösung die Daten der 2 Sensoren erfasst werden müssen. Müssen die Daten noch lokal bearbeitet werden? Es muss abgeklärt werden, ob nur ein IO-Link Slave für die 2 Sensoren ausreichend ist oder ob es ein IO-Link Slave pro Sensor braucht. Andere Aspekten wie die mechanische Integration im Laserschweisssystem, die Verkabelung, die Kosten, die Speisung, den Arbeitsbereich (Temperatur, Feuchtigkeit, Vibrationen, usw.) müssen auch diskutiert werden. Wie lange müssen die Bauteile der Lösung in die Zukunft noch lieferbar sein?

**Wahl der Elektronik**

In diesem Arbeitspaket müssen die elektronischen Komponenten und die Softwareentwicklungsumge-bung gewählt werden. Idealerweise wird eine Demonstrationsboard gefunden, die die Entwicklung eines IO-Link Slave Prototyps vereinfacht.

**Programmieren**

Der IO-Link Slave muss in der gewählten Softwareentwicklungsumgebung programmiert werden. Die Daten der Sensoren werden über I2C und/oder SPI empfangen, bearbeitet und der IO-Link Schnittstelle zur Verfügung gestellt.

**Test mit einem Beckhoff IO-Link Master**

Der neue IO-Link Slave wird mit einem Beckhoff IO-Link Master verbunden. Dieser Master nimmt die Form einer EtherCAT Klemme, die über eine Beckhoff SPS konfiguriert und angesprochen wird. Es wird ein SPS Programm geschrieben um den neuen Slave zu testen. Es muss geprüft werden, dass der neue Slave die Anforderungen vom Pflichtenheft erfüllt.

**Design, Layout und Realisation der Elektronik**

Wenn genug Zeit vorhanden ist, wird einen ersten Hardware Prototyp für den Slave entworfen. Eine eigene Schaltung wird zu diesem Zweck gezeichnet, produziert, bestückt, programmiert und in Betrieb genommen.

**Termine**

Start der Arbeit: Montag 22.2.2016

Zwischenpräsentation: Zu vereinbaren im Zeitraum 11.4. -13.5.2016

Abgabe Broschüre-Doku: Freitag 27. Mai 2016, per Mail an Betreuer und H. R Andrist

Abgabe Schlussbericht: Freitag 10. Juni 2016, vor 16:00 im Sekretariat

Abgabe Poster-File: Montag 20. Juni 2016 per Mail an Betreuer und H. R. Andrist

Abschlusspräsentation: Zu vereinbaren im Zeitraum 13.6. – 1.7.2016

**Dokumentation**

Der gebundene Schlussbericht ist in 4-facher Ausführung zu erstellen. Er enthält zudem zwingend

- die folgende Selbstständigkeitserklärung auf der Rückseite des Titelblattes:

*„Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt und keine ande-ren als die angegebenen Hilfsmittel verwendet habe. Sämtliche verwendeten Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.*

*Horw, Datum, eigenhändige Unterschrift"*

- einen englischen Abstract mit maximal 2000 Zeichen.

- Ein Titelblatt mit: Name des Studierenden, Titel der Arbeit, Abgabedatum, Dozent, Experte,  
 Abteilung, Klassifikation (Einsicht/Rücksprache/Sperre)

- Eine CD-Hülle, innen, auf der Rückseite des Berichtes

Alle Exemplare des Schlussberichtes müssen termingerecht abgeben werden. Zusätzlich muss zu jedem Exemplar eine CD mit dem Bericht (inkl. Anhänge), dem Poster und den Präsentationen, Messdaten, Programmen, Auswertungen, usw. unmittelbar nach der Präsentation abgeben werden.

Ein Poster sowie Unterlagen für eine Diplomarbeitsbroschüre sind gemäss den offiziellen Layout-Vorgaben termingerecht einzureichen.

