SpaceWire-CPU-Interface Handbuch (v.1.0 Beta, Rev. 64)

1. Memory-Mapped IO

Das SpaceWire-Link-Interface ist in drei Bereiche aufgeteilt: Transmitting (TX), Receiving (RX) und Register (REG). Jedem davon wird in einem ARM-AXI-basierten Memory-Mapped-Interface mit 32-Bit Wort- und Adressierungsbreite eine eigene Offset-Adresse zugewiesen. Des weiteren existieren zwei über GPIOs angebundene Ports, über welche der Reset der Logik und das versenden von Time-Codes gesteuert wird. Außerdem besteht die Möglichkeit, über Interrupts bei eingehenden Time-Codes, abgeschlossenen Paketen, Statusänderungen des Links und/oder Übertragungsfehler die CPU zu benachrichtigen. Innerhalb des zugewiesenen Adressbereichs, ist die Adressverteilung konstant und wie folgt aufgebaut:

1.1. Transmitting (TX)

Offset_TX + 0x0000_0000 : Adresse des Transmit-FIFOs *(write-only)*
Offset_TX + 0x0000_0004 : Anzahl freier Plätze im TX-FIFO *(read-only)*

1.2. Receiving (RX)

Offset_RX + 0x0000_0000 : Adresse des Receive-FIFOs *(read-only)*
Offset_RX + 0x0000_0004 : Anzahl verfügbarer Elemente im RX-FIFO *(read-only)*

1.3. Register (REG)

Offset_REG + 0x0000_0000 : Link-Konfiguration *(read/write)*
Offset_REG + 0x0000_0004 : Transmit-Rate *(read/write)*
Offset_REG + 0x0000_0008 : Time-Codes (ausgehend) *(read/write)*
Offset_REG + 0x0000_000C : Time-Codes (eingehend) *(read-only)*
Offset_REG + 0x0000_0010 : Link-Status *(read-only)*

2. Speicherzugriff

Die Verwaltung und der Zugriff auf die Transmit bzw. Receive-Funktionalität erfolgt über das AXI4-Full-Interface. Die Register werden dagegen über ein AXI4-Lite-Interface angesprochen. Der grundlegende Unterschied ist, dass das Full-Interface Burst-Transfers ermöglicht, während die Lite-Version nur mit Single-Transfers umgehen kann und dementsprechend langsamer ist. Die Übersetzung einer Transaktion von Software in AXI-Signalen erfolgt innerhalb der CPU. Ein einfacher Transfer wird dabei über einen Zeiger auf die entsprechende Speicheradresse initiiert.

2.1. Transmitting (TX)

Um Daten in den Transmit-FIFO zu schreiben, muss die unter 1.1. angegebene Adresse als Zieladresse eines Write-Requests angegeben werden. Bei Burst-Transfers ist darauf zu achten, dass die Zieladresse nicht inkrementiert wird, sondern konstant bleibt (Fixed Burst). (Siehe 4.2)

2.2. Receiving (RX)

Um Daten aus dem Receive-FIFO zu lesen, muss die unter 1.2. angegebene Adresse als Quelladresse eines Read-Requests angegeben werden. Bei Burst-Transfers ist darauf zu achten, dass die Quelladresse nicht inkrementiert wird, sondern konstant bleibt (Fixed Burst).

2.3. Register (REG)

Zugriff auf einzelne Register sind mit einem einzelnen Read-/Write-Request und der entsprechenden Angabe der Ziel-/Quelladresse des Registers möglich. Die Konfiguration erlaubt nicht, Read-only-Register versehentlich zu überschreiben, es ist dahingehend also keine besondere Vorsicht walten zu lassen.

3. Technische Beschreibung

Im Folgenden werden die Elemente hinter den unter 1. angegeben Adressen näher beschrieben und erläutert.

3.1. Transmitting (TX)

Daten, die in diese FIFO-Struktur geschrieben werden, werden nach und nach durch ein inbetriebgenommenes Link-Interface verschickt (siehe 2.1). Befindet sich das CPU-Interface in einem nicht-verbundenen Zustand, werden die Daten innerhalb des FIFOs gesammelt und verschickt, so wie eine Verbindungsaufnahme des Links erfolgreich war.

