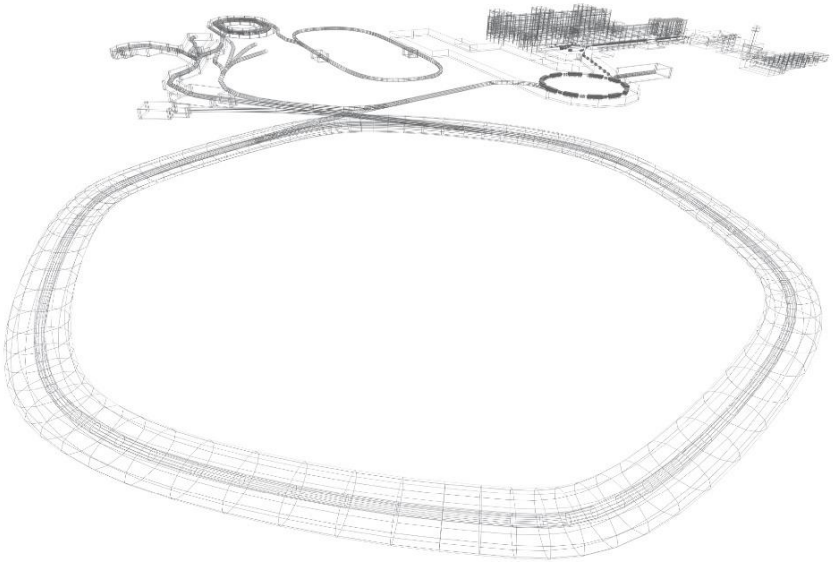
Elektrische
Energie
Technik

GSI Darmstadt



Universal Serial Interface verstehen (USI 1.1)

Derek Schupp

Version vom: Mittwoch, 11. September 2019, 07:48:00

Aktuelle Versionen immer unter:

https://wiki.gsi.de/foswiki/bin/view/EPS/ACUDocuments

Inhaltsverzeichnis

1. Änderungsliste ... 3
2. Begriffe ... 4
3. Beschreibung des USI (universelles Serielles Interface) ... 5
3.1. Überblick ... 5
3.2. Definition des Host ... 5
3.3. Details ... 6
3.3.1. Stromversorgung für Sekundärseite des USI ... 7
3.3.2. Summeninterlockleitung ... 7
3.3.3. Datenversorgung ... 8
3.3.4. Plug & Play an der MFU ... 8
3.3.5. Verbindung MFU ↔ Modul ... 8
4. Transferprinzip ... 9
4.1. FIFO Startpunkte (FSP) ... 9
4.2. Wichtige Vereinbarung ... 9
4.3. FSP-Struktur im USI Modul ... 10
4.4. Tiefe von FSP ... 11
5. Das USI Protokoll ... 12
5.1. Definition der Datenübertragungsrichtung ... 12
5.2. Protokolle der USI ... 12
5.3. Datenformat ... 12
5.3.1. ASCII oder Hex ... 12
5.4. Steuerzeichen ... 13
5.5. Package Identifier ... 14
5.6. Prüfsummenbildung ... 14
5.7. USI Pakete bei Normaler/Standard USI Übertragung ... 15
5.7.1. Phasen einer Standard USI Transaktion zwischen Host und angeschlossenem Teilnehmer ... 15
5.7.1.1. Direkter Datentransfer zwischen Host und Modul ... 15
5.7.1.1.1. Daten vom Host an ein Modul senden ... 16
5.7.1.1.2. Modul bestätigt den erfolgreichen Empfang der Daten ... 17
5.7.1.1.3. Modul hat die Daten nicht erfolgreich empfangen ... 17
5.7.1.1.4. Host will Daten von einem Modul lesen ... 17
5.7.1.1.5. Daten vom Modul an den Host senden ... 18
5.7.1.2. Datentransfer zwischen PC/Kontrollsystem und ACU System ... 18
5.7.1.2.1. Daten vom PC/Kontrollsystem an das ACU System senden ... 19
5.7.1.2.2. MFU, bzw. Modul bestätigt den erfolgreichen Empfang der Daten ... 20
5.7.1.2.3. MFU, bzw. Modul hat die Daten nicht erfolgreich empfangen ... 20
5.7.1.2.4. PC/Kontrollsystem will Daten von der MFU, bzw. einem Modul lesen ... 20
5.7.1.2.5. Daten von der MFU, bzw. einem Modul an den PC/das Kontrollsystem senden ... 21
5.7.2. Fehlernummern ... 21
5.8. USI Pakete bei Hochgeschwindigkeits-/HighSpeed USI Übertragung ... 22
5.8.1. Format des HS USI Paket ... 22
5.8.2. Die HSByte ... 22
5.8.3. Das Tunnelbyte ... 22
5.8.4. Das Strobe/Acknowlege Byte ... 23
5.8.4.1. Freie Bit im Strobe/Acknowledge Byte ... 24
5.8.5. Beispiel eines Hochgeschwindigkeits- USI Pakets ... 25
6. MDS (Modul Descriptor Structure) ... 26
6.1. Vereinbarung ... 26
6.2. Standard Descriptoren ... 26
6.3. Lage der MDS und deren Aufbau ... 26

6.4. Interpretationszyklus der MDS für den Host 27
6.5. Beschreibung der Standard Descriptoren 28
6.5.1. Modul-Descriptor 28
6.5.2. FSP-Descriptor 31
6.5.3. End Descriptor 31
7. Enumeration und Plug & Play 32
7.1. Genereller Ablauf 32

1. Änderungsliste

Datum	Name	Kommentar
11.06.2008	D. Schupp	Dokument erstellt
01.08.2008	D. Schupp	PC<-> Host Kommunikation
08.09.2008	D. Schupp	Bereinigt, Kapitel 9 Treiberstrukturen ist nicht gültig!
05.01.2009	D. Schupp	Div. kleine Änderungen, Rechtschreibkorrekturen
29.04.2009	D. Schupp	MDS MinorVersion von 'w' auf 'l' geändert
22.06.2009	D. Schupp	HS auf 4 Byte korr.
26.01.2010	D. Schupp	APC² in ACU und ACU in MFU umbenannt
17.05.2010	D. Schupp	Kapitel: “Das Strobe/Acknowlege Byte“ hinzugefügt Tabellen mit Details des Strobe/Acknowledge Bytes
20.05.2010	D. Schupp	Prüfsummenbildung bei USI-Übertragung, PID Modulreset (RS) entfernt, Kapitel „Treiberstrukturen“ entfernt.
25.05.2010	D. Schupp	Prüfsummenbeispiel
08.06.2010	D. Schupp	Formatierungen und Strobe/Acknowledgebyte bei Tunnelbetrieb
14.01.2011	D. Schupp	Dokument in USI Spezifikation umbenannt und Kapitel „1. Ausbaustufe“ entfernt und als Version 1.00 den Vermerk ‚Entwurf’ gelöscht.
16.02.2011	D. Schupp	‚SerialNumber’ in MDS von 16 auf 48 Bit vergrößert
06.02.2011	D. Schupp	USI Modulklassen, Danfysik hinzugefügt.
29.05.2012	D. Schupp	USI 1.1 in Version eingefügt
12.06.2014	D. Schupp	Grundlegend überarbeitet
-	-	Übersprungen um mit dem englischen Dokument konform zu sein
03.09.2014	D.Schupp	Formatierungen und kleine Syntaxkorrekturen
16.09.2014	D. Schupp	Kleine Syntaxkorrekturen
16.09.2015	D. Schupp	Druckversion
08.03.2017	D. Schupp	Modul-Klassen ergänzt MDS Attribute um Bit[7] ergänzt
07.02.2019	D. Schupp	Maximale FSP Tiefe neu definiert. Tatsächlich sind nur 4095 Bytes, also 4M-1 Byte möglich.
11.09.2019	D. Schupp	Umstieg auf Office 2016, Schrift „Universe“ durch „Calibri“ ersetzt und die damit verbundenen Neuformatierungen durchgeführt. USI Attribute um Bit[0] ergänzt. Baud durch Bit ersetzt. Dokument überarbeitet im Hinblick auf Host/MFU.

2. Begriffe

ACU	Adaptive Control Unit, Verbund mehrerer Netzgerätesteuermodule, i.d.R. mit einer MFU als Zentraleinheit.
ADC	ADC-Karte/ADC-Modul (Analogue to Digital Converter)
DCCT	DC Current Transformer
FSP	FIFO Start Punkt, legt eine eindeutige Adresse fest an der im Modul eine bestimmt Speichertiefe mit einer bestimmten Funktion hinterlegt ist.
ICM	Interlock- and Control-Modul, Steuer und Interlocksystem, dient innerhalb der AC dazu Steuersatzinformationen an den Leistungsteil der SVE zu senden, bzw. Interlock (Fehler) Signale aus der SVE zu empfangen und diese im Fehlerfall abzuschalten.
MDS	Modul Descriptor, ist in jedem Modul der FSP000 und beinhaltet eine Modulbeschreibung
MFU	Multi Function Unit, Zentrale Bedien- und Steuereinheit innerhalb eines ACU Systems
PC	Personal Computer
SCU	Scalable Control Unit (Interface zwischen ACU- und Control System)
SVE	Strom-Versorgungs- Einheit, Netzgerät
USI	Universelles Serielles Interface, Universal Serial Interface, verbindet die einzelnen Netzgerätesteuermodule durch serielle Kommunikation

3. Beschreibung des USI (universelles Serielles Interface)

3.1. Überblick

Das USI dient zur Anbindung von Sensoren und Aktoren innerhalb einer SVE.

Funktionell ist das USI eine asynchrone serielle Schnittstelle im Vollduplex die Bitraten von 115 kBit/s bis 20 MBit/s auf Basis einer RS485 unterstützt. Im Weiteren wird darüber auch von der Modulseite her eine Versorgungsspannung für die Sekundärseite der USI Schnittstelle geliefert. Außerdem beinhaltet es eine Hardware Interlockleitung die als WIRED OR ausgeführt ist und das Netzgerät zum sofortigen ausschalten zwingt. Als Steckverbinder werden RJ45 Stecker verwendet. Alle Leitungen sind isoliert und die Betriebsspannung ist über DC/DC Wandler geführt um zu gewährleisten, dass die Sensor- und Aktormodule potentialfrei sind.

3.2. Definition des Host

Alle Module müssen mit einem Host verbunden werden über den diese miteinander kommunizieren können. Es ist i.d.R. nicht möglich, dass Module den Host umgehen und direkt miteinander kommunizieren.

In einem ACU System ist üblicher Weise die MFU der Host. Wird diese über USB an einen PC oder über die Backplane an ein Kontrollsystem angebunden, kann der PC oder das Kontrollsystem den Host bilden.

Es ist auch möglich Module mittels RS485/USB Kabel, ohne MFU z.B. direkt an einen PC anzuschließen. Dies wird vor allem aus Debug- und Entwicklergründen geschehen. In diesem Fall ist der PC als Host anzusehen.

Im praktischen Betrieb ist jedoch normaler Weise eine MFU im System und bildet dabei den Host.

Je nachdem ob Module über eine MFU oder direkt mittels RS485/USB Kabel mit einem PC/Kontrollsystem verbunden sind unterscheidet sich die Adressierung des Moduls geringfügig. Infos hierzu finden sich im Kapitel 5.7. „USI Pakete bei Normaler/Standard USI Übertragung“.

