Bachelorarbeit

Sensoranbindung mit IO-Link

Stefanie Schmidiger

Dozent:  
Thierry Prud‘homme

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

Inhaltsverzeichnis

[1 Einleitung 2](#_Toc448842034)

[2 IO-Link 3](#_Toc448842035)

[2.1 Physikalische IO-Link Schnittstelle 3](#_Toc448842036)

[2.2 Betriebsarten 4](#_Toc448842037)

[2.3 Datenübertragung 6](#_Toc448842038)

[2.4 IO-Link Protokoll Stack 9](#_Toc448842039)

[3 Sensoren 12](#_Toc448842040)

[3.1 I2C Schnittstelle 12](#_Toc448842041)

[3.2 SPI Schnittstelle 13](#_Toc448842042)

[4 Hardware 15](#_Toc448842043)

[Abbildungsverzeichnis 17](#_Toc448842044)

[Tabellenverzeichnis 17](#_Toc448842045)

[Literaturverzeichnis 17](#_Toc448842046)

[Anhang 18](#_Toc448842047)

Abstract

Einleitung / Ausgangslage

Das vorliegende Dokument befasst sich mit der Umsetzung des Lösungskonzepts im Rahmen des Moduls Produktentwicklung PREN 1 von Team 35 zur Aufgabenstellung an der Hochschule Luzern Technik und Architektur. Die Anforderung an das interdisziplinare Team besteht darin, ein Gerät zu entwickeln, welches autonom fünf Tennisbälle in einen Eimer schiessen kann. Die Position des Eimers soll beim Start ermittelt werden.

Problemdarstellung / Vorgehen

Das Lösungskonzept vom PREN 1 wurde in die Realität umgesetzt. Während der Realisierung und Testphase des Konzepts waren kleinere Änderungen am Konzept nötig, um das Ziel erfolgreich zu erreichen.

Beschreibung der Teilsysteme / Umsetzung

Für die Lösung der Aufgabenstellung wird eine Konstruktion, welche die Bälle in den Korb schiesst, entwickelt. Die Erkennung des Eimers erfolgt mit einer Kamera und der berechnete Winkel wird via USB Kabel an einen Mikrocontroller gesendet. Jener steuert dann einen Schrittmotor, der die Konstruktion horizontal um die gewünschte Anzahl Schritte dreht. Sobald die Ausrichtung des Gerätes beendet ist, werden die Bälle mit einem Hubteller zu den zwei drehenden Rollen befördert und so in den Eimer geworfen.

Ergebnisse / Ausblick

Das im PREN 1 erstellte Lösungskonzept wurde mit kleinen Anpassungen erfolgreich in die Realität umgesetzt. Das Ziel, alle Bälle unter einer Minute zu versenken wurde erreicht, denn die benötigte Zeit beträgt nur rund 20 Sekunden. Einzig beim Gewicht ist das Team nicht unter die 4 Kilogrammmarke gekommen, die Konstruktion wiegt 5.9kg.

Nun gilt es, den Roboter im Wettbewerb mit dem der anderen Teams zu messen.