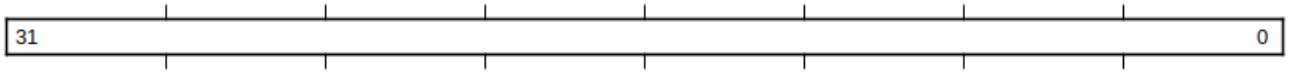
Die Breite des FIFOs beträgt dabei 9 Bits, die wie folgt aufgeteilt sind:
Bit **8** : Flag Bit (1 : EOP/EEP, 0 : Daten)
Bit **7** – Bit **0** : zu übertragenes Byte (0x00 für EOP, 0x01 für EEP)

In der Tiefe passen 2049 Datensätze in das FIFO. Ist das FIFO komplett befüllt und wird als nächstes ein Read-Request gefolgt von einem Write-Request ausgeführt, kann der Write-Request scheitern. Dies ist technisch bedingt und kann dadurch umgangen werden, dass die Größe des FIFOs um 1 geringer angenommen wird als sie tatsächlich ist. Anders kann dieses Problem nicht zuverlässig abgefangen werden.

Daten, die über die 9 Bits hinaus an das FIFO gesendet werden, werden ignoriert und nicht in den Speicher geschrieben. Ein letztendlich gültiger Datentransfer sieht wie folgt aus: 0x0000_0**xvv**, wobei **x** entweder 0x0 oder 0x1 ist und **vv** ein beliebiges Datenbyte sein kann. Gilt für **x** gleich 0x1 und für **vv** ungleich 0x00 oder 0x01, wird dieser Eintrag ignoriert und nicht vom Link-Interface versendet.

Zurückgesetzt werden kann das TX-FIFO mittels eines Resets über die CPU (AXI Reset). Dabei wird das FIFO pseudo-entleert und kann neu mit Daten befüllt werden. Der Link wird keine von vor dem Zeitpunkt des Resets eingefügte Daten mehr versenden.

Um zu verhindern, dass in einen vollen FIFO geschrieben wird, gibt es ein Register, welches die Anzahl freier Plätze im FIFO ausweist (s. 1.1):



Für die Darstellung des Platzes im FIFO sind lediglich 11 Bits notwendig, das ganze Register ist jedoch dafür reserviert. De facto sind allerdings die Bits 31 bis 12 unbelegt. Das Schreiben in einen vollen FIFO verursacht keinen Schaden an der Hardware oder am System jedoch gehen diese Daten verloren.

Obwohl diese Adresse als write-only markiert ist, kann über eine Read-Request das zuletzt in den FIFO geschriebene Datenwort abgefragt werden.

3.2. Receiving (RX)

Über SpaceWire empfangene Daten werden durch den SpaceWire Link nach und nach in den Rx FIFO geschrieben und können durch die CPU abgerufen werden. (siehe 2.2.)

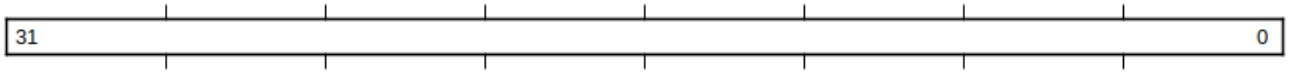
Die Breite des FIFOs beträgt dabei 9 Bits, die wie folgt aufgeteilt sind:
Bit **8** : Flag Bit (1 : Prozess ID/EOP/EEP, 0 : Daten)
Bit **7** – Bit **0** : empfangenes Datenbyte (0xFF für EOP, 0xFE für EEP)

In der Tiefe passen 2049 Datensätze in den FIFO. Ist das FIFO komplett befüllt und wird als nächstes ein Read-Request gefolgt von einem Write-Request ausgeführt, kann der Write-Request scheitern. Dies ist technisch bedingt und kann dadurch umgangen werden, dass bei vollem FIFO mindestens zwei Elemente gelesen werden, also keine Single-Transaktion durchgeführt wird.

Aus dem FIFO gelesene Datensätze enthalten somit neun gültige Bits (0x0000_0**xvv**), wobei **x** entweder 0x0 oder 0x1 und **vv** ein beliebiges Datenbyte sein kann. Ist **x** = 0x1, muss **vv** unterschieden werden. Gilt für **vv** = 0xFF handelt es sich um ein EOP (End of Packet), für 0xFE um ein EEP (Error End of Packet) und ansonsten um eine Prozess ID (0x00 – 0xFD, 0 – 253) oder eine logische Adresse des SpaceWire Packets. [***Hier gibt es noch Unklarheiten!]***

Zurückgesetzt werden kann der FIFO-Speicher mittels des Resets des Links (der Logik) (über einen GPIO). Dabei wird der FIFO pseudo-entleert und kann neu mit Daten befüllt werden. Das Processing System kann dabei keine Daten mehr abrufen, die vor dem Zeitpunkt des Resets in den Speicher geschrieben wurden.

