1.3 Acronyms and abbreviations

Acronym/Abbreviation	***Description***
ASIC	Application Specific Integrated Circuit
CODEC	COder-DECoder.
COLE	**CO**COS (Computer Core Support) I/O + **LE**ON2-FT processor
EDAC	Error Detection And Correction
ESA	European Space Agency
FCT	Flow Control Token
FDIR	Failure Detection, Isolation and Recovery.
FIFO	First In First Out
IMA	Integrated Modular Avionics
ISR	Interrupt Status Register
JAXA	Japan Aerospace Exploration Agency
NASA	National Aeronautics and Space Administration
NCHAR	Normal Character
OBC	On-Board Computer
RKA	Russian Federal Space Agency
RTC	Remote Terminal Controller
PnP	Plug and Play
SpW	SpaceWire
SUAI	St Petersburg State University of Aerospace Instrumentation
TAPI	Telephony Application Programming Interface
TMR	Triple Modular Redundancy
TSP	Time/Space Partitioning

2 User Requirements for Interrupts Distribution support over SpW

This section gives a brief introduction to the solution proposed for Interrupts Distribution by SUAI and contains the requirements for Interrupts Distribution which 4Link collected by the Space Industry.

2.1 *Introduction to the SUAI proposal for Distributed Interrupts over SpW*

SpaceWire (SpW) 1.0 is a networking technology for on-board communications in spacecraft that are used for the interconnection of mass-memory, OBC, telemetry and etc. SpaceWire technology was proposed by ESA and is widely adopted in many ESA, NASA, JAXA and RKA missions.

SpaceWire standardizes full duplex links capable of transferring data at high speeds and employs flow control in order to reduce the memory requirements in the intermediate Switches and end node controllers. However, the feature of propagating critical events, for example alarms, is not provided by the SpaceWire 1.0 standard. In case a SpW node needs to notify another node for the occurrence of a critical event, a SpW packet is transmitted to the destination node. This packet can be delivered to the destination node with minimum latency only if the links between the packet source and the destination are not blocked. In case the links are blocked the SpW packet will reach the destination node with significant delay, which may be unacceptable for hard real time applications. It is therefore necessary to devise a mechanism for end-to-end distribution of critical information, which is not affected by the SpW flow control mechanism.

The University of St Petersburg (SUAI) has proposed a solution to overcome the problem of blocked links. The main concept of this proposal is to use unassigned time-code characters to distribute interrupts and interrupt acknowledgements through the SpW network. These codes are already supported in the current SpW standard but are reserved for future use.

2.1.1 *Definition of interrupts*

The characters that are broadcasted in a SpW network and have highest priority (can pre-empt NCHARs) are shown in Figure 1 and are thereafter called Signalling Codes. The information carried by Signalling Codes is separated in two fields. The T_6-T_7 bits and the T_0-T_5 bits. According to SpW 1.0 standard the only allowed combination regarding the T_6-T_7 bits is the "00" which defines a Time-Code. The rest three combinations are reserved for future use and SUAI proposal is based on using one of the unassigned combinations for the implementation of the proposed Interrupts mechanism.

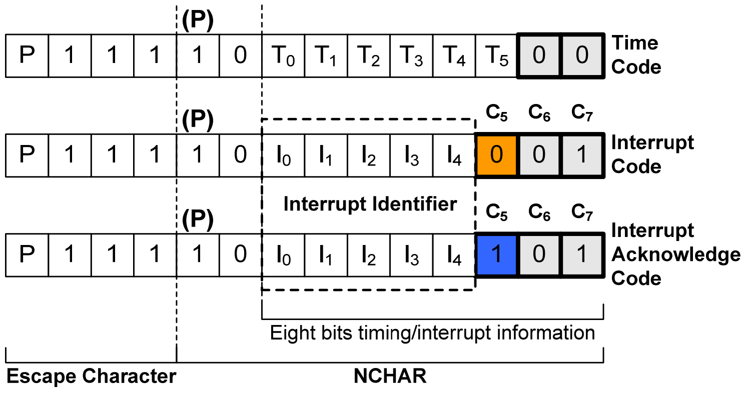

(P)
P 1 1 1 1 0 T0 T1 T2 T3 T4 T5 0 0
Time Code
(P)
C5 C6 C7
P 1 1 1 1 0 I0 I1 I2 I3 I4 0 0 1
Interrupt Code
Interrupt Identifier
(P)
C5 C6 C7
P 1 1 1 1 0 I0 I1 I2 I3 I4 1 0 1
Interrupt Acknowledge Code
Eight bits timing/interrupt information
Escape Character
NCHAR

Figure 1: Time-codes and interrupts characters

According to the SUAI proposal the "01" combination of bits T_6 and T_7 is used for the interrupts mechanism. The proposal defines two Signalling Codes for the interrupts mechanism, the Interrupt Codes and the Interrupt Acknowledge Codes. Bit C_5 is used to distinguish between the Interrupt Code (C_5 set to "0") and the Interrupt Acknowledge Code (C_5 set to "1"). The remaining five bits (I_0-I_4) of the respective NCHAR hold the interrupt identifier. Thus, there is a possibility to define 32 interrupt codes and 32 interrupt acknowledgement codes.

An **interrupt code** is defined as the signal that represents a request to handle an event of high priority. On the other hand, **interrupt acknowledge code** is defined as the signal that acknowledges the interrupt code acceptance by an interrupt handler. The interrupt acknowledge code represents a confirmation that the respective interrupt code has been accepted for processing by the interrupt handler.

For the propagation of Interrupt Codes and Interrupt Acknowledge Codes, SUAI proposes to use the same propagation mechanism as the one used for Time-Codes in order to ensure minimal propagation latency, propagation through redundant links (useful in case the primary link has failed) and propagation through blocked links.

The distribution of the interrupt code is explained with the help of Figure 2.

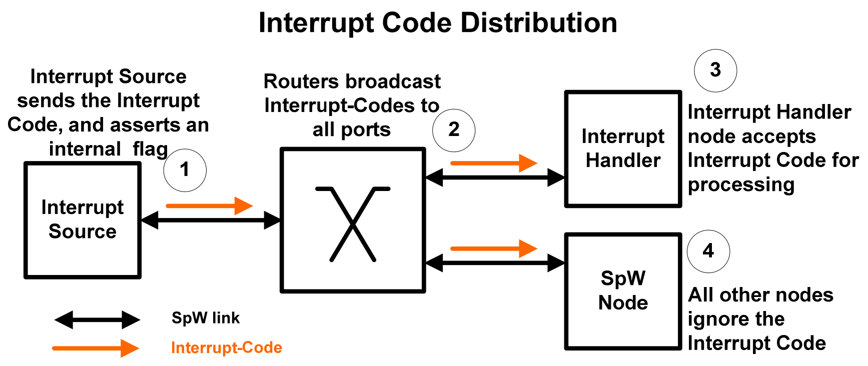

Interrupt Code Distribution
Interrupt Source sends the Interrupt Code, and asserts an internal flag
1
Interrupt Source
Routers broadcast Interrupt-Codes to all ports
2
3
Interrupt Handler
Interrupt Handler node accepts Interrupt Code for processing
4
SpW Node
All other nodes ignore the Interrupt Code
SpW link
Interrupt-Code

Figure 2: Interrupt code distribution

1. A **node** called the **Interrupt Source** transmits an Interrupt Code with a specific Interrupt identifier to signal a critical event.
2. Upon the reception of the Interrupt Code by an intermediate **Switch**, it is broadcasted to all links except for the link from which the interrupt code is received.
3. Another **node**, called **Interrupt Handler**, accepts the interrupt code and handles the request identified by the respective interrupt identifier.
4. All other SpW **nodes** ignore the Interrupt Code.

The Interrupt Handler processes the request received by the Interrupt Code and responds with an Interrupt Acknowledge Code to indicate that the request has been accepted for processing. The distribution over a SpW network of the interrupt acknowledge code is similar to the distribution of an Interrupt Code and is explained with the help of Figure 3.

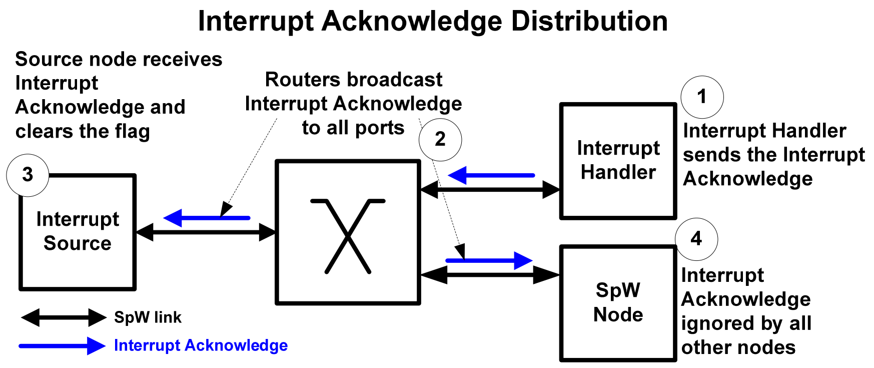

Interrupt Acknowledge Distribution
Source node receives Interrupt Acknowledge and clears the flag
3
Interrupt Source
Routers broadcast Interrupt Acknowledge to all ports
2
1
Interrupt Handler
Interrupt Handler sends the Interrupt Acknowledge
4
SpW Node
Interrupt Acknowledge ignored by all other nodes
SpW link
Interrupt Acknowledge

Figure 3: Interrupt acknowledge code distribution

1. The Interrupt Acknowledge Code is transmitted by the Interrupt Handler and shall carry the same Interrupt Identifier as the one of the received Interrupt Code.
2. Upon the reception of the Interrupt Acknowledge Code by an intermediate **Switch**, it is broadcasted to all links except for the link from which the interrupt code is received.
3. The Interrupt Source receives the Interrupt Acknowledge Code and is therefore informed that its request for servicing the critical event has reached the appropriate handler.
4. All other nodes of the SpW network ignore the Interrupt Acknowledge Code.

2.1.2 *Timeouts and the Interrupt status register*

SpW networks may have circular connections (loops) and it is therefore possible that an Interrupt Code (or Interrupt Acknowledge Code) may be broadcasted indefinitely in the network. To this respect SUAI also proposes the introduction of a register called **Interrupt Status Register** (ISR) which is used by the Interrupts mechanism to:

- store information about the interrupt codes that are pending for acknowledgement
- initiate a time-out for each interrupt code broadcasted in the network.
- prevent repeated transmission of the same interrupt code or the same interrupt acknowledge code in circular networks.

