Salzbrenner Media

Documentation UART-SPI-CPLD

Verfasser:

Wjatscheslaw Rein

Änderungshistorie

Version	Verfasser	Änderungsbeschreibung	Freigabedatum
1.0	Wjatscheslaw Rein	Initiale Erstellung	13.06.2022
1.1	Wjatscheslaw Rein	Zwischenstand	15.07.2022
1.2			

Inhaltsverzeichnis

1. Einführung

1.1 Übersicht

OSI-Schichtenmodells ermöglicht eine Kommunikation über unterschiedlichste technische Systeme hinweg zu beschreiben und die Weiterentwicklung zu begünstigen. Dazu sind aufeinanderfolgende Schichten (engl. layers) mit jeweils entsprechend zugeschnittenen Aufgaben. In unserem Beispiel sind nur drei Schichten dargestellt. Die Daten werden schrittweise verschachtelt und mit den Service - Informationen (Header, CRC usw.) versehen.

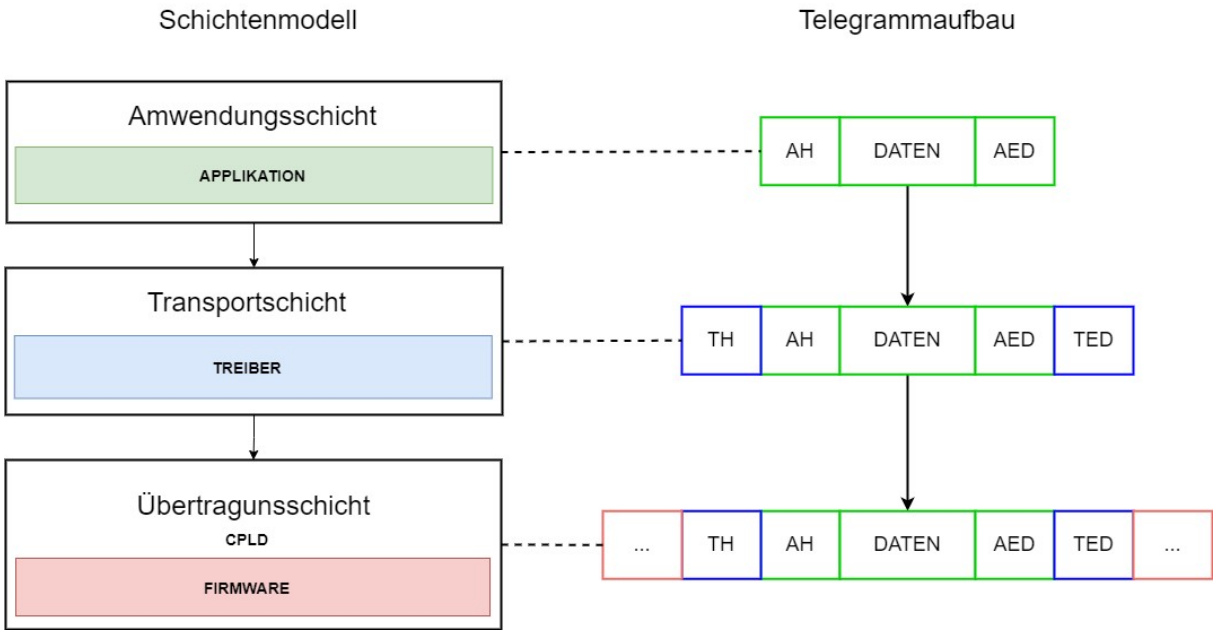

Abbildung 1.1-1: OSI Schichtenmodell. Ein Ausschnitt.

Legende:

AH – Anwendungsschicht Header

AED – Anwendungsschicht *End Delimiter* (den Abschluss eines Telegramms)

TH – Transportschicht Header

TED – Transportschicht *End Delimiter*

1.1.1 Anwendungsschicht

Anwendungsschicht ist in USER-Space von Betriebssystem Linux angesiedelt und beinhaltet alle Benutzer-Programme, die eine logische höhere Ebene präsentieren und für die Datenverarbeitung verantwortlich sind. Nur in diesem Schicht haben die Daten eine "Bedeutung" und werden von entsprechenden Applikationen (deswegen "application layer" genannt) interpretiert. Hier ist die Vielfältigkeit der möglichen Varianten von Programen quasi unbegrenzt und ist nur durch eine Schnittstelle zur Transportschicht bestimmt.

1.1.2 Transportschicht

Transportschicht repräsentiert ein Kernel-Space von Betriebssystem und schließt sehr spezifische, meistens auf Hardware abgestimmte Programme ins sich ein, die eine Brücke (so genannte devices) zwischen logischen und physikalischen Welt darstellen. Das sind die Treibers, die eine vordefinierte, öfters von Betriebssystem vorgeschriebene, Schnittstelle besitzen. Die device können z.B. geöffnet (OPEN) bzw. geschlossen (CLOSE) werden, sowie die Daten per (READ/WRITE) in beide Richtungen transferieren. Die spezifischen Funktionen werden von IO Control (IOCTL) unterstützt.

1.1.3 Übertragungsschicht

Übertragungsschicht ist durch Complex Programmable Logic Device (CPLD) vertretet und stellt eine direkte Kommunikation mit den Geräten her. Durch in Firmware implementierte Algorithmus werden von Transportschicht empfangene Daten verarbeitet und an den physikalische Teilnehmer weitergeschickt. Genauso funktioniert der Datenfluss in eine andere Richtung, und zwar von der Übertragungsschicht über Transportschicht an die Anwendungsschicht.

1.2 Ziel

Ziel der vorliegenden Dokumentation ist eine grobe Prozessbeschreibung zu geben und die wichtigen Eckpunkte festzuhalten. Sie hat kein Anspruch auf Vollständigkeit, bietet nur beschränkt detaillierte Beschreibung und dient nicht als Bedienungsanleitung. Auf vertiefte Informationen, die in beiliegenden Unterordner vorhanden sind, wird verwiesen.

2. Prozessbeschreibung

2.1 Anwendungsschicht – Transportschicht

Da die Schnittstelle zur Transportschicht fest definiert und nur allein durch den Treiber (*UART-TTY*) bestimmt ist, werden hier mögliche Varianten vom Datentransfer zwischen beide Ebenen kurz vorgestellt. Außerdem ist es wichtig eine synchrone Übertragung von einer asynchrone zu unterscheiden. Synchron bedeutet zeitgleich bzw. der Reihen nach ((zeitlich) folgend, sequenziell) und asynchron heißt quasi parallel, von scheduling der Betriebssystem gesteuert (siehe Abbildung 2.1-1).