# 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.

# IO-Link

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

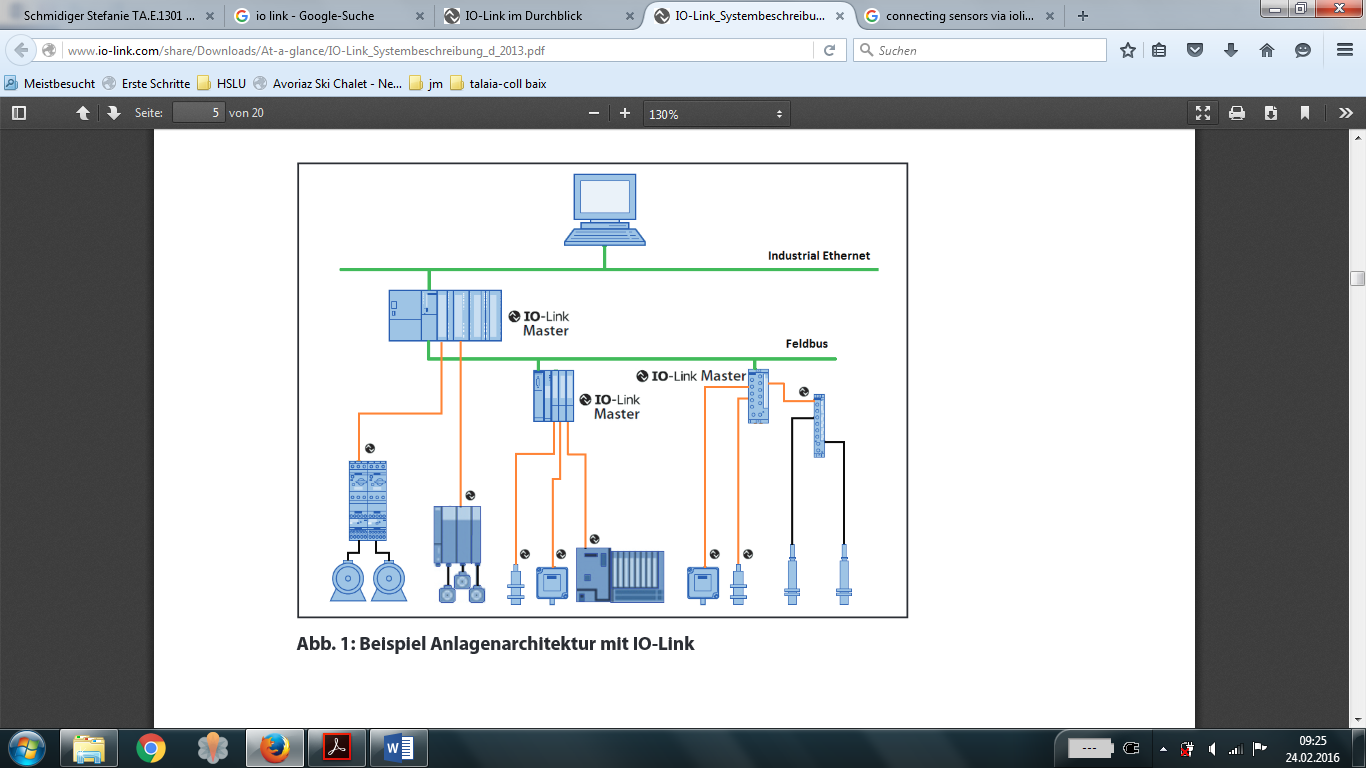


Abbildung 1 - Beispiel Anlagenarchitektur mit IO-Link [1]

## Physikalische IO-Link Schnittstelle

Bei IO-Link handelt es sich um eine serielle und bidirektionale Punkt-zu-Punkt Verbindung. Sensoren haben eine 3-polige Verbindung und Aktoren eine 5-polige Verbindung. Der Master verfügt über eine 5-polige Buchse (M5, M8 oder M12), 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 aus Abbildung 2 entnommen werden.

Die Energieversorgung ist mit Pin 1 und Pin 3 gewährleistet und liefert maximal 200mA. Die maximale Leiterlänge für die ungeschirmte 3- bzw. 5-polige Verbindung zwischen Device und Master ist 20 m.

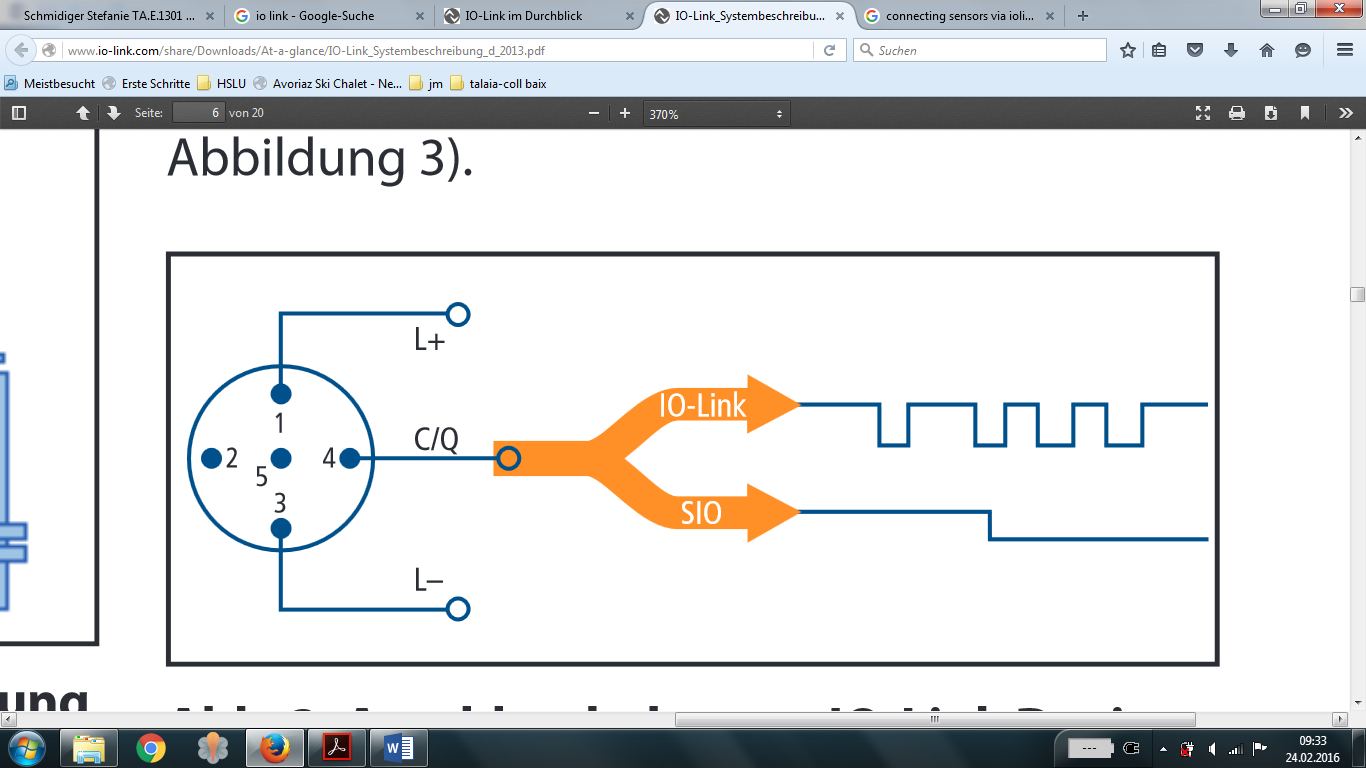


Abbildung 2 - Anschlussbelegung IO-Link Device [1]

Für die IO-Link Schnittstelle gibt es zwei verschiedene Porttypen:

### Port Class A

Die Funktion von Pin 2 und Pin 5 ist nicht vorgegeben und wird vom Hersteller definiert. Typischerweise wird Pin 2 zu einem zusätzlichen Digitalpin. Die Buchsengrösse variiert zwischen M5, M8 und M12.

### Port Class B

Dieser Porttyp ist für energieintensive Devices geeignet da Pin 2 und Pin 5 für eine zusätzliche galvanisch getrennte Versorgerspannung eingesetzt werden. Port Class B Steckverbindungen müssen eindeutig als solche gekennzeichnet werden und sind immer M12 Buchsen.