Um zu verhindern, dass aus einem leeren FIFO gelesen wird, gibt es ein Register, welches die Anzahl verfügbarer Elemente im FIFO ausweist (s. 1.2.):



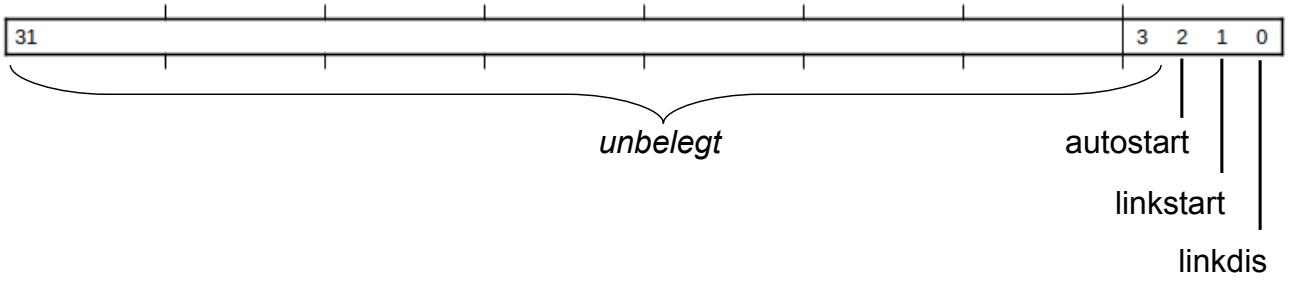
Für die Darstellung der Anzahl Elemente im FIFO sind lediglich 11 Bits notwendig, das gesamte Register ist jedoch für die Darstellung reserviert. De facto sind die Bits 31 bis 12 damit unbelegt.
Das Lesen aus einem leeren FIFO verursacht keinen Schaden an der Hardware oder am System.

Ist das FIFO leer, befindet sich das letzte Element an dessen Ausgabe, bis wieder Daten in das FIFO geschrieben werden und die Ausgabe aktualisiert wird. Es ist also darauf zu achten, dass das Element-Register (s. 1.2.) einbezogen wird.

3.3. Register (REG)

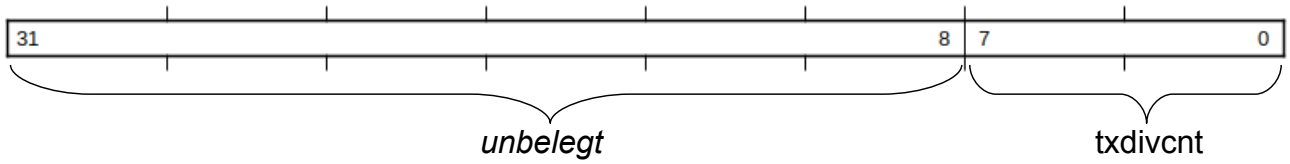
Die Breite des gesamten Registers beträgt 32 Bit. Je nach Adresse ist die Belegung der einzelnen Bits unterschiedlich.

3.3.1. - Link-Konfiguration: (read/write) ── 0x0000_0000



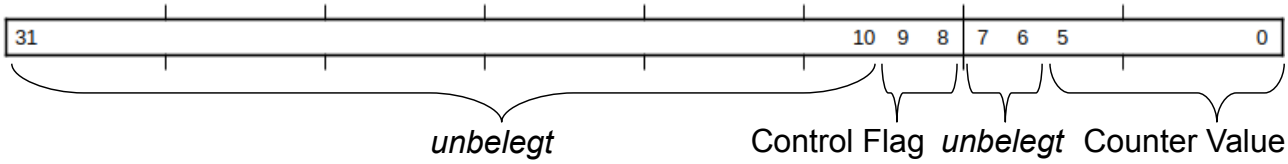
linkdis	Deaktiviert den Link, der Empfang und das Versenden von Nachrichten via SpaceWire ist nicht möglich. Standardwert: 1. 1 : Link deaktiviert (linkstart/autostart werden überschrieben) 0 : Link nicht deaktiviert (linkstart/autostart werden nicht überschrieben)
linkstart	Aktiviert den Link. Es wird aktiv versucht, eine Verbindung mit der gegenüberliegenden Stelle aufzubauen. Dabei wird in regelmäßigen Intervallen ein Verbindungsversuch unternommen. Standardwert: 0. 1 : Link starten 0 : Link nicht starten
autostart	Aktiviert den Link. Es wird nicht aktiv versucht, eine Verbindung mit der gegenüberliegenden Stelle aufzubauen. Auf Verbindungsversuche der gegenüberliegenden Stelle wird jedoch verbindungsaufbauend/kooperierend reagiert. Standardwert: 0. 1 : Automatischer Verbindungsaufbau aktiviert 0 : Automatischer Verbindungsaufbau deaktiviert

