SpaceWire encoder-decoder Functional Verification

Jorge TONFAT

August 17, 2018

Project: PLATO RDCU

Change Record

Revision	Date	Author(s)	Description
0.1	04.06.18	JTO	Created Latex version from Word
0.2	13.06.18	JTO	First version with traceability matrix

Contents

1 Introduction

According to the clause 12.1 of the ECSS-E-ST-50-12C SpaceWire Standard, a SpaceWire encoder-decoder is a device with CMOS level DS signals and with the LVDS drivers and receivers implemented by external sources. The SpaceWire Light IP can be considered such device.
According to the clause 12.2.5, a SpaceWire encoder-decoder shall conform to all of the requirements given in all clauses listed in Table 12-5 of the SpaceWire standard.
The SpaceWire encoder-decoder IP is functionally verified using the coverage package of OSVVM. This package allows the generation of coverage points and bins to organize the functional verification.
In the next sections, it is described the clauses from the standard, the error and corner cases defined, the coverage points and the relation between them and the clauses from the SpaceWire standard.

Table 12-5 of the ECSS-E-ST-50-12C SpaceWire standard

Relevant clauses	Title
6.3	Signal coding
6.5	SpaceWire link
6.6	Data signaling rate
7	Character level
8	Exchange level
9	Packet level

2 List of Relevant Clauses from the SpaceWire Standard

Clause 6.3.1.a (*)
SpaceWire shall use Data-Strobe (DS) encoding.
Clause 6.3.1.b (*)
The data bit stream to transmit shall be encoded using two signals Data and Strobe as follows.
Clause 6.3.1.c
The Data signal shall follow the data bit stream, i.e. be high when the data bit is 1 and low when the data bit is 0.
Clause 6.3.1.d
The Strobe signal shall change state whenever the Data does not change from one bit to the next.
Clause 6.3.2.a
As data corruption following simultaneous transitions on the Data and Strobe lines is expected, the SpaceWire receiver shall be tolerant of simultaneous transitions on the Data and Strobe lines, i.e. the receiver shall not hang up.
Clause 6.3.2.b
When the SpaceWire transmitter is reset it shall be a controlled reset avoiding simultaneous transitions of Data and Strobe signals.
Clause 6.5.a (*)
A SpaceWire link shall comprise two pairs of differential signals, one pair transmitting the D and S signals in one direction and the other pair transmitting D and S in the opposite direction.
Clause 6.6.1.a
The minimum data signalling rate at which a SpaceWire link shall operate is 2 Mb/s.
Clause 6.6.2.a
The maximum data signalling rate shall be defined. For PLATO mission is 100 Mb/s.
Clause 6.6.3.a
The link in one direction can operate at a different data signalling rate to the same link in the opposite direction.
Clause 6.6.3.b (*)
Links within a system can operate at different data signalling rates.
Clause 6.6.4 (*)
Effects of skew and jitter
Clause 6.6.5.a
After a reset or disconnect (see clause 8.9.2.1) the SpaceWire link transmitter shall initially commence operating at a data signalling rate of $(10 \pm 1)Mb/s$.
Clause 6.6.5.b
The SpaceWire link transmitter shall operate at $(10 \pm 1)Mb/s$ until commanded to operate at a different data signalling rate.
Clause 6.6.6.a

The transmitter operating rate shall not be changed before the link connection has been made fully.

Clause 7.1 (*)

The character level protocol defined in this clause takes into consideration the DS-SE and DS-DE character level encoding given in IEEE Standard 1355-1995 [1], but it additionally includes Time-Codes for sending system time information across a SpaceWire link. The host interface to the SpaceWire encoder-decoder is specified.

Clause 7.2.a (*) As illustrated in Figure 7-1, a data character shall contain a parity bit, a data-control flag and eight bits of data as follows.

Clause 7.2.b

The data-control flag shall be set to zero to indicate that the current character is a data character.

Clause 7.2.c

The eight-bit data value shall be transmitted least significant bit first.

Clause 7.3.a (*)

A control character shall be formed from a parity bit, a data-control flag and a two-bit control code with the data-control flag set to one to indicate that the current character is a control character.

Clause 7.3.b

The NULL control code shall be formed from ESC followed by the flow control token (FCT).

Clause 7.3.c

The time control code (Time-Code) shall be formed from ESC followed by a single data character.

Clause 7.3.d (*)

Six bits of time information shall be held in the least significant six bits of the Time-Code (T0-T5) and the two most significant bits (T6, T7) shall contain control flags that are distributed isochronously with the Time-Code.

Clause 7.3.e

An escape character (ESC) followed by ESC, EOP or EEP is an invalid sequence and shall be noted as an escape error (see clause 8.9.2.3).

Clause 7.4.a (*)

A parity bit shall be assigned to each data or control character to support the detection of transmission errors.

Clause 7.4.b (*)

The parity bit shall cover the previous eight bits of a data character or two bits of a control character, the current parity bit, and the current data-control flag.

Clause 7.4.c

The parity bit shall be set to produce odd parity so that the total number of 1‘s in the field covered is an odd number.

Clause 7.5.a

After reset or link error (while in the ErrorReset state, see clause 8) the Data and Strobe signals shall be set to zero.

Clause 7.5.b

When the transmitter is enabled after reset the first bit that is sent shall be a parity bit, this bit shall be set to zero so that the first transition shall be on the Strobe line.

Clause 7.6.a

When the transmit and receive data interfaces to the host comprise eight data bits and one control flag, the coding given in Table 7-1 shall be used.

Table 7-1 of the ECSS-E-ST-50-12C SpaceWire standard

Control flag	Data bits (MSB & LSB)	Meaning
0	xxxxxxxx	8-bit data
1	xxxxxxx0 (use 00000000)	EOP
1	xxxxxxx1 (use 00000001)	EEP

Clause 7.6.b
EEP may be written into the transmitter interface.
Clause 7.6.c
For the two control codes (EOP and EEP), Since only the least significant bit is decoded, the following bits should be set:
1. when writing to the transmit interface, set to zero the remaining data bits.
2. when receiving, set the seven most significant data bits to zero when the control bit is set.
Clause 7.7.a (*)
The time interface to the host system shall comprise two signals, TICK_IN and TICK_OUT, a six-bit time output port, a six-bit time input port, a two-bit control flag input port and a two-bit control flag output port.
Clause 7.7.b
When TICK_IN is asserted and the link interface is in the Run state (see clause 8.5) it shall cause the transmitter to send a Time-Code immediately after the current character has been transmitted.
Clause 7.7.c
TICK_OUT shall be asserted whenever the link interface is in the Run state and the receiver receives a valid Time-Code.
Clause 7.7.d (*)
Only one node in a SpaceWire network should have an active TICK_IN signal.
Clause 7.7.e (*)
All other nodes should keep the TICK_IN signal not asserted.
Clause 7.7.f (*)
A six-bit time output shall be provided from the link receiver to the local time counter.
Clause 7.7.g (*)
A six-bit time input shall be provided to the link transmitter from the local time counter.
Clause 7.7.h (*)
The two control flags are reserved for future use and shall both be set to zero.
Clause 8.1 (*)
The exchange level protocol takes into consideration the DS-SE and DS-DE exchange level protocol given in clause 5.7 of the IEEE Standard 1355-1995 [1], but it also includes additional features in the state machine in order to eliminate problems with the ResetLinkCommand and several ambiguities within the IEEE Standard 1355-1995 have been resolved. See annex A for details of the differences between SpaceWire and IEEE Standard 1355-1995 and the reasons for those differences. The exchange level is responsible for making a connection across a link and for managing the flow of data across the link.
Clause 8.2.1 (*)
At the exchange level, data and control characters are separated into two types: link-characters (L-Char) and normal-characters (N-Char). L-Chars are those that are used in the exchange level and which are not passed on to the packet level. The flow control token (FCT) character and escape (ESC) character are L-Chars. The NULL control code (ESC + FCT) and the Time-Code (ESC + data character) are escape sequences and may be regarded as L-Chars. They are not passed on to the packet level. N-Chars are the characters that are passed on to the packet level: data characters and end-of-packet markers (EOP and EEP).
Clause 8.2.2.a
Only N-Chars shall be passed from the host interface to the link for transmission.
Clause 8.2.2.b
A received character shall not be acted upon until its parity has been checked.
Clause 8.3.a (*)
To avoid host receive buffer overflow and subsequent loss of data, data flow across a link shall be controlled using flow control tokens sent from one end of the link (end A) to the other end (end B) to signify that end A is ready to receive some more data.
Clause 8.3.b

A FCT sent out by a link interface shall be used to indicate that there is space for eight more N-Chars in the host receive buffer.

Clause 8.3.c

For each FCT sent the host system shall reserve room in its receive buffer to accommodate eight N-Chars.

Clause 8.3.d

The transmitter shall not transmit any N-Chars until it has received one or more FCTs to indicate that the receiver is ready.

Clause 8.3.e

The transmitter shall keep a credit count of the number of N-Chars that it has been authorized to send, as follows:

1. Each time a link interface receives an FCT its transmitter increments the credit count by eight.
2. Whenever the transmitter sends an N-Char it decrements the credit count by one.

Clause 8.3.f

If the credit count reaches zero the transmitter shall cease sending N-Chars until it receives another FCT increasing the credit count again to eight.

Clause 8.3.g

When the credit count is zero the transmitter shall continue to send L-Chars.

Clause 8.3.h

In state ErrorReset the credit count shall be set to zero.

Clause 8.3.i

If an FCT is received when the credit count is at or close to its maximum value (i.e. within eight of the maximum value) then the credit count shall not be incremented and a credit error shall be flagged (see clause 8.5.3.6 and 8.9.2.4).

Clause 8.3.j

The credit counter shall hold a maximum credit count of 56 (i.e. seven FCTs).

Clause 8.3.k

On reset (or after a link disconnect) the initial number of FCTs to transmit shall be set according to the size of the receive buffer (one FCT for every eight N-Chars that can be held in the receive buffer) up to a maximum of seven.

Clause 8.3.l (*)

A link interface shall keep a count of the number of outstanding N-Chars it expects to receive, i.e. the number it has asked for by sending FCTs, as follows:

1. Increment this outstanding count by eight each time an FCT is transmitted
2. Decremented this outstanding count by one each time an N-Char is received.

Clause 8.3.m

After a link reset or after a link disconnect, the initial value of the outstanding count shall be zero.