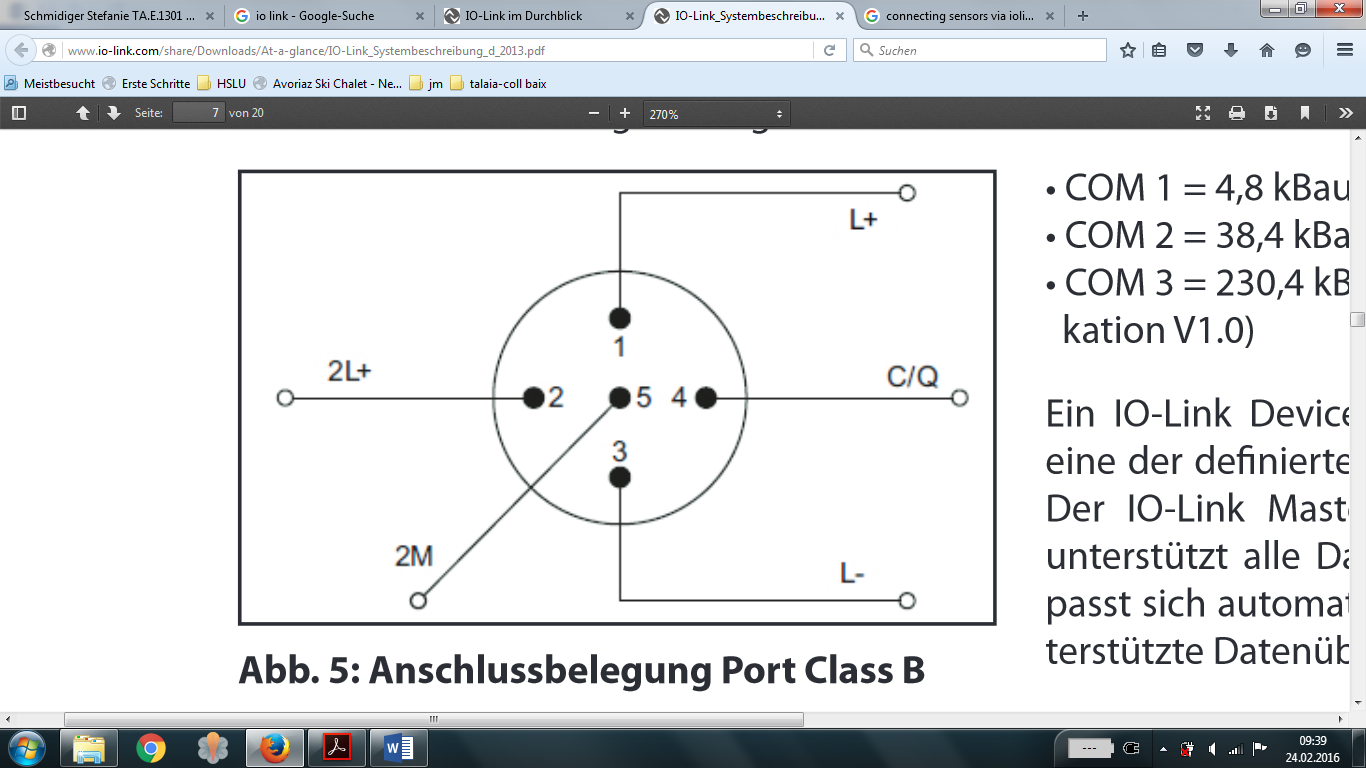


Abbildung 3 - Anschlussbelegung Port Class B

## 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 binäre Devices die am Eingang nur High oder Low erwarten be-  
   ziehungsweise nur High oder Low senden.
* Deaktiviert Diese Betriebsart ist für unbenutzte Ports verwendbar

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 38.4k Baud möglich. Beim neuen IO-Link V1.1 kann zusätzlich noch mit 230.4 kBaud übertragen werden.

Ein IO-Link Device unterstützt nur eine Übertragungsgeschwindigkeit und der Master passt sich deren automatisch an. Ein Master kann für jeden seiner Ports eine andere Übertragungsgeschwindigkeit setzen.

### Erstverbindung von Device und Master

Ist der Port des Masters auf IO-Link eingestellt, so wird er versuchen, mit dem angeschlossenen Device zu kommunizieren. Dazu sendet der Master zuerst in der höchsten definierten Übertragungsgeschwindigkeit einen Wake Up Impuls und wartet auf die Antwort des Devices. Bei 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 Device übertragen. Anschliessend folgt die Übertragung der zyklischen Daten.

### IO-Link Device Description Datei

Die IO-Link Device Description (IODD) Datei ist eine elektronische Gerätebeschreibung und hält für die Systemintegration vielfältige Informationen bereit. Der Aufbau dieser IODD Datei ist dabei immer derselbe und wird von den IO-Link Konfigurationstools immer gleich dargestellt.

### Reaktionszeit und Zykluszeit

In der Gerätebeschreibungsdatei IODD des Devices ist ein Wert für die minimale Zykluszeit angegeben, welcher bestimmt, in welchen Abständen der Master das Device ansprechen darf um zyklische Daten zu erfragen bzw. übertragen. Zusätzlich benötigt der Master noch eine Bearbeitungszeit. Diese beiden Zeiten ergeben die Reaktionszeit des Systems. Da jedes Device eine andere minimale Zykluszeit haben kann, ist die Reaktionszeit verschiedener Devices am selben Master unterschiedlich. Mit der maximalen Übertragungsgeschwindigkeit der minimalen Protokollgrösse beträgt die minimale empfohlene Zykluszeit 400 µs.