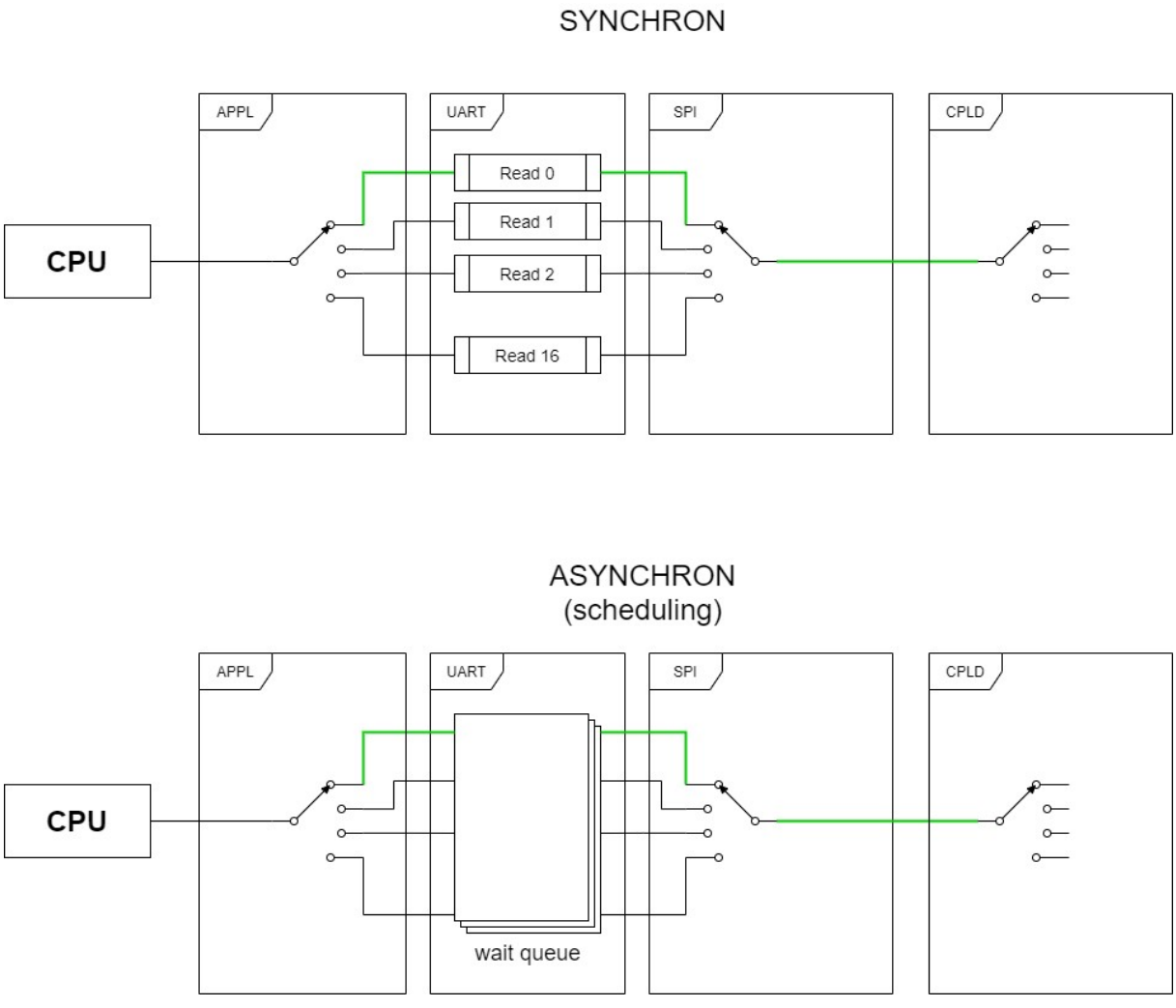

Abbildung 2.1-2 Synchrone und asynchrone Konzept

Die entwickelte UART-SPI Treiber, der auf Transportebene agiert, ist in bestehende UART-Treibersystem von Linux integriert (siehe Abbildung 2.1-2) und nutzt zum Teil die eingebaute Funktionen sodass ein Nutzer gar kein Unterschied zu den "echten" TTY-Treibern erkennt. Ausgenommen sind die spezifischen Abläufe, die eine Hardware bezogene Konfiguration benötigen. Eine USER-Applikation arbeitet mit dem Treiber, als ob es ein UART Device wäre. Ein UART-Treibersystem von Linux ist äußerst umfangsreich und komplex (siehe dazu die Bilder in Unterordner: .../Bilder) und fordert das Anhalten von bestimmten Regeln.