In addition, SUAI proposal specifies that:

- each link controller of a SpW node[1] shall contain one ISR
- each switch shall contain one 32-bit ISR
- each ISR bit corresponds to one of 32 available interrupt identifiers
- an ISR bit is set to "1" upon the transmission or the reception of an interrupt code containing the corresponding interrupt source identifier
- an ISR bit is cleared to "0" upon the transmission or the reception of an interrupt acknowledge code containing the corresponding interrupt source identifier

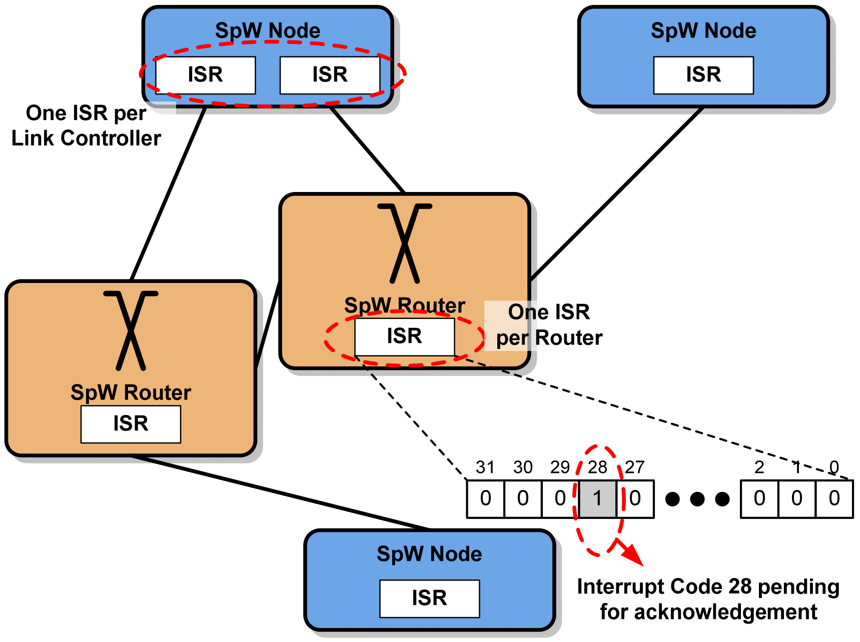


Figure 4: The Interrupt status register

2.1.2.1 Functionality of an Interrupt Source

Figure 5 shows the ISR functionality of an Interrupt Source node.

1. Initially at the Interrupt Source node, the host issues a request for transmission of an interrupt code and passes the requested Interrupt Identifier that shall be propagated through the SpW network.

[1] This means that there is a different implementation between Nodes and Switches. A node with N redundant ports shall have N ISRs, whereas a Switch with N ports shall have one ISR.

2. Upon the reception of the request, and if the Link is in the RUN state, the link controller of the interrupt source node checks if the ISR bit corresponding to the Interrupt Identifier is ‘0’ and if it is ‘0’:
    a. it asserts the ISR bit
    b. it starts a timeout timer waiting for the Interrupt Acknowledge Code
    c. it transmits the Interrupt Code
3. If upon the reception of the request the respective ISR bit is found to be asserted already, the request is just ignored, since the ISR indicated that another Interrupt Code with the same Interrupt Identifier has been sent and is pending for acknowledgement.
4. The request is also ignored in case the Link is not in the RUN state, when the request is received.

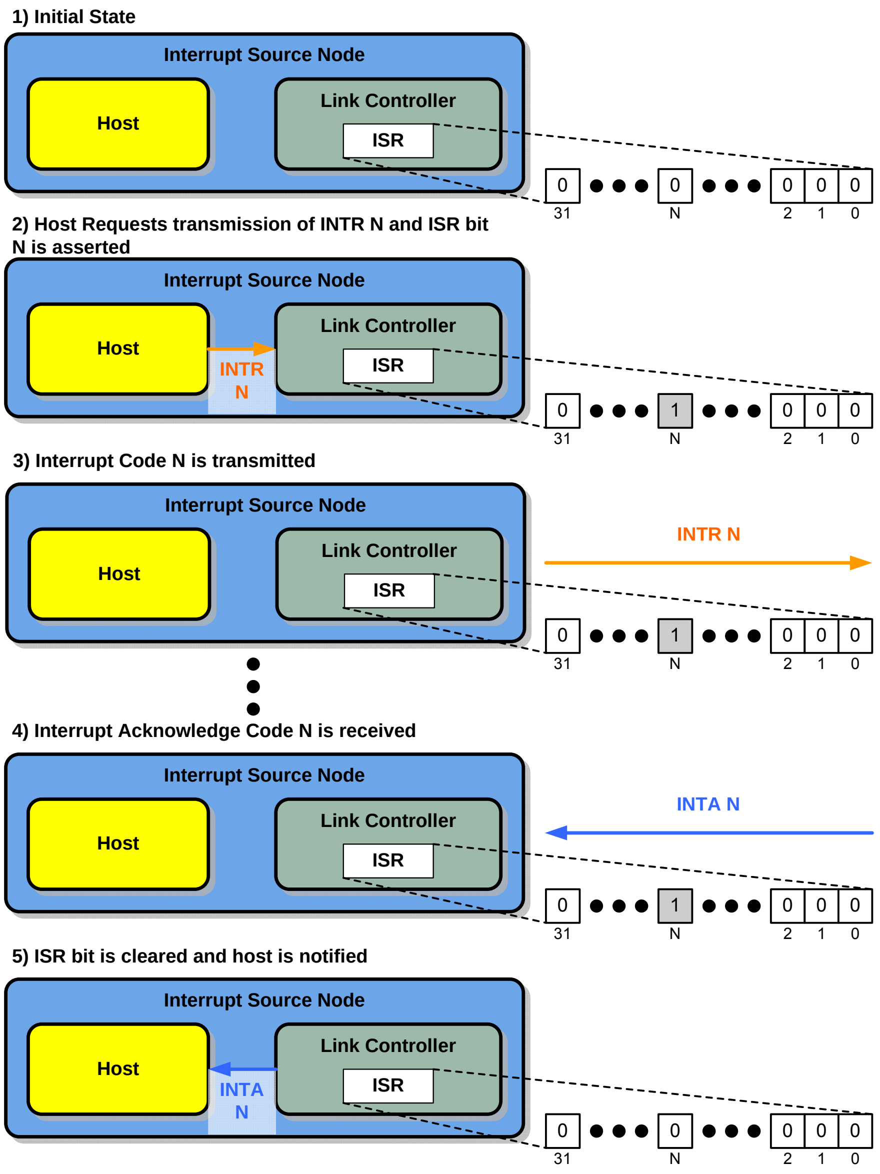


Figure 5: Interrupt Source node functionality

Upon the reception of an Interrupt Acknowledge Code the Link Controller checks if the ISR bit corresponding to the received Interrupt Identifier is set and if it is set

1. it clears the ISR bit.
2. it cancels the respective time-out timer.

3. it signals to the host that an Interrupt Acknowledge Code has been received and passes its Interrupt Identifier

If an Interrupt Acknowledge Code is not received in time and the time-out timer expires, a signal is issued to the host along with an Identifier which corresponds to the expired ISR bit (and therefore Interrupt Code) and an Interrupt Acknowledge Code with the same Interrupt Identifier is transmitted to clear all ISRs in the network

2.1.2.2 Functionality of an Interrupt Handler

At the Interrupt Handler side upon the reception of an Interrupt Code:

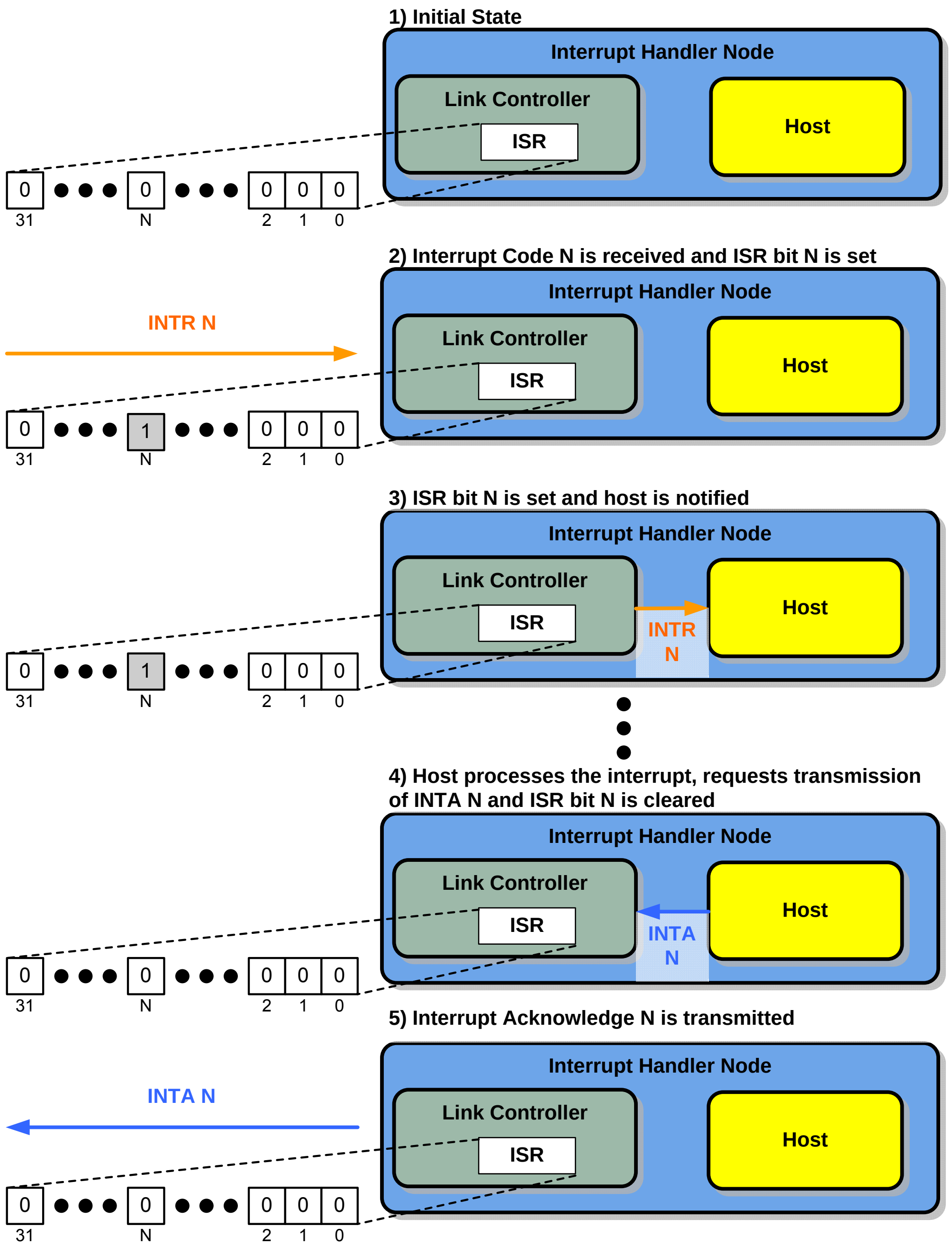


Figure 6: Interrupt Handler node functionality

1. the link controller of the interrupt destination node checks if the ISR bit corresponding to the Interrupt Identifier is '0' and if it is '0':

   a. it asserts the ISR bit

    b. it starts a time-out timer
    c. it notifies the Host that an Interrupt Code has been received passing its identifier

2. if the ISR bit is already asserted the Interrupt Code is just ignored
3. after a period of time the host indicates that an Interrupt Acknowledge Code shall be transmitted and the link controller checks if the respective ISR bit is '1' and if it is:
    a. it clears the ISR bit
    b. it cancels the time-out timer
    c. it transmits the Interrupt Acknowledge Code
4. if the ISR bit is cleared upon the host's request for Interrupt Acknowledge Code transmission, the host request is ignored.