### IO-Link V1.0 und IO-Link V1.1

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.

Die Datenübertragungsrate von 230.4 kBaud ist nur mit der Version V1.1 möglich. Deshalb wird ein IO-Link Master der Version V1.1 mit angehängtem V1.1 Device benötigt für diese schnelle Übertragung. 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.

Prozessdaten (Process Data Innput / Process Data Output)  
Die Prozessdaten sind zyklische Daten wie beispielsweise Sensorwerte. Die Datenmenge kann dabei vom Device einmalig beim Einschalten festgelegt werden und liegt zwischen 0 und 32 Bytes.

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 könne. Für diesen Zweck gibt es ein Adressbereich, auf den mit einem Index zugegriffen werden kann. Innerhalb dieses Adressbereiches gibt es die DirectParameterPage1, welche vorgegebene Daten enthalten muss, die der Master beim Aufsetzen der Kommunikation ausliest (VendorID, DeviceID, Revisionsnummer, Baudrate etc). Die DirectParameterPage2 kann beliebige Informationen enthalten und wird vom Device Hersteller festgelegt. Bei Bedarf kann jetzt der Master auf einen Ort im Adressbereich zugreifen und die darin enthaltenen Daten (Parameter, Identifikationsdaten oder Diagnoseinformationen) auslesen oder überschreiben und somit azyklisch den Status des Devices verändern oder abfragen.

Ereignisse (Events)  
Wenn ein Ereignis auftritt, signalisiert das Device dies dem Master und der Master liest es daraufhin aus. Ereignisse können Fehlermeldungen und Warnungen oder Wartungsdaten sein. Diese Ereignisse werden zur Steuerung weitergeleitet. Auch der Master kann Ereignisse haben und diese weiterleiten.

### Übertragungsart verschiedener Daten

Eine Übersicht über die Übertragung der oben beschriebenen Daten kann aus Abbildung 4 entnommen werden. Die Prozessdaten und der Wertestatus bilden zusammen die Kategorie „Operation“ und werden zyklisch übertragen.

Events und Gerätedaten sind on-request Daten und werden nur auf Anfrage oder bei Bedarf übertragen.

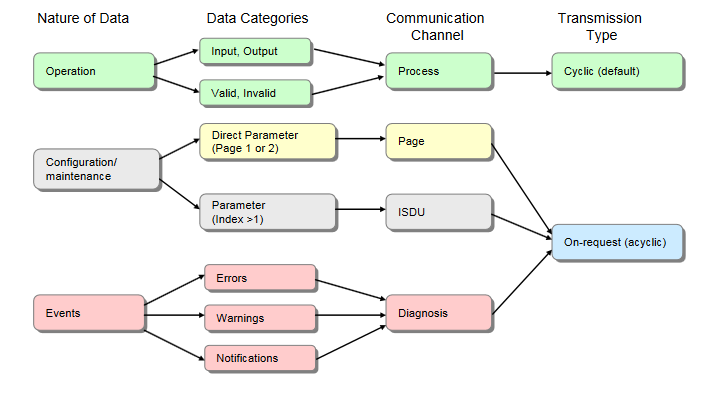


Abbildung 4 - Übertragungsart verschiedener Daten [2]

### Datenspeicherung

Aus der Abbildung 5 kann das Memory Management innerhalb eines Devices entnommen werden. Dabei hat das Device irgendwo einen Buffer für die vom Master empfangenen Prozessdaten und einen Buffer für die zu sendenden Daten. Die Buffergrösse wird dabei bei Kommunikationsbeginn einmalig festgelegt und liegt zwischen 0 und 32 Bytes (octets).

Gerätedaten (Configuration/Maintainance Daten) des IO-Link Devices werden einem festgelegten Adressbereich zugeordnet. Dabei ist der DirectParameterPage1 der Adressbereich 0x00 bis 0x0F zugeordnet und der DirectParameterPage2 der Adressbereich von 0x10 bis 0x1F.

Auch für Events ist ein Buffer angelegt dem beim Aufstarten eine Grösse von 0 bis 32 Bytes zugeordnet werden kann.

### Der Inhalt jeder Speicherstellen der DirectParameterPage1 ist festgelegt. Beispielsweise ist an der Speicherstelle 0x04 die RevisionsID hinterlegt, an der Speicherstelle 0x06 die Anzahl vom Master versendeten Bytes etc. Eine detaillierte Auflistung der Inhalte der directParameterPage1 kann dem Anhang X entnommen werden. Der Inhalt der Speicherstellen der DirectParameterPage2 und allen anderen Speicherstellen bis 0xFFFF hingegen kann vom Hersteller der IO-Link Devices selber bestimmt werden. Beispielsweise kann ein Devicehersteller an die erste Speicherstelle der directParameterPage2 (Index 0x10) den Status des Devices implementieren, liest aus dem Index 0x11 Befehle aus usw. Jedem Hersteller ist die Implementierung freigestellt und der User muss das Datenblatt des jeweiligen Devices beiziehen um an diese Informationen zu gelangen.

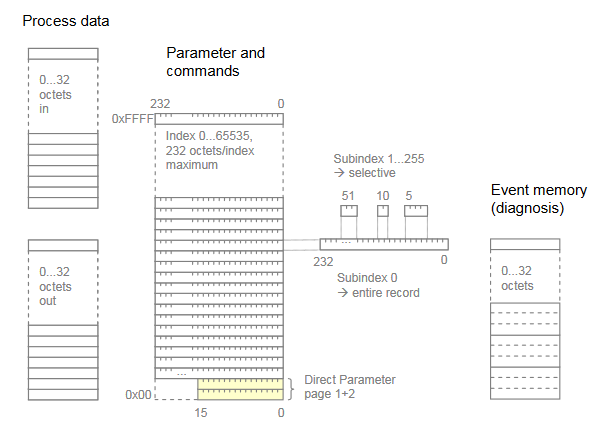


Abbildung 5 - Datenspeicherung bei IO-Link [2]

Die Speicherstellen ausserhalb der DirectParameterPages werden ISDU genannt und umfassen die Speicherstellen 0x20 bis 0xFFFF.