Clause 8.3.n

The outstanding counter shall contain a maximum count of 56, corresponding to seven FCTs.

Clause 8.3.o

An FCT shall not be transmitted unless there is room in the outstanding counter to record eight more outstanding N-Chars and there is room in the receive buffer to reserve space for those eight more N-Chars (see clause 8.4.8).

Clause 8.3.p

When transmission of a Time-Code or an FCT is requested then it shall be sent immediately, as soon as the transmitter has finished sending the current character (or NULL or Time-Code).

Clause 8.3.q

When no Time-Code or FCT is requested and N-Chars are available from the host interface and the flow control credit count is above zero, the transmitter shall send N-Chars.

Clause 8.3.r

If no Time-Code, FCT or N-Char are sent, then the transmitter shall send NULL to indicate that the link is still active (and to prevent the disconnect detection mechanism from being triggered at the other end of the link).

Clause 8.3.s

The order of priority for transmission of characters shall be as follows:

1. Time-Code, highest priority;
2. FCTs;
3. N-Chars;
4. NULL, lowest priority.

Clause 8.4.1 (*)
An example block diagram of a SpaceWire encoder-decoder is illustrated in Figure 8 1.
Clause 8.4.2 (*)
The transmitter is responsible for encoding data and transmitting it using the DS encoding technique. It receives N-Chars for transmission from the host system. If there is neither a Time-Code, FCT nor an N-Char (data, EOP or EEP) to transmit, the transmitter sends NULL. The transmitter sends N-Chars only if the host system at the other end of the link (end B) has room in its host receive buffer. This is indicated by the link interface at end B sending an FCT, showing that it is ready to accept another 8 N-Chars. The transmitter is responsible for keeping track of the FCTs received and the number of N-Chars sent to avoid input buffer overflow at the other end of the link. To do this the transmitter holds a credit count of the number of characters it has been given permission to send.
Clause 8.4.3 (*)
The transmitter can operate at any data signalling rate from the minimum to the maximum possible (see clause 6.6). The transmit clock is responsible for producing the variable data signalling clock signals used by the transmitter. The transmit clock signals are typically derived by dividing down the local system clock or a phase locked loop multiple of the local system clock.
Clause 8.4.4 (*)
The receiver is responsible for decoding the DS signals (Din and Sin) to produce a sequence of N-Chars (data, EOP, EEP) that are passed on to the host system. It also receives NULLs, FCTs and Time-Codes. NULLs represent an active link. They are flagged to the exchange-level state machine (see clause 8.5) but are ignored otherwise. When an FCT is received the receiver informs the transmitter so that it can update its credit count accordingly. All other control characters received are flagged to the state machine. The receiver ignores any N-Chars, L-Chars, parity errors or escape errors until the first NULL is received. The disconnection detection mechanism within the receiver is enabled as soon as the first bit arrives (i.e. first transition detected on D or S inputs to receiver). Time-Codes hold system time information. A valid Time-Code causes the assertion of the TICK_OUT signal from the receiver. The value of the Time-Code is placed on the TIME_OUT and CONTROL_FLAGS_OUT outputs when the TICK_OUT signal is asserted. These signals are used by the host system to update or regulate its system clock.
Clause 8.4.5 (*)
The receive clock (RX_CLOCK) is recovered by simply XORing the received Data and Strobe signals together. The receive clock recovery circuit provides all the clock signals used by the receiver with the exception of the local clock signal use for disconnect timeout.
Clause 8.4.6 (*)
The state machine controls the overall operation of the link interface. It provides link initialization, normal operation and error recovery services. The operation of the state machine is described in the form of a state diagram in clause 8.5.
Clause 8.4.7 (*)
The timer provides the After $6,4\mu s$ and After $12,8\mu s$ timeouts used in link initialization. See the state diagram of Figure 8 2. The timer is started when the state machine moves into particular states. When the state machine moves into the ErrorReset state the After $6,4\mu s$ timer is started. When it moves into the ErrorWait state, Started state or the Connecting state the After $12,8\mu s$ timer is started.
Clause 8.4.8 (*)

The host system is responsible for data buffer management. This makes the SpaceWire interface more versatile and eases partitioning of the error recovery mechanism (see clause 11.4) across the various levels of this Standard. Several different types of host receiver buffering may be implemented:
- FIFO buffering - where the size of the FIFO buffer depends upon the particular application.
- Memory buffering - where direct memory access (DMA) is used to transfer data to host system memory. As soon as the DMA channel has been set up, several FCTs can be requested immediately to allow the transfer of the data as fast as possible.
- No buffering - where the host system is able to accept data at the highest rate that the link interface can provide it. In this case several FCTs can be sent initially, followed by one more every time eight normal characters are received.
Due to the NULL or FCT handshake used during initialization (see clauses 8.5 and 8.7), it is expected that the host system flags that it is ready to receive eight more N-Chars before or during link initialization (see clause 8.3m). If this is not the case, link initialization fails until the host system is ready to receive data - the link interface is not able to send an FCT. When the host systems at both ends of the link have indicated that they are ready to receive data, then the link connection is made.

Clause 8.4.9 (*)

Most implementations of a SpaceWire interface are likely to include transmit and receive FIFOs. This clause describes one way in which these FIFOs can be used. After system reset the transmit and receive FIFOs are empty. When link connection is made any data written to the transmit FIFO can be transmitted if FCTs have been received to enable transmission. The receive FIFO is able to accept data while it still has space available. Data is read from the receive FIFO by the host system. Using a FIFO simplifies the interface to the host system. FIFO half-full or half-empty flags can be used to trigger DMA or processor intervention to read from or write to the FIFO before it becomes full or empty. This helps smooth the flow of data across a link and maintain high data throughput.

Clause 8.5.1 (*)

The complete state transition diagram for the SpaceWire link interface is illustrated in Figure 8-2. The state diagram notation is explained in clause 3.4.3.

Clause 8.5.2.1 (*)

A table listing the exit conditions from each state is included in Annex B for clarification purposes.

Clause 8.5.2.2.a

The ErrorReset state shall be entered after a system reset, after link operation is terminated for any reason or if there is an error during link initialization.

Clause 8.5.2.2.b

In the ErrorReset state the Transmitter and Receiver shall all be reset.

Clause 8.5.2.2.c

When the reset signal is de-asserted the ErrorReset state shall be left unconditionally after a delay of $6,4\mu s$ (nominal) and the state machine shall move to the ErrorWait state.

Clause 8.5.2.2.d

Whenever the reset signal is asserted the state machine shall move immediately to the ErrorReset state and remain there until the reset signal is de-asserted.

Clause 8.5.2.3.a

The ErrorWait state shall be entered only from the ErrorReset state.

Clause 8.5.2.3.b

In the ErrorWait state the receiver shall be enabled and the transmitter shall be reset. NOTE: This allows the receiver to start the disconnection detection mechanism (after registering a transition on the D or S line) and to begin looking for the arrival of a NULL.

Clause 8.5.2.3.c

If a NULL is received then the gotNULL condition shall be set.

Clause 8.5.2.3.d

The ErrorWait state shall be left unconditionally after a delay of $12,8\mu s$ (nominal) and the state machine shall move to the Ready state.

Clause 8.5.2.3.e

If, while in the ErrorWait state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move back to the ErrorReset state. NOTE: The ErrorReset and ErrorWait state with their $6,4\mu s$ and $12,8\mu s$ delays ensure that the receivers at both ends of a link are enabled before either end begins transmission, following an exchange of silence (see clause 8.9.4).

Clause 8.5.2.4.a

The Ready state shall be entered only from the ErrorWait state.

Clause 8.5.2.4.b

In the Ready state the link interface is ready to initialize as soon as it is allowed to do so. The receiver shall be enabled and the transmitter shall be reset.

Clause 8.5.2.4.c

If a NULL is received then the gotNULL condition shall be set. NOTE: This condition is acted upon in the Started state.

Clause 8.5.2.4.d

The state machine shall wait in the Ready state until the [Link Enabled] guard becomes true (see clause 8.6) and then it shall move on into the Started state.

Clause 8.5.2.4.e

If, while in the Ready state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move to the ErrorReset state.

NOTE: In the Ready state the two receivers are enabled and the state machine is waiting for the local host system to command the link to start.

Clause 8.5.2.5.a

The Started state shall be entered from the Ready state when the link interface is enabled. NOTE: In the Started state the state machine begins making a connection with the link interface at the other end of the link by sending one or more NULLs.

Clause 8.5.2.5.b

When the Started state is entered a $12,8\mu s$ (nominal) timeout timer shall be started.

Clause 8.5.2.5.c

In the Started state the receiver shall be enabled and the transmitter shall send NULLs.

Clause 8.5.2.5.d

If a NULL is received then the gotNULL condition shall be set.

Clause 8.5.2.5.e

The state machine shall move to the Connecting state if the gotNULL condition is set. NOTE: The NULL that set the gotNULL condition can have been received in the ErrorWait, Ready or Started states.

Clause 8.5.2.5.f

In the Started state the sending from the transmitter of at least one NULL shall be requested before moving to the Connecting state.

Clause 8.5.2.5.g

If, while in the Started state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move to the ErrorReset state.

Clause 8.5.2.5.h

If the $12,8\mu s$ timeout timer referred to in point b. above expires (i.e. no NULL received since leaving the ErrorReset state) then the state machine shall move to the ErrorReset state.

NOTE: In the Started state the attempt to make a connection across the link is started. NULLs are transmitted and the receiver is waiting to receive a NULL.