2.1.2.3 Functionality of target nodes

The functionality of nodes that are not an Interrupt Source nor Interrupt Handlers (target nodes) is a combination of the two descriptions provided above and is graphically shown in Figure 7.

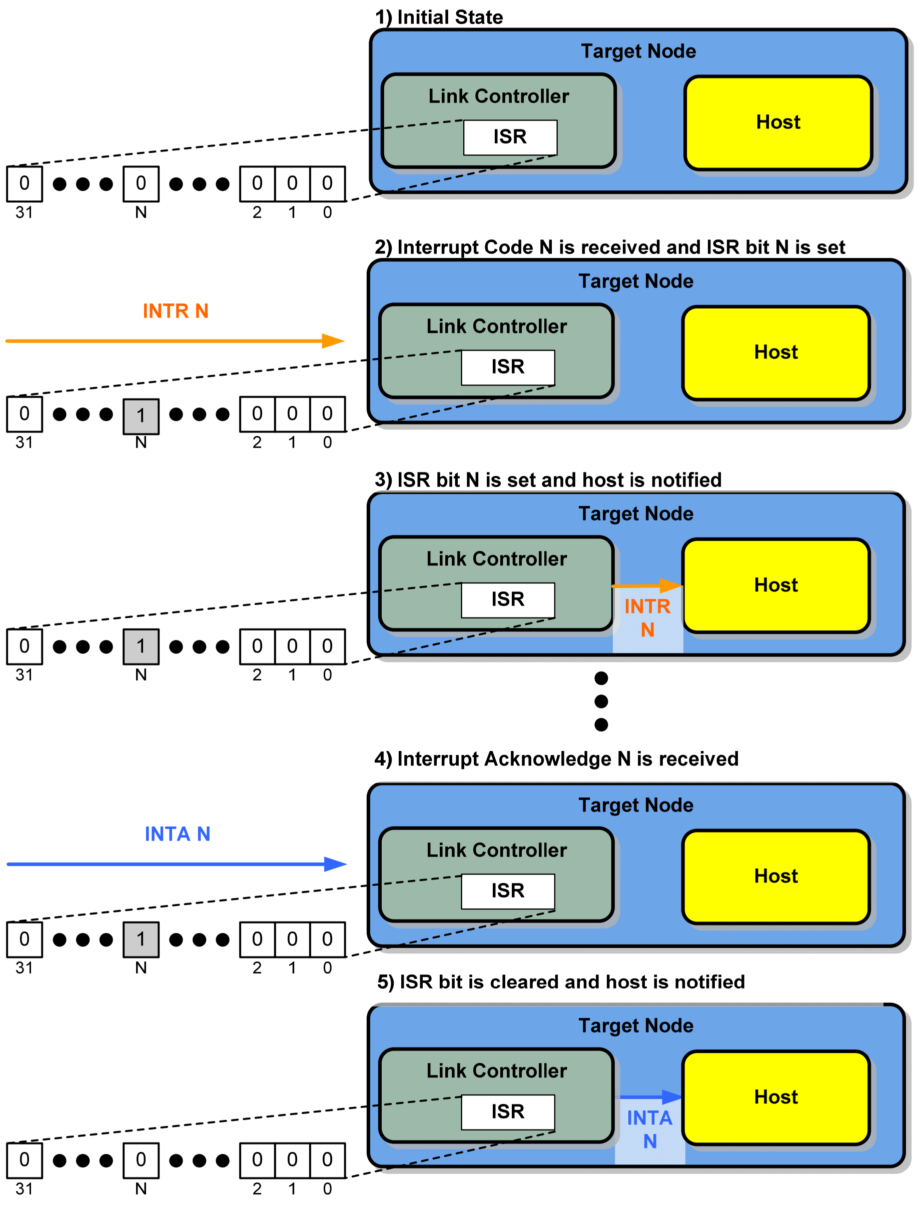

1) Initial State
Target Node
Link Controller
ISR
Host
0 ● ● ● 0 ● ● ● 0 0 0
31 N 2 1 0
2) Interrupt Code N is received and ISR bit N is set
INTR N
0 ● ● ● 1 ● ● ● 0 0 0
3) ISR bit N is set and host is notified
INTR N
4) Interrupt Acknowledge N is received
INTA N
5) ISR bit is cleared and host is notified
INTA N

Figure 7: Non-Interrupt Handler node functionality

1. Upon the reception of an Interrupt Code the link controller of the interrupt target node checks if the ISR bit corresponding to the Interrupt Identifier is ‘0’ and if it is ‘0’:
    a. it asserts the ISR bit.
    b. it starts a timeout timer waiting for the Interrupt Acknowledge Code.
    c. it notifies the host that an Interrupt Code has been received passing its Interrupt Identifier.
2. Since the node is not an Interrupt Handler the host may take action related to the Interrupt Code but will never request for the transmission of an Interrupt Acknowledge Code. Therefore either an Interrupt Acknowledge Code, transmitted by a different node, will be received, or the time-out timer will expire, and in both cases the ISR bit will be cleared.
3. If an Interrupt Acknowledge Code is received the link controller checks if the respective ISR bit is ‘1’ and if it is:
    a. It clears the ISR bit.
    b. It cancels the time-out timer.
4. If the time-out timer expires, the ISR bit is cleared and the host is notified.

2.1.2.4 Functionality of SpW Switches

The functionality of a switch is explained with the help of Figure 8.

1. Upon the reception of an Interrupt Code the switch checks if the ISR bit corresponding to the Interrupt Identifier is ‘0’ and if it is not, the switch ignores the Interrupt Code.
2. Upon the reception of an Interrupt Code the switch checks if the ISR bit corresponding to the Interrupt Identifier is ‘0’ and if it is ‘0’:
    a. it asserts the ISR bit.
    b. it starts a timeout timer waiting for the Interrupt Acknowledge Code.
    c. It broadcasts the Interrupt Code to all its ports, except for the port on which the Interrupt Code was received.

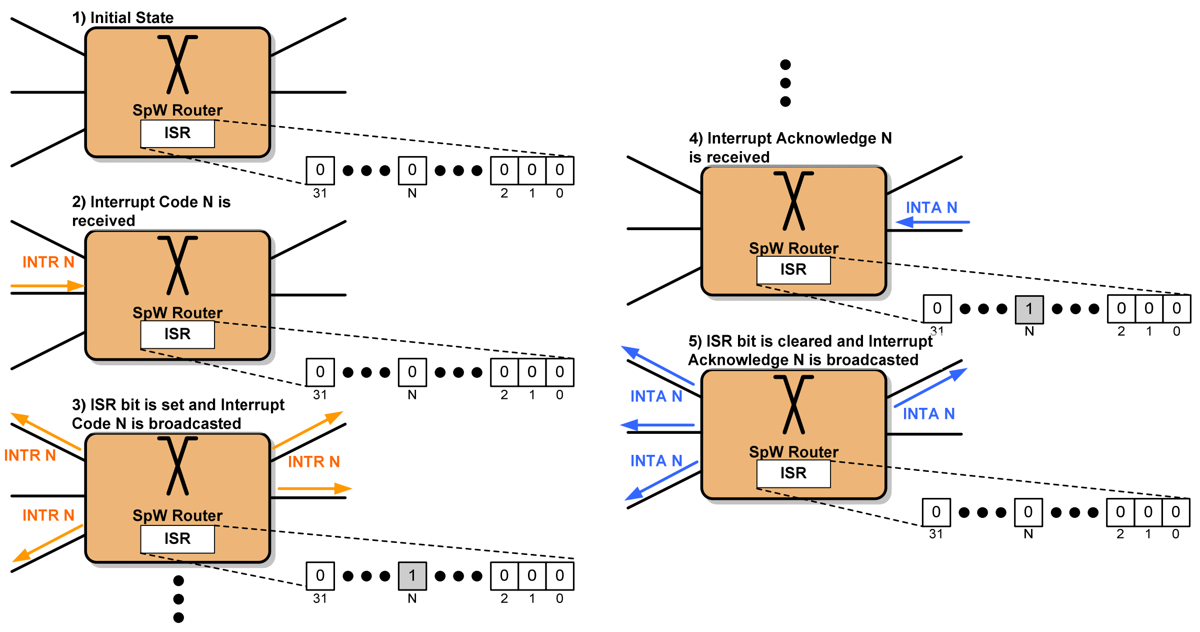

1) Initial State
SpW Router
ISR
2) Interrupt Code N is received
INTR N
3) ISR bit is set and Interrupt Code N is broadcasted
4) Interrupt Acknowledge N is received
INTA N
5) ISR bit is cleared and Interrupt Acknowledge N is broadcast
0
1
31
N
2
1
0

Figure 8: Switch functionality

3. If an Interrupt Acknowledge Code is not received in time and the a time-out timer expires, the respective ISR bit is cleared in order to allow subsequent broadcasting of the same Interrupt Code
4. If an Interrupt Acknowledge Code is received the Switch checks if the ISR bit corresponding to the Interrupt Identifier is ‘1’ and if it is not, the Interrupt Acknowledge Code is ignored.

5. If an Interrupt Acknowledge Code is received the Switch checks if the ISR bit corresponding to the Interrupt Identifier is ‘1’ and if it is ‘1’:
    a. The Switch clears the ISR bit
    b. It cancels the respective time-out timer
    c. It broadcasts the Interrupt Acknowledge Code to all its ports, except for the port on which the Interrupt Acknowledge Code was received.

2.2 User Requirements Collection Methodology

Initial requirements were summarized in the proposal. These were extracted and sent to the State University of Aerospace Instrumentation in St Petersburg (SUAI), who proposed the Distributed Interrupts mechanism, and to companies who use SpaceWire and have contributed to activities of the SpaceWire Working Group.

Response was not, unfortunately, received from SUAI. Responses were received, however, from Honeywell, NEC, RUAG and Thales Alenia Space.

Astrium were not included in the gathering of requirements phase because Astrium were to review the collected requirements, and this document includes additional requirements and comments from Astrium, based on the inputs from the other users.

The requirements resulted in the need for a number of additional Definitions, such as Interoperability, Reserved, and Options. Initial suggested definitions are included in this document, but these need to be discussed and considered carefully within the project, and so the important aspect concerning the definitions is that they are required for unambiguous understanding of the revised standard.

Some of the Requirements are fundamental, others are intended to satisfy these fundamental requirements and are essentially implementation requirements. In tabulating the Requirements, this distinction is made explicit.

