# Revised Requirements of the 4SECURail Case Study

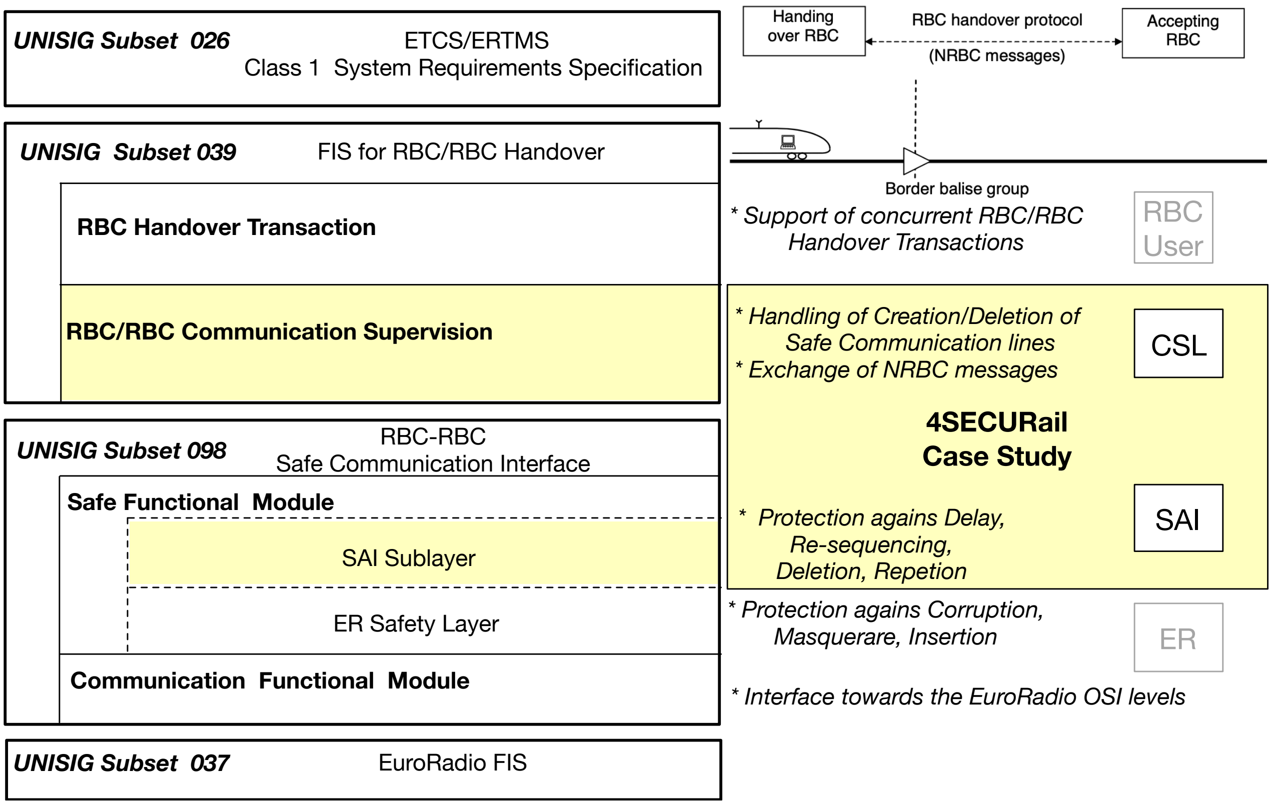
WP2 Task 2.3 Internal Report

| | |
|---|---|
| **Project acronym:** | 4SECURail |
| **Starting date:** | 01/12/2019 |
| **Duration (in months):** | 24 |
| **Call (part) identifier:** | H2020-S2R-OC-IP2-2019-01 |
| **Grant agreement no:** | 881775 |
| **Due date:** | |
| **Actual submission date:** | |
| **Responsible/Author:** | Franco Mazzanti, Dimitri Belli / CNR |
| **Dissemination level:** | PU |
| **Status:** | Draft |

## Table of Contents

# 1 Introduction

The transit of a train from an area supervised by a Radio Block Centre (RBC) to an adjacent area supervised by another RBC occurs during the so-called RBC-RBC handover phase and requires the exchange of information between RBCs according to a specific protocol.
This exchange of information is supported by the communication layer specified within the UNISIG SUBSET_39 and SUBSET_98. Figure 1 hints the overall structure of the UNIG standards supporting the handover of a train.
The 4SECURail case study is focused on two subcomponents of the communication layer underlies the RBC-RBC handover communications. The considered components are the RBC/RBC Communication Supervision layer (CSL) of the SUBSET_039, and on the SAI component, of the Safe Functional Module of the SUBSET_098 (SAI). These two components are those that support the creation/deletion of safe communications lines (over an unreliable OSI Transport layer) and the protected transmission of messages..