Clause 8.5.2.6.a
The Connecting state shall be entered from the Started state after a NULL is received (gotNULL condition set).
Clause 8.5.2.6.b
On entering the Connecting state a $12,8\mu s$ timeout timer shall be started.
Clause 8.5.2.6.c
In the Connecting state the receiver shall be enabled and the transmitter shall be enabled to send FCTs and NULLs.
Clause 8.5.2.6.d
If an FCT is received (gotFCT condition true) the state machine shall move to the Run state.
Clause 8.5.2.6.e
If, while in the Connecting state, a disconnect error, parity error or escape error is detected, or if any character other than NULL or FCT is received, then the state machine shall move to the ErrorReset state.
Clause 8.5.2.6.f
If the $12,8\mu s$ timeout referred to in point b. above occurs then the state machine shall move to the ErrorReset state.
NOTE: The Connecting state is entered when the link interface (end A) receives a NULL, waiting then for the reception of an FCT indicating that the other end of the link (end B) has also received a NULL. When the link interface receives a NULL and an FCT it means that communication is established in both directions. If an FCT fails to arrive within $12,8\mu s$ then something is wrong with the link connection and so the link interface is reset once more (ErrorReset state) and connection is attempted once again.
Clause 8.5.2.7.a
The Run state shall be entered from the Connecting state.
NOTE: In the Run state the receiver is enabled and the Transmitter is enabled to send Time-Codes, FCTs, N-Chars and NULLs.
Clause 8.5.2.7.b
If the link interface is disabled, or if a disconnect error, parity error, escape error or credit error is detected (see clause 8.5.3), while in the Run state, then the state machine shall move to the ErrorReset state. NOTE: The Run state is the state for normal operation where link connection has been made and L-Chars and N-Chars can flow freely in both directions across the link. The link remains in the Run state until an error occurs or until the link is disabled.
Clause 8.5.3.1.1 (*)
Reset represents power on reset, other hardware reset or software commanded reset.
Clause 8.5.3.1.2 (*)
After $6,4\mu s$ or after $12,8\mu s$ represents a delay of the specified time measured from when the current state is entered. The actual time intervals are nominal delays (see clause 8.11).
Clause 8.5.3.1.3 (*)
[Link Enabled] is a condition for the transition to occur (i.e. a guard). [Link Enabled] can be set true by software or hardware (see clause 8.6).
Clause 8.5.3.2.a
A gotNULL condition shall be asserted when a NULL is received. NOTE: GotNULL means that a NULL detection sequence (see Figure 8-3) has been received.
Clause 8.5.3.2.b
NULL detection shall be enabled whenever the receiver is enabled.
Clause 8.5.3.2.c
Any sequence of bits encountered after reset (ErrorReset state) prior to the first NULL being received shall be ignored.
Clause 8.5.3.2.d (*)
NULL detection shall include all three parity bits related to the NULL, i.e. the parity bit that covers the data-control flag of the ESC character, the parity bit that covers the ESC character, and the parity bit that covers the FCT character.
Clause 8.5.3.2.e

Hence the NULL shall be detected and gotNULL asserted, when the 011101000 sequence of bits is received as illustrated in Figure 8-3. NOTE: During initialization the character following a NULL is a control character (either another NULL or an FCT) so the last parity bit of the NULL is zero.

Clause 8.5.3.2.f

If a parity error occurs within the NULL detection sequence then the gotNULL condition shall not be asserted.

Clause 8.5.3.3.a

FCTs shall be valid only when received in the Connecting and Run states. NOTE 1: If received in any other state they represent an error. NOTE 2: gotFCT means that an FCT has been received.

Clause 8.5.3.4.a

An N-Char received when the exchange-level state machine is not in the Run state shall be interpreted as an error. NOTE: gotN-Char means that an N-Char has been received.

Clause 8.5.3.5.a

A time-code received when the exchange-level state machine is not in the Run state shall be interpreted as an error. NOTE: gotTime-Code means that a time-code has been received.

Clause 8.5.3.6

[Link Disabled] is a condition set by external hardware or software in order to disable and stop the link interface (see clause 8.6).

Clause 8.5.3.7.1 (*)

RxErr or receiver error is shorthand for disconnect error, parity error or escape error.

Clause 8.5.3.7.2.a Disconnect error

The disconnect error shall be detected by the receiver in the link interface. NOTE: Disconnect error is an error condition asserted when the length of time since the last transition on the D or S lines was longer than 850 ns nominal (see clause 8.11).

Clause 8.5.3.7.2.b

The disconnect detection mechanism shall be activated after leaving the ErrorReset state as soon as the first edge is detected on the D or S line.

Clause 8.5.3.7.3.a Parity error

The parity error shall be detected by the receiver in the link interface. NOTE: The parity error event occurs if a parity error is detected (see clause 7.4).

Clause 8.5.3.7.3.b

Parity detection shall be enabled whenever the receiver is enabled after the first NULL is received.

Clause 8.5.3.7.4.a Escape error

The Escape error shall be detected by the receiver in the link interface. NOTE: The escape error event occurs if an ESC character is followed by any control character other than an FCT (ESC followed by FCT is a NULL, see clause 7.3). ESC followed by a data character is a Time-Code (clause 7.3).

Clause 8.5.3.7.4.b

Escape error detection shall be enabled whenever the receiver is enabled after the first NULL is received.

Clause 8.5.3.8.a Credit error

The credit error shall be detected by the receiver in the link interface. NOTE: Credit error occurs if data is received when the host system is not expecting any more data, i.e. when all the N-Chars expected, according to the requested eight more N-Chars and subsequent transmitted FCTs, have been received. A credit error also occurs if an FCT is received when the credit error is at or close to its maximum value (see clause 8.3i). A credit error indicates that some undetected error has occurred on the link, which has caused the corruption of the credit count.

Clause 8.5.3.9.a Character sequence error

The character sequence error shall be detected by the state machine in the link interface.

Clause 8.5.3.9.b

Any characters received before a NULL is received shall be ignored.

Clause 8.5.3.9.c

Once a NULL is received, an FCT received before a NULL is sent shall indicate an error (i.e. FCT received in ErrorWait, Ready or Started state).

Clause 8.5.3.9.d

An N-Char should only be expected after both a NULL and an FCT are received otherwise an error occurs (i.e. N-Char can only be received in the Run state). NOTE: In the state diagram of Figure 8-2, the invalid gotFCT or gotN-Char events are shown explicitly, rather than as a general character sequence error event.

Clause 8.6.a (*) AutoStart (normative)

A link interface should be able to be commanded to start automatically on receipt of a NULL.

Clause 8.6.b

In the case specified in 8.6a, the Link Enabled condition in the state machine should be set as follows:
$[LinkEnabled] = (NOT[LinkDisabled])AND([LinkStart]OR([AutoStart]ANDgotNULL))$
Where:

- LinkDisabled is the flag set by software or hardware to indicate that the link is disabled. This corresponds to the Link Disabled condition in the state diagram.
- LinkStart is a flag set by software or hardware to start a link, i.e. to cause the transition from the Ready state to the Started state.
- AutoStart is a flag set by software or hardware to request the link to start automatically on receipt of a NULL.
- gotNULL is the flag indicating that the link interface has received a NULL detection sequence as defined in clause 8.5.3.2.

Clause 8.6.c

LinkStart and AutoStart should only be acted upon when the link interface is not disabled i.e. [LinkDisabled] = False. NOTE: The AutoStart facility enables the setting up of a system where one end (end A) of the link is held waiting for the other end (end B) to attempt connection. As soon as end B tries to connect end A responds immediately. This allows the control of the connection of a link from one end of the link only.

Clause 8.7 Link initialization

This clause explains how the state diagram given in clause 8.5 handles link initialization, going from the reset of one end of a link through to the link operating normally and sending data in both directions.

Clause 8.8 Normal operation

In normal operation both ends of the link are in the Run state and are sending and receiving Time-Codes, FCTs, N-Chars and NULLs. An example is a host system with buffer space sufficient to hold 16 N-Chars. This host system at one end of a link (end A) indicates that it is ready to receive N-Chars by twice flagging that it has room for 8 more characters to the link interface. The link interface sends two FCTs to the other end of the link (end B), which increments its credit count accordingly (from zero to 16). The link interface at end B indicates to its host system that it is ready to transmit data (N-Chars). When the host system at end B has data to transfer, it passes it to the link interface, which sends it across the link to end A. As each character is transmitted by the link interface (end B), it decrements its credit count until it reaches zero, at which point the link interface (end B) indicates to its host system that it is not ready to transfer any more data. The data received at end A is passed on to its host system, which places it in its 16 character buffer. As the host system uses the data out of this buffer it makes space for the reception of more data. As soon as there is space for another 8 more characters it flags this to the link interface, which then sends out another FCT informing end B that 8 more normal characters may be sent.

Clause 8.9.1.a

Whenever one of the following errors occur both character synchronization and flow-control status shall cease to be valid: disconnect errors, parity errors, escape errors, credit errors and character sequence errors.
NOTE: This are the five forms of receiver error that can be detected and acted upon at the exchange level.

Clause 8.9.1.b

The error detecting end shall be reset and re-initialized to recover character synchronization and flow control status. NOTE: This forces the remote end to do the same.

Clause 8.9.1.c

Empty packets, which can be received in addition to these errors, shall be discarded.

Clause 8.9.2.1.1 (*)

An operational link interface sends Time-Codes, FCTs, N-Chars or NULLs continuously, and thus the Data, Strobe or both signals are always changing.

Clause 8.9.2.1.2.a Disconnect error

The receiver shall detect a disconnection when the time interval from the last transition on either the Data or Strobe signal exceeds the disconnect-detection time.

Clause 8.9.2.1.2.b

The disconnect-detection time shall be 850 ns nominal. NOTE: See clause 8.11.2 for the disconnect timing specification. A disconnection cannot be detected unless the receiver has previously received at least one bit.

Clause 8.9.2.1.2.c

The detection of a disconnection shall provoke a disconnect error. NOTE: A disconnect error can either be caused when one end of the link is disabled or when the link is physically disconnected (intentionally or unintentionally).

Clause 8.9.2.1.2.d

If a physical disconnection is the cause of the disconnect error then both ends of the link shall try repeatedly to make a connection until the link is reconnected or until the link interfaces are disabled.

Clause 8.9.2.1.2.e

If a disconnect error is detected then the link interface shall follow the exchange of silence error recovery procedure described in clause 8.9.4.

Clause 8.9.2.1.2.f

If the disconnect error occurs in the Run state then the disconnect error shall be flagged up to the network level as a link error (see clause 8.9.5).

Clause 8.9.2.2.a Parity error

When a parity bit is received it shall be checked (see clause 7.4).

Clause 8.9.2.2.b

If a parity error occurs after the first NULL is received, then the link interface shall follow the error recovery procedure described in clause 8.9.4.

Clause 8.9.2.2.c

If the parity error occurs in the Run state then the parity error shall be flagged up to the network level as a link error (see clause 8.9.5).

Clause 8.9.2.3.a Escape error

An ESC character shall only be used to form either the NULL i.e. ESC followed by FCT, or the Time-Code, i.e. ESC followed by a single data character (see clause 7.3).