2.3 Collected User Requirements and Analysis (4Links)

2.3.1 Fundamentals: Form of Standard, Interoperability and Definitions

2.3.1.1 Form of SpaceWire 1.1 additions and changes to the SpaceWire standard

Source	**Astrium**
Requirement	A compliance test set should be defined for ECSS which allows to assess compliance of a device. The same for SpaceWire 1.1.
Analysis	
The current ECSS-E-ST-50-12C standard includes several instructions of how something should be done, but without compliance criteria that can be measured. Correcting these is outside the scope of the current project, but all changes and additions to the standard for SpaceWire 1.1 should be written such that compliance criteria can be measured.	

2.3.1.2 Interoperability

Source	**SUAI, summarized in the proposal**
Requirement	SpaceWire 1.1 implementations shall be interoperable with implementations that comply with ECSS-E ST-50-12C
Analysis	
(4Links) This requirement is universal across all the SpaceWire evolutions. It does depend, however on the definition of interoperability, and there is no such definition in ECSS-E-ST-50-12C.	

2.3.1.2.1 Offered Definition of Interoperability

Source	**4Links**
Offered Definition	A device built to SpaceWire 1.1 1. may generate traffic that is ignored by a device built to ECSS-E-ST-50-12C; 2. may generate (as part of an initialization or recovery procedure) traffic that causes a device built to ECSS-E-ST-50-12C to disconnect, but shall not, except in such an initialization or recovery procedure, generate such traffic. 3. shall not generate traffic that a device built to ECSS-E-ST-50-12C will interpret as valid traffic but with different semantics from SpaceWire 1.1.
Analysis	
(4Links) The offered definition is specific to SpaceWire Evolutions and would be better if made totally generic to SpaceWire standards. The wording may not be adequately precise and needs to be considered carefully. (Astrium) “Totally agree” with this reservation. Also comment: (Astrium) The compliance definition is not clear since the implementation will have some limits. E.g Infinite packet. (4Links) Agree entirely with the lack of clarity of the suggested definition. The Infinite packet issue is considered in a Discussion item below. (Astrium) A compliance test set should be defined for ECSS which allows to assess compliance of a device. The same for SpaceWire 1.1. (4Links) This comment generates the first Requirement, 2.3.1.1: Form of SpaceWire 1.1 additions and changes to the SpaceWire standard (4Links) If there is a general ECSS definition of Interoperability which covers differences 1. within the Normative clauses of a standard? 2. between Normative and Optional clauses of a standard? 3. between Normative and Reserved features of a standard? 4. between different versions of a standard ? or if there is a definition in other standards, elsewhere than ECSS, that we could use, it might be preferable to inventing our own	

2.3.1.2.2 Interoperability with existing SpaceWire implementations

Source	**Honeywell, TAS, RUAG**
Implementation Requirement	SpaceWire 1.1 implementations should, in special cases, be interoperable with existing implementations of ECSS-E-ST-50-12C Interoperability with existing implementations of ECSS-E-ST-50-12C will be aided if SpaceWire 1.1 switches can disable propagation of all Time Codes, and can disable propagation of those “Time Codes” whose control flags are non-zero, from each output port independently.
Analysis	
(4Links) This enables SpaceWire 1.1 networks to include legacy nodes that treat non-zero control flags for Time Codes as if they were zero. The requirement comes from the following requests: (Honeywell) Aeroflex-Gaisler have raised some backward compatibility issues about time code character propagation in legacy switches, but (as far as I can determine) those issues are related to specific implementations rather than the SpaceWire standard (i.e. the implementations are more restrictive than the standard in propagating time code characters). (TAS) Ok the mechanism is not simple but seems to work (in addition it seems backward compatible with the ATMEL AT7910E which can be configured to discard incoming time codes whose control flags are not 00 (4Links) In principle, interoperability should not depend on implementation choices of existing devices. The issue is ambiguity of the interpretation of “Reserved” in ECSS-E-ST-50-12C, not helped by there being no definition. As a result, incompatible implementations can be built which all have valid claims to comply with the standard, and yet which need special features of third-party implementations (such as described by TAS) to make them interoperable. 4Links suggests defining “Reserved” in SpaceWire 1.1, and suggests that, in this specific case only, the implementation described by TAS in the ATMEL AT7910E is adopted. These are included as this Requirement and in the offered Definitions of Reserved and Option (Astrium) Does it means that SpaceWire 1.1 functionalities stay usable in a ECSS-E-ST50-12C compliant network ? or they could be degraded ? (4Links) Legacy devices not designed for SpaceWire 1.1 cannot, in general, use SpaceWire 1.1 capabilities. This requirement ensures that devices which fail to decode the Time Code control bits fully need not be corrupted by Distributed Interrupts nor by multiple Time Codes.	

2.3.1.2.3 Offered Definition of Reserved

Source	**4Links, derived from comments from Honeywell, TAS, RUAG**
Offered Definition	(4Links) The following definition of Reserved is offered To be compliant with this version of this standard, implementations: 1. Shall not generate codes that are defined as Reserved 2. Shall ignore and discard any received codes that are defined as Reserved (without signalling that an error has been received)
Analysis	
(4Links) As above, the issue raised is ambiguity of the interpretation of “Reserved” in ECSS-E-ST-50-12C, not helped by there being no definition. As a result, incompatible implementations can be built which all have valid claims to comply with the standard, and yet which need special features of third-party implementations (such as described by TAS) to make them interoperable.	

2.3.1.2.4 Placeholder Definition of Option

Source	**4Links, derived from comments from Honeywell, TAS, RUAG**
Placeholder Definition	"Option" TBD
Analysis	
(4Links) This is inserted as a placeholder, because it has similarities with Reserved but Options apply to different implementations within a version of a standard whereas Reserved is intended for different versions of a standard.	

2.3.1.2.5 Discussion of limitations on packet size

Source	**Astrium, Teletel, 4Links**
Discussion	Infinite packets and limitation of packet size
Analysis	
(4Links) 4Links may have misunderstood this point, but Astrium were concerned about links blocked by long or infinite packets. While these long or blocked packets can cause problems in current SpaceWire systems, they can be pre-empted both by Distributed Interrupts and by Virtual SpaceWire Networks. Various limitations were suggested, such as 64kByte, but it was pointed out that RMAP has a maximum packet length of 16MBytes, and concern was expressed that an 8-bit checksum may be inadequate for a packet so long. There are systems flying (SDO at least) using packets of 34MBytes. There could be good reason for using such large or even larger packets, and the project should consider carefully before retrospectively banning them. The project could define a maximum transfer unit (MTU) to alleviate this issue, but it is a system issue and such issues are the domain of the system designers rather than for the standard itself. As both Distributed Interrupts and Virtual SpaceWire Networks make it possible to interrupt the long or blocked packet, it should not be necessary to define a global MTU. If individual system designers or implementers choose to design a gatekeeper function that limits packet lengths to a certain length (which could be configured) then that could provide an additional check for system malfunction.	

2.3.2 Functional Requirements

2.3.2.1 Broadcast

Source	**SUAI, summarized in the proposal**
Requirement	Low-latency signals often need to be broadcast
Analysis	
(4Links) No one has commented on this requirement, and it is not clear to what extent broadcast is required.	

2.3.2.2 No deadlock from Broadcast

Source	**Astrium**
Requirement	There shall be no deadlock arising from the broadcast of Interrupts
Analysis	
(4Links) The Time Code mechanism allows circulating Time Codes because there is a sequence of valid Time Code values and repeated receipt of the same Time Code value is ignored. Distributed interrupts do not have such a mechanism, and so an alternative mechanism is required.	

2.3.2.2.1 Corollary of no deadlock from Broadcast: No Echoing of Interrupt

Source	**Astrium requirement for no deadlock resulting from broadcast**
Corollary Requirement	A port, either of a Node or of a switch, shall not echo a received Distributed Interrupt or Acknowledge
Analysis	
(4Links) There may be other mechanisms but this is simple at first sight.	

2.3.2.3 Priority over normal packet transfers to pre-empt blocked links

Source	**SUAI, summarized in the proposal**
Requirement	Signals need to have priority over normal packet transfers to pre-empt blocked links
Analysis	
No comments have been received from outside the project on this requirement (4Links) this allows urgent broadcast signals to travel over blocked links, but does not provide a control path for unblocking the blocked link for continuing normal traffic, which Virtual SpaceWire Networks do provide.	

2.3.2.4 No loss of Interrupts in switches

Source	**Astrium**
Requirement	When several interrupts arrive at a switch at the same time, none of them shall be discarded.
Analysis	
(4Links) This means that they need to be queued, but it does not demand that they are handled in any particular order.	

2.3.2.5 Low-layer of the protocol

Source	**SUAI, summarized in the proposal**
Requirement	Low-latency signals need to be at a low layer of the protocol
Analysis	
(4Links) The Distributed Interrupts mechanism is at a low level. No one has commented on this requirement.	

2.3.2.6 Number of Time-Code sets and number of Distributed Interrupts required

Source	**SUAI, following request from ESA, summarized in the proposal**
Requirement	Use only a single Reserved Time Code set for Distributed Interrupts and acknowledgement
Analysis	
(4Links) SUAI originally proposed to use two Time Code sets, one for 64 Interrupts and one for 64 Acknowledges. ESA, or possibly the SpW WG, requested that this was reduced to a single Time Code set with 32 Interrupts and 32 Acknowledges. The comments below question whether 32 is adequate, and suggest that a solution similar to Virtual SpaceWire Networks would better meet the requirements: (NEC) Leaving a spare for future extension is proposed. (TAS) The 5-bit interrupt code allows no more than 32 active interrupt source nodes in the same SpW network. This may result a limit in the near future (currently the SpW network of Bepi Colombo has 20 nodes which become 31 considering nominal and redundant SpW I/F) (ASTRIUM) Same remarks as TAS. The foreseen architecture includes more nodes than 31 with the redundancy. Sharing one interrupt between more than one node raises the problem of acknowledge. (Discussion: Astrium, Teletel, 4Links) There is a need for some form of priority interrupts on the network, so that such packets provide information along with the interrupt and are not limited to a fixed number of interrupts. This is similar to 4Links proposed Virtual SpaceWire Networks, and 4Links were asked if this technology could be included in the project. 4Links response was that requests for the technology have been received for other programmes and that their intention is to have the technology developed on a similar time scale to the project. It is, however, outside the current project scope and 4Links would currently prefer not to commit to delivering it within the project. If implemented, 4Links will change the signalling mechanism so that it does not use Time Codes (there are already too many demands on these codes), and so that the extra information carried for the Virtual Network features is invisible to any legacy SpaceWire device.	