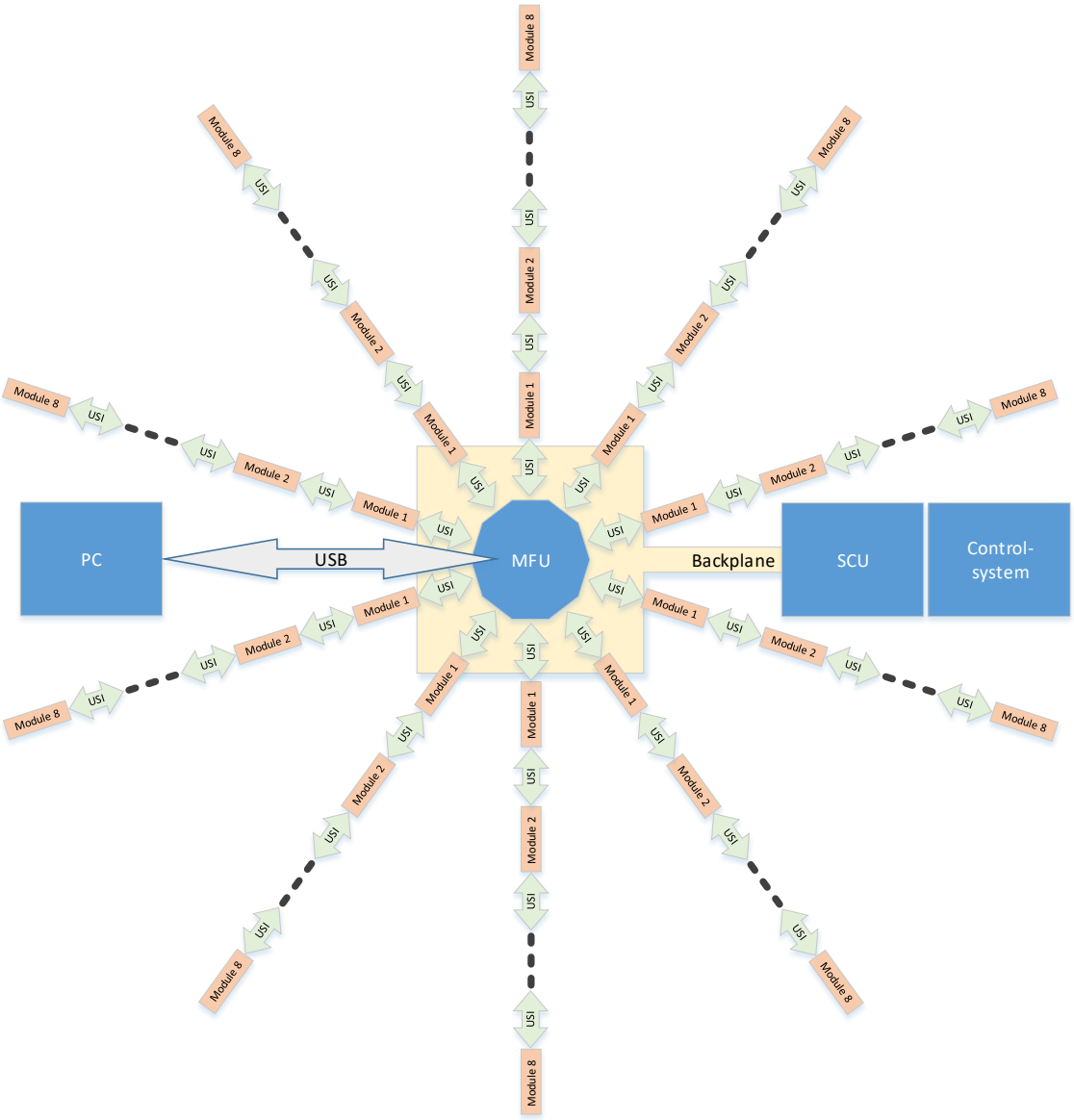


Abbildung 3.1: Verbindungen via USI zwischen MFU als Host und den Modulen im System

Durch USI ist der Aufbau einer Sternarchitektur rund um die MFU möglich. Die MFU unterstützt bis zu 10 externe USI Kanäle gleichzeitig.

3.3. Details

Im Nachfolgenden werden die einzelnen USI Bestandteile genauer beschrieben.

Da in einem ACU System Module i.d.R. an einer MFU als Host angeschlossen sind, wird nachfolgend diese Topologie betrachtet.

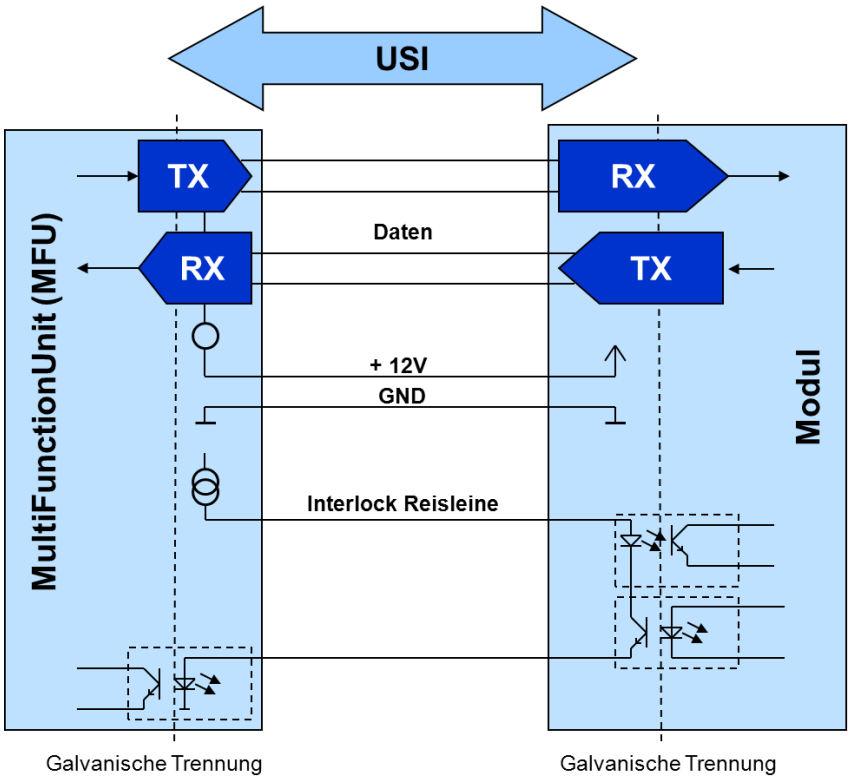


Abbildung 3.2: Detailaufbau des USI zwischen MFU und Modul

3.3.1. Stromversorgung für Sekundärseite des USI

Die galvanische Trennung des USI macht eine beidseitige Spannungsversorgung der Schnittstelle nötig. Die Primärseiten der USI Schnittstellen werden auf der MFU Backplane seitens der MFU Spannungsversorgung gespeist. Die Sekundärseite muss vom Modul mit Spannung versorgt werden. Dazu liefert mindestens ein Modul am USI über 2 Leitungen des Netzwerkkabels diese Versorgungsspannung. Auf der MFU Backplane werden 3,3 Volt benötigt, die mittels Längsregler gewonnen werden. Die vom Modul ausgegebene Spannung muss hoch genug sein um Leitungsverluste und den Spannungsabfall am Längsregler zu kompensieren. In Abbildung 3.2 liegt diese Spannung bei 12 Volt. Bei sehr kurzer Leitungsführung können durchaus 5 Volt ausrechend sein.

Hinweis: In USI 1.0 wurden diese beiden Leitungen für die Funktion „PowerOverUSI“ benutzt. Dabei wurde die Spannung von der MFU ausgegeben um daran angeschlossene Module evtl. darüber fremd zu speisen. Ab USI 1.1 ist die Spannungsrichtung also umgekehrt!

3.3.2. Summeninterlockleitung

Die Summeninterlockleitung dient dazu allen Modulen die mit der MFU verbunden sind zu ermöglichen das Netzgerät auszuschalten. Die Leitungen sind konstruktiv als Konstantstromquellen auf der MFU Backplane ausgeführt.

Wenn ein Modul einen Fehler in der Hardware entdeckt, wird über einen Optokoppler auf dem Modul die Leitung unterbrochen um allen Teilnehmern den Fehler mitzuteilen. Die MFU detektiert diese Unterbrechung und kommuniziert dies den übrigen Modulen, indem sie die Reißleinen aller USIs öffnet. Die eigentliche Abschaltung der SVE übernimmt i.d.R. das ICM.

Wichtig dieser Vorgang muss unabhängig von der Software des Gerätes funktionieren.

Alle Summeninterlockleitungen sind auf der MFU Backplane miteinander verknüpft. Dadurch sind z.B. auch Kabelbrüche detektierbar.

3.3.3. Datenversorgung

Das USI ist eine serielle Voll Duplex Schnittstelle die der Kommunikation der Module mit dem Host (i.d.R. die MFU) dient.

Auf dieser Schnittstelle werden sowohl Kommandos wie auch Daten oder Statusinformationen der Module gesendet und empfangen.

Die Schnittstelle ist eine reine Master-Slave Schnittstelle, d.h. Module können keine Informationen beim Host abfragen.

Die Schnittstelle ist elektrisch eine RS485, die im Falle der MFU die Module von dieser über iCoupler galvanisch trennt. Die Trennung erfolgt dabei auf dem Modul.

Die Bitrate ist beim Einschalten des Moduls per Standard auf 115,2kBit mit 8 Bit, 1 Stoppbit und kein Paritätsbit gesetzt. Anschließend fragte der Host beim Modul an welche Bitrate möglich ist und setzt diese dann, um das Modul mit der maximal möglichen Bitrate zu bedienen. Die möglichen Bitraten liegen zwischen 115,2kBit und 20 MBit.

3.3.4. Plug & Play an der MFU

Die USI Schnittstelle ist eine Plug & Play Schnittstelle, das heißt es kann jedes Modul auf eine beliebige USI Schnittstelle der MFU gesteckt werden. Die MFU erkennt automatisch durch Zuhilfenahme einer Systembeschreibung um welches Modul es sich handelt. Anschließend konfiguriert diese ihre internen Datenkanäle automatisch so, dass die Zuordnung mit der in der MFU hinterlegten Konfiguration übereinstimmt.

Die Systembeschreibung benötigt die MFU weil es möglich ist, dass mehrere Module gleicher Art im System vorhanden sind, diese aber unterschiedliche Aufgaben ausführen können. Die Systembeschreibung wird in der MFU in einem nicht flüchtigen Speicher hinterlegt, damit die MFU den gleichen Modulen die evtl. unterschiedlichen Aufgaben zuordnen kann.

Um diese Funktionalität zu gewährleisten muss jedes Modul über einen Standartregistersatz und Standartkommandos (MSP, FSP) verfügen.

3.3.5. Verbindung MFU ↔ Modul

Als Verbindung zwischen MFU und Modulen dienen Standard CAT 5 Netzwerkkabel mit Eins-zu-Eins-Verbindung.

4. Transferprinzip

4.1. *FIFO Startpunkte (FSP)*

FIFO Startpunkte (FSP) sind Speicher (Register) mit einer festgelegten Datentiefe an einem definierten Startpunkt. Die Datentiefe der FSP ist dabei als ein Vielfaches von „Byte“ definiert.

Ein FSP kann aus einem einzigen Byte bestehen, d.h. die festgelegte Speichertiefe ist 1.

