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

1. Memory-Mapped IO

The SpaceWire Link Interface is split up into three sections: Transmitting (TX), Receiving (RX) and Register (REG). Each one is, in an ARM-AXI based Memory-Mapped Interface with 32-bit word width and address width, assigned to its own offset address. Furthermore there are two ports, accessible via GPIOs, which are used to control the logic reset as well as the transmitting of Time-Codes. Additionally, in case of incoming Time-Codes, completed packages or status changes of the link, there is the possibility to inform the CPU via interrupt signals.

1.1. Transmitting (TX)

Offset_TX + 0x0000_0000 : Address of transmit FIFO *(write-only)*
Offset_TX + 0x0000_0004 : Number of free FIFO spaces *(read-only)*

1.2. Receiving (RX)

Offset_RX + 0x0000_0000 : Address of receive FIFO *(read-only)*
Offset_RX + 0x0000_0004 : Number of available elements in RX-FIFO *(read-only)*

1.3. Register (REG)

Offset_REG + 0x0000_0000 : Link Configuration *(read/write)*
Offset_REG + 0x0000_0004 : Transmit Rate *(read/write)*
Offset_REG + 0x0000_0008 : Time-Codes (outgoing) *(read/write)*
Offset_REG + 0x0000_000C : Time-Codes (incoming) *(read-only)*
Offset_REG + 0x0000_0010 : Link-Status *(read-only)*

2. Memory Access

The controlling and access of the transmit/receive functionality is done via the AXI4-Full-Interface. However, the registers are accessed via a AXI4-Lite-Interface. The fundamental difference is that the Full-Interface supports burst transfers while the Lite-Interface is only capable of doing single transfers and is therefore slower.

2.1. Transmitting (TX)

To write data into the transmit-FIFO, the address specified under 1.1. must be the destination address of a write request. When using burst transfers it is important to ensure that the address is not incremented but remains constant (fixed burst). (See 4.2.)

2.2. Receiving (RX)

To read data from the receive-FIFO, the address specified under 1.2. must be the source address of a read request. When using burst transfer it is important to ensure that the source address is not incremented but remains constant (fixed burst).

2.3. Register (REG)

Access to a single register are performed with a single read/write request and the using of the specific destination/source address. The configuration does not allow to overwrite read-only registers by mistake so it is not necessary to take special care in this respect.

3. Technical Specification

In the following section the elements specified under 1. are described in more detail.

3.1. Transmitting (TX)

Data which is written into the TX-FIFO is sent successively through a running Link-Interface (see 2.1.). Is the interface in a non-connected state the data within the FIFO is gathered and sent as soon as a connection attempt of the interface was successful.

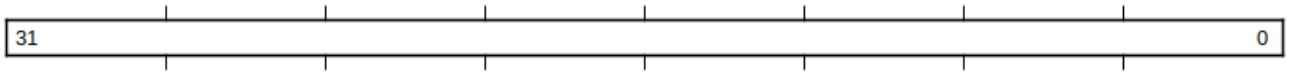
The FIFOs width is nine bits wide which are divided as follows:
Bit 8 : flag bit (1: End of Packet (EOP)/ Error End of Packet (EEP), 0 : Data)
Bit 7 – Bit 0 : byte to send (0x00 for EOP, 0x01 for EEP)

In depth, 2049 records fit into the FIFO. Is the FIFO completely full and is then a write request followed by a read request performed, the write request may fail. This is due to technical reasons and can be circumvented by assuming the depth of the FIFO to be one less than actual.

Data larger than 9 bits sent to the FIFO is ignored and not written into the memory. An ultimately valid data transfer looks like this: 0x0000_0**xvv**, where **x** can be either 0x0 or 0x1 and **vv** any data byte. Is x equal to 0x1 and vv unequal to 0x00 or 0x01 the record is ignored and not sent by the Link-Interface.

The TX-FIFO can be reset via the CPU (AXI reset). Thereby the FIFO is pseudo-emptied and can re-filled with data. The link will no longer send any data inserted before the time of the reset.

To prevent that data is written into a full FIFO there is a register which shows the number of free record spaces in the FIFO (s. 1.1.):



For the representation of the FIFO depth only 11 bits are necessary but the whole register is reserved for it. De facto the bits 31 down to 12 are unplaced. The writing into a full FIFO does not cause any harm on hardware or system but the data is lost though.