### M-Squenz

Je nach Anzahl zu empfangener und übermittelnder Bytes an Prozessdaten wird ein anderer M-Sequenz Typ für die Einbettung der Daten gewählt. Die Auswahl an M-Sequenz Typen ist aus Abbildung 6 ersichtlich.

Die Kommunikation zwischen IO-Link Master und Device besteht also aus einem Byte M-Sequenz-Control (MC), gefolgt von einem Byte Check/Type (CKT) und danach die gewünschte Anzahl Prozessdaten (PD) und on-request Daten (OD). Das letzte Byte (CKS) repräsentiert den Wertestatus und enthält eine Checksumme.

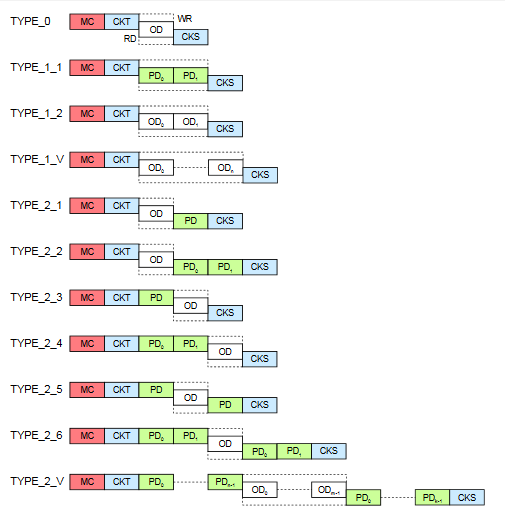


Abbildung 6 - M-Sequenz Typen [2]

Der Sequenztyp TYPE\_1\_1 kann nicht mehr verwendet werden, da er keinen Zugriff auf on-request Daten zulässt.

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

Grundsätzlich gilt, dass bis zum TYPE\_2\_6 der Versand von maximal 2 Byte an Prozessdaten in die jeweilige Richtung möglich war. Mit dem TPYE\_2\_V kann die Grösse der Prozessdaten frei gewählt werden (zwischen 0 und 32 Bytes). Auch kann dabei die Anzahl Bytes an on-request Daten frei gewählt werden.

Im folgenden Abschnitt wird auf den Inhalt der einzelnen Bytes 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 Byte handelt es sich um das Control Byte. Dieses beinhaltet die Offset-Adresse für einen ISDU-Zugriff, den Kommunikationskanal (Prozessdaten, DirectParameterPage 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.

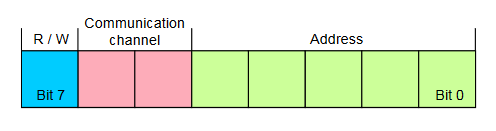


Abbildung 7 - M-Sequenz Control

M-Sequenz Checksumme / Typ

Dieses Byte 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 Bytes an zyklischen Prozessdaten er bekommt und senden muss und wieviele Bytes für die on-request Daten reserviert sind. Die Implementierung des Bytes kann aus Abbildung 8 entnommen werden.

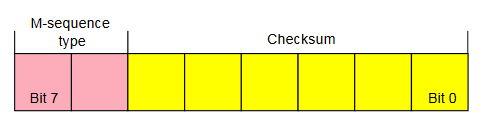


Abbildung 8 - M-Sequenz Checksumme / Typ

Prozessdaten

Prozessdaten sind z.B. Messwerte von einem Sensor oder Ausgabewerte an einen Aktor.

Eine bestimmte Menge an Bytes ist bei jedem Zyklus für die Prozessdaten reserviert. Diese können entweder vom Typ „ProcessData\_in“ sein, wenn sie an den Master gesendet werden oder vom Typ „ProcessData\_out“, 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 directParameterPage oder Events. Obwohl diese Daten nur auf Anfrage (on request) versendet werden, muss trotzdem eine feste Anzahl Bytes 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. 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.

Aufgrund Performanceoptimierung kann die zyklisch zu übertragende Menge an on-request Daten minimal gehalten werden (1 Byte). Dies bedeutet dann jedoch, dass Events oder Eingriffe in die DirectParameterPages über mehrere Zyklen übertragen werden müssen, da sie nicht in einem Byte Platz haben.

M-Sequenz Wertestatus / Checksumme

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

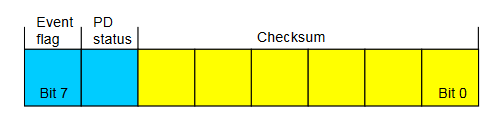


Abbildung 9 - M-Sequenz Wertestatus / Checksumme

## IO-Link Protokoll Stack

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

### ISO/OSI Modell

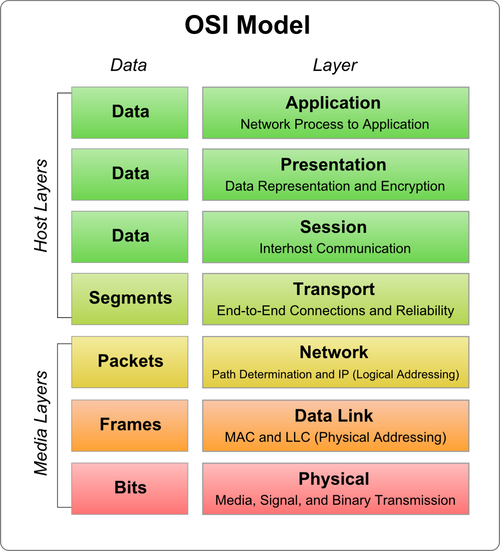
Das IO-Link Protokoll stützt sich ab auf das ISO/OSI Modell. Nach dem ISO/OSI Modell wird der Datenaustausch immer in sieben Layer unterteilt, die jeweils einen Teil der Kommunikation übernehmen.