Clause 8.9.2.3.b

If a ESC character is received followed by any control character other than an FCT then the link interface shall follow the error recovery procedure described in clause 8.9.4.

Clause 8.9.2.3.c

If the escape error occurs in the Run state then the escape error shall be flagged up to the network level as a link error (see clause 8.9.5).

Clause 8.9.2.4.a Credit error

A credit error shall be identified when in the Run state, a normal character is received when the host system is not expecting any N-Chars. NOTE: A credit error can be caused if an error occurs undetected by the parity bit (e.g. two bits in error) which results in one or more spurious FCTs.

Clause 8.9.2.4.b

In the event of a credit error the link interface shall follow the error recovery procedure described in clause 8.9.4.

Clause 8.9.2.4.c
If the credit error occurs in the Run state then the credit error shall be flagged up to the network level as a link error (see clause 8.9.5).
Clause 8.9.2.5.1 Character sequence error
During initialization it is possible for a link interface to receive FCTs or N-Chars when they are not expected. Any unexpected characters are caught by the exchange-level state machine resulting in the link being reset and re-initialized.
Clause 8.9.2.5.2.a
As a character sequence error can only occur during link initialization, it shall not be flagged up to the network level as a link error (see clause 8.9.5).
Clause 8.9.3.1 (*) Handling empty packets
Empty packets, i.e. an EOP or EEP followed immediately by another EOP or EEP, are not expected, but they can occur if a packet terminated by an EEP comprises just the header and the EEP (see clause 11.4) and the header characters are stripped off as the packet progresses through a network leaving just the EOP (see clauses 10.1.2.7 and 10.2.3).
Clause 8.9.3.2.a
In the Run state, if the next N-Char received after an EOP or EEP is received is another EOP or EEP, then the link interface may discard the second EOP or EEP (see clause 10.5.4.3).
Clause 8.9.3.2.b
This error shall not be flagged up to the network level as a link error (see clause 8.9.5). NOTE: In this way empty packets are discarded quietly.
Clause 8.9.4.1 (*) Exchange of silence error recovery procedure
When one end of the link (end A) is disabled or detects an error, it ceases transmission. As described in clause 8.9.2.1, this causes a disconnect error at the other end of the link (end B). End B then ceases transmission, resulting in a disconnect error at end A. This procedure is known as an âĂIJexchange of silenceâĂİ (see Figure 4-5).
Clause 8.9.4.2.a
After an exchange of silence, both ends of the link shall cycle through the reset sequence (ErrorReset, ErrorWait, Ready) ending up in the Ready state ready to begin operation once enabled, proceeding then as follows.
Clause 8.9.4.2.b
If both ends are enabled then they shall move to the Started state and re-initialize.
Clause 8.9.4.2.c
If one end (end A) is disabled and the other end (end B) is enabled then the following series of events shall continue until either end A is enabled or end B is disabled:
1. end B, to move from the Ready state to the Started state, and send NULLs for $12,8\mu s$.
2. Since end A is disabled it cannot respond, but have started its disconnect timer and also have registered that a NULL was received.
3. When end B completes the $12,8\mu s$ timeout, to move to the ErrorReset state and disconnect (stop its output).
4. End A can detect the disconnection so end A to move also to the ErrorReset state.
5. Both ends, to move once again through the reset sequence.
Clause 8.9.5.a Reporting link errors to network level
Receiver errors (i.e. disconnect error, parity error, escape sequence error, character sequence error and credit error) during link initialization (i.e. ErrorReady, ErrorWait, Ready, Started and Connecting states) shall not be reported to the network level.
Clause 8.9.5.b
Once a link connection is established (Run state), a receiver error represents a failure of the link connection and shall be reported to the network level so that appropriate action for error recovery or reporting, as described in clause 10.5, can be taken.
Clause 8.9.5.c
A link error shall be reported to the network level whenever any of the following errors occur while a link interface is in the Run state: disconnect error, parity error, escape sequence error, or credit error.

NOTE: The exclusion of character sequence error from this list is because a character sequence error can only occur during initialization.

Clause 8.10.1 (*) Exception conditions

The following clauses describe the exception conditions where events do not follow the expected sequence.

Clause 8.10.2 Disconnect error while waiting to start

"Waiting to start" means that a link interface is in either the ErrorReset, ErrorWait, Ready or Started state. A disconnect error cannot be detected while waiting to start, unless the other end of the link (end B say) has sent at least one bit, so that the disconnect detect mechanism at end A can be activated. End B then gives up waiting for end A to send a NULL and moves to the ErrorReset state and stops its transmitter, thus causing the disconnect. An alternative possibility is that the link becomes physically disconnected. Table 8-2, Table 8-3, Table 8-4 and Table 8-5 illustrate the various sequences of events starting from when end B has just moved to the ErrorReset state. If a physical disconnection has occurred then both ends of the link continue to try to make a connection, cycling around the reset sequence, until they are disabled or until the connection is re-established.

Clause 8.10.3 Link connected in one direction but not in the other

A link can be connected in one direction and not in the other while a link is in the process of being plugged in or if there is a break in the link cable. In this case events follow the sequence listed in the Table 8-6. For convenience it is assumed that both links are in the Started state and that end A is connected to end B, but end B is not connected to end A.

Clause 8.10.4 Parity error while waiting to start

Parity errors are only recognized after a NULL is received. If a parity error occurs during link initialization the effect is identical to a disconnect error.

Clause 8.10.5 One end starts as other end disconnects

One end (end A) arrives at the Start state $12,8\mu s$ before the other end (end B) arrives at the Start state. The sequence of event is illustrated Table 8-7.

Clause 8.10.6 D connected, S disconnected

If D is connected and S disconnected then the clock generated in the receiver follows the Data signal, i.e. there is a clock edge every time the Data signal changes. This results in a continuous sequence of 0101010101. During initialization this sequence is ignored as it does not produce a NULL so initialization fails until the Strobe signal is properly connected. In this case the link interface cycles round ErrorReset, ErrorWait, Ready, Started until full link connection is achieved or until the link is disabled. If S becomes disconnected after a NULL is received, this sequence produces a parity error for both control characters (4 bits) extracted from the 01010101010 sequence, but does not produce a parity error for data characters (10 bits). If the disconnection occurs so that the 01010101010 sequence is regarded as control characters a parity error is detected and the link interface cycles through ErrorReset, ErrorWait, Ready and Started until S is connected or the link is disabled. If the disconnection occurs so that the 010101010 sequence is regarded as data characters then no parity error is detected and the link interface receives a continuous series of data characters with the value AA hex (NOTE: that SpaceWire is least significant bit first).

Clause 8.10.7 S connected, D disconnected

If S is connected and D disconnected then the clock generated in the receiver follows the Strobe signal, i.e. there is a clock edge every time the Strobe signal changes. This results in a continuous sequence of 1111111111 since the Data signal input goes to 1 when the data line is disconnected. If the data line is shorted to ground then the continuous sequence 0000000000 is received. During initialization either sequence is ignored as it does not produce a NULL so initialization fails until the Data signal is properly connected. In this case the link interface cycles round ErrorReset, ErrorWait, Ready, Started until full link connection is achieved or until the link is disabled. If D becomes disconnected after a NULL is received the received data sequence quickly produces a parity error because the parity is even for both control characters (4 bits) and data characters (10 bits) extracted from the received sequence, whereas the expected parity is odd (see clause 7.4). The parity error causes the link interface to cycle through ErrorReset, ErrorWait, Ready and Started until D is connected or the link is disabled.

Clause 8.10.8 (*) One side of differential pair disconnected

The effect of disconnecting one side of the differential pair depends upon the values of the internal bias resistors at the interface, on the length of cable and on the grounding arrangements. Three cases are possible: The Data and Strobe signals are still received correctly but with a much reduced noise margin. The link then continues operating with a significant increase in the number of detected parity errors. The Strobe signal sits at logic 0 or logic 1. This is similar in effect to the D connected, S disconnected case of clause 8.10.6 The Data signal sits at logic 0 or logic 1. This is similar in effect to the S connected, D disconnected case of clause 8.10.7.

Clause 8.11.1.a D and S reset timing

The delay between the reset of the Strobe signal and the Data signal shall be between 555 ns (i.e. the period of slowest permitted transmit clock, 2 MHz - 10 %) and the shortest clock cycle time for the transmitter (i.e. the period of maximum clock, dependent upon implementation).

Clause 8.11.2.a Disconnect timing

The disconnect timeout of 850 ns nominal shall be from 727 ns (i.e. 8 cycles of 10 MHz + 10 % clock) to 1000 ns (i.e. 9 cycles of 10 MHz - 10 % clock).

Clause 8.11.3.a Exchange timeout periods

The $6,4\mu s$ (nominal) timeout period shall be from $5,82\mu s$ (i.e. 64 cycles of 10 MHz + 10 % clock) to $7,22\mu s$ (i.e. 65 cycles of 10 MHz - 10 % clock).

Clause 8.11.3.b

The $12,8\mu s$ (nominal) timeout period shall be from $11,64\mu s$ (i.e. 128 cycles of 10 MHz + 10 % clock) to $14,33\mu s$ (i.e. 129 cycles of 10 MHz - 10 % clock).

Clause 8.12.1 (*)

As defined in clause 7.4, a Time-Code comprises the ESC character followed by a single 8-bit data character. The data character contains two control flags and a sixÂŋbit time-count which can be the least significant six bits of system time.

Since the SpaceWire IP core only transmit and receive Time-Codes, all clauses related to time counters logic are not evaluated.

Clause 8.12.2.a (*)

Each node or router shall contain one six-bit time counter.

Clause 8.12.2.b (*)

A single link interface shall manage the distribution of time.

Clause 8.12.2.c (*)

The time-master interface shall have a TICK_IN input, which is asserted periodically by its host system. NOTE: For example, every millisecond.

Clause 8.12.2.d (*)

When the time master link interface receives a tick (TICK_IN asserted) it shall increment its time-counter and then send out a Time-Code with the six-bit time field of the data character set to the new value of the time-counter and the other two bits set to the value of the control flags. NOTE: The frequency at which the TICK_IN signal is driven is called the tick rate.

Clause 8.12.2.e

The Time-Code shall be sent out as soon as the current character or control code is transmitted.