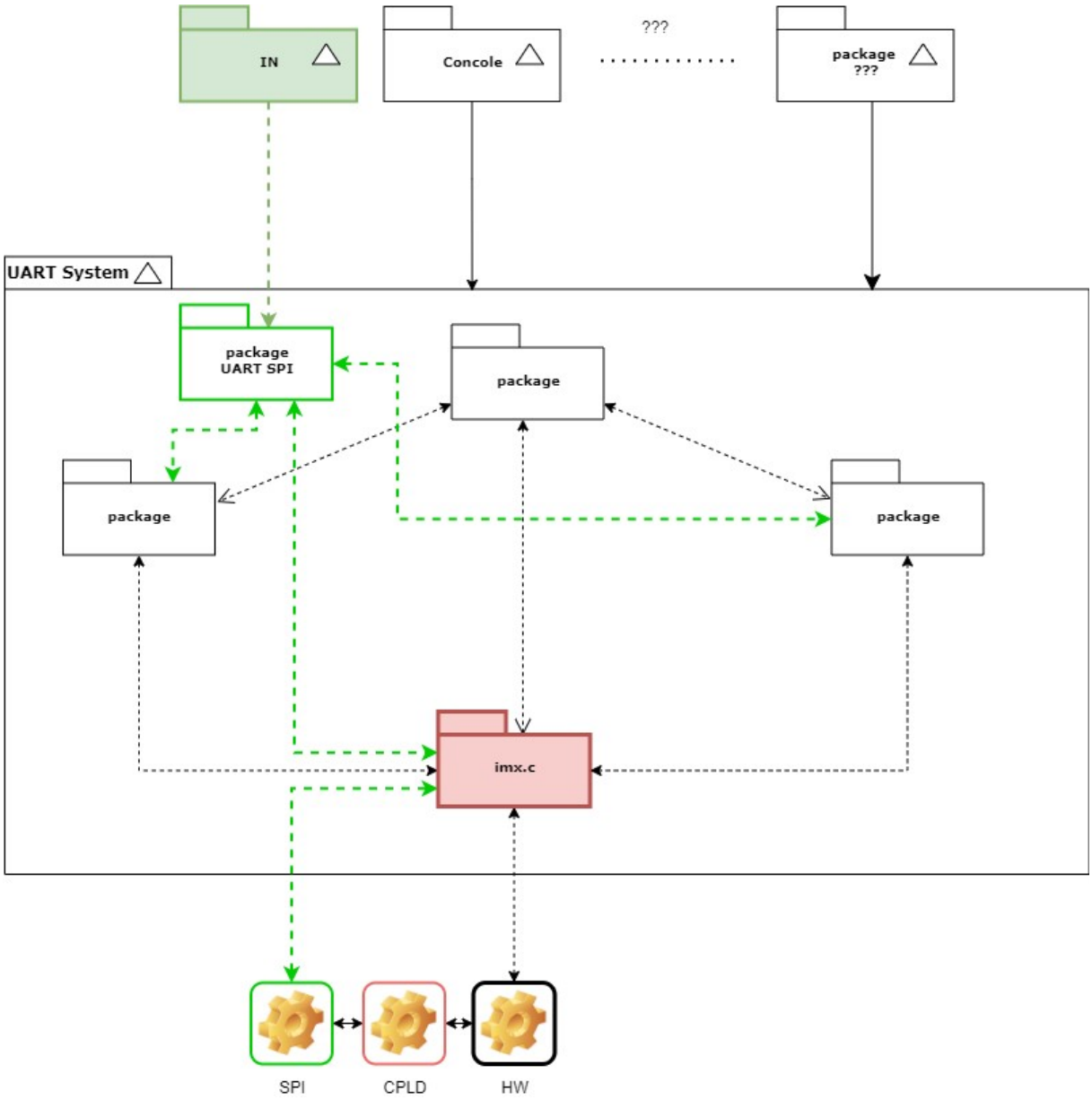

Abbildung 2.1-2: UART-SPI Integration in Treibersystem

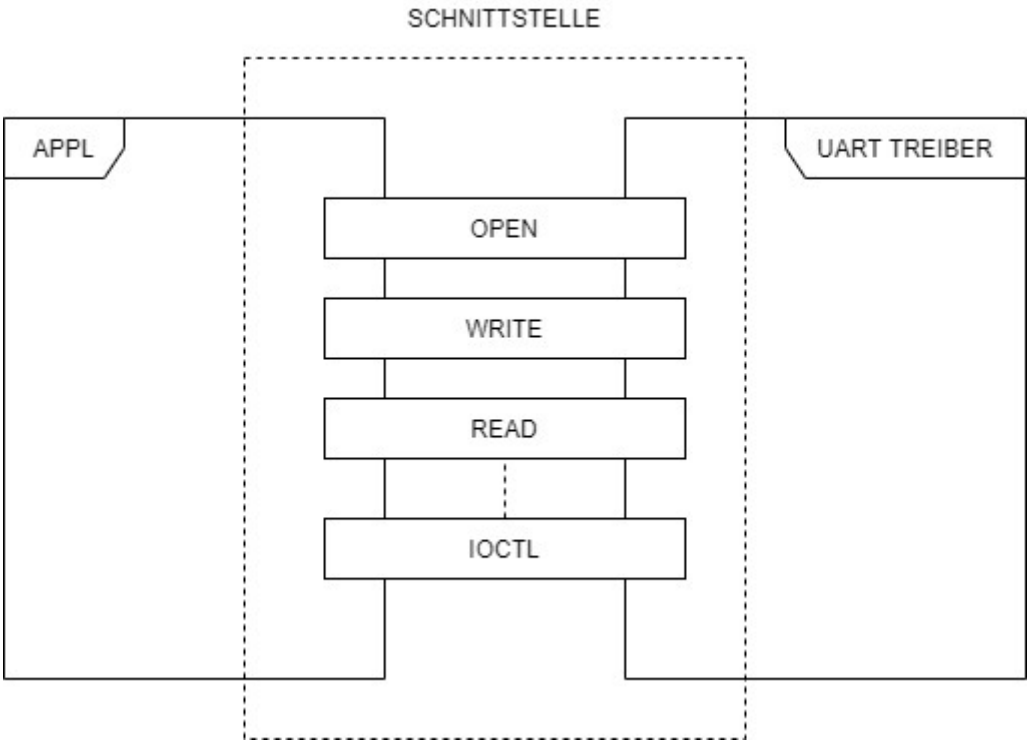

Abbildung 2.11-3: Schnittstelle Anwendungsschicht – Transportschicht

2.1.1 Variante 1. "dezentralisiert" / vollständig asynchron

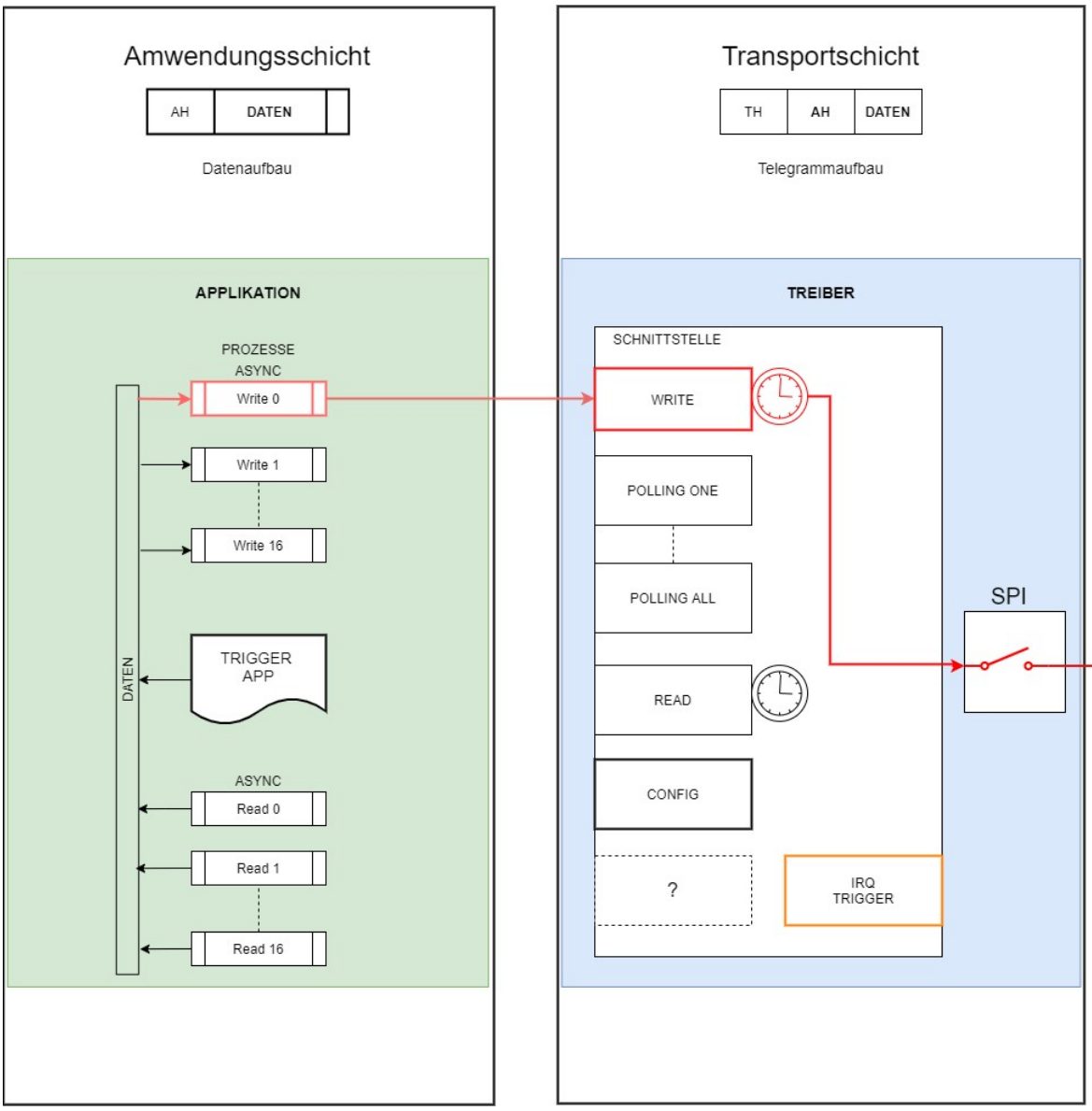

Abbildung 2.1.1-1: Variante 1. "dezentralisiert" / vollständig asynchron WRITE

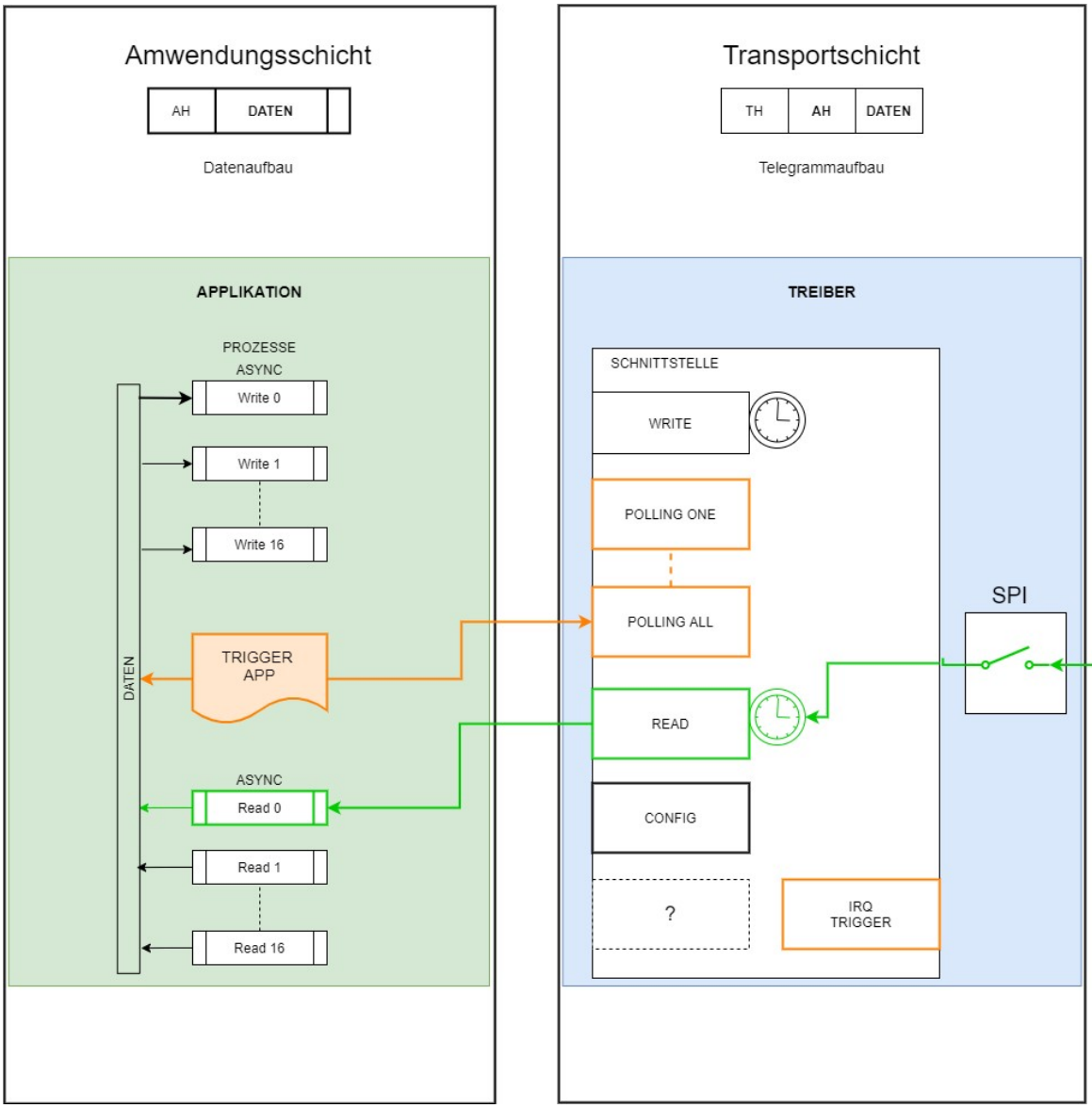

Abbildung 2.1.1-2: Variante 1. "dezentralisiert" / vollständig asynchron READ

2.1.2 Variante 2. "Zum Teil dezentralisiert" / asynchron

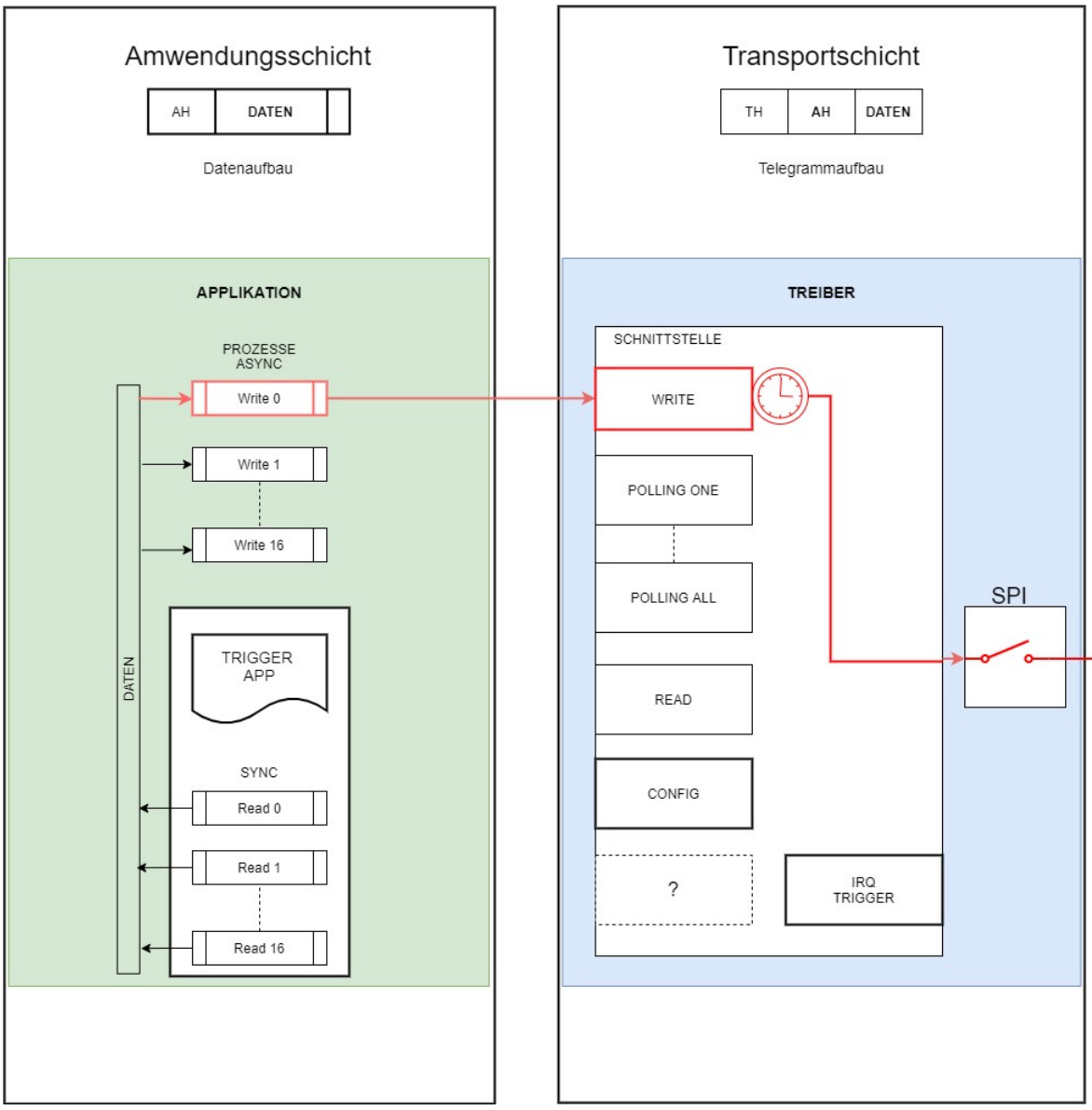