Abbildung 10 - 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 (deaktivierter Port, SIO-Modus, COMx-Modus).

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 7 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.

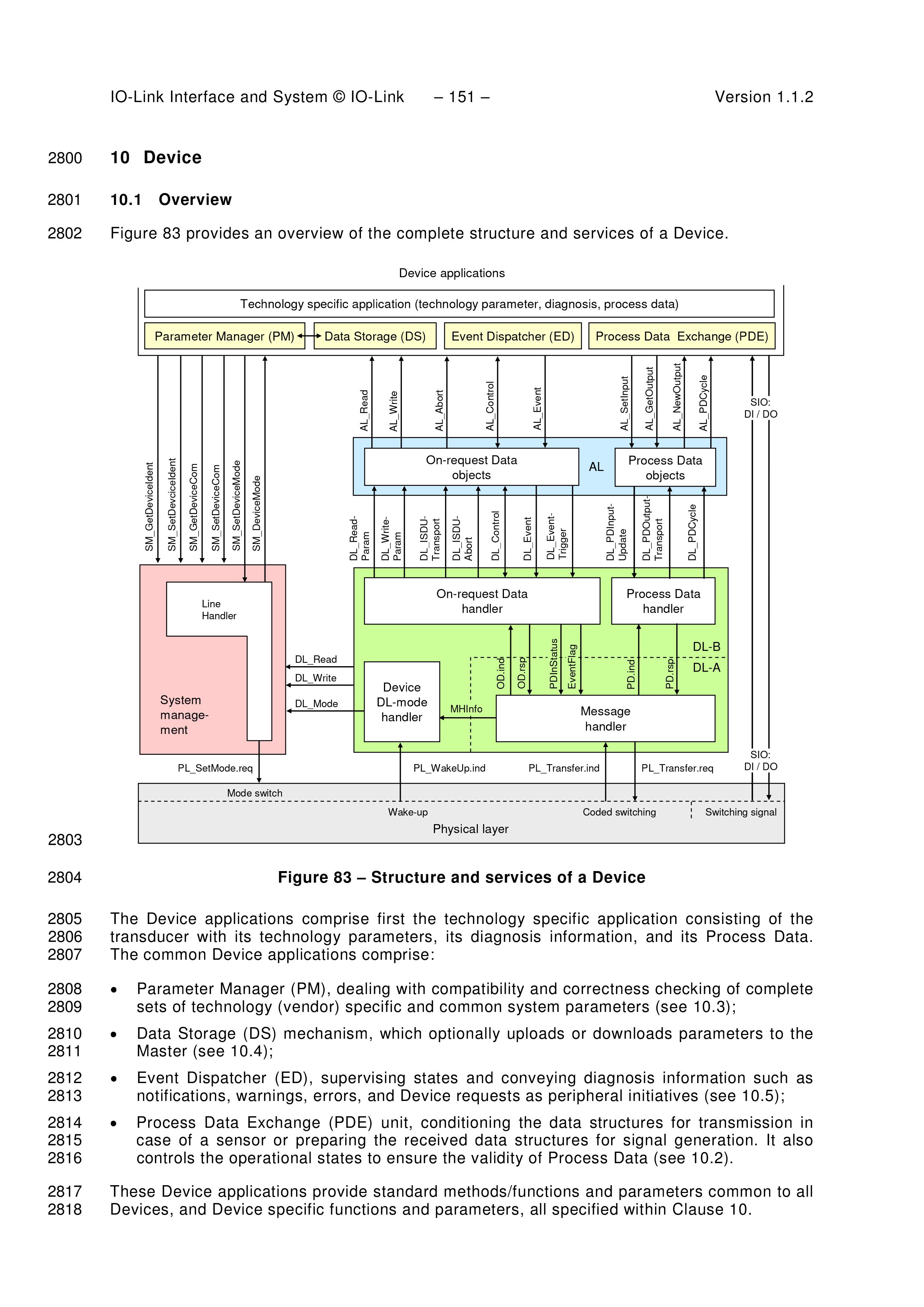


Abbildung 11 - Struktur und Service eines IO-Link Devices [2]

Die Aufgabe eines Funktionsaufrufes kann der jeweiligen Endung des Funktionsnamens entnommen werden. Dabei gibt es vier Möglichkeiten:

.req   
Beim Funktionsaufruf handelt es sich um einen Request. Der Initiator des Funktionsaufrufes bittet also um Daten.

.ind  
Die .ind Endung einer Funktion bedeutet, dass der Initiator des Funktionsaufrufes auf etwas Hinweisen möchte, beispielsweise einen Statuswechsel.

.resp  
Jede Funktion, die mit .resp endet, ist eine Antwort auf eine Anfrage oder ein Ereignis.

.cnf  
Wenn eine Funktion mit .cnf endet, so handelt es sich um eine Bestätigung.

Sofern die Funktionsnamen keine der vier Endungen aufweisen, so kann es sie in mehreren Ausführungen geben.

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 9 entnommen werden.