2.3.2.7 Robustness in the presence of network failures

Source	**SUAI, following request from ESA, summarized in the proposal**
Requirement	Ensure that the Distributed Interrupt mechanism is robust in the presence of network failures
Analysis	
(4Links) This matter raised serious concern which, combined with the difficulty of defining minimum or maximum latency, may be difficult to address. (Honeywell) The basic concept seems reasonable and I believe has been captured by the requirements you identified. I think the requirement about assuring robustness in the presence of network errors is ambiguous (it apparently means more than best effort, but it is not clear how much more). Since the SpaceWire standard uses best effort delivery for all other traffic types, it isn't obvious that a more stringent definition is acceptable to the user community. I am concerned about the complexity described in RD12 to address robustness. a. I think the mechanisms introduce the potential for coherency issues. They essentially create a distributed Interrupt Status Register (ISR) with copies throughout the network. The goal is obviously to keep every ISR copy in the same state (ignoring delivery latency) at all times. Without doing a detailed analysis it is hard be sure, but my intuitive reaction is that the currently described mechanisms aren't sufficient. b. The interrupt-acknowledge character is clearly necessary in the context of the error recovery mechanisms described, but other methods for acknowledging interrupts are feasible (and might be preferable by some). Their existence (and use) obviously increase the number of error opportunities for any specific interrupt/interrupt-acknowledge event pair and, correspondingly, the opportunities for coherency errors between the distributed ISRs. (RUAG) Uncorrelated time-out timers: What to do when a switch receives an interrupt code and one of its links is in the process of restarting? When reading 8.13.1g it seems that the switch shall wait until the Run state is reached and that it then send the interrupt. This means that the timer in the other end of the links may be off by several microseconds with other timers in the system and this may then affect the operation when a second interrupt is generated. (Astrium) Usually the interrupt acknowledge is automatic and done at CPU level (cf SPARC processor) The acknowledge as it is described is done by the Interrupt Service Handler when the CPU take in account the message by reading or writing a specific register in the device. If the a packet cannot pass because of network congestion then it is non-sense to acknowledge using high-priority message A simpler mechanism to avoid echoing on the network and a time to repeat in the device are enough to insure robustness. The impact is that it is not compliant with old switch behaviour. (4Links) This clearly needs to be discussed within the project and with ESA.	

2.3.2.7.1 Time-out mechanism

Source	**e.g. ASTRIUM**
Requirement/Comment	e.g. A mechanism must prevent an interrupt to be held in the network if the Interrupt Acknowledge from the sink is lost or not transmitted. A source should be able to retry an interrupt message after some delay in case of missing acknowledge
Analysis	
(4Links) The SUAI implementation includes such timeouts, but the concern raised in the robustness requirement above is equally valid here. With the difficulty in calculating latency, extra margin may need to be allowed such that the minimum required timeout for one criterion is longer than the maximum required timeout for another criterion. There may be a small benefit here from the Virtual SpaceWire Network technology, where timeout values can be based on the traffic type, rather than needing a single timeout value to cover a wide range of different traffic types.	

2.3.2.8 Priority over normal packet transfers to pre-empt blocked links

Source	**SUAI, summarized in the proposal**
Requirement	Signals need to have priority over normal packet transfers to pre-empt blocked links
Analysis	
No comments have been received from outside the project on this requirement (4Links) this allows urgent broadcast signals to travel over blocked links, but does not provide a control path for unblocking the blocked link for continuing normal traffic, which Virtual SpaceWire Networks do provide.	

2.3.2.9 Compatibility with PnP protocol

Source	Astrium
Requirement	The proposed mechanism shall be compatible with a PnP protocol. I.E. the value of the interrupt used by a source link, and any timeout value, shall be configurable.
Analysis	
(4Links) It is necessary for the timeout values for the Distributed Interrupts to be configured. It is possibly not completely clear whether this requirement applies to nodes or to switches or to both. For example it would be expected that switches have timeouts, but do nodes need them? A SpaceWire1.1 node would be expected to generate the Interrupt and so would need to be configurable. It might be convenient if a SpaceWire 1.1 switch had an interrupt and interrupt acknowledge pin for some of its ports --- so that a legacy node might be able to generate Interrupts, but is this a requirement of the project?	

2.3.2.10 Allocate values of control flags to (multiple) Time Codes and to Distributed Interrupts etc.

Source	NEC, Honeywell
Requirement	The project shall allocate values of the two-bit control flags to Time Codes (including multiple Time Codes), to Interrupts, and to Reserved if appropriate
Analysis	
(NEC) Coexistence with multi-time code scheme is proposed with mode distinction (Honeywell) There is no mention in the requirements of the proposed generalization of time code characters into signalling code characters where the set of signalling code characters can be allocated as desired to communicate time events, interrupt events or any other event needed by the specific application. (4Links) The allocation of control flags for Time Codes and for Interrupts (and for any further Reserved code) needs to be defined. This was done, at least partially, in the SUAI documents, but needs to be confirmed	

2.3.3 *Performance Requirements*

2.3.3.1 Latency

Source	**Honeywell, RUAG, NEC**
Requirement	The delivery latency for Interrupt characters shall be within the following limits: Minimum: (TBD) Maximum: (TBD)
Analysis	
(General) The actual value of latency is not defined and is difficult to define. The following comments show a clear need for defined latency at the same time as showing the difficulty in defining it. (Honeywell) Aeroflex-Gaisler have identified interrupt character propagation issues that I consider critical (RD12 appears to address them indirectly, but the requirements do not). Specifically, since interrupt characters are likely to be transmitted from multiple sources within a SpaceWire network and are inherently asynchronous events, a switch is likely to occasionally receive multiple interrupt characters simultaneously and must provide some mechanism for serializing them to the output ports. Because Time Code characters are defined as having a single source and are expected to be spaced in time, the SpaceWire standard doesn't address this problem. The delivery latency for interrupt characters is clearly a function of the number (and bit rate) of SpaceWire links to be traversed, but is also affected by the other network traffic encountered. Requirements for the maximum latency and jitter (latency uncertainty) introduced by a switch may be difficult to define, but are necessary. (NEC) Synchronization with Time Code is proposed as a merit inherent in existing mechanism of SpaceWire specification. (4Links) And if used for synchronization, the latency needs to be determinate. (RUAG) Fast acknowledge: There is no minimum time requirement for how soon an acknowledge may be generated. Depending on the topology of the network you could imagine a situation where the acknowledge is made very quickly to the nearest switch and where that switch receives a delayed interrupt via some other path (e.g. due to a link restart) and thus that switch generates a second interrupt since its ISR bit is already reset by the acknowledge. (4Links) If Interrupts are lower priority than all other Time Codes, there can still be a queue of 32 interrupts. Assuming that a switch can transmit these all without gaps, the delay at 10Mbits/s is 1.4μs per interrupt, 44.8μs for all 32. It may be difficult to design for no gaps, and also difficult to design for absolute first-in-first-out. Allowing for margins on this may need a permissible delay exceeding 100μs and possibly 100μs per switch in a path. (4Links) Distributed Interrupts are not the only way to provide low latency, as 4Links' Virtual SpaceWire Networks also provide low latency.	

2.3.4 Other Categories

2.3.4.1 Use the word “Signalling to describe both Time Codes and Interrupts

Source	**Honeywell, NEC**
Requirement	Use the word “Signalling” to describe both Time Codes and Interrupts
Analysis	
(NEC) Polling function is mentioned in addition to interrupt distribution, and sideband does not seem as a requirement. So "sideband" and "interrupt distribution" might not be necessary in the title of 2.1.3. "Signalling" seems to be core requirement. (Honeywell) There is no mention in the requirements of the proposed generalization of time code characters into signalling code characters where the set of signalling code characters can be allocated as desired to communicate time events, interrupt events or any other event needed by the specific application.	

2.3.4.2 Changes to ECSS Standard

Source	**RUAG**
Suggestions for changes to the ECSS standard	The changes that must be made to the current SpaceWire standard are not identified except for: *Clause 7.3d: 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.* *Proposed to change to:* *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 (C6=0, C7=0) shall contain control flags that are distributed isochronously with the Time-Code.* The latter change is a bit unclear: how can control flags be distributed if the corresponding bits are all zero? Then there are several parts in chapter 8.12.2 in the SpaceWire standard that must be updated. The most obvious ones are: *Clause 8.12.2d: 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* *Clause 8.12.2j: When a link interface on a switch or node receives a Time-Code it shall check that it is one more (modulo 64) than its current time setting.* *Clause 8.12.2k. The switch or node shall then increment the switch’s time-count and emit a tick signal.* *NOTE This tick signal propagates to all the output ports of the switch so that they all emit the Time-Code. This Time-Code is the same value as that received by the switch since the switch time-counter has been incremented. If there is a circular connection then the switch receives a Time-Code with the same time value as the switch 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 switch.* *Clause 8.12.2n: 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.* *Several of these changes may be minimized by the redefinition that there are now four different Control Codes instead of only two in the current standard.*
Analysis	
4Links did not consider these to be Requirements, but they may be very useful to the project.	

3 Technical Solution for Interrupts Distribution Support over SpW

3.1 Problems and proposed solutions

The interrupts and interrupts acknowledgements technique is well defined in the SUAI proposal. However in order to conclude to a specification, certain actions, although they are intuitively clear, need further clarification and some modifications/additions are proposed herein in order to handle special cases.

3.1.1 Switch

In a SpW network that uses interrupts and interrupts acknowledgements to distribute critical events there is a significant probability for a switch (*ref:* Figure 8) of this network to receive many interrupts and interrupt acknowledgements codes simultaneously.

Thus storing devices (e.g. FIFO memory) shall be used, one for the interrupt codes and one for the interrupt acknowledgement codes, in order to store the codes and prevent overflow in the case of simultaneous arrivals of Interrupt Codes and/or Acknowledge Code over more than one ports.

The FIFOs depth is implementation dependent. Specifically, in case all the links of the switch operate at the same link speed then the maximum number of Signalling Code that can be simultaneously received is equal to the number of ports. There is a special theoretic case in which one of the ports may be very slow and all the other may be continuously be receiving Interrupt Codes (or Acknowledge Codes). For this theoretic case each FIFO shall be 32 entries deep in order to prevent overflow. In order to support handling of simultaneous pending Interrupt Codes (and Acknowledgements) the Link Controller interface should be extended and provide one more signal which controls the injection of Interrupt Codes, Acknowledge Codes and Time-Codes to the Link Controller[1].

3.1.2 Functionality in circular networks

In circular networks an Interrupt Code (and Acknowledge Code) arrives at the SpW ports of at least one switch more than once. Under certain conditions a problem may be present which is explained with the help of Figure 9.

In the case of a SpW network having a topology similar to the network depicted in Figure 9, interrupt routines may be executed more than once. This may happen for example in the following scenario:

1. An Interrupt Code is transmitted by the interrupt source and arrives at the switch closest to the interrupt source.
2. The switch propagates the Interrupt Code to both the next switch to the switch that is closest to the interrupt handler.
3. The interrupt handler executes the interrupt service routine and immediately sends back the respective Interrupt Acknowledge Code.
4. The Interrupt Acknowledge Code is propagated to the switch that is closest to the Interrupt Source and to the farthest switch in the network in which the interrupt code has not been yet received. Since an the Interrupt Code has not been received, the respective ISR bit is cleared and therefore the Interrupt Acknowledge Code, sent by the Handler, will be ignored.
5. At a later point in time, the Interrupt Code reaches the farthest switch of the network and then the interrupt code is propagated again to the interrupt handler.
6. **The interrupt service is executed for second time**.