1.3.2. - Transmit Rate (read/write) ── 0x0000_0004



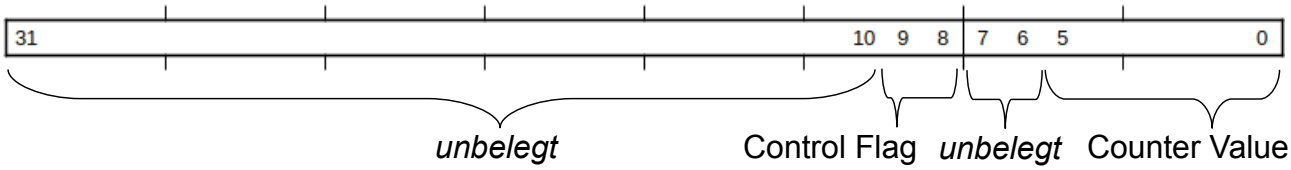
txdivcnt<7:0>	Skalierungsfaktor minus 1. Wird benutzt um die Sendebitrate aus dem Transmit-Takt abzuleiten. Standardwert: 0x01.

1.3.3. - Time-Codes (ausgehend) (read/write) ⟶ 0x0000_0008



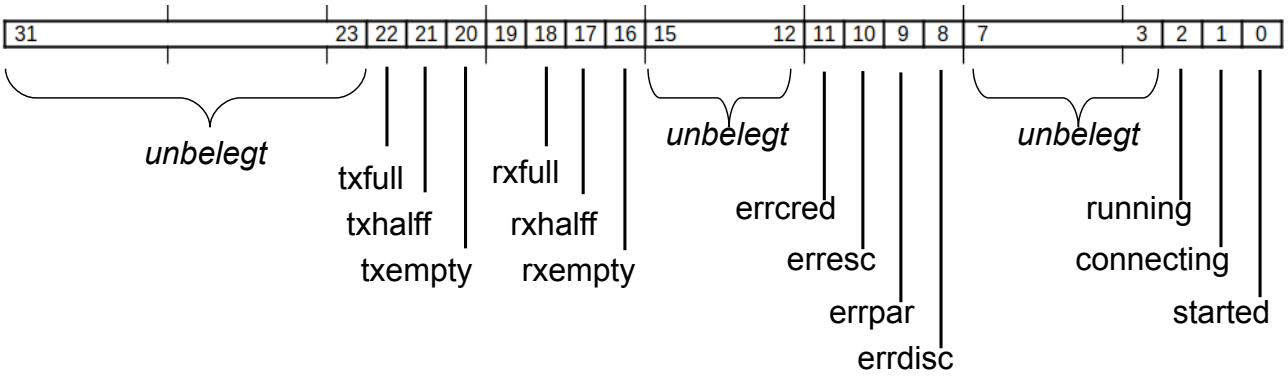
Counter Value <5:0>	Wert des Time-Codes, der zur Synchronisierung benutzt wird (Wertebereich: 0-63). Standardwert: 0b00000.
Control Flag <1:0>	Flag des Time-Codes. Standardwert: 0b00.

1.3.4. - Time-Codes (eingehend) (read only) ⟶ 0x0000_000C



Counter Value <5:0>	Wert des Time-Codes, der zur Synchronisierung benutzt wird (Wertebereich: 0-63)
Control Flag <1:0>	Flag des Time-Codes. (Standard: 0b00)

1.3.5. - Link-Status (read only) ⟶ 0x0000_0010



started	Zeigt an, ob der SpaceWire Link im started-Zustand ist. (Nicht verbunden, aber bereit eine Verbindung aufzunehmen)
connecting	Zeigt an, ob der SpaceWire Link im connecting-Zustand ist. (Verbindungsaufnahme im Gange)