Transmitter

Receiver

**IO-Link Master**

Transmitter

Receiver

**IO-Link Slave**

Abbildung 12 - Halbduplexe Kommunikation

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  
Der 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 der 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 10bit AD-Wandler abgetastet und im selben Baustein in ein I2C Protokoll verpackt. Dabei ist der Sensor als Slave implementiert und benötigt einen Konstantstrom 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 kontinuierich ein Clock generiert wird. Der AD-Wandler wird dann kontinuierlich wandeln (typische Wandlungszeit: 8.96us, Analog Input Zeit: 1.12us, total 10.08us pro Wandlung) 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 ist ein 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 Bit 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 also 90 us.

## 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. 20mA.

### SPI Konfiguration

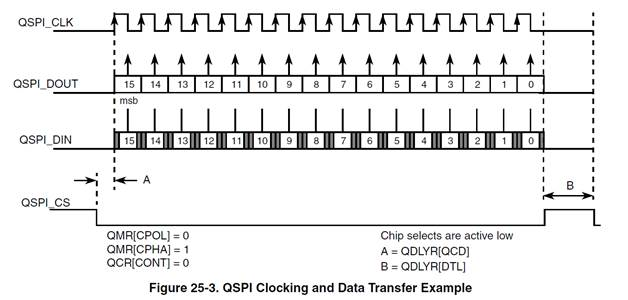
Die Aufsetzung der SPI Schnittstelle kann aus Abbildung 7 entnommen werden.

Abbildung 13 - SPI Konfiguration

Ein Datenpaket entspricht jeweils 16 Bit. Das MSB wird zuerst übermittelt. Die Leitung SlaveSelect ist low-aktiv und muss 3 Systemclockzyklen vor der Datenübertragung auf 0V gezogen werden. Zwischen zwei SlaveSelects müssen mindestens 600 Clockzyklen liegen um eine korrekte Datenverarbeitung im Mikrocontroller zu gewährleisten.



Es ist eine Baudrate von 1 MHz möglich.

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

### SPI Kommandos

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

|  |  |  |
| --- | --- | --- |
| Kommandoname | Wert | Bedeutung |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 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 14 entnommen werden.

Pyrometer

Leistungs-messung

Mikrocontroller

PHY

IO-Link Master

Demoboard

3.3V

24V

3.3V

3.3V

Abbildung 14 - 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 zwischen Master und Slave nicht mehr als xx % betragen darf.

## 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
* Header 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 6-10 GPIO Pins zur Verf  
   gung stehen.
* 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 2 Wochen lieferbar sein
* Kosten Bei der HSLU stehen CHF 600 zur Verfügung für die Bachelorarbeit

Die Auswahl an IO-Link Demoboards ist klein, eine Auflistung:

Auflistung HW Vorschläge

Da die Firma HMT ein Demoboard gratis zur Verfügung stellt und ein IO-Link Stack inkl. Demoprogramm dazuliefert, wird mit ihrem Demoboard gearbeitet.

### 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 Resourcekritischen Anwendungen ein PHY nötig sein.

Das 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. Master und Slave kommunizieren über das UART Protokoll. Dies ist sehr resourcenintensiv, 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, kann Ressourcen gespart werden. Jetzt kann der MultiByteModus 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 SingleByteModus 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 AtMega Mikrocontroller 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 SingleWireDebug unterstützt. Auf der Webseite von Atmel kann eine Auflistung an Programmern gefunden werden für den verwendeten Mikrocontroller AtMega328.



Tabelle 2 - Debugger Auswahl

Aus Tabelle 2 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.

## 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 Demoboar  
IQ2 Application Layer und Physical Layer muss noch implementiert warden

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

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.

## 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.

## Demoapp

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

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

Im Kapitel 2.3 wurde auf den Unterschied zwischen zyklischen und azyklischen Daten eingegangen und im Zusammenhang damit auch auf die Direct Parameter Page.  
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 (MirrorOutput und MirrorInput). Der Wert an der Adresse 0x12 bestimmt, welcher der oben angegebenen 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 2 entnomen werden.

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.

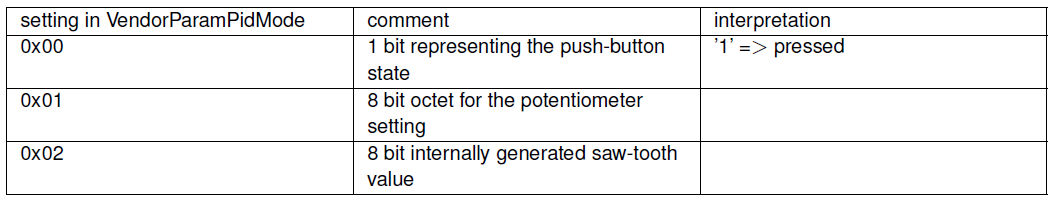
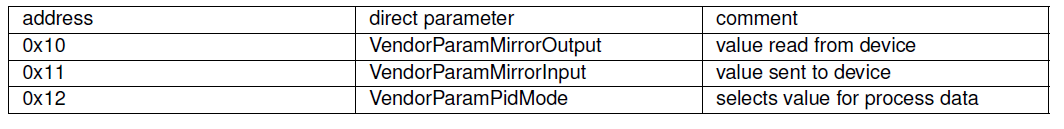


Tabelle 3 - Direct Parameter Page 2 der Demoapp

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.

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 7, 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 Absatz 2.3.3 zum Thema on-request Daten).

### Inbetriebnahme der Hardware mit Demoapp

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

Stackmode wählen  
Ein Merkmal des PHY Bausteines der HMT ist, dass es einen Teil des IO-Link Protokolls selber übernehmen kann.

Die Demoapp kann in drei verschiedenen Modi laufen:

* SingleByte Sobald ein ganzes Byte beim PHY angekommen ist, wird dieses an   
   den Mikrocontroller weitergeleitet.
* MultiByte 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 Levelkonverter verwendet.