Clause 8.12.2.f

Time-Codes shall not be sent out until a link interface is in the Run state NOTE: For the Run state, see clause 8.5.

Clause 8.12.2.g (*)

When the link interface at the other end of the link receives the Time-Code, it shall update its associated time-counter with the new time and assert its TICK_OUT output signal and copy the values of the two control flags in the Time-Code to the control-flag outputs.

Clause 8.12.2.h (*)

The new time should be one more than the time-counter‘s previous time-value. NOTE: This fact can be used for checking on Time-Code validity.

Clause 8.12.2.i (*)

If a link interface receives a Time-Code that is equal to its current six-bit time value then it shall not emit a TICK_OUT output signal. NOTE: This prevents repeated Time-Code propagation in a circular network.

Clause 8.12.2.j (*)

When a link interface on a router or node receives a Time-Code it shall check that it is one more (modulo 64) than its current time setting.

Clause 8.12.2.k (*)

The router or node shall then increment the router's time-count and emit a tick signal. NOTE: This tick signal propagates to all the output ports of the router so that they all emit the Time-Code. This Time-Code is the same value as that received by the router since the router time-counter has been incremented. If there is a circular connection then the router receives a Time-Code with the same time value as the router time-counter. The control-flags of the Time-Codes that are emitted are set to the control flag values of the received Time-Code that caused the subsequent emission of Time-Codes by the router.

Clause 8.12.2.l (*)

If the router or node receives a Time-Code with the same time value as the router time-counter the Time-Code shall be ignored. NOTE 1: In this way, time flows forwards through a network reaching all nodes, but is suppressed if it flows backwards due to a circular connection. NOTE 2: With the provision of this basic time-distribution function, application level protocols can be used, for example, to distribute specific time values at full resolution (not just 6 bits) and to issue time dependent commands.

Clause 8.12.2.m (*)

After reset or disconnect-reconnect (state machine in ErrorReset state) the time-counter shall be set to zero and any control-flag outputs shall be set to zero.

Clause 8.12.2.n (*)

If a received Time-Code is not one more than (modulo 64) or equal to the current time-count at the receiving link interface, then either the Time-Code or the time-count shall be considered invalid. NOTE: This can happen if a Time-Code is lost, or if a link is reset or restarted after a disconnect.

Clause 8.12.2.o (*)

If the Time-Code is invalid then the time-count shall be updated to the new value but the tick-output shall not be asserted. NOTE 1: This prevents propagation of invalid Time-Codes across a network. When the next Time-Code is received it is expected that the time-counter matches the Time-Code and normal operation resumes. NOTE 2: Nodes using the system time distribution function can either use the TICK_OUT signal as a periodic timing signal or use the value of the time-count as an indication of the least significant 6 bits of system time.

Clause 8.12.2.p.1 (*)

As a missing tick results in a timing discrepancy the following should be done: 1. The TICK_OUT signal not be used to increment a counter with the expectation that this counter corresponds to the system time.

Clause 8.12.2.p.2 (*)

2. Rather a time-lock or phase-lock technique be used where a free running local time-counter is updated to be an exact multiple of the system tick rate every time the TICK_OUT signal is asserted.

NOTE 1: The reason for this is that when using the TICK_OUT signal as a periodic timing signal the Time-Code can be missed so that a TICK_OUT signal is missed. NOTE 2: The accuracy with which system time can be distributed is dependant upon the number of links over which it is distributed and the operating rate of each of those links. A delay of at least 14 bit-periods (ESC + data character = (4 + 10) bits) is encountered for each link that the Time-Code traverses, due to the time taken for each link interface on the way to receive a Time-Code. This gives rise to a time-skew across a network of STskew = 14N/R, where N is the number of links traversed and R is the average link operating rate. Jitter is also introduced at each link interface due to the variation in time spent waiting for the transmitter to finish transmitting the current character or control code. At each link interface a delay of 0 bit periods to 10 bit periods can be encountered. Across a network, this gives rise to a total jitter of STjitter = 10N/R. For an average rate of 100 Mb/s and 10 links traversed, the time skew is 1,4 Îijs and the jitter 1,0 Îijs. The skew and jitter may be higher than indicated above depending

on the implementation of the link interface. A time accuracy across a network of better than 10 Îijs can be difficult to achieve.

Clause 9.1 (*)

The packet level protocol agrees with the principles of the DS-SE and DS-DE character level encoding given in clause 9 of the IEEE Standard 1355-1995 [1]. Some ambiguities in the IEEE Standard 1355-1995 are resolved in this Standard. See annex A for details of the differences between SpaceWire and IEEE Standard 1355-1995 and the reasons for the differences.

Clause 9.2.1 (*)

A packet shall comprise a destination address, a cargo and an end_of_packet (EOP or EEP) marker, i.e. <destination address><cargo><end_of_packet>

Clause 9.2.2.a (*)

The destination address shall consist of a list of zero or more destination identifiers (dest_id): <destination address>= <dest_id1><dest_id2>0 <dest_idN>

Clause 9.2.2.b (*)

A destination identifier shall comprise one data character.

Clause 9.2.2.c (*)

The destination identifier list shall not be delimited. NOTE 1: The case of zero destination identifiers in the destination list (i.e. the destination list is empty) is intended to support a network which is simply a single point-to-point link from source to destination. NOTE 2: The case of one or more destination identifiers in the destination list is intended to support routing of a packet across a network.

Clause 9.2.3.a (*)

The cargo shall comprise the data characters that are to transfer from the source to the destination.

Clause 9.2.3.b (*)

The cargo shall contain one or more characters. NOTE: Clause 8.9.3 describes what happens to empty packets.

Clause 9.2.3.c (*)

Empty cargoes shall be allowed to move across the network. NOTE: A cargo can become empty if there is an error on the link.

Clause 9.2.4.a (*)

There shall be two possible end_of_packet markers: EOP and EEP. NOTE: The EOP and EEP control character formats are described in clause 7.3 of this Standard.

Clause 9.2.4.b (*)

EOP (end_of_packet) shall be used as the normal end of packet marker indicating the end of a packet.

Clause 9.2.4.c (*)

EEP (error end_of_packet) shall be used to indicate the end of a packet in which an error occurs. The data in this packet can be valid, but the packet shall be prematurely terminated at the point that the error occurred.

Clause 9.2.4.d (*)

The first data character following either end_of_packet marker shall be taken as the first character of the next packet.

Clause 9.3.a (*)

N-Chars from one packet shall not be interleaved with N-Chars from another packet. NOTE: N-Chars can be interleaved with FCTs, NULLs and Time-Codes as explained in 8.3s.

Clause 11.4 Link error recovery

If any form of error is detected within the link interface then the following sequence of events occurs to recover from the error (see Figure 11-1):

1. Detect error (disconnect, parity, escape sequence, character sequence, credit).
2. Disconnect link.
3. If previous character was NOT EOP then add EEP (error end of packet) to the receiver buffer (i.e. the receive FIFO in Figure 11-1).

4. Delete data in the transmitter buffer (i.e. transmit FIFO in Figure 11-1) until the next EOP (End of Packet).
5. Reconnect.
6. Send next packet.

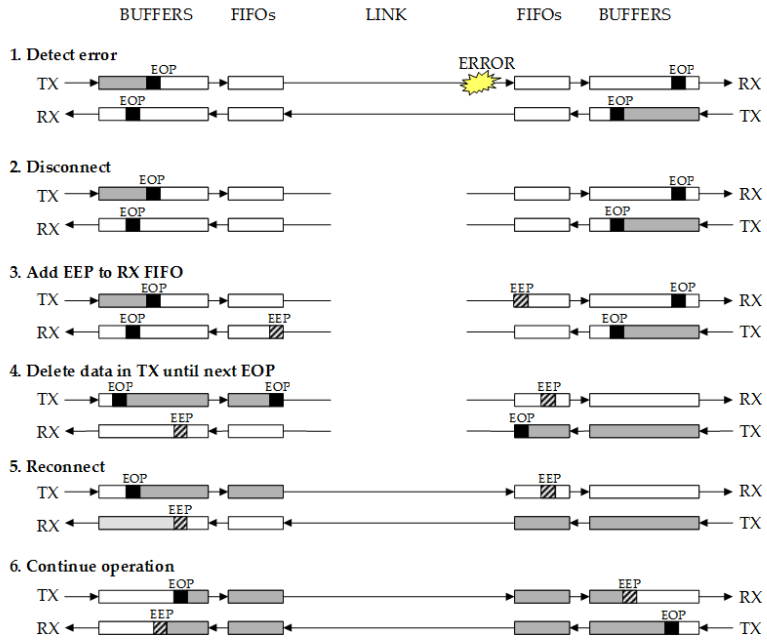


Figure 11-1 from the SpaceWire standard. Link interface error detection and recovery operation.

3 Error and Corner Cases

The error and corner cases are grouped in TestCases (TC) in the testbench.

1. TC Reset

   (a) Corner Case: Reset
   Evaluates the Link FSM response and signals after reset is asserted in each FSM state. The evaluated signals are: Data_out and Strobe_out. The evaluation includes: the timing of transitions and the checking of simultaneous transitions).

   When the SPWIP is reset by the rst_n signal, it is not possible to guarantee that Data_out and Strobe_out will not switch simultaneously to zero. On the other side, when the transmitter is disabled by the link FSM, it exists a controlled mechanism to reset Data_out and Strobe_out signals.