A similar situation may occur in the case of the interrupt source has redundant links if Interrupt Source sends a second interrupt code during the propagation of the first interrupt acknowledge code. This duplicate execution of the same interrupt may be fatal in many cases of events that are distributed throughout a SpW network.

[1] This is not included in the next section. The reason for not including it is that during the SpW WG #16 it was agreed that the Time-Codes interface (which is the one used for the interrupts mechanism as well) shall not be standardized in the next revision of the SpW standard.

Two additions in the SUAI proposal will assist in avoiding cases as the one described above:

- Specify a minimum time after which the interrupt handler has to respond to an interrupt code. This time depends on the network diameter i.e. the network configuration, which shall be the double of the largest latency from an Interrupt Source to a Handler. This way the interrupt handler of the example of Figure 9 will not send the interrupt acknowledge code before the second interrupt code arrives from the farthest switch of the network.
- Specify a minimum time for the transmission of an Interrupt Code after the reception of an Interrupt Acknowledge Code with the same interrupt identifier. This will eliminate the possibility of simultaneous existence of an Interrupt Code and an Interrupt Acknowledge Code with the same interrupt code identifier in the network.

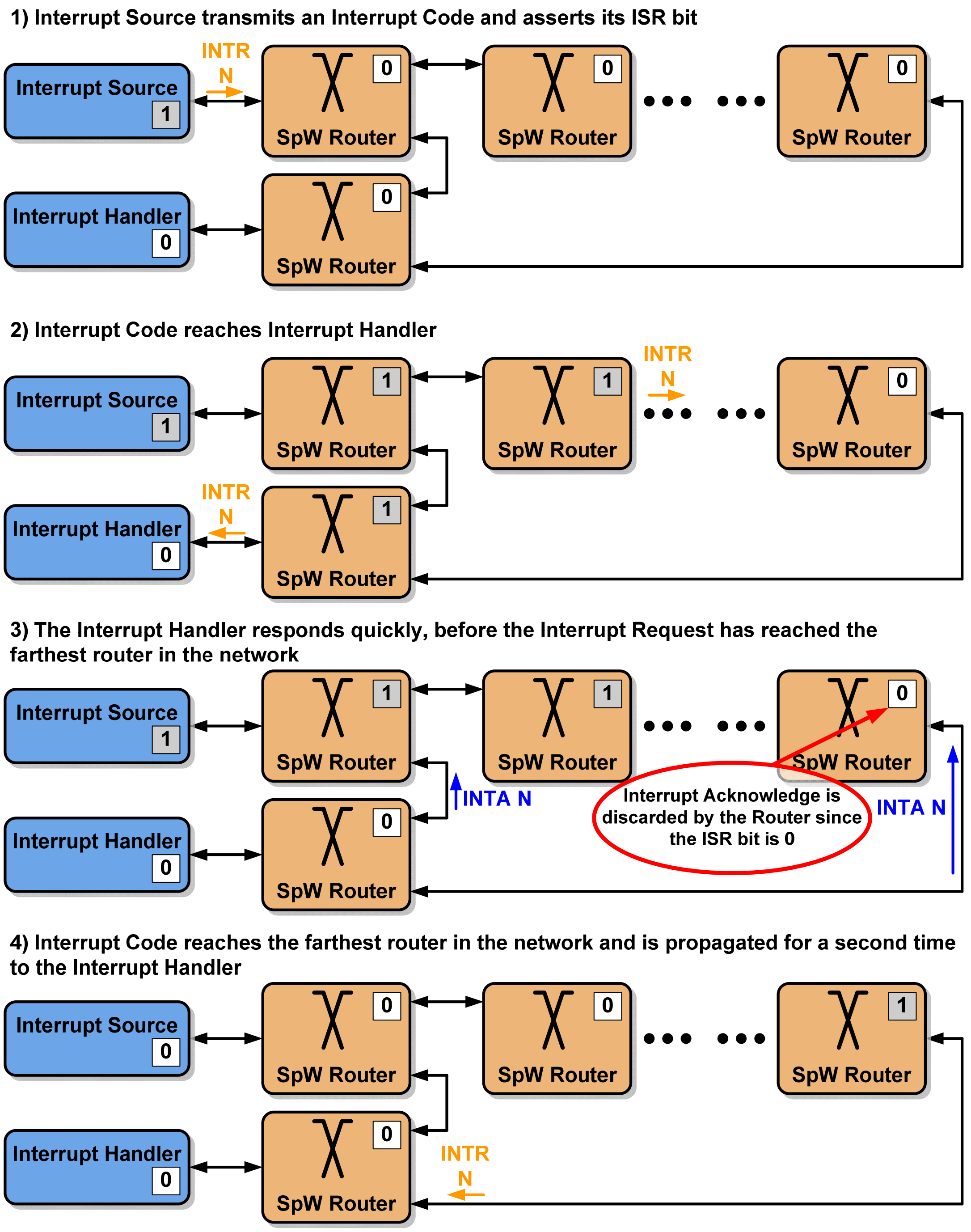

1) Interrupt Source transmits an Interrupt Code and asserts its ISR bit
Interrupt Source
Interrupt Handler
SpW Router
INTR N
2) Interrupt Code reaches Interrupt Handler
3) The Interrupt Handler responds quickly, before the Interrupt Request has reached the farthest router in the network
INTA N
Interrupt Acknowledge is discarded by the Router since the ISR bit is 0
4) Interrupt Code reaches the farthest router in the network and is propagated for a second time to the Interrupt Handler

Figure 9: Simultaneous existence of Interrupt Code and Acknowledgement in a circular network

3.1.3 Reset on disconnect

SUAI proposes (*ref:* [AD,1] sections 8.13.1 w and 8.13.3 g) that after a link disconnect-reconnect the ISR bits and the ISR timers shall be set to zero. This however may result in the scenario shown in Figure 10.

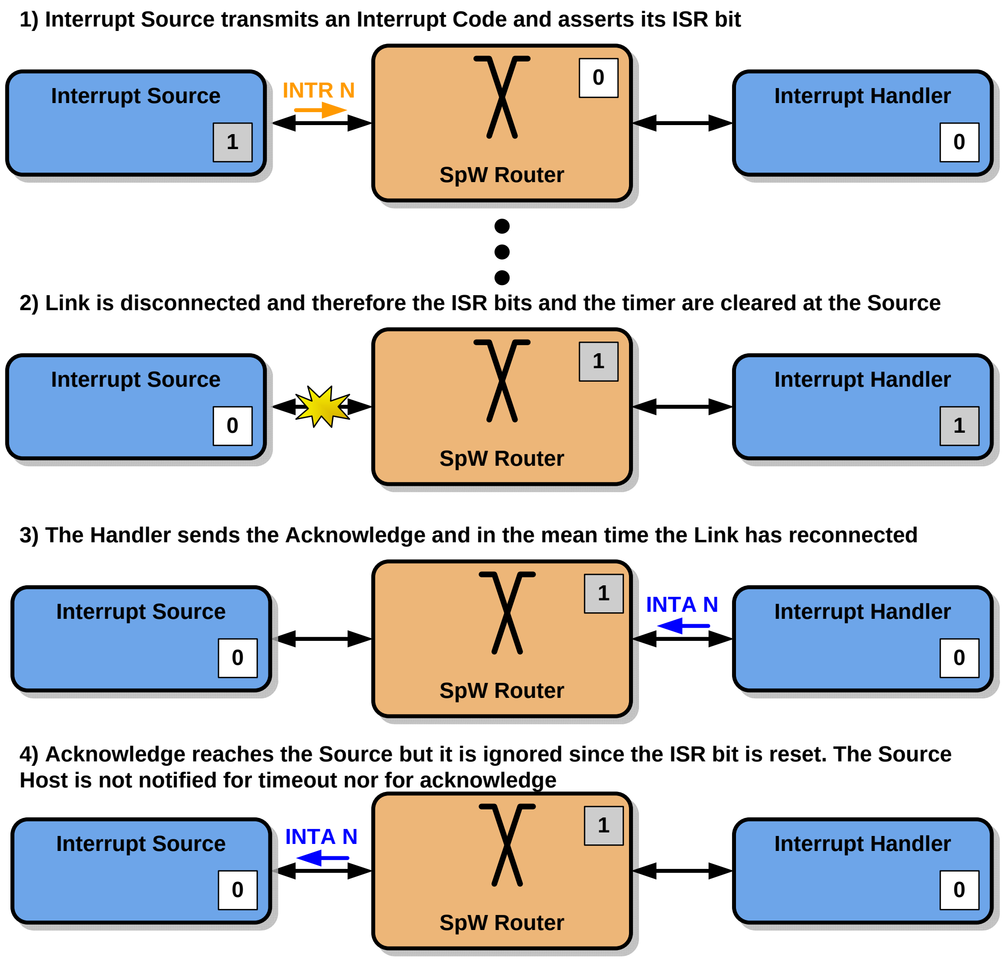

1) Interrupt Source transmits an Interrupt Code and asserts its ISR bit
Interrupt Source
1
INTR N
SpW Router
0
Interrupt Handler
0
2) Link is disconnected and therefore the ISR bits and the timer are cleared at the Source
Interrupt Source
0
SpW Router
1
Interrupt Handler
1
3) The Handler sends the Acknowledge and in the mean time the Link has reconnected
Interrupt Source
0
SpW Router
1
INTA N
Interrupt Handler
0
4) Acknowledge reaches the Source but it is ignored since the ISR bit is reset. The Source Host is not notified for timeout nor for acknowledge
Interrupt Source
0
INTA N
SpW Router
1
Interrupt Handler
0

Figure 10: Reset on disconnect

1. The interrupt source transmits an interrupt code, asserts the respective ISR bit and starts the time-out timer.
2. After the interrupt code reaches the interrupt handler the interrupt source link is disconnected. The ISR bit and respective timer of the link controller of the interrupt source are set to zero.
3. The link is reconnected and has now all its ISR bits reset and time-out timers cancelled.
4. The interrupt handler sends the interrupt acknowledge.
5. The interrupt acknowledge reaches the interrupt source. Since the respective ISR bit is "0" the interrupt acknowledge is ignored by the interrupt source.

This results in the host of the interrupt source not being notified either for the acknowledge indication or for the corresponding acknowledge timeout. In conclusion, an infinite timeout situation is occurred at the host of the interrupt source, or, in order to avoid that, a time-out shall exist in SW. Although the problem can be mitigated with the later approach, this does not result in optimal implementation, since there is a time-out block already in the Link Controller which should cover this case as well.

The same situation applies in the case of the interrupt acknowledge is sent while the interrupt source link is still disconnected. In this case the Interrupt Acknowledge is lost indeed and the Source Controller never issues a time-out to its host.