running	Zeigt an, ob der SpaceWire Link mit einem anderen Link erfolgreich verbunden und lauffähig ist.
errdisc	Zeigt an ob ein Disconnect entdeckt wurde. Der Link cleart diesen Fehler automatisch und versucht eine erneute Verbindungsaufnahme.
errpar	Zeigt an ob ein Paritätsfehler während einer Datenübertragung entdeckt wurde. Der Link cleart diesen Fehler automatisch und versucht eine erneute Verbindungsaufnahme.
erresc	Zeigt an ob ein Escape-Fehler während der Datenübertragung entdeckt wurde. Der Link cleart diesen Fehler automatisch und versucht eine erneute Verbindungsaufnahme.
errcred	Zeigt an ob ein Credit-Fehler entdeckt wurde. Der Link cleart diesen Fehler automatisch und versucht eine erneute Verbindungsaufnahme.
rxempty	Zeigt an ob der interne Eingangs-FIFO des Links (nicht der RX FIFO unter 1.2.) **nicht** befüllt ist.
rxhalff	Zeigt an ob der interne Eingangs-FIFO des Links (nicht der RX FIFO unter 1.2.) **zur Hälfte** befüllt ist.
rxfull	Zeigt an ob der interne Eingangs-FIFO des Links (nicht der RX FIFO unter 1.2.) **vollständig** befüllt ist.
txempty	Zeigt an ob der interne Ausgangs-FIFO des Links (nicht der TX FIFO unter 1.1.) **nicht** befüllt ist.
txhalff	Zeigt an ob der interne Ausgangs-FIFO des Links (nicht der TX FIFO unter 1.1.) **zur Hälfte** befüllt ist.
txfull	Zeigt an ob der interne Ausgangs-FIFO des Links (nicht der TX FIFO unter 1.1.) **vollständig** befüllt ist.

4. Steuerung

4.1. Inbetriebnahme

Um den Link in Betrieb zu nehmen sind folgende Schritte erforderlich:

1. Herstellen einer physischen Verbindung der SpaceWire-Verbindungen (spw_do, spw_so und spw_di, spw_si) zu einem externen SpaceWire-Link.

2. Aktivieren des Interfaces:

2.1. Durchführen eines initialisierenden Resets sowohl der Logik wie auch des AXI Busses (Reihenfolge irrelevant) über mindestens fünf Taktzyklen des langsameren Taktes der beiden beteiligten (Bsp.: AXI Clock: 50 MHz; Logik Clock: 100 MHz; mindestens 100 ns Reset durchführen)
-- Kann ggf. entfallen. --

2.2. Schreiben des Wertes 0x01 in das Transmit-Rate-Register.

2.3. Schreiben des Wertes 0x06 in das Link-Konfigurationsregister.

2.4. Warten bis Verbindungsaufnahme hergestellt wurde. Feststellbar ist dies über entsprechendes Interrupt oder über das Statusregister.

3. Ab jetzt können Daten gem. Kap. 2 verschickt und empfangen werden.

4.2. Steuerung

Um Daten in den TX-FIFO zu schreiben, genügt es, einen Zeiger auf dessen Adresse (s. 1.1.) zu richten und diesem entsprechend Daten zuzuweisen. Diese Transaktion wird intern über den AXI-Bus abgewickelt.

Selbes Vorgehen ist beim RX-FIFO notwendig, nur das hier Daten gelesen werden sowie der Zeiger dereferenziert wird. Dadurch wird auch – sofern sich mehr als ein Element im FIFO befindet – das nächste Elemente an den FIFO Ausgang geladen.

Ankommende, gültige Time-Codes werden mittels eines Interrupts an die CPU gemeldet. Es muss also eine entsprechende ISR (Interrupt Service Routine) geschrieben und am Processing System (PS) registriert werden. Die Werte des Time-Codes sind dann über das entsprechende Register (s. 1.3.) abrufbar.

Sowohl der TX als auch der RX Zweig sind über AXI4-Full angebunden und unterstützen Burst Transfer. (Bis zu 255 Datenworte in einer Transaktion lesen/schreiben). Durchgeführt werden kann dies allerdings ausschließlich über einen DMA Controller und nicht über Programmcode.

Soll ein Time-Code verschickt werden, ist es zuerst nötig, die entsprechenden Werte innerhalb des dafür zuständigen Registers (s. 1.3.) zu schreiben. Danach wird der dafür vorgesehene GPIO auf HIGH gesetzt. So wie das Link-Interface eine steigende Flanke dieses GPIOs erkennt, wird mittels der Daten im Register (s. 1.3.) ein Time-Code generiert und priorisiert verschickt. Dabei genießt es Vorrang gegenüber den Daten im Tx-FIFO. Allerdings kann durch den Link gerade ein Datenwort verschickt werden, was die Versendung des Time-Codes um einige Takte verzögern kann. Wichtig ist, den GPIO nach diesem Prozess wieder auf LOW zu setzen.

4.3. Außerbetriebnahme

Um den Link zu deaktivieren oder abzuschalten genügt es, in das Link-Konfigurationsregister den Wert 0x0000_0001 zu schreiben. Damit wird linkdis aktiviert, welche alle anderen Einstellung innerhalb dieses Registers überschreibt. Um beide FIFOs zurückzusetzen, müssen sowohl das AXI-Interface als auch das Link-Interface über einen Zeitraum von mindestens 5 Taktzyklen resettet werden.