   Also is checked the timing for the $6.4\mu s$ and $12.8\mu s$ counters.
   The case covers the following clauses: 6.3.2.b, 7.5.a, 7.5.b, 8.3.h, 8.3.m, 8.5.2.2.a, 8.5.2.2.b, 8.5.2.2.c, 8.5.2.2.d, 8.11.1.a, 8.11.3.a, 8.11.3.b

2. TC Link Initialization (clause 8.7)

   (a) Error Case: Wrong parity in first NULL and timeout in Started state
   After reaching the Started state, the input pattern generator is set to 11, so the first NULL bit pattern is generated but with bad parity in each of the parity bits of the first NULL. Since this first NULL should not be acted, the case waits for the timeout of $12.8\mu s$. If this happens, the bpb1n (bad parity before 1st NULL) testbench signal is asserted for one clock cycle and the coverage bin BPB1N_BIN is covered.
   The case covers the following clauses: 8.2.2.b, 8.5.2.2.a, 8.5.2.5.b, 8.5.2.5.h, 8.5.3.2.f, 8.5.3.7.3.b, 8.11.3.b

   (b) Corner Case: Wrong sequence of bits instead of first NULL
   Test different sequence of bits that are not first NULL bit pattern.
   The case covers the following clauses: 8.5.3.2.c, 8.5.3.9.b

   (c) Error Case: Timeout in the Connecting state
   Send NULLs up to the connecting state and wait for timeout $12.8\mu s$.
   The case covers the following clauses: 8.5.2.6.b, 8.5.2.6.f, 8.11.3.b

   (d) Corner Case: Disable in Ready state
   Evaluate link disable in Ready state. The FSM should stay in Ready state when link_disable = '1' even if link_start or link_auto_start are '1'.
   The case covers the following clauses: 8.6.c

   (e) Corner Case: Disable in Run state
   Send NULLs, FCT and data up to the run state and disable the link.
   The case covers the following clauses: 8.5.2.7.b, 8.5.3.6

   (f) Corner Case: Relationship of FCT to data (Tx and Rx)
   Evaluate if the UUT is able to Rx 7 FCTs (maximum number of FCTs after reset) and should only send a maximum of 7 FCTs. Also if the UUT only Tx data when is allowed (Rx FCTs).
   The case covers the following clauses: 8.3.d, 8.3.e, 8.3.f, 8.3.g, 8.3.j, 8.3.k, 8.3.n

3. TC Handling Receiver Errors (clause 8.9.2)

   The testcase covers the following clauses: 8.5.2.2.a, 8.5.2.3.e, 8.5.2.4.e, 8.5.2.5.g, 8.5.2.6.e, 8.5.2.7.b, 8.9.1.a, 8.9.1.b, 8.9.5.a, 8.9.5.b, 8.9.5.c

(a) Error Case: Disconnect Error
Evaluates the FSM response and signals after a disconnect error in each FSM state.
The case covers the following clauses: 8.5.3.7.2.a, 8.5.3.7.2.b, 8.9.2.1.2.a, 8.9.2.1.2.b, 8.9.2.1.2.c, 8.9.2.1.2.f, 8.10.2, 8.11.2.a

(b) Error Case: Parity Error
Evaluates the FSM response and signals after a parity error in each FSM state. Not possible to evaluate in Started state.
The case covers the following clauses: 8.5.3.7.3.a, 8.5.3.7.3.b, 8.9.2.2.a, 8.9.2.2.c, 8.10.4

(c) Error Case: Escape Error
Evaluates the FSM response and signals after an escape error in each FSM state. Not possible to evaluate in Started state.
The case covers the following clauses: 7.3.e, 8.5.3.7.4.a, 8.5.3.7.4.b, 8.9.2.3.a, 8.9.2.3.c

(d) Error Case: Credit Error
Evaluates the FSM response and signals after a credit error in each FSM state. Only possible in the Run state. There are two types of credit error: when Rx more data than expected or when Rx more FCT than expected.
The case covers the following clauses: 8.3.i, 8.5.3.8.a, 8.9.2.4.a, 8.9.2.4.c

(e) Error Case: Character sequence Error
Evaluates the FSM response and signals after a character sequence error in each FSM state. There are three types of character sequence error: when Rx FCT unexpected, when Rx N-char (data, EOP or EEP) unexpected or when Rx timecode unexpected.
The case covers the following clauses: 8.5.3.3.a, 8.5.3.4.a, 8.5.3.5.a, 8.5.3.9.a, 8.5.3.9.c, 8.5.3.9.d, 8.9.2.5.1, 8.9.2.5.2.a

4. TC Handling Empty Packets (clause 8.9.3)

(a) Corner Case: Empty packets
Evaluate if the UUT is discarding received empty packets. An empty packet is defined as an EOP or EEP followed immediately by another EOP or EEP.
The case covers the following clauses: 8.9.1.c, 8.9.3.2.a, 8.9.3.2.b

5. TC Exchange of silence error recovery procedure (clause 8.9.4)

(a) Corner Case: Exchange of silence: both links enabled
Evaluating exchange of silence procedure. Both links enabled.
The case covers the following clauses: 8.9.4.2.a, 8.9.4.2.b

(b) Corner Case: Exchange of silence: one link enabled
Evaluating exchange of silence. One link enabled.
The case covers the following clauses: 8.9.4.2.a, 8.9.4.2.c

(c) Corner case: The exchange of silence is triggered by a disconnect error.
The case covers the following clauses: 8.9.2.1.2.e

(d) Corner case: The exchange of silence is triggered by a parity error.
The case covers the following clauses: 8.9.2.2.b

(e) Corner case: The exchange of silence is triggered by an escape error.
The case covers the following clauses: 8.9.2.3.b

(f) Corner case: The exchange of silence is triggered by a credit error.
The case covers the following clauses: 8.9.2.4.b

6. TC Exception Conditions (clause 8.10)

   (a) Error Case: Simultaneous switching in data and strobe input signals
   The case covers the following clauses: 6.3.2.a

   (b) Error Case: Link connected in one direction but not in the other
   The case covers the following clauses: 8.9.2.1.2.d, 8.10.3

   (c) Corner Case: one end starts while the other end disconnects
   The case covers the following clauses: 8.10.5

   (d) Error Case: D connected, S disconnected
   The case covers the following clauses: 8.10.6

   (e) Error Case: S connected, D disconnected
   The case covers the following clauses: 8.10.7

   (f) Error Case: link error while transmitting/receiving a packet
   The case covers the following clauses: 11.4

7. TC Normal Operation (clause 8.8)
   Normal operation including: normal link initialization, send of data, timecodes, EOP. The testcase covers the following clauses: 6.3.1.c, 6.3.1.d, 6.6.5.a, 6.6.5.b, 6.6.6.a, 7.2.b, 7.2.c, 7.3.b, 7.3.c, 7.4.c, 7.6.a, 7.7.b, 7.7.c, 8.2.2.a, 8.3.b, 8.3.c, 8.3.o, 8.3.p, 8.3.q, 8.3.r, 8.3.s, 8.5.2.3.a, 8.5.2.3.b, 8.5.2.3.c, 8.5.2.3.d, 8.5.2.4.a, 8.5.2.4.a, 8.5.2.4.b, 8.5.2.4.c, 8.5.2.4.d, 8.5.2.5.a, 8.5.2.5.c, 8.5.2.5.d, 8.5.2.5.e, 8.5.2.5.f, 8.5.2.6.a, 8.5.2.6.c, 8.5.2.6.d, 8.5.2.7.a, 8.5.3.2.a, 8.5.3.2.b, 8.5.3.2.e, 8.5.3.3.a, 8.5.3.4.a, 8.6.b, 8.7, 8.8, 8.12.2.e, 8.12.2.f

   (a) Corner Case: RX FIFO Full
   Evaluate the correct operation when RX FIFO is full.
   The case covers the following clauses: 8.8

   (b) Corner Case: RX FIFO Empty
   Evaluate the correct operation when RX FIFO is empty.
   The case covers the following clauses: 8.8

   (c) Corner Case: TX FIFO Full
   Evaluate the correct operation when TX FIFO is full.
   The case covers the following clauses: 8.8

   (d) Corner Case: TX FIFO Empty
   Evaluate the correct operation when TX FIFO is empty.
   The case covers the following clauses: 8.8

   (e) Corner Case: Minimum data signalling rate
   Evaluate the correct operation when the minimum data signalling rate is set. It should evaluate transmission and receiving at this data rate.
   The case covers the following clauses: 6.6.1.a

   (f) Corner Case: Maximum data signalling rate
   Evaluate the correct operation when the maximum data signalling rate is set. It should evaluate transmission and receiving at this data rate.
   The case covers the following clauses: 6.6.2.a

   (g) Corner Case: One link TX and RX at different rates
   Evaluate the correct operation when the link has different transmission and receiving data rates.
   The case covers the following clauses: 6.6.3.a

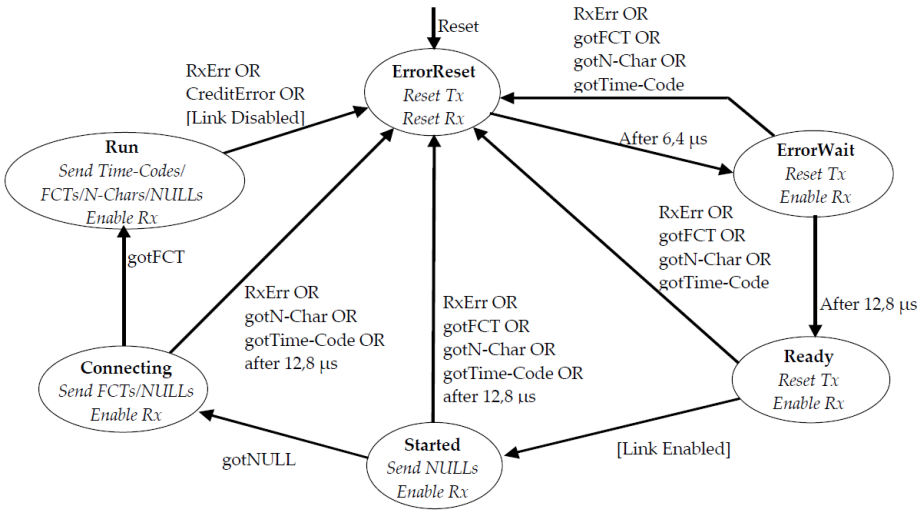
(h) Corner Case: Send special N-chars: EOP, diff EOP, EEP and diff EEP. A diff EOP/EEP means that a different byte from 0x00 (EOP) or 0x01 (EEP) is sent from the host interface. The UUT should only evaluate the LSB bit to determine which type of end of packet should be sent.
Evaluate the correct operation when the link sends EOP, diff EOP, EEP and diff EEP.
The case covers the following clauses: 7.6.a, 7.6.b, 7.6.c

4 Description of Functional Coverage Points

Coverage points used in more than one testcase (TC)

4.1 cp_link_fsm

Functional coverage model of the link FSM. Along with FSM conditions, it is possible to model the complete FSM. This coverpoint is covered when all possible state transitions and conditions are evaluated. The FSM state diagram appears in Fig. 8-2 from the SpaceWire standard.



RxErr = Disconnect error OR Parity error OR Escape error (ESC followed by EOP or EEP or ESC).