Die Startpunkte der FSP beginnen bei Adresse 0. An dieser Stelle (FSP000) befindet sich die Modul Description Structur (MDS). Jedes Modul muss eine MDS unterstützen. Die MDS ist nur lesbar. Alle übrigen FSP, beginnend bei FSP001, wahlwiese nur gelesen oder geschrieben und rückgelesen werden. Eine entsprechende Information über die möglichen Datentransferrichtungen ist in der MDS im zugehörigen FSP-Descriptor hinterlegt. Ein Modul kann maximal 256 FSP besitzen. Daraus folgt, dass die Adresse eines FSP die Größe Byte besitzt. Die FSP000 bis FSP059 sind allgemeingültig und in ihrer Funktion weitestgehend vereinheitlicht. Ab FSP060 sind die FSP modulspezifisch und deshalb auch in den Moduldefinitonen zu beschrieben. Siehe hierzu auch das Dokument „ACU-FSPs_mUSIC“. In diesem sind die FSP der bisherigen Module beschrieben.

Bei der Adressierung eines Moduls über das USI wird neben der Modulnummer [1…8] auch noch eine eindeutige FSP-Nummer gesendet, wodurch der angesprochene FSP in einem Modul eindeutig adressierbar ist.

FSP Nummer	Startpunkt-Adresse	Tiefe	Richtung
000 (MDS)	0x00	modulabhängig	lesbar
001	0x01	3 Byte/24 Bit	lesbar
…			
010	0x0A	2 Byte/16 Bit	beschreibbar/rücklesbar

Tabelle 4.1: FSP Beispiele

Tabelle 4.1 zeigt 3 Beispiele für Modul FSPs. Das FSP000, die MDS ist an der Startpunkt-Adresse 0x00 zu finden. Die Tiefe dieses FSP ist modulabhängig, da dieses FSP als MDS das Modul selbst beschreibt (siehe hierzu Kapitel: 6 „MDS (Modul Descriptor Structure)“. Dieses FSP ist nur lesbar. Das FSP001 ist ebenfalls nur lesbar und befindet sich an Startpunkt-Adresse 0x01. Es hat in diesem Beispiel eine Tiefe von 3 Byte/24 Bit. D.h. bei jedem Lesevorgang werden 3 Byte/24 Bit Daten gesendet. Diese Daten können u. A. Statusinformationen des Moduls sein. Das FSP010 liegt an der Startpunkt-Adresse 0x0A, hat hier eine Tiefe von 2 Byte/16 Bit und ist beschreibbar und rücklesbar. Bei einem Schreibzugriff müssen 2 Byte/16 Bit Daten an dieses FSP gesendet werden. Bei einem Rücklesezugriff werden diese 2 Byte/16 Bit Daten zurück geliefert. Ein solches FSP dient z.B. zur Konfiguration des Moduls.

4.2. *Wichtige Vereinbarung*

Bei der Betrachtung von FSP in einem Modul wird die Richtung des Datentransfers an dieser Stelle immer aus Sicht des Moduls bestimmt.

D.h. Ausgabe (nur lesbare) FSP in Modulen haben Transferrichtung OUT, bidirektionale (beschreibbare/rücklesbare) FSP folglich INOUT.

FSP Nummer	Transferrichtung	Startpunkt-Adresse	Erklärung
FSP000*) [MDS]	OUT	0x00	Die FSP ist als unidirektional reserviert. D.h. dieser FSP kann vom Host nur gelesen werden und beinhaltet die MDS (Module Description Structure)..
FSP001*)	OUT	0x01	FSP001 dient allgemein als Modulstatus und kann nur Daten zum Host senden
FSP010*)	INOUT	0x0A	der zum FSP010 gehörende FIFO kann gelesen oder geschrieben werden und dient der Kommandoübermittlung
FSP064	OUT	0x40	FSP064 ist ein modulspezifischer FSP und kann vom Host nur gelesen werden.
…	…	…	…

Tabelle 4.2: Beispielhafte Übersicht

*) Die FSP000 bis FSP59 sind allgemeingültig und daher in ihrer Transferrichtung festgelegt.

4.3. FSP-Struktur im USI Modul

Bezugnehmen auf Tabelle 4.2 ist nachfolgend die FSP-Struktur im USI Modul dargestellt:

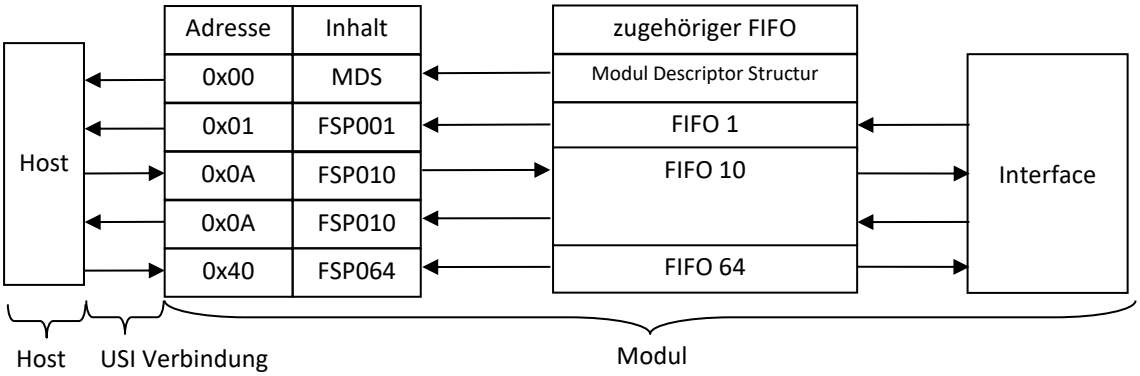


Abbildung 4.1: Speicheraufbau im USI Modul

Dieses Konzept hat den Vorteil, dass die Speichertiefe die sich hinter einem FSP verbirgt variiert werden kann, sofern die anfängliche FSP Speichertiefe für spätere Anwendungen nicht ausreichend sein sollte.

Der FSP000 beinhaltet die MDS (Modul Descriptor Structur). Die Speichertiefe ist variabel und abhängig von den darin befindlichen Standard Descriptoren.

FSP001 gehört zu FIFO 1, ist allgemeingültig, hat eine Speichertiefe von 3 Byte (24 Bit) und gehört z.B. zum Status eines ADC Moduls. Dieser FSP kann vom Host nur gelesen werden. Wichtig hierbei ist, dass das Modul, sofern der Host den FSP001 adressiert und lesen möchte, nachfolgend auch drei Byte als Daten sendet. D.h. der Host kann und muss einen FSP immer vollständig abfragen und empfangen.

FSP010 gehört zu FIFO 10, ist allgemeingültig, hat eine frei definierbare Speichertiefe von z.B. 2 Byte (16 Bit) und beinhaltet Modul Kommandos, die der Host schreiben und lesen kann. Wichtig hierbei ist, dass das der Host, sofern dieser den FSP010 adressiert und beschreiben möchte, nachfolgend auch 2 Byte als Daten sendet. Im Lesefall gilt gleiches wie zuvor, das Modul wird 2 Byte Daten senden, die der Host auch erwarten und empfangen muss.

FSP064 gehört zu FIFO 64, ist modulspezifisch, hat eine frei definierbare Speichertiefe von z.B. 15 Byte (120 Bit) und repräsentiert z.B. bestimmte modulspezifische Konfigurationsdaten. Dieser FSP kann vom Host nur gelesen werden.

4.4. Tiefe von FSP

Frei definierbare FSP können im Prinzip eine Tiefe von bis zu 4095 Byte haben. Die Zählung der Byte-tiefe beginnt mit 1 und endet bei 4095 (2^12 Bit), es sind also maximal (4MByte – 1 Byte) möglich.

Die FSP000 bis FSP059 sind vordefiniert in Bezug auf deren Funktionalität. Deren Tiefe ist den Modulbe-dingungen anpassbar.

5. Das USI Protokoll

5.1. Definition der Datenübertragungsrichtung

Die Richtung des Datentransfers wird nachfolgend immer aus Sicht des Host bestimmt.

FSP für Interfaces in Modulen die nur Daten empfangen können sind Empfänger-FSP. Als Beispiel sei ein DAC Modul genannt, das Daten vom Host empfangen und weiterverarbeiten soll. Seitens des Hosts wird dieser FSP beim Schreiben über ein OUT bedient. Im Modul selbst ist dieser FSP ein INOUT, weil sämtliche beschreibbaren FSP in den Modulen rücklesbar sind. Liest der Host das FSP des Moduls zurück tut er dies über ein IN, was den Transferkanal im Host letztlich ebenfalls zu einem INOUT werden lässt.

Soll der Host hingegen Daten von einem nur lesbaren Modul FSP empfangen, liegt im Modul ein OUT vor. Analog zum DAC ist hier als Beispiel der ADC, da dieser Daten von außen sammelt, wandelt und an den Host weiterleitet. Seitens des Host liegt also ein IN vor.

5.2. Protokolle der USI

Das USI Protokoll verfügt über 2 unterschiedliche Formate. Dies ist zum einen die „Normale USI“ oder auch „Standard USI“ Übertragung und zum anderen die USI Hochgeschwindigkeits- oder auch „HighSpeed USI“ Übertragung.

Bevor ein Datentransfer stattfinden kann müssen alle Module am USI eindeutig vom Host (i.d.R. der MFU) identifiziert sein, d.h. der Host muss über die notwendigen Informationen/Treiber verfügen um eine Kommunikation mit den Modulen durchführen zu können.

An dieser Stelle wird davon ausgegangen, dass alle Module am USI erkannt und erfolgreich vom Host (der MFU) in das System eingebunden wurden.

Informationen über das Suchen und Finden von Modulen am USI können Kapitel 7: „Enumeration und Plug & Play“ ab Seite 32 entnommen werden.

5.3. Datenformat

Die Übertragung sämtlicher Daten erfolgt in ASCII d.h. im Klartext lesbar. Dies soll die Diagnose im Fehlerfall erleichtern. Die Übertragung als interpretierbares ASCII Zeichen bedingt allerdings auch, dass sämtliche Daten als Byte übertragen werden.

5.3.1. ASCII oder Hex

Modulintern werden alle Daten i.d.R. hexadezimal verarbeitet. Dies bedeutet, dass z.B. ein FSP mit der Speichertiefe von 4 Byte (32 Bit) im Modul die Daten $0xCA38F8F6_H$ in der Form vorhalten wird wie die nachfolgende Abbildung zeigt.

4. Byte (MSB)				3. Byte				2. Byte				1. Byte (LSB)			
C				A				3				8			
1	1	0	0	1	0	1	0	0	0	1	1	1	0	0	0
31			28	27			24	23			20	19			16

F				8				F				6			
1	1	1	1	1	0	0	0	1	1	1	1	0	1	1	0
15			12	11			8	7			4	3			0

Abbildung 5.1: modulinterne FSP Datenverwaltung

Auf dem USI werden diese Daten aber als ASCII Zeichen übertragen. Um dies zu realisieren werden zum Transfer stets 4 Bit (1 Nibble) der FSP Daten zusammengefasst und in ein ASCII Zeichen konvertiert.

Diese 4 Bit können 16 verschiedene hexadezimale Zustände annehmen, wobei jeder dieser Zustände ein hexadezimales ASCII Zeichen repräsentiert.