Abbildung 2.1.2-1: Variante 2. "Zum Teil dezentralisiert" / asynchron WRITE

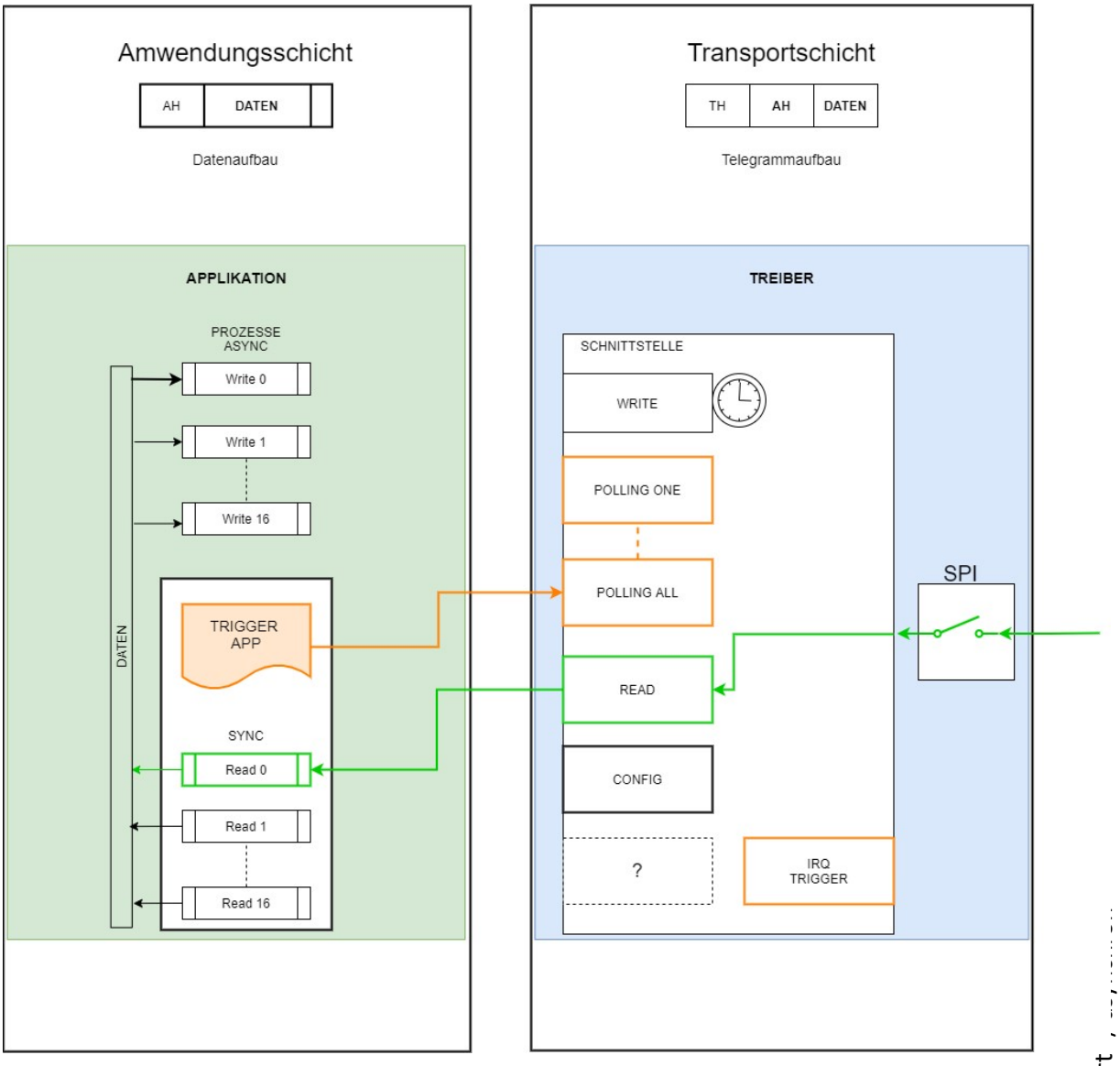

Abbildung 2.1.2-2: Variante 2. "Zum Teil dezentralisiert" / asynchron READ

2.1.3 Variante 3. "zentralisiert“ / synchron

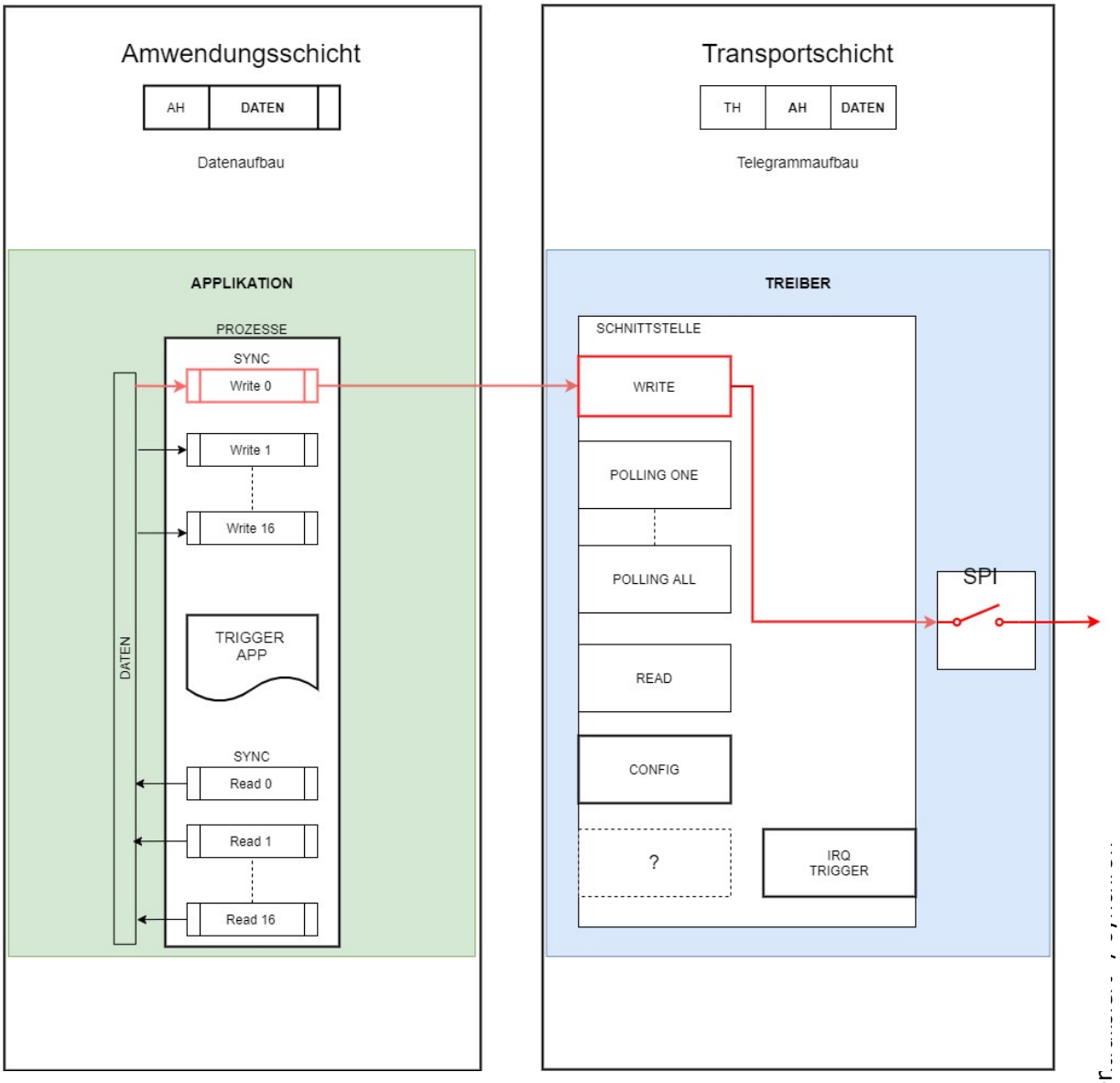

Abbildung 2.1.3-1: Variante 3. "zentralisiert“ / synchron WRITE

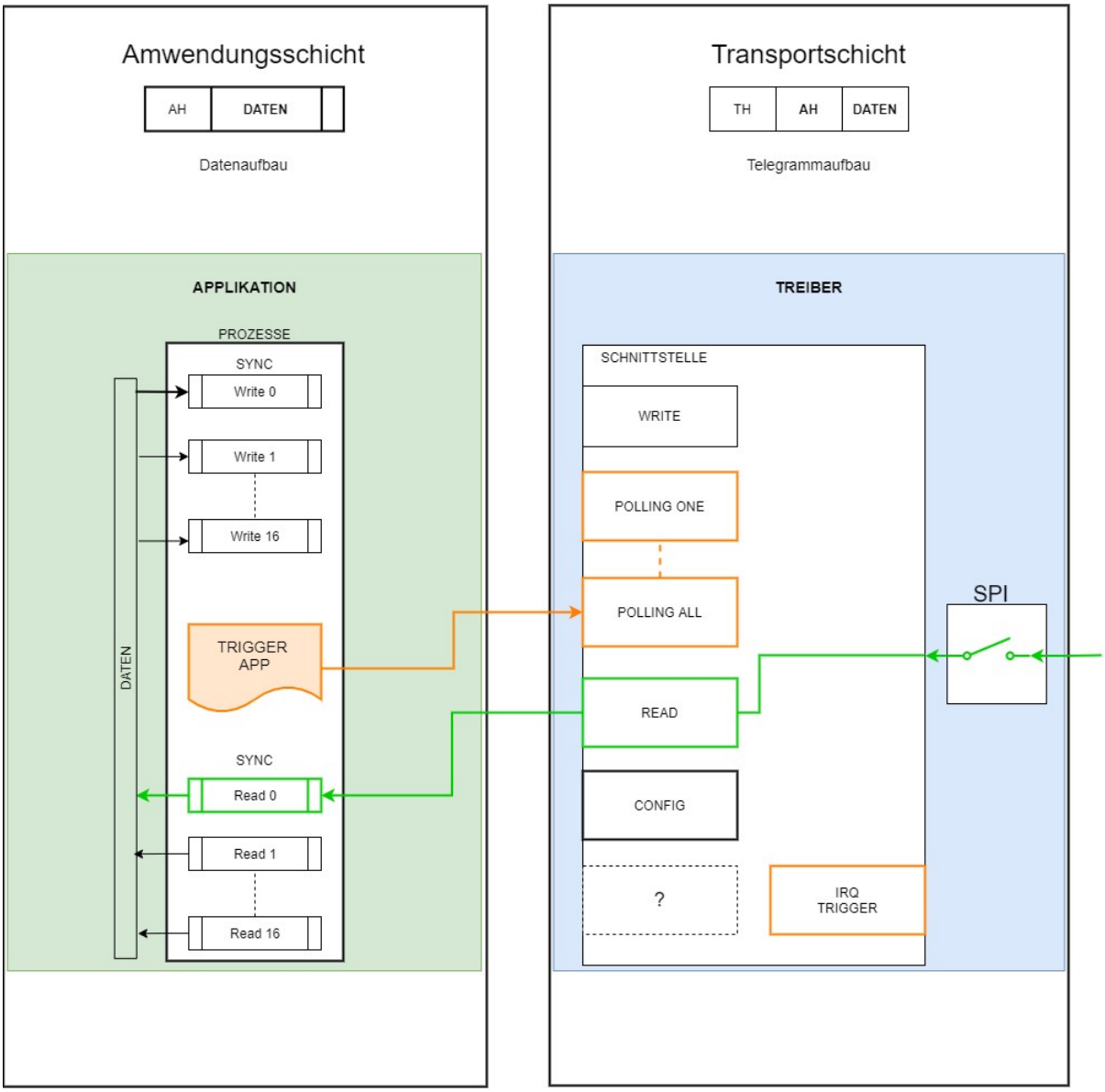

Abbildung 2.1.3-2: Variante 3. "zentralisiert“ / synchron READ

2.1.4 Variante 4. "dezentralisiert" / zum Teil asynchron

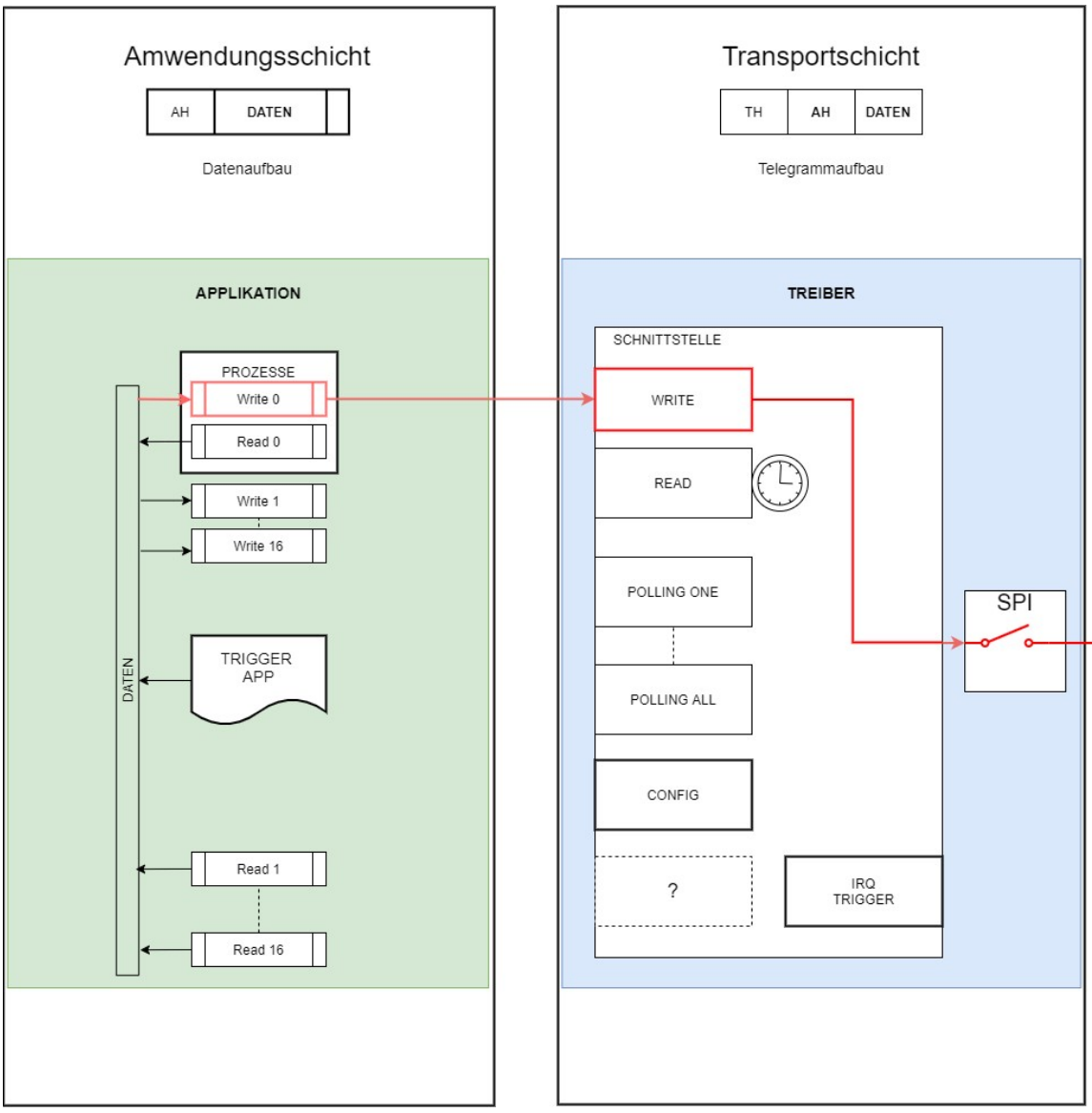