**Fachliteratur/Web-Links/Hilfsmittel**

**Geheimhaltungsstufe:**  Einsicht/Rücksprache/Sperre

**Verantwortlicher Dozent/Betreuungsteam, Industriepartner**

**Dozent**  Prof. Dr. Thierry Prud‘homme thierry.prudhomme@hslu.ch

**Industriepartner** Leister Technologies AG

Laser Plastic Welding

Galileostrasse 10

CH-6056 Kaegiswil

Dr. Jens Rauschenberger

jens.rauschenberger@leister.com Tel. 041 662 7537

**Experte**

Dr. Piotr Myszkorowski

piotr.myszkorowski@sigmatek.ch Tel +41 52 354 50 50

Hochschule Luzern

Technik & Architektur

Prof. Dr. Thierry Prud‘homme

1. Direct Parameter Page 1

Der Inhalt der Direct Parameter Page 1 ist von der IO-Link Community vorgegeben und kann aus Tabelle 10 entnommen werden. Für mehr Informationen zum Inhalt der jeweiligen Speichestelle kann im Dokument „IO-Link Interface Specs“ [2] der Anhang B.1 studiert werden.

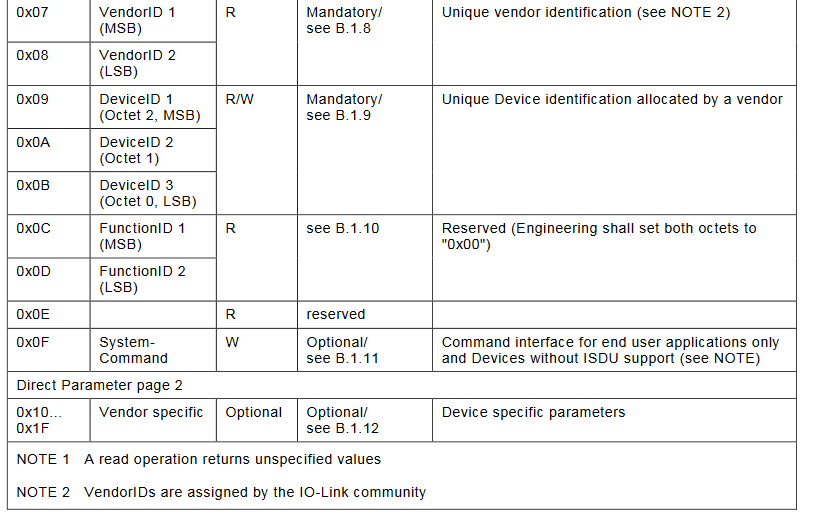
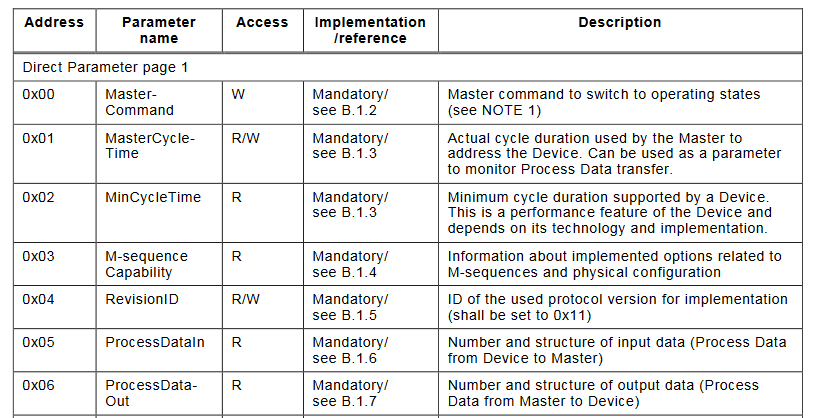


Tabelle 10 - Direct Parameter Page 1 [2]

1. Software

Main.c

Alle anderen Files

1. Projektplan

Hier Projektplan einfügen

Blablabla