Nibble		ASCII	
Binär	Hex	Zeichen	Hex
0000	0x0	0	0x30
0001	0x1	1	0x31
0010	0x2	2	0x32
0011	0x3	3	0x33
0100	0x4	4	0x34
0101	0x5	5	0x35
0110	0x6	6	0x36
0111	0x7	7	0x37
1000	0x8	8	0x38
1001	0x9	9	0x39
1010	0xA	A	0x41
1011	0xB	B	0x42
1100	0xC	C	0x43
1101	0xD	D	0x44
1110	0xE	E	0x45
1111	0xF	F	0x46

Tabelle 5.1: Nibble in hexadezimal als ASCII Zeichen in hexadezimal

Daraus folgt, dass die 32 Bit hexadezimal im FSP vorgehaltenen Daten auf dem USI als hexadezimale ASCII Zeichen in der folgenden Form übertragen werden

C	A	3	8	F	8	F	6
0x43	0x41	0x33	0x38	0x46	0x38	0x46	0x36

Abbildung 5.2: Beispiel Datenübertragung in ASCII

Es ist ersichtlich, dass sich das Datenaufkommen im Verhältnis zu deren Inhalt durch die Konvertierung der Daten von hexadezimalen Nibble in hexadezimale ASCII Zeichen verdoppelt.

Um den Dekodieraufwand zu minimieren wird vereinbart, dass alle Buchstaben im hexadezimalen ASCII Code prinzipiell Großbuchstaben sein müssen. (A, B, C, D, E, F).

5.4. Steuerzeichen

Ein USI Paket besteht nicht nur aus ASCII Daten, sondern auch aus ASCII Steuerzeichen. Diese signalisieren z.B. den Beginn und das Ende eines Pakets.

ASCII Steuerzeichen	Typ	hexadezimal	Bemerkung	USI Betrieb
STX	Start of Text	0x02	Start einer Übertragung	Normal/ HighSpeed
ETX	End of Text	0x03	Ende einer Übertragung	
ENQ	Enquiry	0x05	Start eines Tunneltransfer	Tunnelmarker im HighSpeed Paket
EOT	End of Transmission	0x04	Ende eines Tunneltransfer	
ACK	Acknowledge	0x06	Das Modul bestätigt eine erfolgreiche Datenübertragung	Normal/ HighSpeed
NAK	No Acknowledge	0x15	Das Modul bestätigt die Übertragung als fehlgeschlagen. Anschließend folgt ein Fehlercode.	

Tabelle 5.2: ASCII Steuerzeichen

5.5. *Package Identifier*

Einer Sende-, bzw. Empfangsanforderung an ein USI Modul wird seitens des Host ein eindeutiger Package-Identifier (PID) vorangestellt. Dadurch signalisiert der Host, ob er Daten vom Modul empfangen oder an das Modul senden will.

Ein PID besteht aus einem 16 Bit Identifikationsfeld das den PID als ASCII Zeichen in Klartext enthält.

PID-Name	Typ	PID Darstellung		Bemerkung
		ASCII Zeichen	hexadezimal	
RD	Read	RD	0x52 0x44	Host erwartet Daten von einem Modul
WR	Write	WR	0x57 0x52	Host sendet Daten an ein Modul

Tabelle 5.3: PID Übersicht

5.6. *Prüfsummenbildung*

Um die Korrektheit der übertragenen Daten zu verifizieren wird den Daten bei der Übertragung eine Prüfsumme angehängt. Diese Prüfsumme bildet ein exklusives Oder (XOR). Dabei werden alle ASCII Datenbytes XOR verknüpft und das daraus resultierende Ergebnis als 1 Byte Hexadezimalwert in Form von 2 Byte hexadezimal ASCII als Prüfsumme ausgegeben.

Nachfolgende Tabelle 5.4 zeigt das Beispiel einer Prüfsummenbildung bei 8 ASCII Byte Nutzdaten. Zu beachten ist, dass die Daten zwar als ASCII Zeichen vorliegen, diese aber hexadezimal betrachtet werden. Durch die XOR Verknüpfung der Daten entsteht ein 1 Byte hexadezimales Ergebnis welches aber als 2 Byte hexadezimale ASCII Zeichen gesendet wird.

Wichtig: Die Prüfsumme wird ausschließlich aus den Nutzdaten gebildet, d.h. weder Steuerzeichen noch PID, USI- oder Moduladressen und FSP Adressierungen befinden sich in der Prüfsumme.

Daten		Prüfsummenbildung in hexadezimal (Summe XOR nächster hex. ASCII Wert)							Prüfsumme ASCII
Hex.	ASCII in Hex.								
1	0x31	0x03							
		XOR							
2	0x32		0x30						
			XOR						
3	0x33			0x04					
				XOR					
4	0x34				0x31				
					XOR				
5	0x35					0x37			
						XOR			
6	0x36						0x30		
							XOR		
7	0x37							0x08	
							XOR		
8	0x38								
								Ergebnis	0x30 0x38

Tabelle 5.4: Beispiel einer Prüfsummenbildung

5.7. USI Pakete bei Normaler/Standard USI Übertragung

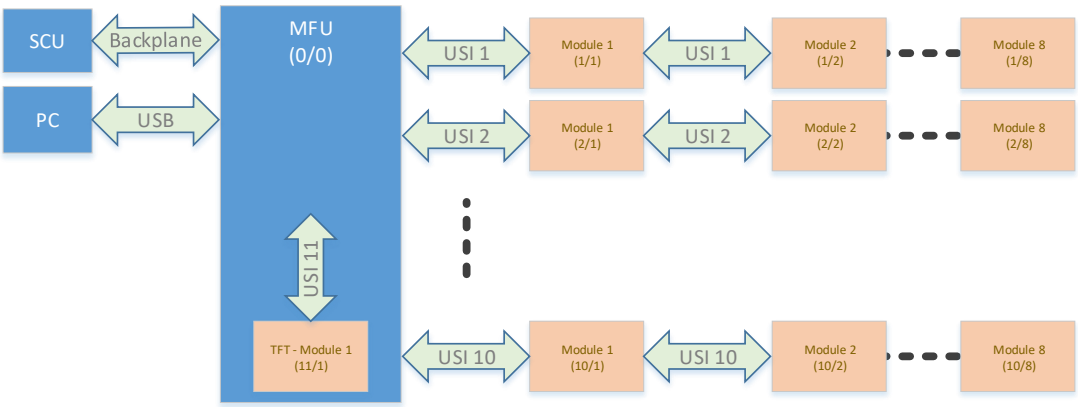


Abbildung 5.3: ACU Topologie

Die obige Abbildung zeigt eine typische ACU Topologie.

Die normale oder auch Standard USI Übertragung findet zwischen der MFU und den einzelnen Teilnehmern an den USI Bussen, aber auch zwischen dem PC und der MFU statt.

Abbildung 5.4: Direkte Verbindung eines PC mit Modulen

Es ist ebenso möglich mittels USB/USI Adapterkabel einen PC direkt mit einem/mehreren Modul(n) zu verbinden. In diesem Fall wird i.d.R. ebenfalls die Standard USI Übertragung angewendet.

5.7.1. Phasen einer Standard USI Transaktion zwischen Host und angeschlossenem Teilnehmer

In Abbildung 5.3 bildet i.d.R. die MFU den Host. Ist ein PC oder ein übergeordnetes Kontrollsystem mittels SCU angebunden und wollen diese auf Module über USI zugreifen bilden diese den Host. In diesem Fall ist die MFU quasi als Modul am USI zu betrachten.

In Abbildung 5.4 bildet der PC den Host.

5.7.1.1. Direkter Datentransfer zwischen Host und Modul

Nachfolgend wird der direkte Datentransfer über USI zwischen der MFU als Host (Abbildung 5.3) oder einer PC↔Modul Direktverbindung (Abbildung 5.4), mit einem PC als Host, über ein USB/USI Kabel betrachtet. Der Datentransfer vom PC über die MFU an ein Modul wird in Kapitel 5.7.1.2: „Datentransfer zwischen PC/Kontrollsystem und ACU System“ beschrieben.

Jeder Datentransfer über das USI besteht aus mehreren Phasen.

In der ersten Phase wird vom Host ein Transfer explizit initiiert, da nur dieser einen Transfer starten darf.

Dies geschieht, in dem er ein StartOfText (STX) und einen nachfolgenden WR/RD PID sendet.

Anschließend folgen die Moduladresse und die Nummer/Adresse des zu adressierenden FSP.

Möchte der Host Daten empfangen wird die erste Phase hier vom Host beendet. Dazu sendet er ein EndOfText (ETX) und wartet auf Antwort des Moduls.

Möchte der Host Daten senden folgen in der zweiten Phase nun die Daten mit denen der FSP im Zielmodul beschrieben werden soll. Wichtig hierbei ist, dass die Datenlänge in jedem Fall der FSP Tiefe entsprechen muss. Ist der Ziel FSP z.B. 4 Byte tief, folgen nun 8 ASCII-Zeichen. Jedes Zeichen enthält dabei ein Daten-Nibble). Dies ist die zweite Phase.

Die zweite Phase wird abgeschlossen durch das Senden einer Prüfsumme. Mit einem anschließenden ETX.

Sofern der Host Daten gesendet hat, ist die dritte Phase die Bestätigung des Moduls, dass es alle Daten einwandfrei erhalten hat.

Will der Host hingegen Daten empfangen, besteht die zweite Phase aus der Antwort des Moduls. Dieses leitet die zweite Phase mit dem Senden von STX ein. Anschließend folgen die Moduladresse und die Nummer/Adresse des FSP. Daran schließen sich die Daten des FSP an. Die Phase wird abgeschlossen durch das senden der Prüfsumme und ETX.

5.7.1.1.1. Daten vom Host an ein Modul senden

Format: STX WR MA FSP DD... PP ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"WR" Ist das Kommando um dem Modul mitzuteilen das Daten geschrieben werden. Das PID als 2 Byte ASCII

"MA" Die Moduladresse in Hex als 1 Byte ASCII

"FSP" Das gewünschte FSP in Hex als 2 Byte ASCII

"DD..." mind. 2 Byte ASCII welche in das zuvor adressierte FSP geschrieben werden sollen. Sofern dem FSP ein FIFO mit einer Speichertiefe größer 1 zugehörig ist, werden die folgenden ASCII Byte unmittelbar anschließend übertragen.

"PP" Prüfsumme, die Prüfsumme bildet sich aus der XOR Verknüpfung der Datenworte, wird in HEX dargestellt und als 2 Byte ASCII übertragen.

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync.	PID	MA	FSP	DD...	PP	Sync.
STX	WR	1	1	87654321	08	ETX
0x02	0x57 0x52	0x31	0x30 0x31	0x38 0x37 0x36 0x35 0x34 0x33 0x32 0x31	0x30, 0x38	0x03

Tabelle 5.5: Senden von Hostdaten an ein Modul

Beispiel "STX WR 1 01 87654321 08 ETX"

Wenn die Übertragung ordnungsgemäß war antwortet das Modul mit "ACK". Siehe hierzu Kapitel 5.7.1.1.2: "Modul bestätigt den erfolgreichen Empfang der Daten**Fehler! Verweisquelle konnte nicht gefunden werden.**".

Antwortet das Modul mit "NAK" (siehe hierzu Kapitel 5.7.1.1.3: "Modul hat die Daten nicht erfolgreich empfangen") oder sollte die Rückmeldung ausbleiben muss der Host nach einer Wartezeit einen erneuten Versuch starten die Daten zu senden.