Although this address is marked as write-only it is able to read the last written record via a read request.

3.2. Receiving (RX)

Via SpaceWire received data is written successively into the RX-FIFO and can be read by the CPU. (s. 2.2.)

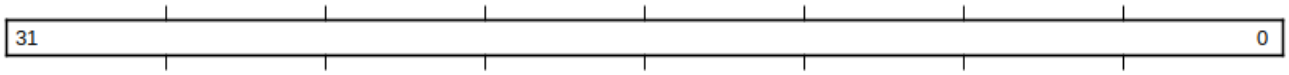
The FIFOs width is nine bits wide which are divided as follows:
Bit 8 : flag bit (1 : Process ID/EOP/EEP, 0 : data)
Bit 7 – Bit 0 : received data byte (0xFF for EOP, 0xFE for EEP)

In depth, 2049 records fit into the FIFO. Is the FIFO completely full and is then a read-request followed by a write-request executed the write-request may fail. This is due to technical reasons and can be circumvented in that situation by reading at least two elements out of the FIFO.

Data read out of the FIFO contains nine valid bits (0x0000_0**xvv**), where **x** is either 0x0 or 0x1 and **vv** any data byte. Is x equal to 0x1, **vv** must be differentiated: If vv = 0xFF the read record is an EOP (End of Packet), for 0xFE it is an EEP (Error End of Packet) and otherwise it is a process id (0x00 – 0xFD, 0x – 253) or a logical address of a the SpaceWire packet.

Reset of the FIFO is done via the reset of the link (or the logic) (via a GPIO). Thereby the FIFO is pseudo-emptied and can be re-filled with data. The Processing System is then not able to read data that was received before the time of the reset.

To prevent that data is read out of an empty FIFO there is a register which shows the number of available elements currently stored in the FIFO (s. 1.2.):



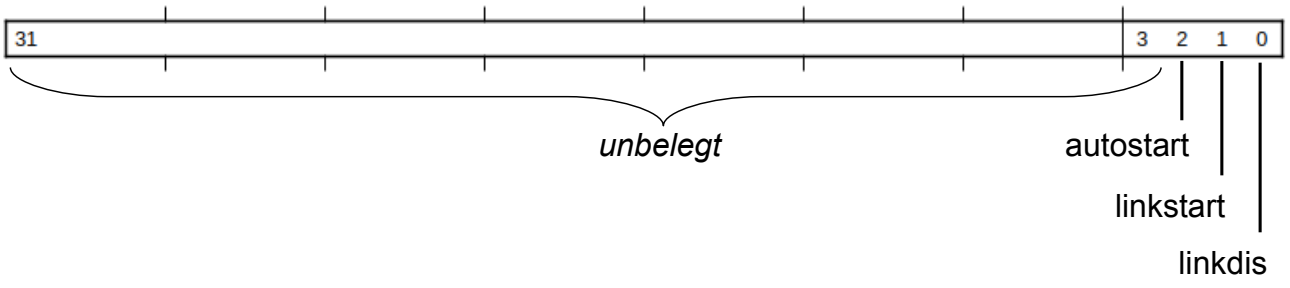
For the representation of the number of elements stored in the FIFO are just 11 bits necessary but the entire register is reserved for it. So de facto the bits 31 down to 12 are unplaced.

If the FIFO is empty the last valid element remains at its output as long as new data is written into the memory. So its recommended to include the validation of elements register (s. 1.2.).

3.3. Register (REG)

The width of the entire register is 32 bits. The placement of individual bits varies depending on the address.