UNISIG Subset 026
ETCS/ERTMS
Class 1 System Requirements Specification
UNISIG Subset 039
FIS for RBC/RBC Handover
RBC Handover Transaction
RBC/RBC Communication Supervision
UNISIG Subset 098
RBC-RBC
Safe Communication Interface
Safe Functional Module
SAI Sublayer
ER Safety Layer
Communication Functional Module
UNISIG Subset 037
EuroRadio FIS
Handing over RBC
RBC handover protocol
(NRBC messages)
Accepting RBC
Border balise group
* Support of concurrent RBC/RBC Handover Transactions
RBC User
* Handling of Creation/Deletion of Safe Communication lines
* Exchange of NRBC messages
CSL
4SECURail Case Study
* Protection agains Delay, Re-sequencing, Deletion, Repetion
SAI
* Protection agains Corruption, Masquerare, Insertion
ER
* Interface towards the EuroRadio OSI levels

In particular the CSL is responsible for requesting the activation, and in case of failure the rehestablishment, of the communication line, and for the forwarding of RBC Handover Transaction messages on the active line.

The SAI is responsible for ensuring the absence of excessive delays, message repetitions or message rehordering during the transmission.  This is achieved by adding sequence number and time related information to the RBC messages.
The two sides of the communication line are configured as an "initiator" side and a "called" side.

The 4SECURail case study does not include the rigorous specification and  formal modelling of the ER Safety Layer (responsible to prevent corruption, masquerade and insertion issues during the communications), not of the lower  EuroRadio levers and the interface to it (Communication Functional Module). All this parts are abstracted an ER environment component which behaves according to the requested assumptionsa and allows the two SAI side to interact.

On the upper leverl the 4SECURail case study does not include the rigorous specification and formal modelling of the interactions of the trains with the RBC controlling the zone in which they are moving, nor the activation of multiple, possible concurrent,  RBC-RBC handover sessions when trains have to move from a zone controlled by an RBC to another. From the point of view of the CSL only RBC messages are forwarded to/from the upper level, toghether with information on the current status of the communication line. Again. these upper parts are abstracted an RBC environment component which behaves according to the requested assumptions.

Section 2 (derived from Appendix B of Deliverable 2.5) presents, in the form of natural language, the requirements for these CSL and SAI components, distinguishing the initiator (ICSL, ISAI) and called (CCSL, CSAI) sides. These requirements are a rewriting of the original standard definition to make the specification more self-contained and understandable.
While performing the formal modelling and analysis, these requirements have been incrementally generated and kept aligned with the developed formal and semi-formal models when missing parts or inconsistencies are detected.

It is important to remark that the overall goal of the 4SECURail project is NOT the validation/verification of some parts of standard documents, but the observation of the impact (in terms of qualitative and quantitative and costs and benefits) of the use of formal methods during the requirements definition phase. In our specific case  we are interested to observe the imprevements that can be achived in the natural language specification of the components of interest, while passing from the initial free-style (subject to ambiguities, incompletenes and inconsistences) version to a more rigorous style backed by underlying formal models.

Section 3  (derived from Appendix A of Deliverable 2.5) presents a graphical (abstract, semi-formal) representation of the UML state machines describing the expected behavior of the SAI and CSL components.

Section 4 presents one complete executable and verifiable version of these components specified in the syntax accepted by UMC for the specification of UML state machines.

This UMC notation is used as a basis for the mechanical translation of the UMC forml models into other notations like CADP/LNT  and ProB, allowing the use of different formal frameworks for the analysis of the system properties. All the three different formal models can be proved, inspite of the the differences in the underlying theories and syntax, to be strongly equivalent.

In order to perform the analysis of the specified components it is necessary to build closed scenarios in which the designed CSL and SAI components and composed with a stimulating environment. Different kind of scenarios can be imagined depending on the kind of system properties we are interest to analyse.
Since the modelled system is heavily time dependent, but our formalisms are not explicitly support time, out formal models include an additiona Timer component that allowes the various system components to synchronously proceed in parallel, but relatively at the same speed.
Also these environent components (Timer, ER and two RBC sides) can be designed as UML state machines and encoded in UMC syntax so that the full resulting system can be mechanically translated in the various formalisms and analysed.