5.7.1.1.2. Modul bestätigt den erfolgreichen Empfang der Daten

Format: STX MA ACK FSP ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"MA" Die Modul Adresse in Hex als 1 Byte ASCII

"ACK" Ist das Steuerzeichen um dem Host mitzuteilen, dass die Daten vom Modul verstanden und verarbeitet worden sind, als 1 Byte ASCII.

"FSP" Das gewünschte FSP in Hex als 2 Byte ASCII

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync.	MA	ACK	FSP	Sync.
STX	1	6	1	ETX
0x02	0x31	0x06	0x30 0x31	0x03

Tabelle 5.6: Modul bestätigt den erfolgreichen Empfang der Daten

5.7.1.1.3. Modul hat die Daten nicht erfolgreich empfangen

Format: STX NAK MA ERR ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"MA" Die Moduladresse in Hex als 1 Byte ASCII

"NAK" Ist das Steuerzeichen um dem Host mitzuteilen, dass die Daten vom Modul nicht verstanden und/oder nicht verarbeitet worden sind, als 1 Byte ASCII.

"ERR" Fehlercode in Hex als 2 Byte ASCII

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync.	MA	NAK	ERR	Sync.
STX	1	21	04	ETX
0x02	0x31	0x15	0x30 0x34	0x03

Tabelle 5.7: Modul hat die Daten nicht erfolgreich empfangen

Tabelle 5.7 zeigt die Rückmeldung eines Moduls welches Daten nicht erfolgreich empfangen hat und die Fehlernummer 4 zurück sendet. Alle verfügbaren Fehlernummern können der Tabelle 5.15 auf Seite 21 entnommen werden.

5.7.1.1.4. Host will Daten von einem Modul lesen

Format: STX RD MA FSP ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"RD" Ist das Kommando um dem Modul mitzuteilen das Daten gelesen werden sollen. Das PID als 2 Byte ASCII

"MA" Die Moduladresse in Hex als 1 Byte ASCII

"FSP" Das gewünschte FSP in Hex als 2 Byte ASCII

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync	PID	MA	FSP	Sync.
STX	RD	1	1	ETX
0x02	0x52 0x44	0x31	0x30 0x31	0x03

Tabelle 5.8: Senden von Hostdaten an ein Modul

Beispiel "STX RD 1 01 ETX"

Wenn die Übertragung ordnungsgemäß war antwortet das Modul mit den angeforderten Daten.

Antwortet das Modul mit "NAK" (siehe hierzu Kapitel 5.7.1.1.3: "Modul hat die Daten nicht erfolgreich empfangen") oder sollte die Rückmeldung ausbleiben muss der Host nach einer Wartezeit einen erneuten Versuch starten die Anforderung zu senden.

5.7.1.1.5. Daten vom Modul an den Host senden

Vorbedingung ist, dass der Host Daten beim Modul angefordert hat, siehe Kapitel 5.7.1.1.4: "Host will Daten von einem Modul lesen".

Format: STX MA FSP DD... PP ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"MA" Die Moduladresse in Hex als 1 Byte ASCII

"FSP" Das gewünschte FSP in Hex als 2 Byte ASCII

"DD..." mind. 2 Byte ASCII welche aus dem zuvor adressierte FSP gesendet werden. Sofern dem FSP ein FIFO mit einer Speichertiefe größer 1 zugehörig ist, werden die folgenden ASCII Byte unmittelbar anschließend übertragen.

"PP" Prüfsumme, die Prüfsumme bildet sich aus der XOR Verknüpfung der Datenworte, wird in HEX dargestellt und als 2 Byte ASCII übertragen.

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync	MA	FSP	DD...	PP	Sync
STX	1	1	87654321	08	ETX
0x02	0x31	0x00, 0x31	0x38 0x37 0x36 0x35 0x34 0x33 0x32 0x31	0x30 0x38	0x03

Tabelle 5.9: Senden von Moduldaten an einen Host

Beispiel "STX 1 01 87654321 08 ETX"

5.7.1.2. Datentransfer zwischen PC/Kontrollsystem und ACU System

Nachfolgend wird der Datentransfer über USI zwischen einem PC, bzw. dem Kontrollsystem als Host (Abbildung 5.3) und den ACU System beschrieben. Der direkte Datentransfer wischen einem Host und einem Modul wird in Kapitel 5.7.1.1: „Direkter Datentransfer zwischen Host und Modul“ beschrieben.

Der Datentransfer zwischen PC/Kontrollsystem und ACU System erfolgt analog dem direkten Datentransfer zwischen Host und Modul.

Der einzige Unterschied besteht darin, dass die MFU in diesem Falle als Gateway fungiert und die Daten des PC/Kontrollsystems entweder an die adressierte USI und das daran adressierte Modul weiterleitet oder bei sich behält, sofern sie selbst der Empfänger ist.

Zu diesem Zweck ist das USI Paket um ein weiteres ASCII Zeichen erweitert. Vor der Moduladresse erfolgt zusätzlich die Adressierung der USI, bzw. der MFU selbst.

Die Transferphasen gestalten sich dabei wie folgt:

In der ersten Phase wird vom PC/Kontrollsystem ein Transfer explizit initiiert, da nur dieser/dieses einen Transfer starten darf. Dies geschieht, indem ein StartOfText (STX) und ein nachfolgendes WR oder RD PID gesendet wird.

Jetzt folgt die Gatewayadresse (GW). Dies ist entweder eine USI Nummer (1_D..11_D, $0x1_H$..$0x9_H$, $0xA_H$, $0xB_H$) oder eine 0_D ($0x0_H$) sofern die MFU der Empfänger ist.

Anschließend folgen die Moduladresse und die Nummer/Adresse des zu adressierenden FSP. Sind sowohl GW als auch MA = 0_D ($0x0_H$) wird die MFU adressiert

Möchte der PC/das Kontrolsystem Daten empfangen wird die erste Phase hier vom PC/Kontrollsystem beendet. Dazu wird ein EndOfText (ETX) gesendet und auf Antwort des ACU Systems gewartet.

Diese Antwort erfolgt, indem die MFU, sofern eine USI adressiert wurde, die Anfrage an das adressierte Modul weiterleitet und dessen Antwort umgekehrt an den PC/das Kontrollsystem schickt. Die MFU fungiert also als Gateway.

Möchte der PC/das Kontrollsystem Daten senden folgen in der zweiten Phase nun die Daten mit denen der FSP im Zielmodul beschrieben werden soll. Wichtig hierbei ist, dass die Datenlänge in jedem Fall der FSP Tiefe entsprechen muss. Ist der Ziel FSP z.B. 4 Byte tief, folgen nun 8 ASCII-Zeichen. Jedes Zeichen enthält dabei ein Daten-Nibble.

Die zweite Phase wird abgeschlossen durch das senden einer Prüfsumme. Mit einem anschließenden ETX.

Sofern der PC/das Kontrollsystem Daten gesendet hat, ist die dritte Phase die Bestätigung der MFU, bzw. Moduls an welches die MFU die Daten weitergeleitet hat, dass alle Daten einwandfrei empfangen wurden, bzw. eine Fehlermeldung, wenn dies nicht zutrifft.

Will der PC/das Kontrollsystem hingegen Daten empfangen, besteht die zweite Phase aus der Antwort der MFU, bzw. des Moduls an welches die MFU die Anfrage weitergeleitet hat. Die MFU, bzw. das Modul leitet die zweite Phase mit dem Senden von STX ein. Anschließend folgt als GW die USI Nummer (1_D..11_D, 0x1_D..0x9_H, 0xA_H, 0x0B_H) oder eine 0_D (0x0_H) sofern die MFU adressiert wurde. Danach die Moduladresse (MA) und die Nummer/Adresse des FSP. Daran schließen sich die Daten des FSP an.

Die Phase wird abgeschlossen durch das senden einer Prüfsumme und ETX.

5.7.1.2.1. Daten vom PC/Kontrollsystem an das ACU System senden

Format: STX WR GW MA FSP ZZ... PP ETX

"STX"	StartOfText, Paket Start als 1 Byte ASCII
"WR"	Ist das Kommando um der MFU, bzw. einem Modul mitzuteilen das Daten geschrieben werden. Das PID als 2 Byte ASCII
"GW"	Die Gatewayadresse als 1Byte ASCII
"MA"	Die Moduladresse in Hex als 1 Byte ASCII
"FSP"	Das gewünschte FSP in Hex als 2 Byte ASCII
"DD..."	mind. 2 Byte ASCII welche in das zuvor adressierte FSP geschrieben werden sollen. Sofern dem FSP ein FIFO mit einer Speichertiefe größer 1 zugehörig ist, werden die folgenden ASCII Byte unmittelbar anschließend übertragen.
"PP"	Prüfsumme, die Prüfsumme bildet sich aus der XOR Verknüpfung der Datenworte, wird in HEX dargestellt und als 2 Byte ASCII übertragen.
"ETX"	EndOfText, Paket Ende als 1 Byte ASCII

Sync.	PID	GW	MA	FSP	DD...	PP	Sync.
STX	WR	1	1	01	87654321	08	ETX
0x02	0x5752	0x31	0x31	0x00, 0x31	0x38 0x37 0x36 0x35 0x34 0x33 0x32 0x31	0x30 0x38	0x03

Tabelle 5.10: Senden von Daten an das ACU System

Beispiel "STX WR 1 1 01 87654321 08 ETX"

Wenn die Übertragung ordnungsgemäß war antwortet die MFU, bzw. das Modul mit "ACK". Siehe hierzu Kapitel 5.7.1.2.2: "MFU, bzw. Modul bestätigt den erfolgreichen Empfang der Daten"

Antwortet die MFU, bzw. das Modul mit "NAK" (siehe hierzu Kapitel 5.7.1.2.3: "MFU, bzw. Modul hat die Daten nicht erfolgreich empfangen") oder sollte die Rückmeldung ausbleiben muss der PC/das Kontrollsystem nach einer Wartezeit einen erneuten Versuch starten die Daten zu senden.

5.7.1.2.2. MFU, bzw. Modul bestätigt den erfolgreichen Empfang der Daten

Format: STX GW ACK MA FSP ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"GW" Die Gatewayadresse als 1Byte ASCII

"MA" Die Moduladresse in Hex als 1 Byte ASCII

"ACK" Ist das Steuerzeichen um dem PC/Kontrollsystem mitzuteilen, dass die Daten von der MFU, bzw. vom Modul verstanden und verarbeitet worden sind, als 1 Byte ASCII.

"FSP" Das gewünschte FSP in Hex als 2 Byte ASCII

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync.	GW	MA	ACK	FSP	Sync.
STX	1	1	6	01	ETX
0x02	0x31	0x31	0x06	0x30 0x31	0x03

Tabelle 5.11: MFU, bzw. Modul bestätigt den erfolgreichen Empfang der Daten

5.7.1.2.3. MFU, bzw. Modul hat die Daten nicht erfolgreich empfangen

Format: STX GW NAK MA ERR ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"GW" Die Gatewayadresse als 1Byte ASCII

"MA" Die Moduladresse in Hex als 1 Byte ASCII

"NAK" Ist das Steuerzeichen um dem PC/Kontrollsystem mitzuteilen, dass die Daten von der MFU, bzw. vom Modul nicht verstanden und/oder nicht verarbeitet worden sind, als 1 Byte ASCII.