Abbildung 2.1.4-1: Variante 4. "dezentralisiert" / zum Teil asynchron WRITE

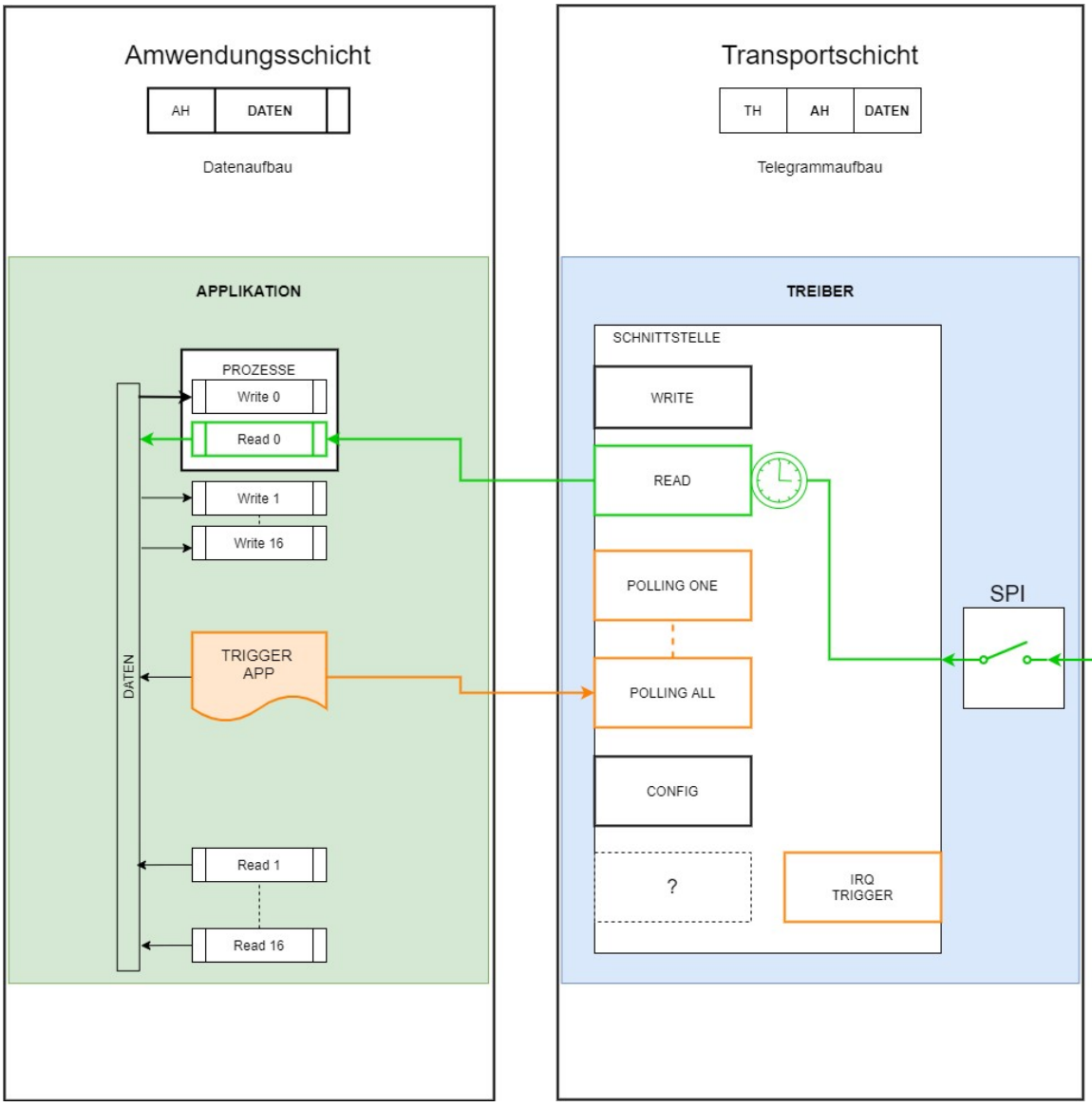

Abbildung 2.1.4-2: Variante 4. "dezentralisiert" / zum Teil asynchron READ

2.1.5 Telegrammaufbau für Transportschicht

Ein Daten-String wird von einer Applikation vorbereitet (siehe Abbildung 2.1.5-1) und als Pointer von User-Space an die UART-SPI Treiber übergeben. Diese Datenmenge holt der Treiber in Kernel-Space und versieht es entsprechend von CPLD-Protokoll vereinbarten Algorithmus mit Header bzw. End-Abschlüssen. Dabei werden die Daten selbst vom Treiber nicht interpretiert oder ausgewertet, die Aufgabe ist es wie der Name sagt - Transportieren.