The proposed modification to the SUAI proposal is to specify that the ISR bits and timers shall not be affected by a link disconnect.

3.1.4 ISR operations

A theoretic scenario possible with the proposed SUAI specification is the one shown in Figure 11.

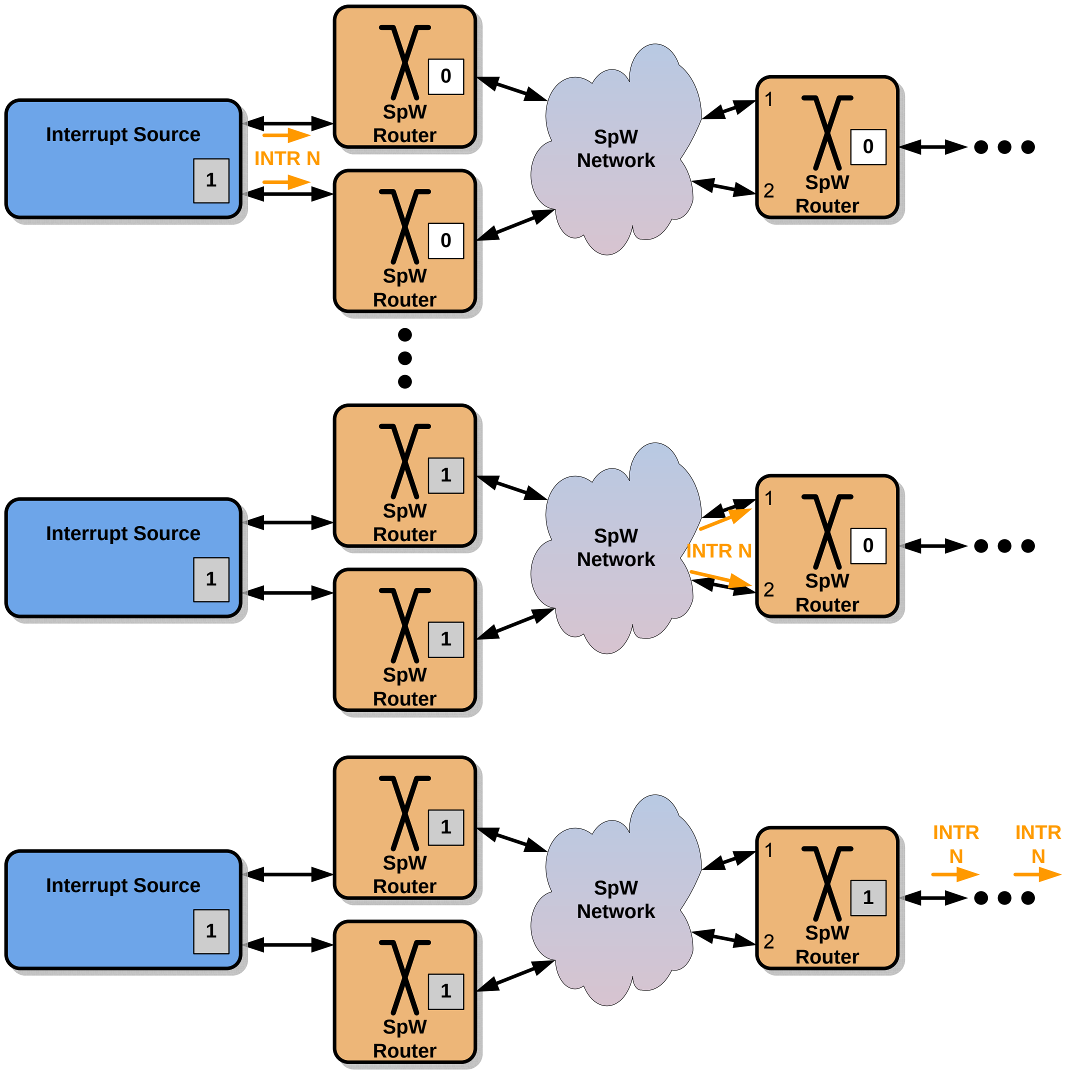


Figure 11: Two equal interrupt codes are propagated in the same node

This issue is depicted in the above figure and can be described using the following points:

1. An interrupt source node with redundant links transmits an interrupt code.
2. The interrupt code arrives using two different paths in the same switch over ports 1 and 2.
3. The Port 1 logic checks the ISR and find it ‘0’.
4. The Port 2 logic checks the ISR and find it ‘0’ since Port 1 logic has not asserted it yet.
5. Port 1 logic places the Interrupt Code in the respective FIFO for transmission, asserts the ISR bit and starts the respective time-out timer.
6. Port 2 logic does the same
7. The same Interrupt Code is broadcasted twice.

Although implied in the SUAI proposal it shall be explicitly states that all operations on ISRs and time-out timers shall be performed with ATOMIC operations.

3.1.5 *Babbling idiot*

The scenario shown in Figure 12 reveals the inefficiency of the current SUAI proposal upon the presence of an erroneous node in the network.

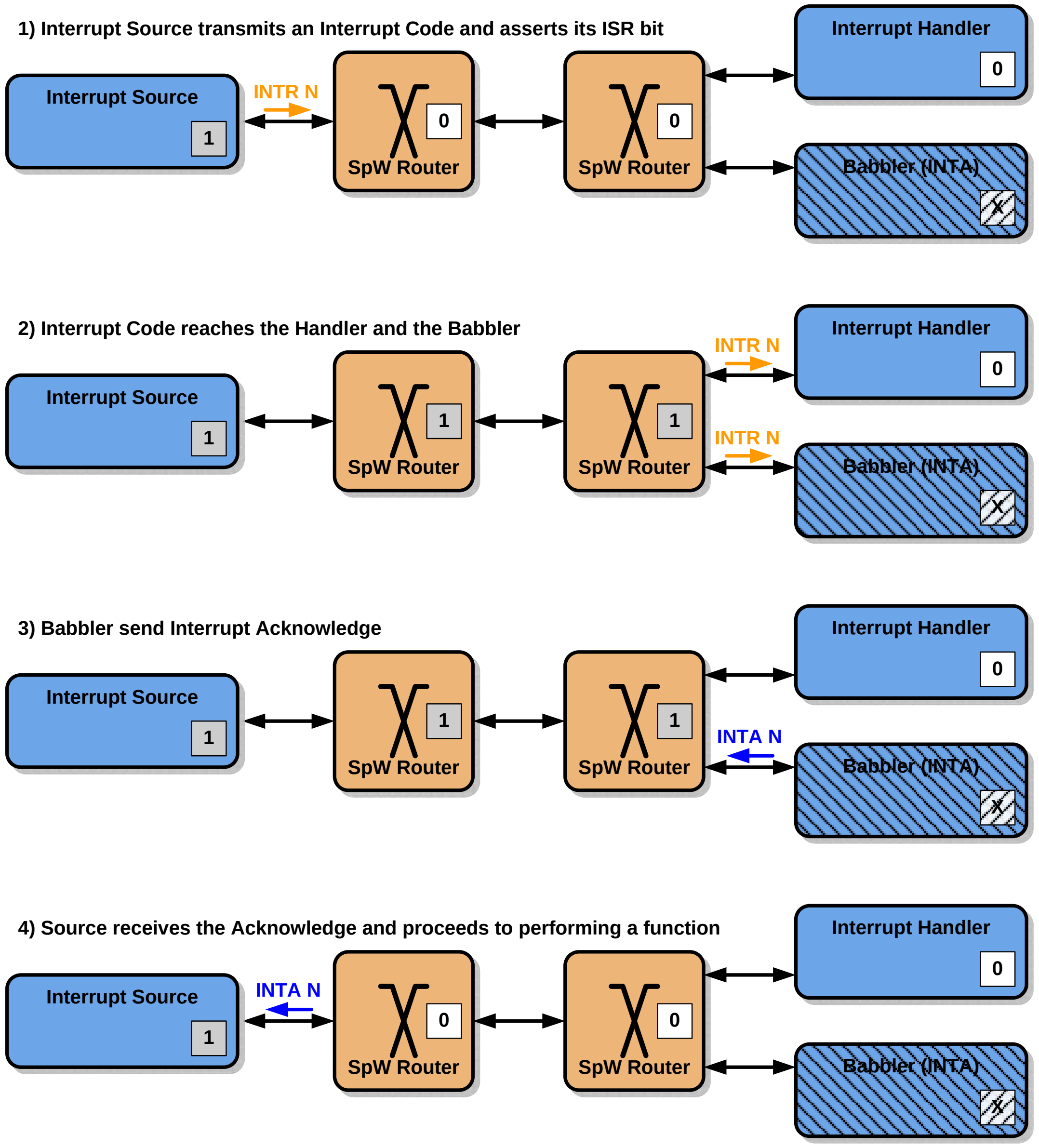


Figure 12: Babbling idiot

In this scenario we assume that there is a faulty node (babbler) in the SpW network of the above figure that responds to all or some of the interrupt codes.

1. The interrupt source sends an interrupt code and this code is received from the interrupt handler and the babbler.
2. The Interrupt receives the Interrupt Code but does not acknowledge it since it shall not be acknowledged
3. The babbler erroneously responds with an interrupt acknowledge.
4. The interrupt acknowledge reaches the interrupt source and confirms the execution of further critical actions.
5. The Interrupt Source is erroneously granted to perform further actions, a situation which may be fatal in certain cases.

A similar situation may occur in case the babbler sends interrupt codes.

A solution of the above situation can be provided by the edge switches. Specifically, the interrupts distribution will be protected by assigning allowed/expected interrupt and interrupt acknowledge codes to each port of each edge switch. Additionally, error information may be provided in the switches of the network for FDIR i.e. remote node deactivation using PnP. This way the erroneous code will be blocked and its propagation will be avoided.

3.2 Conclusions and alternative solutions for Interrupts Distribution

3.2.1 Summary of proposed modifications

The so far described modifications and additions to the SUAI proposal can be submitted in the following list. If no reference follows then this is a generic modification/addition:

- If the link controller of the interrupt source is not in the RUN state upon request of the interrupt source host for transmission of an interrupt code, the link controller shall notify the host and ignore the request
- The link controller of an interrupt source starts a timeout timer upon transmission of an interrupt code for the reception of the respective interrupt acknowledge code. In order this feature to be implemented the SpW CODEC must be modified.
- An interrupt mask register per node that defines for which interrupt identifiers the node is the interrupt handler will alleviate the processor from handling Interrupt Requests that it does not handle.
- Specify a minimum time after which the interrupt handler has to respond to an interrupt code. This time is related to the diameter of the SpW network, which may correspond to a circular connection.
- Specify a minimum time for the transmission of an interrupt code after the reception of an interrupt acknowledge code with the same interrupt identifier.
- Cold redundancy implementation (where possible) at both interrupt source and handler in circular networks will prevent the propagation of duplicate interrupt codes.
- ISR bits and timers shall not be affected by a link disconnect.
- The switches shall check and manage (set/reset) ISR bits through atomic operations.
- Protection can be provided by edge switches (*ref:* 3.1.5) in order to overcome the problem of babbling nodes.
- Assign allowed/expected interrupt and interrupt acknowledge codes to each port of each edge switch.
- Consecutive range of allowed interrupt and interrupt acknowledge codes to each port of each edge switch.
- Provision of error information in the switches of the network for FDIR i.e. remote node deactivation using PnP.