"ERR" Fehlercode in Hex als 2 Byte ASCII

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync.	GW	MA	NAK	ERR	Sync.
STX	1	1	21	04	ETX
0x02	0x31	0x31	0x15	0x30 0x34	0x03

Tabelle 5.12: MFU, bzw. Modul hat die Daten nicht erfolgreich empfangen

Tabelle 5.12 zeigt die Rückmeldung der MFU, bzw. Moduls welche Daten nicht erfolgreich empfangen hat und die Fehlernummer 4 zurücksendet. Alle verfügbaren Fehlernummern können der Tabelle 5.15 auf Seite 21 entnommen werden.

5.7.1.2.4. PC/Kontrollsystem will Daten von der MFU, bzw. einem Modul lesen

Format: STX RD GW MA FSP ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"GW" Die Gatewayadresse als 1Byte ASCII

"RD" Ist das Kommando um der MFU, bzw. einem Modul mitzuteilen das Daten gelesen werden sollen. Das PID als 2 Byte ASCII

"MA" Die Moduladresse in Hex als 1 Byte ASCII

"FSP" Das gewünschte FSP in Hex als 2 Byte ASCII

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync	PID	GW	MA	FSP	Sync.
STX	RD	1	1	01	ETX
0x02	0x52 0x44	0x31	0x31	0x00 0x31	0x03

Tabelle 5.13: Lesen von Daten

Beispiel "STX RD 1 1 01 ETX"

Wenn die Übertragung ordnungsgemäß war antwortet die MFU, bzw. das Modul mit den angeforderten Daten.

Antwortet die MFU, bzw. das Modul mit "NAK" (siehe hierzu Kapitel 5.7.1.2.3: "MFU, bzw. Modul hat die Daten nicht erfolgreich empfangen") oder sollte die Rückmeldung ausbleiben muss der PC/das Kontrollsystem nach einer Wartezeit einen erneuten Versuch starten die Anforderung zu senden.

5.7.1.2.5. Daten von der MFU, bzw. einem Modul an den PC/das Kontrollsystem senden

Vorbedingung ist, dass der PC/das Kontrollsystem Daten bei der MFU, bzw. einem Modul angefordert hat, siehe Kapitel 5.7.1.2.4 „PC/Kontrollsystem will Daten von der MFU, bzw. einem Modul lesen“".

Format: STX GW MA FSP DD... PP ETX

"STX" StartOfText, Paket Start als 1 Byte ASCII

"GW" Die Gatewayadresse als 1Byte ASCII

"MA" Die Moduadresse in Hex als 1 Byte ASCII

"FSP" Das gewünschte FSP in Hex als 2 Byte ASCII

"DD..." mind. 2 Byte ASCII welche aus dem zuvor adressierte FSP gesendet werden. Sofern dem FSP ein FIFO mit einer Speichertiefe größer 1 zugehörig ist, werden die folgenden ASCII Byte unmittelbar anschließend übertragen.

"PP" Prüfsumme, die Prüfsumme bildet sich aus der XOR Verknüpfung des/der Datenworts/e, wird in HEX dargestellt und als 2 Byte ASCII übertragen.

"ETX" EndOfText, Paket Ende als 1 Byte ASCII

Sync.	GW	MA	FSP	DD...	PP	Sync.
STX	1	1	1	87654321	08	ETX
0x02	0x31	0x31	0x00, 0x31	0x38 0x37 0x36 0x35 0x34 0x33 0x32 0x31	0x30 0x38	0x03

Tabelle 5.14: Senden von Daten an den PC/das Kontrollsystem

Beispiel "STX 1 1 01 87654321 08 ETX"

5.7.2. Fehlernummern

Schlägt ein USI Transfer fehl und das Modul sendet eine NACK Antwort, wird dieser eine Fehlernummer angehängt. Die möglichen Fehlernummern lauten hierbei wie folgt:

Fehlernummer	ASCII	Bemerkung
87_D / 57_H	0x35 0x37	FSP nicht vorhanden
85_D / 55_H	0x35 0x35	Datenlänge stimmt nicht mit FSP Tiefe überein
80_D / 50_H	0x35 0x30	Keine Daten erlaubt, FSP soll gelesen werden, PID war „RD“
04_D / 04_H	0x30 0x34	PID nicht verstanden, MA stimmt aber PID ist nicht bekannt
75_D / 75_H	0x37 0x35	Fehler bei der ASCII nach Hex Konvertierung, Daten nicht ASCII
83_D / 53_H	0x35 0x33	FSP kann nur gelesen und nicht beschrieben werden
12_D / 12_H	0x31 0x32	Fehler bei der CRC Bildung

Tabelle 5.15: Fehlernummern

5.8. USI Pakete bei Hochgeschwindigkeits-/HighSpeed USI Übertragung

Sollen Daten sehr schnell von einem Modul an die MFU oder umgekehrt kommuniziert werden, können Hochgeschwindigkeits-USI Pakete dafür verwendet werden. Diese Kommunikation ist der USI zwischen der MFU und den Modulen vorbehalten und findet zwischen dem PC/Kontrollsystem und dem ACU System nicht statt. Dafür wird nur das Standard USI Protokoll genutzt.

5.8.1. Format des HS USI Paket

Ein Hochgeschwindigkeitspaket kann die gesamte USI Bandbreite für sich in Anspruch nehmen und hat dabei das folgende Format:

Format: STX HSByte1 HSByte2 HSByte3 HSByte4 Tunnelbyte Strobe/Ack ETX

"STX"	StartOfText, Paket Start als 1 Byte ASCII
"HSByte1"	Das Erste von 4 hexadezimalen Datenbytes welches mit jedem Paket übertragen wird, als 2 Bytes hexadezimales ASCII Zeichen
"HSByte2"	Das Zweite von 4 hexadezimalen Datenbytes welches mit jedem Paket übertragen wird, als 2 Bytes hexadezimales ASCII Zeichen
"HSByte3"	Das Dritte von 4 hexadezimalen Datenbytes welches mit jedem Paket übertragen wird, als 2 Bytes hexadezimales ASCII Zeichen
"HSByte4"	Das Vierte von 4 hexadezimalen Datenbytes welches mit jedem Paket übertragen wird, als 2 Bytes hexadezimales ASCII Zeichen
"Tunnelbyte"	Ein Datenbyte, eingebettet in ein HS USI Paket zur Einbettung einer Standard USI Kommunikation innerhalb des HS USI Paketes, als 1 Byte hexadezimales ASCII Zeichen
"Strobe/Ack"	Enthält Strobe bzw. Acknowledge Informationen als Handshakesignale zum „Tunnelbyte“, als 1 Bytes hexadezimales ASCII Zeichen. Zusätzlich können hier zwei weitere Nutzbit enthalten sein.
"ETX"	EndOfText, Paket Ende als 1 Byte ASCII

Ein Hochgeschwindigkeitspaket besteht in der Summe aus 12 Byte. Jedes Byte wird durch ein Start Bit gestartet und durch ein Stoppbit beendet. Bei 20 MBit Übertragungsgeschwindigkeit bedeutet dies, das ein komplettes Paket in 50ns * 10 Bit * 12 Byte = 6us übertragen wird.

5.8.2. Die HSByte

Die vier hexadezimalen HSByte_ (HighSpeedByte) werden mit jedem Paket übertragen. Genau genommen handelt es sich dabei sogar um 8 Byte, da durch die Wandlung der hexadezimalen Daten in hexadezimale ASCII Zeichen zur Übertragung jedes hexadezimale Datenbyte zu 2 hexadezimalen ASCII Zeichen wird. Jedes dieser hexadezimalen ASCII Zeichen enthält also stets nur ein Nibble eines hexadezimalen Datenbytes.

In der Regel werden in diesen Bytes Werte übertragen, die der Host oder der Slave in sehr rascher Folge mit aktuellen Werten für die weitere Verarbeitung benötigt. Dies sind z.B. ADC Werte eines schnellen ADC Moduls zur Erfassung des Strom-Istwertes. Oder Steuersätze vom Host für die IGBT Steuerung. Durch die Limitierung auf 4 Byte sind bei jedem Transfer 32 Bit Daten, Status u. ä. übertragbar. Zusätzlich stehen 2 weitere Bit im Strobe/Ack zur Verfügung die ebenfalls als Datenbit benutz werden können. Dies erhöht den Transfer auf 34 Bit.

5.8.3. Das Tunnelbyte

Damit die MFU mit einem Modul auch bei Hochgeschwindigkeitsbetrieb noch kommunizieren kann, wird innerhalb des HS USI Paketes ein Tunnelbyte übertragen. Dieses Tunnelbyte enthält ein ASCII Byte einer regulären Standard USI-Anfrage an einen Modul FSP, bzw. umgekehrt ein ASCII Byte einer regulären Standard USI-Antwort eines Moduls. Es wird also eine konventionelle USI Standard Kommunikation innerhalb eines HS USI Paketes getunnelt.

Da das HS USI Paket allerdings schon vom STX "StartOfText" eingeleitet und auch mittels ETX "EndOfText" beendet wird benutzt der Tunnel für die Startidentifikation das ASCII Zeichen ENQ (0x05) anstelle STX und zur Abschlussidentifikation EOT (0x04) anstelle ETX.

Da sichergestellt werden muss, dass das Modul von der MFU getunnelte Byte auch verstanden hat und umgekehrt, folgt das Strobe/Acknowlege Byte auf das Tunnelbyte. Dieses dient als Handshake für die Tunnelkommunikation.

5.8.4. Das Strobe/Acknowlege Byte

Dieses Byte wird in jedem Stream als ein Byte ASCII übertragen.

Legt der Sender einen neuen Wert in das Tunnelbyte, setzt dieser auch das Strobe Bit und teilt dem Empfänger dadurch mit, dass sich der Wert des Tunnelbyte geändert hat. Der Sender wartet bis der Empfänger das Acknowledge Bit als Handshakesignal gesetzt hat und dadurch signalisiert, dass der neue Wert abgeholt wurde. Der Sender nimmt das Strobe Bit zurück. Dadurch löscht der Empfänger auch das Acknwoledge Bit. Erst dann ändert der Sender den Inhalt des Tunnelbyte und setzt das Strobe Bit erneut.

Durch diesen Vorgang ist die Kommunikation im Tunnel natürlich wesentlich langsamer als der Rest des Streams. Es wird aber sichergestellt, dass Sender und Empfänger sicher Standard USI Pakete innerhalb eines HS USI Pakets zur Kommunikation weiterer Informationen übertragen können.

Ein entscheidender Vorteil ist die dabei durchführbare asynchrone Kommunikation innerhalb des HS USI Pakets.

Da das Strobe/Acknowledge Byte auf dem USI ein 1 Byte hexadezimales ASCII Zeichen ist, also nur ein 4 Bit Nibble repräsentieren kann, ist es auch nur möglich diese 4 Bit innerhalb der Module zu verwenden.

Für die Strobe/Acknowledge Bit werden die beiden unteren Bit[0] (Strobe) und [1] (Acknowledge) verwendet, daraus folgt, dass das übertragende hexadezimale ASCII Zeichen für Storbe/Acknowledge nur die Zustände 0x30, 0x31 oder 0x32 annehmen kann, sofern die beiden frei belegbaren Bit (2) und (3) nicht verwendet werden und immer ‚0' sind. Diese frei belegbaren Bit (2) und (3) können als ergänzende Datenbit der HSBytes verwendet werden.

Die nachfolgende Tabelle 5.16 betrachtet das Strobe/Acknowledge Bytes aus Sicht eines Senders als hexadezimales Nibble.