LBDL								
	7	6	5	4	3	2	1	0
Header	Origin *Master: 0b010* *Slave: 0b101*			Packet-Options-Len *0-3 Byte*		Telegram-Type *Send with ACK: 0b000* *Send/recieve with reply: 0b001* *Send without ACK: 0b010* *XY: 0b100*		
Options[1]								
Options[2]								
Options[3]								
Address	Broadcast: 0x00 Unicast: Device-Addr. [1-126] Default-Device-Addr. 127							
PDU-1	Payload-Length [1 ... 240]							
PDU-2	Application-ID							
...	Payload depends on application							
PDU-n								
Checksum	CRC-16 from Header (included) to PDU-n (included)							
Checksum								
EOT	0xA9							
1 Startbit								
8 Databits	(LSB first, not inverted)							
No parity								
1 Stopbit								

Abbildung 2.1.5-1: Telegrammaufbau für Transportschicht (LaceNet)

2.2 Transportschicht – Übertragungsschicht

Bei diesem Übergang dient der Standard SPI-Treiber als Übertragungsglied zwischen einem *UART-SPI* Treiber und CPLD-Firmware (siehe Abbildung 2.2-1). Bevor ein Datentransfer stattfinden kann, soll SPI-Schnittstelle erstmal konfiguriert werden (z.B. Übertragungsrate usw.). Ein vordefinierten Protokoll beeinflusst das Telegrammaufbau (siehe Abbildung 2.2-2).