3.2.2 Pros and Cons of the proposed mechanism

The proposed solution for Interrupts distribution along with the proposed changes fulfils the basic requirements for Interrupts:

- Interrupt Codes and Acknowledgements are propagated with minimized latency since the same mechanism than Time Code is used.
- Interrupts Codes and Acknowledgements are not affected by the SpW Flow Control mechanism and can therefore travel through blocked links.

On the other hand, the mechanism presents some drawbacks:

- Interrupt codes and interrupt acknowledgement codes may insert more jitter to a time-code than a NCHAR does (14 bits instead of 10).
- Considering that a SpW network may contain up to 224 nodes, the maximum available number of 32 interrupt and interrupt acknowledge identifiers pairs is definitely a significant limitation for future on-board architectures

- In the case of the appearance of simultaneous interrupts the effect on the performance cannot be predicted.
- The design of a switch in the SpW network is challenged in the case of multiple simultaneous interrupts and interrupts acknowledgements.
- The interrupts distribution is compatible with the SpW standard but not with certain existing implementations (GR SpW-RTC, RUAG COLE ASIC).
- The interrupts mechanism does not allow the interrupt source to be interrupt handler for the same interrupt identifier. This may be required by IMA/TSP partitioning in future architectures with SW migration. Specifically, in the theoretic case in which an Interrupt Handler fails and all its tasks are assigned to a node which is the Interrupt Source for an Interrupt Identifier handled by the failed node, the absence of loopback in the switches shall be taken into account during the system design phase.
- Complexity is added in the switch design due to the failsafe operation i.e.:
    - Faulty node protection by edge switches requires up to 32+32 bits per port without taking into account the TMR/EDAC/scrubbing.
    - FDIR statistics by edge switches. For N interrupt and interrupt acknowledge pairs 2xN bits per port are required to count up to 2^N interrupt and interrupt acknowledge codes errors per port without taking into account the TMR/EDAC/scrubbing.

3.2.3 Alternative proposals for Interrupts Distribution

During the requirements collection phase and SpW WG #17 the comments collected on the presented, so far, approach focused on two problems of the mechanism:

- The implementation of the time-outs is complex and requires a lot of HW resources, thus increasing cost and complexity of the switches.
- The proposed mechanism supports up to 32 Interrupts only whereas contemporary architectures (e.g. Bepi Colombo) already have 31 nodes, or more, without taking redundancy into account.

This section briefly presents alternative proposals for Interrupts Distribution which could be taken into account in future evolutions of SpW and related protocols.

3.2.3.1 Interrupts Multicast at the Switches

Interrupts multicast together with switch protection (3.1.5) may mitigate the implementation complexity issue. With this approach each Interrupt Code and Acknowledgement arriving at a switch is not broadcasted but multicasted to a predefined set of switch ports. With this approach there are no circular propagations in a network and thus the time-outs and ISRs at the switch can be removed from the respective implementations.

Assuming that there is a faulty switch in the network that broadcasts the Interrupts and Acknowledgements over ports on which it should not, these can be blocked by the attached switches .(3.1.5).

Regarding the implementation complexity, a memory is required with a 32-bits entry per port for the Interrupt Codes and another one for the Acknowledgements. For a N port switch this makes a 2xNx32 bits memory. With the current proposal, each switch shall have an ISR and a time-out register for each Interrupt Code. Assuming that a 10 bits time-out is required per port this means that 10x32 bits are removed. Although the savings do not seem significant at first sight, it shall be noted that with the current proposal the time-outs are implemented as Flip Flops, whereas with the proposed extension a LUT is required.

3.2.3.2 More than one T7, T6 combinations

Extending the different supported Interrupts can be obtained by using more than one T7, T6 combinations of the unassigned Signalling Codes. This however presents the following drawbacks and is not recommended:

- Using all signalling codes does not allow for future uses of these codes (e.g. redundant Time Code source)

- It has an upper limit of 64 or 96 different Interrupts (depending on whether 2 or 3 T7-T6 combinations will be used) which may also be insufficient for future architectures

3.2.3.3 Virtual Networks

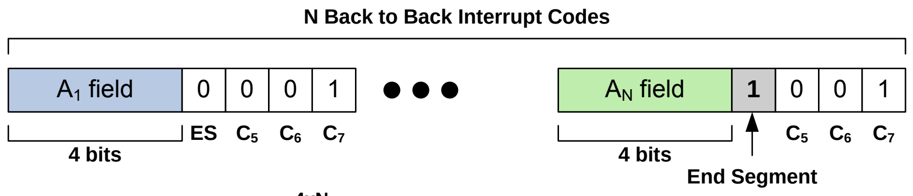
One of the mechanisms to avoid the limitation of 32 different Interrupts is to use the SpW Virtual Networks, an idea proposed by 4Links ([AD,11]). Upon the reception of a Signalling Code, the SpW CODEC makes a "context switch" and uses another path for the reception of the following NCHARs/EOPs/EEPs. This approach has the following pros and cons:

- ☑ Interrupt Vectors are formed by SpW packets.
- ☑ Allows unlimited number of Interrupts.
- ☑ It is possible (although not recommended) to define NACK packets.
- ☒ It violates the SpW 1.0 standard since the Signalling Codes are not broadcasted.
- ☒ It introduces latency and jitter to application packets, which is proportional to the SpW packet length.
- ☒ There is not broadcast or multicast support and in case a link has failed delivery to the destination over redundant paths cannot be supported.
- ☒ The existence of Flow Control in the Virtual Networks may also result in link blockage.
- ☒ It requires CODEC modification in order to synchronize Signalling Code and NCHARs transmission.
- ☒ The effects at system level in case one of the Signalling Codes is lost cannot be predicted accurately.

3.2.3.4 Back-to-back Signalling Codes

Another mechanism to support more than 32 different Interrupts is with the use of back-to-back Signalling Codes. With this approach more than one Signalling Codes form an Interrupt Code and they are sent one after the other with a small time gap in the microseconds range. No other Signalling Code except for a Time Code can be interleaved between them.

Each Signalling Code caries 4 bits of the Interrupt Vector. And there is a bit which defines which is the first part and which is the last part of the Interrupt (or Acknowledge) Code. Upon the reception of a Signalling Code, the Link Controller checks if the ES (End Segment) bit is '0'. If it is not the Signalling Code is rejected, otherwise a time-out timer is started waiting for the second part of the Interrupt Code. If the second segment is received in time (and bit ES = 1) then the Interrupt vector is formed and can be further broadcasted (or multicasted) by a switch, or an Interrupt Code is formed at the receiving node. In the case that the time-out timer has expired the first segment is deleted from the Controller and the respective state machine returns to the initial state.



Up to 2^{4xN} Interrupts can be supported
<u>For N = 2, 256 Interrupts are supported</u>

Figure 13: Back to Back Signalling Codes

This approach has the following pros and cons:

- ☑ Up to 256 Interrupts can be supported
- ☑ The Interrupt Codes and Acknowledgements are broadcasted (or multicasted) allowing for delivery at the Interrupt Handler node in a network with redundant connections, in case a link has failed.
- ☑ It is possible (although not recommended) to define NACK packets, by reducing the number of supported Interrupts

- ☑ Can be extended to support more than 256 different Interrupts with more back-to-back Signalling Codes by using a bit in the $A_1 – A_N$ field as a toggle bit.
- ☒ It violates the SpW 1.0 standard since the Signalling Codes are not broadcasted individually
- ☒ Inserted jitter and latency to application packets is more than twice than those of the SUAI proposal

3.2.3.5 Signalling Code – NCHAR – Signalling Code

A solution similar to the Virtual Networks is to interleave an NCHAR, between two Signalling Codes, which will define the Interrupt Vector. Upon the reception of a Signalling Code at a switch the latter waits for the NCHAR and subsequent Signalling Code in order to broadcast (or multicast) the three characters. It is also possible to extend this approach to more than one characters and result in an approach similar to the Virtual Networks. The main difference with the Virtual networks however, is that there are not FCTs involved in the Interrupts path in order not to result in blocked links.

This approach has the following pros and cons:

- ☑ Up to 256 Interrupts can be supported (Unlimited number of interrupts if more than one NCHARs are used).
- ☑ It is possible to define NACK packet.
- ☑ It violates the SpW standard since the Signalling Codes are not broadcasted upon reception.
- ☑ The Interrupt Codes and Acknowledgements are broadcasted (or multicasted) allowing for delivery at the Interrupt Handler node in a network with redundant connections, in case a link has failed.
- ☑ It is possible (although not recommended) to define NACK packets, by reducing the number of supported Interrupts
- ☑ Can be extended to support more than 256 different Interrupts with more back-to-back Signalling Codes by using a bit in the $A_1 – A_N$ field as a toggle bit.
- ☒ It violates the SpW 1.0 standard since the Signalling Codes are not broadcasted individually
- ☒ Requires modifications to the SpW CODEC in order to synchronize the transmission of Signalling Codes and NCHARs/EOPs/EEPs.
- ☒ Inserted jitter and latency to application packets is more than twice than those of the SUAI proposal
- ☒ Requires modifications at SpW CODEC level in order not to take into account the Flow Control logic over the Interrupts path

3.2.3.6 Summary of Pros and Cons of alternative future evolutions for Interrupts

The following table summarizes the pros and cons of the proposals presented in 3.2.3.2 - 3.2.3.5 for support of more than 32 different Interrupts.

Proposal	T6,T7 bits	Back to Back Interrupts	Virtual Networks	Signalling Code – NCHAR – Signalling Code
Latency	Excellent	Excellent	Excellent/Very Good	Very Good
SpW Link Blockage	No	No	Yes	No
Max Supported Interrupts	64 - 96	At least 256	Unlimited	At least 256
Bounded message size	Yes	Yes	Not specified	Yes
Broadcasted Interrupts	Yes	Yes	No	Yes
Routed Interrupts	No	No	Yes	Can be supported if more than one NCHARs are used

Proposal	T6,T7 bits	Back to Back Interrupts	Virtual Networks	Signalling Code – NCHAR – Signalling Code
NACK packet	No	Possible but reduces max. number of Interrupts	Yes	Yes
Compatibility with SpW standard	Yes	No	No	No
Compatibility with existing devices	No			
Jitter insertion	Minimum	Minimum	Medium	Medium
Single Interrupt reception over Redundant topologies	Yes	Yes	Needs redundancy filtering	Yes
Behaviour upon failure and failure handling	Good	Good	Unknown	Unknown
SpW CODEC modifications	No	Minimum	Significant	Significant
Required memory resources	Low	Low	Medium	Low or Medium