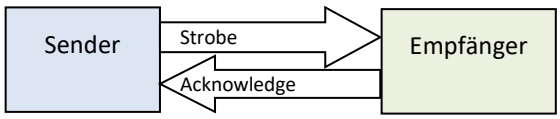


Abbildung 5.5: Das Strobe/Acknwoledge Byte aus Sicht des Senders

Bit	Bedeutung (aus Sendersicht)
3	frei belegbar
2	frei belegbar
1	Wenn '1' → Acknowledge empfangen, d.h. der Empfänger hat ein neues von einem Sender mittels Tunnel übertragenen Datenbyte abgeholt und bestätigt dies dem Sender.
0	Wenn '1' → Strobe senden, d.h. der Sender hat ein neues Tunnelbyte angelegt und teilt dem Empfänger mit, dass sich dieser das Byte abholen kann.

Tabelle 5.16: Details des Strobe/Acknowledge Bytes aus Sicht des Senders in HEX

Aus Sicht eines Empfängers verhält sich das Strobe/Acknowledge Bytes analog wie Tabelle 5.17 zeigt.

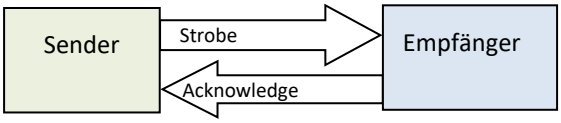


Abbildung 5.6: Das Strobe/Acknwoledge Byte aus Sicht des Empfängers

Bit	Bedeutung (aus Empfängersicht)
3	frei belegbar
2	frei belegbar
1	Wenn '1' → Acknowledge senden, d.h. der Empfänger hat das vom Sender mittels Strobe angekündigte Datenbyte abgeholt und bestätigt dies dem Sender.
0	Wenn '1' → Strobe empfangen, d.h. der Empfänger bekommt vom Sender mitgeteilt, dass ein neues Tunneldatenbyte zur Abholung bereit steht.

Tabelle 5.17: Details des Strobe/Acknowledge Bytes aus Sicht eines Moduls in HEX

Die nachfolgende Abbildung 5.7 zeigt schematisch welche Bit dabei im Handshakebyte bei USI HighSpeed Betrieb gesetzt werden. Zu beachten ist, dass jeder Teilnehmer durch den Vollduplexbetrieb jeweils zeitgleich ein Handshakebyte empfangen und senden kann.

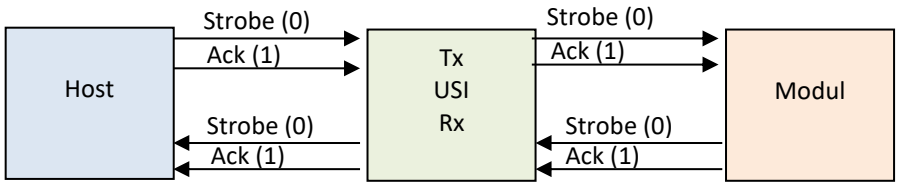


Abbildung 5.7: Voll-Duplex Handshake bei High-Speed USI (Bit)

5.8.4.1. Freie Bit im Strobe/Acknowledge Byte

Wie zuvor beschrieben werden innerhalb des Strobe/Acknowledge HEX Nibble nur die beiden unteren Bit für die Handshakesignale genutzt. Die beiden oberen Bit sind frei belegbar und können im Prinzip als 2 Bit Erweiterung der 4 HSBytes benutzt werden. Zu beachten ist dabei allerdings, dass dadurch das übertragene Handshake ASCII-Byte auch weitere Wert außer 0x30, 0x31 und 0x32 annehmen kann. Werden diese zusätzlichen Bit nicht genutzt, ist es daher sinnvoll diese fest auf GND (‚0') zu legen.

5.8.5. *Beispiel eines Hochgeschwindigkeits- USI Pakets*

Die Daten der HSByte1..4 können sich bei jedem Stream ändern und sind daher nicht explizit angeführt.

#	Richtung	Sync	HSByte1	HSByte2	HSByte3	HSByte4	TunnelByte	Stb./Ack.	Sync
			Jeweils 2 hexadezimale ASCII Zeichen						
Innerhalb des HS USI Pakets sendet der Sender ein ENQ an den Empfänger, dieser kann innerhalb des Vollduplexbetriebs natürlich nicht im gleichen Moment antworten, daher sind das Tunnelbyte und Stb./Ack noch nicht gesetzt.									
1	S → E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	ENQ (0x05)	Strobe (0x31)	ETX (0x03)
	S ← E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	0x30	ETX (0x03)
Der Empfänger bestätigt den Empfang der Tunnelanfrage mit einem Ack.									
2	S → E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	ENQ (0x05)	Strobe (0x31)	ETX (0x03)
	S ← E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	Ack (0x32)	ETX (0x03)
Der Sender nimmt darauf das Strobe zurück									
3	S → E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	ENQ (0x05)	0x30	ETX (0x03)
	S ← E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	Ack (0x32)	ETX (0x03)
Der Sender sendet das erste Byte des Lese PID.									
4	S → E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	R (0x52)	Strobe (0x31)	ETX (0x03)
	S ← E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	0x30	ETX (0x03)
Der Empfänger antwortet mit einem Ack.									
5	S → E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	R (0x52)	Strobe (0x31)	ETX (0x03)
	S ← E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	Ack (0x32)	ETX (0x03)
Der Sender nimmt darauf das Strobe zurück									
6	S → E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	R (0x52)	0x30	ETX (0x03)
	S ← E	STX (0x02)	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	Ack (0x32)	ETX (0x03)
Der Sender sendet das zweite Byte des Lese PID									
7	S → E	STX	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	D (0x44)	Strobe (0x31)	ETX (0x03)
	S ← E	STX	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	0x30	ETX (0x03)
Der Empfänger antwortet mit Ack.									
8	S → E	STX	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	R (0x44)	Strobe (0x31)	ETX (0x03)
	S ← E	STX	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	Ack (0x32)	ETX (0x03)
usw. Der Sender beendet die Anfrage an den Slave mittels EOT									
..	S → E	STX	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	EOT	0x30	ETX
..	S ← E	STX	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0xYY, 0xYY	0x30	Ack	ETX
Der Empfänger sendet jetzt auf die gleiche Weise die angeforderten Daten und der Sender bestätigt den einwandfreien Empfang des jeweiligen Byte mittels Ack.									

Tabelle 5.18: Beispiel eines HS USI Pakets

6. MDS (Modul Descriptor Structure)

Jedes Modul muss für den Host eindeutig identifizierbar sein. Hierzu dient die MDS. Diese ist in eine Reihe einzelner Standard Descriptoren unterteilt. Zusammengefasst bilden die Standard-Descriptoren die MDS. Die MDS liegt immer FSP000 (0_D bzw. $0x00_H$, bzw. $0x30,0x30_{ASCII}$).

Für die MFU als Host gilt: da zu jedem unterstützen Modul weitere Informationen über dessen Nutzen in der MFU hinterlegt sein muss, dient die MDS der grundlegenden Information mit deren Hilfe die MFU entscheiden kann, ob das Modul unterstützt wird oder nicht.

6.1. Vereinbarung

Um die nachfolgende Beschreibung übersichtlich und einfach zu gestalten vereinbaren wir, dass alle Registerwerte in Dezimal oder Hexadezimal (0x..) angegeben werden. Bei der Übertragung über das USI werden diese aber in ASCII übertragen. Daher muss die Schnittstelle der Module oder des Host eine Wandlung der Werte von Hex nach ASCII, bzw. Dez. nach ASCII vornehmen. D.h. der Wert 1_D, bzw $0x1_H$ bedeutet in ASCII ein Byte mit dem Wert $0x31_A$, wohingegen der Wert 255_D, bzw. $0xFF_H$ zwei Byte mit den Werten $0x46_A$, $0x46_A$ bedingen würde.

Ob ein 1 Byte Hex-Wert letztlich als ein oder 2 ASCII Byte übertragen wird, steht in der jeweiligen Descriptorenbeschreibung innerhalb der Spalte Länge.

6.2. Standard Descriptoren

Ein einzelner Standard Descriptor der MDS besteht aus einer größeren Anzahl von 8 Bit Feldern in denen, durch die USI Spezifikation festgelegt, Eigenschaften des Moduls codiert sind.

Es werden zurzeit 3 verschiedene Standard Descriptoren unterschieden:

Deskriptortyp	Code	Bemerkungen
Modul-Descriptor	0x1	einer pro Modul, beginnt immer bei Adresse 0x0000 innerhalb der MDS
FSP-Descriptor	0x2	geben die unterstützten Startpunkte an
End Descriptor	0x9	Descriptoren Ende

Tabelle 6.1: Standard-Descriptoren

Um ein Plug and Play zu realisieren, muss der Host ein neu an das USI angeschlossenes Modul identifizieren können. Die MDS dient hierbei der Kurzbeschreibung der physikalischen und logischen Eigenschaft eines Moduls. Diese MDS wird vom Host während der Enumeration durch eine Standardabfrage (GetMDS) abgefragt.

Ein USI Modul besitzt genau einen Modul-Descriptor, der die globalen Informationen über das Modul enthält.

Für jeden FSP im Modul muss ein FSP-Descriptor vorhanden sein, außer für den FSP000 (die MDS). Jeder FSP-Descriptor beschreibt genau einen der vorhandenen FSP eines Moduls. Für FSP000 (die MDS) gibt es **keinen** FSP_Descriptor. Hat ein Modul z.B. 25 FSP (inkl. FSP000) werden hier nur 24 FSP beschrieben.

Der End Descriptor gibt das explizite Ende der MDS an.

6.3. Lage der MDS und deren Aufbau

Die MDS liegt immer im FSP000.

Möchte der Host die MDS lesen, sendet er eine Leseanforderung an das Modul mit dem Startpunkt FSP000 (0x00). Daraufhin sendet das Modul den gesamten Inhalt des MDS. Dieser muss im Host interpretiert werden und dazu führen, dass im Host, sofern er das Modul unterstützt, die nötigen Treiberinformationen zur Verfügung gestellt werden.

An erster Stelle befindet sich immer der Modul-Descriptor. D.h. der erste Eintrag der MDS muss immer 0x1 sein um den Modul-Descriptor eindeutig zu identifizieren. Ist der Wert nicht 0x1 wird das Modul verweigert. Im nachfolgenden Eintrag steht der Offset bis zum nächsten Descriptor. D.h. wenn der Modul-Descriptor nachfolgend noch 62 Byte Einträge hat steht an dieser Stelle der Offsetwert 62_D bzw. $0x3E_H$.

Offset	Wert	Beschreibung
0x0000	0x1	ID Modul-Descriptor
0x0001	0x3	MSB Offset bis zum nächsten Descriptor
0x0002	0xE [62]	LSB Offset bis zum nächsten Descriptor
...	...	
0x0010	0xy	letzter Eintrag
0x0011	0x2	ID FSP-Descriptor
0x0012	0x0	MSB Offset bis zum nächsten Descriptor
0x0012	0x6	LSB Offset bis zum nächsten Descriptor
...	...	...
0x0018	0xy	letzter Eintrag
0x0019	0xF	ID End-Descriptor

Tabelle 6.2: Prinzipieller Aufbau der MDS.