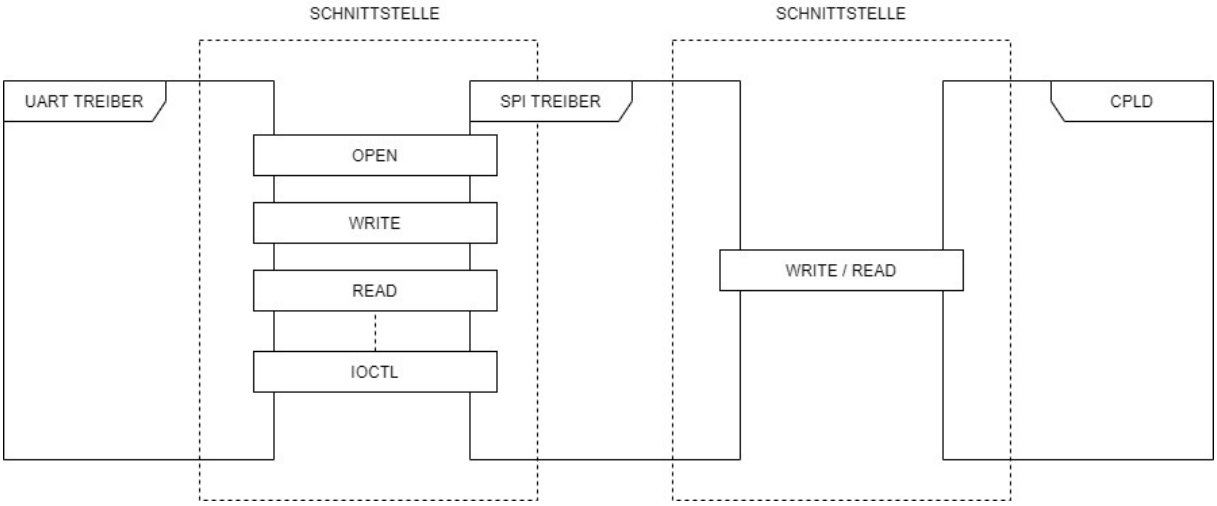

Abbildung 2.2-3: Schnittstelle Transportschicht – Übertragungsschicht

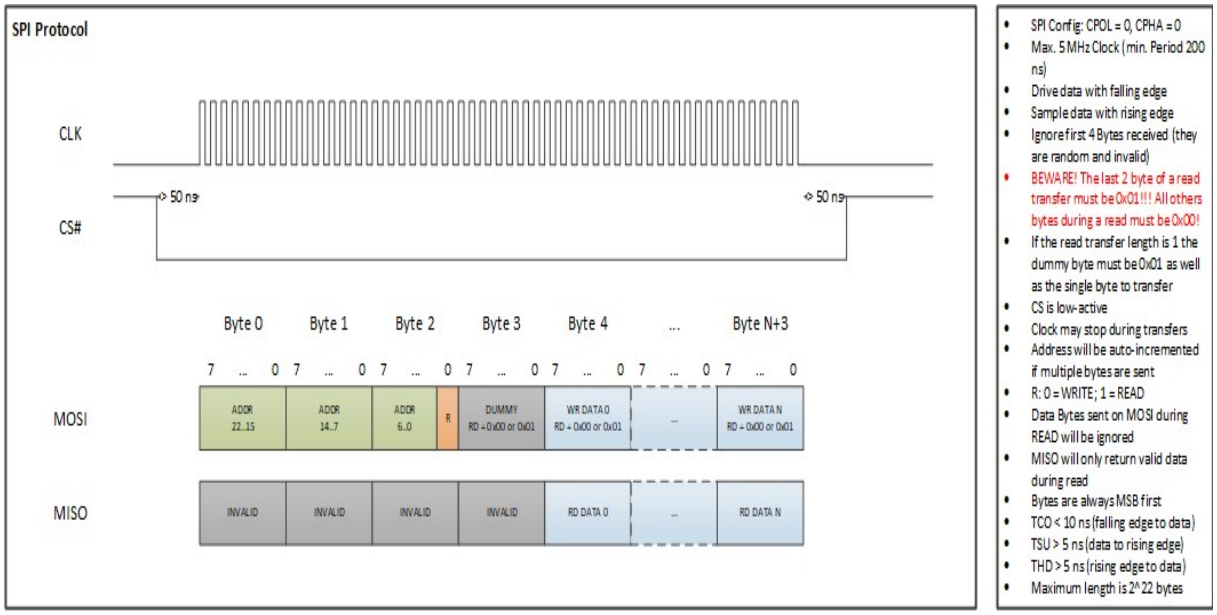

Abbildung 2.2-2: Telegrammaufbau für Übertragungsschicht (LaceNet)

Die vollständige und detaillierte Beschreibung von dem Protokoll sowie die einzustellende CPLD-Parameter als auch Status-Bits sind in einer separaten Datei zu finden (siehe dazu Unterordner.../Config).

3. Timing – Tests

Die durchgeführten Übertragungstests für unterschiedliche Testszenarien unter vorgegebenen Bedingungen und mit Berücksichtigung von der Messunsicherheit liefern eindeutige Ergebnissen, dass die Reaktionszeit sowie die Latenz stark von eingesetzte Variante (siehe Kapitel 2.1) und der Datenmenge abhängen. Außerdem lässt sich auch ein Zusammenhang zwischen optimierten Image (Betriebssystem Image) und verbesserte Übertragungsrate festzustellen, was wiederum auf vom CPU zur Verfügung gestellte Leistung zurückzuführen ist. Das wurde auch bei nicht optimierte Image nachgewiesen indem man eine höhere Priorisierung dem Prozess erteilt.

Die angesprochenen Ergebnisse sind in der Unterorder (.../Tests) abgespeichert.

4. Fazit

Die Übertragungskette (siehe Abbildung 4-1) ist nach einem Bausteinen Konzept aufgebaut und ermöglicht somit Anforderungsbedingte Prozessablauf. Die gekapselten Elemente bleiben für äußerer Betrachter als sogenannte Blackboxes. Die Nutzung bzw. Zusammenbauen von denen ist anhand der vordefinierten Schnittstellen möglich. Dank dieses erwähnten Konzepts lassen sich die einzelnen Bausteine flexibel um die gewünschte neue Funktionen erweitern.

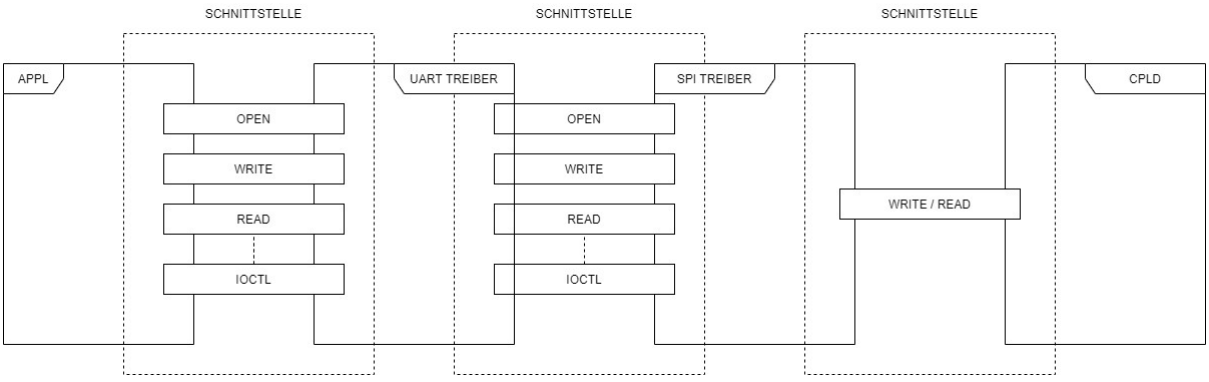

Abbildung 4-1: Prozess

Abbildungsverzeichnis