Um einen Modus auszuwählen, muss die entsprechende Headerdatei eingebunden werden in der Demoapp.

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

PDInSize und PDOutSize  
PD\_IN\_SIZE Anzahl Bytes die vom Slave zum Master übertragen werden  
PD\_OUT\_SIZE Anzahl Bytes 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 DirectParameterPage1 im Adressfeld 0x03.

Eine Übersicht von möglichen M-Sequenz Kompatibilitäten kann aus den folgenden Tabellen gewonnen werden.

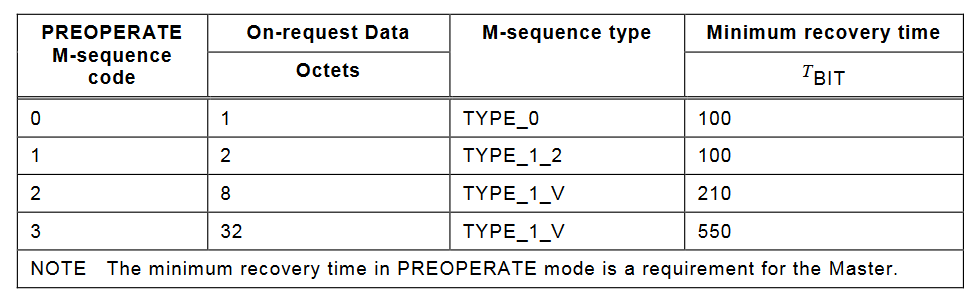


Tabelle 4 - M-Sequenz Preoperate

Die Tabelle 4 listet die möglichen M-Sequenzen auf, die im Preoperate Modus 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.

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 Bytes, die übermittelt werden, muss die passende M-Sequenz gewählt werden.

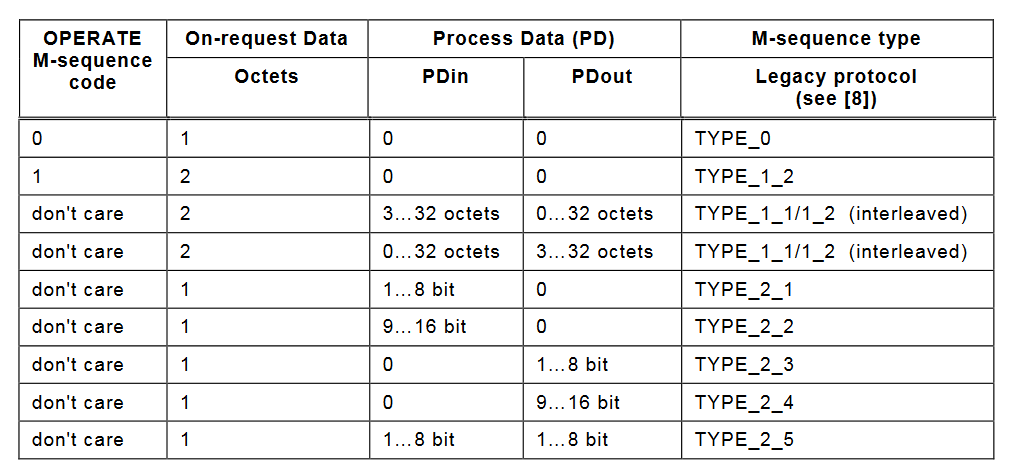


Tabelle 5 - M-Sequenz Operate

In der Demoapp werden 1 Byte an Prozessdaten vom Master an den Slave gesendet und 1 Byte Prozessdaten vom Slave an den Master. Für den Preoperate Modus ist also die M-Sequenz TYPE\_0 implementiert und für den Operate Modus die M-Sequenz TYPE\_2\_5.

Abbildungsverzeichnis

[Abbildung 1 - Beispiel Anlagenarchitektur mit IO-Link [1] 3](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926267)

[Abbildung 2 - Anschlussbelegung IO-Link Device [1] 4](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926268)

[Abbildung 3 - Anschlussbelegung Port Class B 4](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926269)

[Abbildung 4 - Datenspeicherung bei IO-Link [2] 7](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926270)

[Abbildung 5 - M-Sequenz Typen 8](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926271)

[Abbildung 6 - ISO/OSI Modell 9](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926272)

[Abbildung 7 - Struktur und Service eines IO-Link Devices [2] 10](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926273)

[Abbildung 8 - Halbduplexe Kommunikation 11](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926274)

[Abbildung 9 - SPI Konfiguration 13](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926275)

[Abbildung 10 - Hardware Aufbau 15](file:///C:\Users\Stefanie\Documents\Studium\6.Semester\BDA\Dokumentation\Dokumentationsvorlage.docx#_Toc448926276)

Tabellenverzeichnis

[Tabelle 1 - I2C Protokoll Spezifikationen 12](#_Toc448926277)

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,“ Juli 2013. [Online]. Available: http://www.io-link.com/share/Downloads/Spec-Interface/IOL-Interface-Spec\_10002\_V112\_Jul13.pdf. |
| [3] | S. Feuz, HTI Burgdorf, 2007. [Online]. Available: https://prof.hti.bfh.ch/uploads/media/Bericht\_IO-Link.pdf. [Zugriff am April 2016]. |

Anhang

[A Produktanforderungsliste 34](#_Toc421909746)

[B Präferenzmatrix 35](#_Toc421909747)

[C Testfälle 36](#_Toc421909748)

[D Prints 39](#_Toc421909749)

[E CAD Zeichnungen 41](#_Toc421909750)

1. Produktanforderungsliste

In der Produktanforderungsliste werden die Anforderungen an das Produkt und die