# 2 Natural Language specifications

## Requirements Specification for the ***Initiator CSL*** Component

**Configuration Parameters**

System parameters,

- *max connection delay;*
- *max delay between send operations:*
- *max delay between receive operations.*

**External Interactions**

The *Initiator CSL* can receive from the *Initiator RBC* component the following message:

- *RBC_User_Data.request(RBC_data_value);*

and can send to the *RBC* component the following messages:

- *RBC_User_Connect.indication;*
- *RBC_User_Disconnect.indication;*
- *RBC_User_Data.indication(RBC_data_value).*

The *CSL* can receive from the *Initiator SAI* component the following messages:

- *SAI_CONNECT.confirm;*
- *SAI_DISCONNECT.indication;*
- *SAI_DATA.indication(message_type*[1]*, SAI_data_value);*
- *SAI_ERROR.report;*

and can send to the *SAI* component the following messages:

- *SAI_CONNECT.request;*
- *SAI_DISCONNECT.request;*
- *SAI_DATA.request(message_type, SAI_data_value).*

**States**

The *CSL* can be in the following four main states:

- *Disconnected*, when the communication is unactive;
- *Connecting*, when the communication is in the establishment phase;
- *Connected*, when the communication is active;
- *Waiting*, when the communication is between the *Connected* and *Disconnected* states.

**External Guarantees**

- *CSL* sends *RBC_User_Data.indication* messages only after an *RBC_User_Connect.request* not followed by *RBC_User_Disconnect.indication*;
- *CSL* sends to the *RBC* component an *RBC_User_Disconnect.indication* message only after an *RBC_User_Connect.request* message not already followed by *RBC_User_Disconnect.indication*;
- *CSL* sends to the *RBC* component an *RBC_User_Connect.indication* message only as first message or after an *RBC_User_Disconnect.indication* not already followed by *RBC_User_Connect.indication*;

[1] *message_type* may refer to either *life_sign* or *RBC_data.*

- the first message (possibly) sent to the *RBC* component is an *RBC_User_Connect.indication* message;
- the initiator *CSL* periodically sends to the *SAI* component either *SAI_CONNECT.request* or *SAI_DATA.request* messages;
- if the initiator *CSL*, while in *Connected* (*COMMS*) state, does not receive any *SAI_DATA.indication* message from the *SAI* for a certain specified amount of time, a *SAI_DISCONNECT.request* message is sent to the *SAI*;
- the initiator CSL may send a *SAI_DISCONNECT.request* message only when in *Connected* (*COMMS*) state;
- incoming messages are buffered and served with *FIFO* policy.

**External Assumptions**

- The *SAI* always replies with a *SAI_DISCONNECT.indication* message to *SAI_DISCONNECT.request* messages issued by the *CSL*.

**Behavioral Requirements**

**R1:** At startup, the *CSL* is in *Disconnected* state.

*When in Disconnected State*

**R2:** When in *Disconnected* state, the *CSL* immediately sends a *SAI_CONNECT.request* to the *SAI* component, starts a *connTimer*, and moves to the *Connecting* state.

*When in Connecting State*

**R3:** When in *Connecting* state the *connTimer* expires, the *CSL* moves to *Disconnected* state.

**R4:** When in *Connecting* state is received a *SAI_CONNECT.confirm* from the *SAI* component, the *CSL* sends an *RBC_User_Connect.indication* to the *RBC* component, starts both the *sendTimer* and the *recTimer*, and moves to *Connected* state. It is allowed to set the sendTimer so that an initial lifesign is sent without delay.

*When in Waiting State*

**R5:** When in *Waiting* state is received a *SAI_DISCONNECT.indication* from the *SAI* component, the *CSL* moves to *Disconnected* state.

*When in Connected State*

**R6:** When in *Connected* state the *recTimer* expires, the *CSL* sends a *SAI_DISCONNECT.request* to the *SAI* component, an *RBC_User_Disconnect.indication* to the *RBC* and moves to *Waiting* state.

**R7:** Each time that in *Connected* state the *sendTimer* expires, the *CSL* sends a *SAI_DATA.request* with a *life_sign* to the *SAI* component.

**R8:** When in *Connected* state is received an *RBC_User_Data.request* with *RBC_data* from the *RBC* component, the *CSL* sends a *SAI_DATA.request* with such *RBC_data* to the *SAI* component.

**R9:** When in *Connected* state is received a *SAI_DATA.indication* with *SAI_data* from the *SAI* component, the *CSL* sends an *RBC_User_Data.indication* with such *SAI_data* to the *RBC* component and restarts the *recTimer*.

**R10:** When in *Connected* state is received a *SAI_DATA.indication* with *a life_sign* from the *SAI* component, the *CSL* restarts the *recTimer*.