NOTE Disconnect error only enabled after First Bit Received. Parity Error, Escape Error, gotFCT, gotN-Char, gotTime-Code only enabled after First NULL Received (i.e. gotNULL asserted). Thus RxErr OR gotFCT OR gotN-Char OR gotTime-Code is really RxErr OR (gotNULL AND (gotFCT OR gotN-Char OR gotTime-Code)).

Figure 8-2 from the SpaceWire standard.

The coverpoint covers the following clauses: 8.5.2.2.a, 8.5.2.2.d, 8.5.2.2.c, 8.5.2.3.a, 8.5.2.3.b, 8.5.2.3.d, 8.5.2.3.e, 8.5.2.4.a, 8.5.2.4.d, 8.5.2.4.e, 8.5.2.5.a, 8.5.2.5.b, 8.5.2.5.e, 8.5.2.5.g, 8.5.2.5.h, 8.5.2.6.a, 8.5.2.6.d, 8.5.2.6.e, 8.5.2.6.f, 8.5.2.7.a, 8.5.2.7.b, 8.5.3.3.a, 8.5.3.4.a, 8.5.3.6, 8.6.b, 8.6.c, 8.7, 8.8, 8.9.1.a, 8.9.1.b

TC reset coverage points

4.2 cp_xmite

Verify that the transmitter is only enabled in the Started, Connecting and Run states.
The coverpoint covers the following clauses: 8.5.2.2.b, 8.5.2.3.b, 8.5.2.4.b, 8.5.2.5.c, 8.5.2.6.c

4.3 cp_recve

Verify that the receiver is enabled in all the states except the ErrorReset state.
The coverpoint covers the following clauses: 8.5.2.2.b, 8.5.2.3.b, 8.5.2.4.b, 8.5.2.5.c, 8.5.2.6.c

4.4 cp_txcredzero

In ErrorReset state, the transmission credit (tx_credit) should be zero.
The coverpoint covers the following clauses: 8.3.h

4.5 cp_rxcredzero

In ErrorReset state, the receiving credit (rx_credit) should be zero.
The coverpoint covers the following clauses: 8.3.m

4.6 cp_ds_status

Data_out and Strobe_out should be zero ('0') in the ErrorReset state.
The coverpoint covers the following clauses: 7.5.a, 8.11.1.a

4.7 cp_sim_ds_switch

Evaluates the data and strobe output signals never switch simultaneously.
When the SPWIP is reset by the rst_n signal, it is not possible to guarantee that Data_out and Strobe_out will not switch simultaneously to zero. On the other side, when the transmitter is disabled by the link FSM, it exists a controlled mechanism to reset Data_out and Strobe_out signals.
The coverpoint covers the following clauses: 6.3.2.b

4.8 cp_tmout1

Verify that the $6.4\mu s$ timeout is in the range: $[5.82 - 7.22]\mu s$.
The coverpoint covers the following clauses: 8.5.2.2.c, 8.5.2.2.d, 8.11.3.a

4.9 cp_ds_out_init

Monitors output data-strobe signals to check that the first transition after reset is on the strobe signal.
The coverpoint covers the following clauses: 7.5.b

TC link initialization coverage points

4.10 cp_charbadpar

L or N-chars received with wrong parity should not be acted. Checks for bad parity response before the first NULL and after the first NULL. The "after first NULL" case is best covered by the cp_parerr coverage point.
The coverpoint covers the following clauses: 8.2.2.b, 8.5.3.2.f, 8.5.3.7.3.b

4.11 cp_gotNULL

Verify the reception of the first NULL. It is valid if it is received in any state except in ErrorReset.

The coverpoint covers the following clauses: 8.5.2.3.c, 8.5.2.4.c, 8.5.2.5.d, 8.5.3.2.a, 8.5.3.2.b, 8.5.3.2.c, 8.5.3.2.e, 8.5.3.2.f, 8.5.3.9.b

4.12 cp_tmout2

Verify that the $12.8\mu s$ timeout is in the range: $[11.64 - 14.33]\mu s$.
The coverpoint covers the following clauses: 8.5.2.5.b, 8.5.2.6.b, 8.11.3.b

4.13 cp_txdcond

It verifies two scenarios. First, if it is not received a FCT character, no N-chars are transmitted. And second, If it is received FCT characters, only the allowed number of N-chars bytes are transmitted.
The coverpoint covers the following clauses: 8.3.d, 8.3.e, 8.3.f, 8.3.g

4.14 cp_rx7FCT

Verify that the IP should be able to receive 7 FCT chars without credit errors.
The coverpoint covers the following clauses: 8.3.j

4.15 cp_txm7FCT

Verify that after reset or link disconnect the maximum number of FCTs send is 7 and RX credit = 56.
The coverpoint covers the following clauses: 8.3.k, 8.3.n

TC receiver errors coverage points

4.16 cp_discerr

Detect correctly a disconnect error. It is also verified the timing $850ns$ (nominal). Range: $[727 - 1000]ns$.
The coverpoint covers the following clauses: 8.5.2.2.a, 8.5.3.7.2.a, 8.5.3.7.2.b, 8.9.2.1.2.a, 8.9.2.1.2.b, 8.9.2.1.2.c, 8.10.2, 8.11.2.a

4.17 cp_parerr

Detect correctly a parity error.
The coverpoint covers the following clauses: 8.5.2.2.a, 8.5.3.7.3.a, 8.5.3.7.3.b, 8.9.2.2.a, 8.10.4

4.18 cp_escerr

Detect correctly an escape error.
The coverpoint covers the following clauses: 7.3.e, 8.5.2.2.a, 8.5.3.7.4.a, 8.5.3.7.4.b, 8.9.2.3.a

4.19 cp_crederr

Detect correctly a credit error.
The coverpoint covers the following clauses: 8.3.i, 8.5.2.2.a, 8.5.3.8.a, 8.9.2.4.a,

4.20 cp_chserr

Detect correctly a character sequence error.
The coverpoint covers the following clauses: 8.5.2.2.a, 8.5.3.5.a, 8.5.3.9.a, 8.5.3.9.c, 8.5.3.9.d, 8.9.2.5.1

4.21 cp_crlerr

Verify that errors are only reported in the Run state.
The coverpoint covers the following clauses: 8.9.2.1.2.f, 8.9.2.2.c, 8.9.2.3.c, 8.9.2.4.c, 8.9.2.5.2.a, 8.9.5.a, 8.9.5.b, 8.9.5.c

TC empty packets coverage points

4.22 cp_rxepkt

Handling empty packets. Receiving consecutive EEP or EOP chars is not an error, the empty packet should be discarded.
The coverpoint covers the following clauses: 8.9.1.c, 8.9.3.2.a, 8.9.3.2.b

TC exchange of silence coverage points

4.23 cp_exchng_silence

Exchange of silence error recovery procedure. Evaluates when both links are enable. Evaluates when one link is enable. Evaluates using the link disable pin.
The coverpoint covers the following clauses: 8.9.2.1.2.e, 8.9.2.2.b, 8.9.2.3.b, 8.9.2.4.b, 8.9.4.2.a, 8.9.4.2.b, 8.9.4.2.c

TC exception conditions coverage points

4.24 cp_ds_in_simtrans

Data and Strobe inputs simultaneous transitions. Evaluates that the SpaceWire receiver is tolerant to simultaneous transitions on the Data and Strobe lines.
The coverpoint covers the following clauses: 6.3.2.a

4.25 cp_linkcon_onedir

Link connected in one direction but not in the other. Evaluates the behavior of the SpaceWire IP when it is only connected in one direction. If both are set to link start, both FSM should cycle between ErrorReset and Started states.
The coverpoint covers the following clauses: 8.9.2.1.2.d, 8.10.3

4.26 cp_onestarts_otherdiscon

One end starts while the other end disconnects. Evaluates the behavior of the SpaceWire IP when one FSM arrive to the Started state $12,8\mu s$ after the other end arrived to the Started state.
The coverpoint covers the following clauses: 8.10.5

4.27 cp_d_con_s_discon

D connected, S disconnected. Evaluates the behavior of the SpaceWire IP when only the data line is connected and the strobe line is disconnected.
The coverpoint covers the following clauses: 8.10.6

4.28 cp_s_con_d_discon

S connected, D disconnected. Evaluates the behavior of the SpaceWire IP when only the strobe line is connected and the data line is disconnected.
The coverpoint covers the following clauses: 8.10.7

4.29 cp_lnk_err_recov

Link error recovery. Evaluates the behavior of the SpaceWire IP when a link error happens when transmitting/receiving a data packet. The transmitted packet should be spilled from the TXFIFO and after the last received byte from the received data packet should be added an EEP marker.
The coverpoint covers the following clauses: 11.4

TC normal operation coverage points

4.30 cp_tx_full

Verify the operation of IP when the Tx FIFO is full.
The coverpoint covers the following clauses: 8.8

4.31 cp_tx_empty

Verify the operation of IP when the Tx FIFO is empty.
The coverpoint covers the following clauses: 8.8

4.32 cp_rx_full

Verify the operation of IP when the Rx FIFO is full.
The coverpoint covers the following clauses: 8.8

4.33 cp_rx_empty

Verify the operation of IP when the Rx FIFO is empty.
The coverpoint covers the following clauses: 8.8

4.34 cp_illegal_disc

Verify that no disconnection error occurs in TC2SPW.
The coverpoint covers the following clauses: 8.8

4.35 cp_illegal_par

Verify that no parity error occurs.
The coverpoint covers the following clauses: 7.4.c, 8.8

4.36 cp_illegal_esc

Verify that no escape error occurs.
The coverpoint covers the following clauses: 8.8

4.37 cp_illegal_cred

Verify that no credit error occurs.
The coverpoint covers the following clauses: 8.8

4.38 cp_tx_rx_null_chars

Handling null chars. Includes the correct detection of first NULL.
The coverpoint covers the following clauses: 7.3.b

4.39 cp_tx_rx_fct_chars

Handling FCT chars
The coverpoint covers the following clauses: 8.8

4.40 cp_tx_rx_eop_chars

Handling EOP chars
The coverpoint covers the following clauses: 7.6.a, 7.6.c

4.41 cp_tx_rx_eep_chars

Handling EEP chars
The coverpoint covers the following clauses: 7.6.a, 7.6.b, 7.6.c