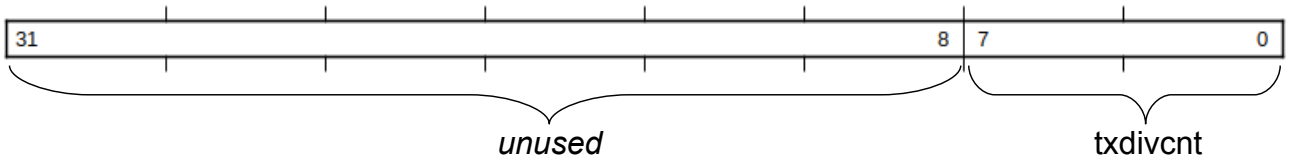
3.3.1. - Link-Configuration: (read/write) — 0x0000_0000



linkdis	Deactivates the link, receiving and transmitting of message is no

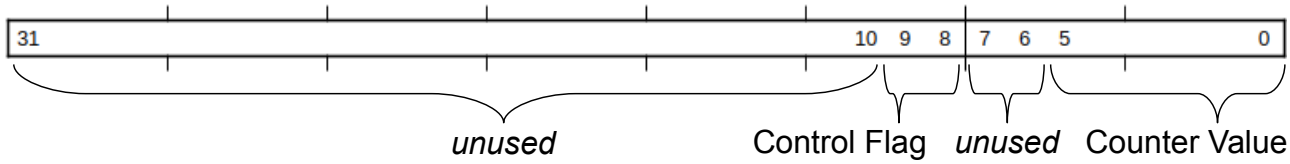
	longer possible. Default value: 1. 1 : Deactivates link (linkstart/autostart will be overwritten) 0 : Link is not deactivated
linkstart	Activates the link. Connection attempts are regularly made with the opposite entity in regular intervals. Default value: 0. 1 : Start link 0 : Link remains silent
autostart	Activates the link. It is not tried to make a connection attempt with the opposite entity. But on incoming connection attempts the link will react cooperatively. Default value: 0. 1 : Automatic connection establishment is activated 0 : Automatic connection establishment is deactivated

3.3.2. - Transmit Rate (read/write) — 0x0000_0004



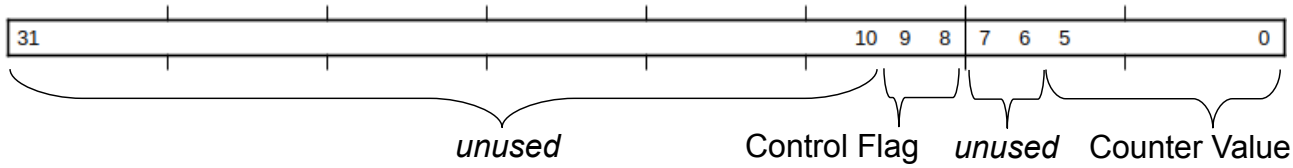
txdivcnt<7:0>	Scaling factor minus 1, used to derive the transmit rate from the transmit base clock. Default value: 0x01.

3.3.3. - Time-Codes (outgoing) (read/write) — 0x0000_0008



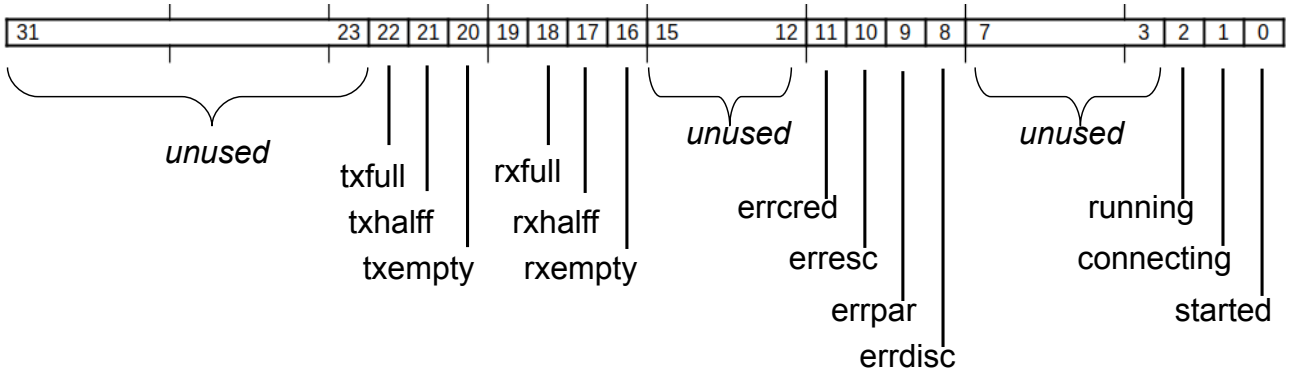
Counter Value <5:0>	SpaceWire Time-Code counter value (0-63) which is used for synchronization. Default value: 0b00000.
Control Flag <1:0>	Flag of the Time-Code. Default value: 0b00.