6.4. Interpretationszyklus der MDS für den Host

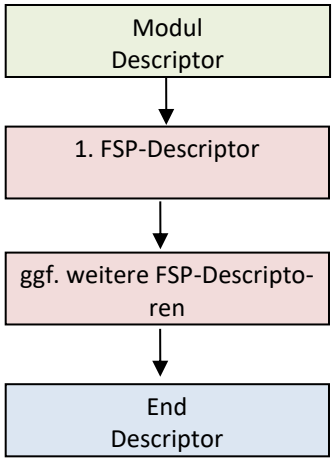


Abbildung 6.1: Zeitlicher Ablauf für das Lesen der MDS

6.5. Beschreibung der Standard Descriptoren

6.5.1. Modul-Descriptor

Jedes USI Modul enthält genau einen Modul-Descriptor. Dieser beinhaltet grundlegende Informationen über das Modul wie z.B. eine Vendor-ID, Produkt- und Release-ID.

Offset	Feldbezeichnung	Länge in Nibble	Beschreibung	Beispiel in Nibble
0	bDescriptorType	1	Descriptor Typ = Modul-Descriptor	0x1
1	wOffset	2	Offset bis zum nächsten Descriptor in Hex	0x46
3	wModuleClass	2	Modul-Klassen-Code	0x01
5	wModuleSubClass	2	Modul-Subklassen-Code	0x00
7	sSerial	12	Serienummer des Moduls	0x123456789ABC
19	wVendorID	2	Vendor-ID des Herstellers	0x01
21	lProductID	4	Produkt-ID des Herstellers	0x0001
25	wHWMajorRelease	2	Haupt Release-Nr. der Hardware (z.B. 2) Vorkommastelle	0x02
27	lHWMinorRelease	4	Unter Release-Nr. der Hardware (z.B. 3) Nachkommastelle	0x0003
31	sHWDate	6	Release Datum der Hardware (21.11.07) TTTTMMMMYYYY	0x230410
37	wFWMajorRelease	2	Haupt Release-Nr. der Firmwarware (z.B. 2) Vorkommastelle	0x02
39	lFWMinorRelease	4	Unter und Sub Release-Nr. der Firmware (z.B. 3.1) Nachkommastelle	0x0301
43	sFWDate	6	Release Datum der Firmware (21.11.17) TTTTMMMMYYYY	0x211117
49	wUSI	2	USI-Spec.Vers. in BCD (z.B.: 1.1) MSB Vorkommastelle LSB Nachkommastelle	0x11
51	bDeviceMaxSpeed	1	Maximale USI Geschwindigkeit des Moduls (3 Bit)	0x1
52	wAttributes	2	ggf. weitere erforderliche Attribute dieses Moduls	0x00
54	lMaxPower	4	I-Aufnahme des Moduls in 2mA Einheiten	0x0010
58	lNmbsOfFSPs	4	Anzahl unterstützter FSP, ohne MDS (FSP000)	0x001E
62	sDescription	beliebig	Modulbeschreibung als Textstring	

Tabelle 6.3: Modul Descriptor

Nachfolgende Modulklassen *wVendorID* werden unterschieden

Hersteller	Bedeutung
1	GSI
2	Danfysik LEM
3	WME
4	I-Tech
	...

Tabelle 6.4: Hersteller

Nachfolgende Modulklassen *wModuleClass* werden unterschieden

Klasse	Bedeutung
0	MFU
1	ADC Modul
2	DAC Modul
3	Digital Interlock Modul
4	Analog Interlock Modul
5	Interlocksystem
6	Load Switch – Lastumschaltermodul, basiert auf Interlocksystem
7	Climate Control System – Klimakontrollsystem, basiert auf Interlocksystem
8	TFT Display
9	Static Converter Module
10	Parallel Feeder Module
11	Water Interlock Module
12	ADC DAC IO Module
13	ADC Dan Physik
14	WME PSU Module
15	SER-Module
16	TS1MU1
17	Zero-Field-Control
20	ACU Testsystem
22	Analog Interlock Modul
23	DoseDeliverySystem Module (Therapie)
24	ACM Module (Ampulz Module)
25	ACU Monitoring Module
26	High Speed Data Channel (Dummy module)
27	ICM_SigmaPhi_QD
28	ICM_SigmaPhi_IngD
29	SR_Injection_Septa
30	FPGA_QuD
31	ADC 24 Bit Class 1
...	
255	Sondermodul

Tabelle 6.5: Unterschiedene Modulklassen

Im Weiteren können in den einzelnen Klassen Unterklassen *wModuleSubClass* definiert sein.

SubKlasse	Bedeutung
0	MFU LE/SE
...	
...	

Tabelle 6.6: Modulunterklassen für MFU

SubKlasse	Bedeutung
0	keine SubKlasse
1	10 Bit
2	12 Bit
3	14 Bit
4	16 Bit
5	18 Bit
6	20 Bit
7	22 Bit
8	24 Bit
…	…

Tabelle 6.7.: Modulunterklassen für ADC, DAC Module

SubKlasse	Bedeutung
0	Slave USI Nr.1
1	Slave USI Nr.2
2	Slave USI Nr.3
3	Slave USI Nr.4

Tabelle 6.8: Modulunterklassen für Static Converter Module

wDeviceMaxSpeed wird als 3 Bit Information übermittelt. Im Resetfall wird immer 115,2kBit als Geschwindigkeit gewählt.

[2..0]	ASCII	Bitrate	
000	0x30	25 MBit	
001	0x31	20 MBit	
010	0x32	16,6 MBit	
011	0x33	10 MBit	
100	0x34	5 MBit	
101	0x35	2 MBit	
110	0x36	1 MBit	
111	0x37	*115,2 kBit*	*Reset*

Tabelle 6.9: Unterstützte Bitraten

wAttributes wird als Bitmap interpretiert und dient weiteren modulspezifischen Angaben.

Bitposition	Bedeutung
7	0-FactoryImage 1-ApplicationImage
6...2	n.u.
1	0-Modul unterstützt nur den Normalen USI Modus 1-Modul unterstützt auch den HighSpeed Tunnel Modus
0	0-Modul erwartet über Standard USI KEINE Kommandos 1-Modul erwartet über Standard USI ebenfalls Kommandos

Tabelle 6.10: Attribute

6.5.2. *FSP-Descriptor*

Es folgt eine Anzahl von FSP-Descriptoren die dem Eintrag „lNmbsOfFSPs“ entsprechen. Dabei enthält jeder Descriptor alle Informationen die der Host benötigt um klar zu unterscheiden welche der im Treiber hinterlegten FSP vom Modul wirklich benutzt und zur Verfügung gestellt werden.

Offset	Feldbezeichnung	Länge in Nibble	Beschreibung	Beispiel in Nibble
0	bDescriptorType	1	Descriptor Typ= FSP-Descriptor	0x2
1	wOffset	2	Offset bis zum nächsten Descriptor	0x08
3	lStartpointAttributes	8	Attribute vom gefundenen FSP	0x0020A01

Tabelle 6.11: FSP-Descriptor

Bitposition	Bedeutung
31..28	FSP Nummer (ext.) z. Z. nicht benutzt
27..20	FSP Nummer (0..255)
19..16	FSP Tiefe (256..4095)
15..8	FSP Tiefe (0..255)
7..1	n.u.
0	0 = FSP ist les-/ und beschreibbar 1 = FSP ist nur lesbar (ReadOnly)

Tabelle 6.12: FSP Startpunkt Attribute

FSP Nr. (ext)				FSP Nr.								FSP Tiefe (ext)				FSP Tiefe								n.u.							
31			28	27							20	19			16	15							8	7						1	0

Abbildung 6.2: Aufbau im Bitmuster

6.5.3. *End Descriptor*

Dieser Descriptor beschreibt das eindeutige Ende der MDS. Er darf nur ein einziges Mal implementiert sein und muss am Ende der MDS stehen.

Vorgeschrieben ist nur der Eintrag *bDescriptorType*, daran muss der Host eindeutig das Ende der MDS erkennen und *wOffset*.

Der Eintrag *sEndOfDescriptor* ist optional und nicht zwingend, dann muss *wOffset* allerdings den Wert 0 enthalten.

Offset	Feldbezeichnung	Länge in Nibble	Beschreibung	Beispiel in Nibble
0	bDescriptorType	1	Descriptor Typ= EndDescriptor	0x9
1	wOffset	2	Offset bis zum Ende	0x03
1	sEndOfDescriptor	3	Klartextstring "EOD"	EOD

Tabelle 6.13: End Descriptor

7. Enumeration und Plug & Play

Enumeration beschreibt die Identifizierung und ggf. Konfiguration eines USI Moduls, das neu an das USI angeschlossen wird. Das Anstecken und Abziehen eines Moduls wird durch den Host detektiert. Der Host übernimmt dann die Enumeration.

Das Anstecken und Abziehen eines Moduls kann, muss aber nicht im spannungsfreien Zustand geschehen.

Da der Host die Enumeration der Module in der Regel aber nach einer Spannungszuschaltung durchführt ist ein Neustart des Systems nach einer Konfigurationsänderung der Module sinnvoll. Andererseits kann der Host auch per Befehl zur Neusuche der Module veranlasst werden.

7.1. Genereller Ablauf

Ein USI Modul besitzt entweder einen Kodierschalter mit dem eine eindeutige Moduladresse eingestellt werden kann. Oder besitzt intern eine feste Modulnummer. Die Zählung von USI-Nummern, als auch Modulnummern beginnen für den Benutzer immer mit 1. Da bis zu 8 Module an einem USI angebunden werden können lauten die einstellbaren Moduladressen 0x1...0x8.

Bei Hochgeschwindigkeitsmodulen, deren Bandbreite ein USI für sich alleine beansprucht, kann die Moduladresse (in der Regel 0x01) auch fest implementiert sein.

Der Benutzer muss vor dem anstecken des USI Moduls an das USI darauf achten, dass an dem ausgewählten USI Zweig noch kein Modul steckt, dass die gleiche Moduladresse aufweist. Sollte dieser Fall eintreten muss der Host dies bemerken und den Benutzer darüber informieren indem er einen eindeutigen Fehler am USI signalisiert.

Um nun eine Plug & Play Funktionalität zu gewährleisten, muss der Host bei der Systeminitialisierung alle USI Schnittstellen abfragen und die Moduldescriptoren der ggf. daran befindlichen Module anfordern.

Hierzu sendet der Host eine Leseanforderung auf das USI wie in Kapitel 5.7.1.1.4 "Host will Daten von einem Modul lesen" auf Seite 17 beschrieben.

Die Anforderung auf dem USI mit der MFU als Host lautet hierbei:

STX - RD - Moduladresse – FSP000 - ETX

0x02 - 0x52 0x44 - 0x3y - 0x30 0x30 - 0x03

y = Modulnummer von 1...8.

Da an einem USI mehrere Module angeschlossen sein können, muss noch die eindeutige Moduladresse "y" angegeben werden. Diese Moduladresse wird bei jedem Zyklus um eins erhöht und beginnt nach Erreichen der maximal zulässigen Modulanzahl wieder bei null.

Auf die Anfrage muss ein Modul mit dem senden des Moduldescriptors reagieren.

Dabei wird zunächst der einleitende Header gesendet:

STX - MA - FSP -MDS... - PP - ETX

0x02 - 0x3y - 0x30 0x30 ..... PP ETX

..... = Inhalt des MDS.

Der Host wertet die MDS aus und analysiert ob er einen Treiber für dessen Unterstützung parat hat. Trifft dies zu, wird das Modul akzeptiert.

Es werden der Reihe nach alle USI Schnittstellen auf Ihre Module hin getestet.