4.42 cp_tx_rx_esc_chars

Handling ESC chars
The coverpoint covers the following clauses: 8.8

4.43 cp_tx_rx_data

Handling data chars
The coverpoint covers the following clauses: 7.2.b, 7.2.c, 7.6.a

4.44 cp_tx_rx_timecode

Tx timecode chars. Send only in Run state. Send immediately. Rx timecode chars. Receive only in Run state.
The coverpoint covers the following clauses: 7.3.c, 7.7.b , 7.7.c, 8.12.2.f

4.45 cp_illegal_link_char

L-chars should not appear in the host data interface.
The coverpoint covers the following clauses: 8.2.2.a

4.46 cp_fct_send_condition

Checks that FCT send when at least 8 bytes are free and not reserved in Rx FIFO.
The coverpoint covers the following clauses: 8.3.b, 8.3.c, 8.3.o

4.47 cp_snullreq

Verify the request to send nulls in the started state.
The coverpoint covers the following clauses: 8.5.2.5.f

4.48 cp_sfctreq

Verify the request to send FCTs in the connecting state.
The coverpoint covers the following clauses: 8.5.2.6.c

4.49 cp_2_min_rate

Evaluates that the link can run at the minimum rate of 2 Mbps.
The coverpoint covers the following clauses: 6.6.1.a

4.50 cp_100_max_rate

Evaluates that the link can run at the maximum rate of 100 Mbps.
The coverpoint covers the following clauses: 6.6.2.a

4.51 cp_tx_rx_diff_rates

TX and RX at different rates. Evaluated at UUT1. Verify that the Tx rate can only be changed after connection. Verify that the initial Tx rate is 10 Mbps.
The coverpoint covers the following clauses: 6.6.3.a, 6.6.5.a, 6.6.5.b, 6.6.6.a

4.52 cp_diff_EOP_tx

Handling a Tx of different code from EOP (X"00").
The coverpoint covers the following clauses: 7.6.a

4.53 cp_diff_EEP_tx

Handling a Tx of different code from EEP (X"01").
The coverpoint covers the following clauses: 7.6.a

4.54 cp_chars_priority_on_tx

Evaluates the tx priority of chars.
The coverpoint covers the following clauses: 8.3.p, 8.3.q, 8.3.r, 8.3.s, 8.12.2.e

5 Traceability Matrix

The following table presents all relevant clauses from SpW standard and which error/corner cases and coverage points covered the respective clause.
(*) Clauses with only general descriptive or informative text

Clause	Covered by Error/Corner Cases	Covered by Coverpoints
6.3.1.a (*)		
6.3.1.b (*)		
6.3.1.c	7	
6.3.1.d	7	
6.3.2.a	6a	4.24
6.3.2.b	1a	4.7
6.5.a (*)		
6.6.1.a	7e	4.49
6.6.2.a	7f	4.50
6.6.3.a	7g	4.51
6.6.3.b (*)		
6.6.4 (*)		
6.6.5.a	7	4.51
6.6.5.b	7	4.51
6.6.6.a	7	4.51
7.1 (*)		
7.2.a (*)		
7.2.b	7	4.43
7.2.c	7	4.43
7.3.a (*)		
7.3.b	7	4.38
7.3.c	7	4.44
7.3.d (*)		
7.3.e	3c	4.18
7.4.a (*)		
7.4.b (*)		
7.4.c	7	4.35
7.5.a	1a	4.6
7.5.b	1a	4.9
7.6.a	7, 7h	4.40, 4.41, 4.43, 4.52, 4.53
7.6.b	7h	4.41
7.6.c	7h	4.40, 4.41
7.7.a (*)		
7.7.b	7	4.44

Clause	Covered by Error/Corner Cases	Covered by Coverpoints
7.7.c	7	4.44
7.7.d (*)		
7.7.e (*)		
7.7.f (*)		
7.7.g (*)		
7.7.h (*)		
8.1 (*)		
8.2.1 (*)		
8.2.2.a	7	4.45
8.2.2.b	2a	4.10
8.3.a (*)		
8.3.b	7	4.46
8.3.c	7	4.46
8.3.d	2f	4.13
8.3.e	2f	4.13
8.3.f	2f	4.13
8.3.g	2f	4.13
8.3.h	1a	4.4
8.3.i	3d	4.19
8.3.j	2f	4.14
8.3.k	2f	4.15
8.3.l (*)		
8.3.m	1a	4.5
8.3.n	2f	4.15
8.3.o	7	4.46
8.3.p	7	4.54
8.3.q	7	4.54
8.3.r	7	4.54
8.3.s	7	4.54
8.4.1 (*)		
8.4.2 (*)		
8.4.3 (*)		
8.4.4 (*)		
8.4.5 (*)		
8.4.6 (*)		
8.4.7 (*)		
8.4.8 (*)		
8.4.9 (*)		
8.5.1 (*)		
8.5.2.1 (*)		
8.5.2.2.a	1a, 2a, 3	4.1, 4.16, 4.17, 4.18, 4.19, 4.20
8.5.2.2.b	1a	4.2, 4.3
8.5.2.2.c	1a	4.1, 4.8
8.5.2.2.d	1a	4.1, 4.8
8.5.2.3.a	7	4.1

Clause	Covered by Error/Corner Cases	Covered by Coverpoints
8.5.2.3.b	7	4.1, 4.2, 4.3
8.5.2.3.c	7	4.11
8.5.2.3.d	7	4.1
8.5.2.3.e	3	4.1
8.5.2.4.a	7, 7	4.1
8.5.2.4.b	7	4.2, 4.3
8.5.2.4.c	7	4.11
8.5.2.4.d	7	4.1
8.5.2.4.e	3	4.1
8.5.2.5.a	7	4.1
8.5.2.5.b	2a	4.1, 4.12
8.5.2.5.c	7	4.2, 4.3
8.5.2.5.d	7	4.11
8.5.2.5.e	7	4.1
8.5.2.5.f	7	4.47
8.5.2.5.g	3	4.1
8.5.2.5.h	2a	4.1
8.5.2.6.a	7	4.1
8.5.2.6.b	2c	4.12
8.5.2.6.c	7	4.2, 4.3, 4.48
8.5.2.6.d	7	4.1
8.5.2.6.e	3	4.1
8.5.2.6.f	2c	4.1
8.5.2.7.a	7	4.1
8.5.2.7.b	2e, 3	4.1
8.5.3.1.1 (*)		
8.5.3.1.2 (*)		
8.5.3.1.3 (*)		
8.5.3.2.a	7	4.11
8.5.3.2.b	7	4.11
8.5.3.2.c	2b	4.11
8.5.3.2.d (*)		
8.5.3.2.e	7	4.11
8.5.3.2.f	2a	4.10, 4.11
8.5.3.3.a	3e, 7	4.1
8.5.3.4.a	3e, 7	4.1
8.5.3.5.a	3e	4.20
8.5.3.6	2e	4.1
8.5.3.7.1 (*)		
8.5.3.7.2.a	3a	4.16
8.5.3.7.2.b	3a	4.16
8.5.3.7.3.a	3b	4.17
8.5.3.7.3.b	2a, 3b	4.10, 4.17
8.5.3.7.4.a	3c	4.18
8.5.3.7.4.b	3c	4.18

Clause	Covered by Error/Corner Cases	Covered by Coverpoints
8.5.3.8.a	3d	4.19
8.5.3.9.a	3e	4.20
8.5.3.9.b	2b	4.11
8.5.3.9.c	3e	4.20
8.5.3.9.d	3e	4.20
8.6.a (*)		
8.6.b	7	4.1
8.6.c	2d	4.1
8.7	7	4.1
8.8	7, 7a, 7b, 7c, 7d	4.1, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35, 4.36, 4.37, 4.39, 4.42
8.9.1.a	3	4.1
8.9.1.b	3	4.1
8.9.1.c	4a	4.22
8.9.2.1.1 (*)		
8.9.2.1.2.a	3a	4.16
8.9.2.1.2.b	3a	4.16
8.9.2.1.2.c	3a	4.16
8.9.2.1.2.d	6b	4.25
8.9.2.1.2.e	5c	4.23
8.9.2.1.2.f	3a	4.21
8.9.2.2.a	3b	4.17
8.9.2.2.b	5d	4.23
8.9.2.2.c	3b	4.21
8.9.2.3.a	3c	4.18
8.9.2.3.b	5e	4.23
8.9.2.3.c	3c	4.21
8.9.2.4.a	3d	4.19
8.9.2.4.b	5f	4.23
8.9.2.4.c	3d	4.21
8.9.2.5.1	3e	4.20
8.9.2.5.2.a	3e	4.21
8.9.3.1 (*)		
8.9.3.2.a	4a	4.22
8.9.3.2.b	4a	4.22
8.9.4.1 (*)		
8.9.4.2.a	5a, 5b	4.23
8.9.4.2.b	5a	4.23
8.9.4.2.c	5b	4.23
8.9.5.a	3	4.21
8.9.5.b	3	4.21
8.9.5.c	3	4.21
8.10.1 (*)		
8.10.2	3a	4.16
8.10.3	6b	4.25
8.10.4	3b	4.17

Clause	Covered by Error/Corner Cases	Covered by Coverpoints
8.10.5	6c	4.26
8.10.6	6d	4.27
8.10.7	6e	4.28
8.10.8 (*)		
8.11.1.a	1a	4.6
8.11.2.a	3a	4.16
8.11.3.a	1a	4.8
8.11.3.b	1a, 2a, 2c	4.12
8.12.1 (*)		
8.12.2.a (*)		
8.12.2.b (*)		
8.12.2.c (*)		
8.12.2.d (*)		
8.12.2.e	7	4.54
8.12.2.f	7	4.44
8.12.2.g (*)		
8.12.2.h (*)		
8.12.2.i (*)		
8.12.2.j (*)		
8.12.2.k (*)		
8.12.2.l (*)		
8.12.2.m (*)		
8.12.2.n (*)		
8.12.2.o (*)		
8.12.2.p.1 (*)		
8.12.2.p.2 (*)		
9.1 (*)		
9.2.1 (*)		
9.2.2.a (*)		
9.2.2.b (*)		
9.2.2.c (*)		
9.2.3.a (*)		
9.2.3.b (*)		
9.2.3.c (*)		
9.2.4.a (*)		
9.2.4.b (*)		
9.2.4.c (*)		
9.2.4.d (*)		
9.3.a (*)		
11.4	6f	4.29