3.3.4. - Time-Codes (incoming) (read only) — 0x0000_000C



Counter Value <5:0>	SpaceWire Time-Code counter value (0-63) which is used for synchronization.
Control Flag <1:0>	Flag of the Time-Codes.

3.3.5. - Link-Status (read only) — 0x0000_0010



started	Shows if the SpaceWire link state machine is in the ‘Started’ state.
connecting	Shows if the SpaceWire link state machine is in the ‘Connecting’ state.
running	Shows if the SpaceWire link state machine is in the ‘Run’ state and able to receive and transmit data.
errdisc	Disconnect detected in the ‘Run’ state. Triggers a link reset; auto-clearing.
errpar	Parity error detected in the ‘Run’ state. Triggers a link reset; auto-clearing.
erresc	Invalid escape sequence detected in the ‘Run’ state. Triggers a link reset; auto-clearing.
errcred	Credit error detected. Triggers a link reset; auto-clearing.
rxempty	High if the receive FIFO is empty. (Not meant is the FIFO specified under 1.2.)
rxhalff	High if the receive FIFO is at least half full. (Not meant is the FIFO specified under 1.2.)
rxfull	High if the receive FIFO is full. (Not meant is the FIFO specified under 1.2.)
txempty	High if the transmit FIFO is empty. (Not meant is the FIFO specified under 1.1.)
txhalff	High if the transmit FIFO is at least half full. (Not meant is the FIFO specified under 1.1.)
txfull	High if the transmit FIFO is full. (Not meant is the FIFO specified under 1.1.)

4. Operation manual

4.1. Commissioning

To initialize the link the following steps are necessary:

1. Establishing a physical SpaceWire connection (spw_do, spw_so and spw_di, spw_si) to an external SpaceWire link.

2. Activate the interface:

2.1. Performing of an initial reset of both the logic and the AXI bus system over at least five clock cycles of the slower clock (e.g.: AXI Clock: 50 MHz; Logik Clock: 100 MHz; assign reset signal for min. 100 ns)
-- May be omitted. --

2.2. Writing the value 0x01 into Transmit-Rate register.

2.3. Writing the value 0x06 into Link-Configuration register.

2.4. Waiting until a connection is successfully established. This can be determined either by the status interrupt or by the Link-Status register.

3. From now on, data can be transmitted and received according to chapter 2.

4.2. Controlling

In order to write data to the TX-FIFO, it is sufficient to direct a pointer to its address (see 1.1.) and assign data to it accordingly. This transaction is handled internally via the AXI bus.

The same procedure is necessary for the RX-FIFO, except that here data is read and the pointer is de referenced. This also means that if there is more than one element in the FIFO – the next element is loaded automatically at the FIFO output after each read transaction.

Incoming, valid Time-Codes are reported to the CPU by means of an interrupt. Therefore, a corresponding ISR (Interrupt Service Routine) must be writte and registered at the Processing System (PS). The values of the Time-Code can then be called up via the corresponding register (see 1.3.)

Both the TX and RX branches are connected via AXI4-Full and support burst transfers. (read/write up to 255 data words). Howevery, this can only be done via a DMA controller and not via programme code.

If a Time-Code shall be sent, it is first necessary to write the corresponding values within the register responsible for this (see 1.3.). Then the GPIO provided for this purpose is set to HIGH. Just as the link interface recognises a rising edge of this GPIO, a Time-Code is generated by means of the data in the register and sent with priority. In doing so, it enjoys priority over the data in the TX-FIFO. However, a data word can be sent through the link which can delay the sending of the Time-Code by a few clock cycles. It is important to set the GPIO back to LOW after this process.

4.3. Decommissioning

To deactivate or shut down the link, its sufficient to write the value 0x0000_0001 into the Link-Configuration register. By doing so linkdis is activated which overwrites other

settings of the register. To reset both FIFO’s it is necessary to keep the reset signal of the logic and AXI HIGH for at least five clock cycles.