**R11:** When in *Connected* state is received a *SAI_DISCONNECT.indication* from the *SAI* component, the *CSL* sends an *RBC_User_Disconnect.indication* to the *RBC* component and moves to *Disconnected* state.

*<u>Discarding of Messages</u>*

**RD1:** When in *Connecting* state, the *CSL* discards any message except for *SAI_CONNECT.confirm* from the *SAI* component.

**RD2:** When in *Waiting* state, the *CSL* discards any message except for *SAI_DISCONNECT.indication* from the *SAI* component.

**RD3:** When in *Connected* state, the *CSL* component discards only *SAI_CONNECT.confirm* and *SAI_ERROR.report* messages from the *SAI* component.

## Requirements Specification for the ***Initiator SAI*** Component

**Configuration Parameters**

Initialization kind: *Execution Cycle* option.

System parameters,

- for *Execution Cycle* procedure:
    - *maximum initialization delay*
    - *Mec* (limit of the execution cycle counters)*;*
    - *K* (max acceptable transmission delay for a message);
- for *ACK* procedure:
    - *ack_request_period;*
    - *ack_response_timeout;*
- for *sequence number*:
    - *N* (limit of acceptable, consecutive message losses, *N* = *1* means no losses)*;*
    - *M* (limit of the sequence number values, which have range *0..M-1*).

**External Interactions**

The *Initiator SAI* can receive from the *Initiator CSL* component the following messages:

- *SAI_CONNECT.request*;
- *SAI_DISCONNECT.request*;
- *SAI_DATA.request (message_type*[2]*, RBC_data_value)*;

and can send to the *CSL* component the following messages:

- *SAI_CONNECT.confirm*;
- *SAI_DISCONNECT.indication*;
- *SAI_DATA.indication(message_type, RBC_data_value)*;
- *SAI_ERROR.report*.

The *SAI* can receive from the *EuroRadio* Safety Layer (henceforth *ER-SL*) the following messages:

- *Sa_CONNECT.confirm*;
- *Sa_DISCONNECT.indication*;
- *Sa_DATA.indication(message_type, data_value, ack_request, ack_response, sequence_number, execution_cycle_number)*;
- *Sa_ExecutionCycleStart(sequence_number, execution_cycle_counter)*;

and can send to the *ER-SL* the following messages:

- *Sa_CONNECT.request*;
- *Sa_DISCONNECT.request*;
- *Sa_DATA.request(message_type, data_value, ack_request, ack_response, sequence_number, execution_cycle_number)*;
- *Sa_ExecutionCycle(sequence_number, execution_cycle_counter)*.

**Internal Variables**

- *sequence_number;*
- *execution_cycle_counter;*

[2] *message_type* may refer to either *life_sign* or *RBC_data.*

- *last_received_sequence_number;*
- *last_received_execution_cycle_counter;*
- *execution_cycle_OFFSET.*

**States**

The *SAI* can be in the following four main states:

- *Connected*, when the communication is active;
- *Connecting*, when the communication is in the establishment phase;
- *Initializing*, while performing the execution cycle start procedure;
- *Disconnected*, when the communication is unactive.

**External Guarantees**

- The *SAI* always replies with a *SAI_DISCONNECT.indication* message to *SAI_DISCONNECT.request* messages issued by the *CSL*;
- the data messages delivered to the *CSL* are valid (i.e., arrived with a limited delay), not duplicated, not reordered messages;
- no more than one data message per execution cycle is sent to the *ER-SL*;
- incoming messages are buffered and served with *FIFO* policy.

**External Assuptions**

- The *ER-SL* always eventually replies either with a *Sa_DISCONNECT.indication* or with a *SAI_CONNECT.confirm* to *Sa_CONNECT.request* messages issued by the SAI;
- the initiator CSL, after having sent a *SAI_CONNECT.request* message to the SAI, does not send a *SAI_DISCONNECT.request* message until *SAI_CONNECT.indication* messages is received.

**Behavioral Requirements**

**R1**: At startup, the *SAI* is in *Disconnected* state.

*When in Disconnected State*

**R2:** When in *Disconnected* state is received a *SAI_CONNECT.request* from the *CSL* component, the *SAI* sends a *Sa_CONNECT.request* to the *ER-SL* and moves to *Connecting* state.

**R3:** When in *Disconnected* state is received a *SAI_DISCONNECT.request* from the *CSL* component, the *SAI* replies with a *SAI_DISCONNECT.indication* to the *CSL* component.

*When in Connecting State*

**R4:** When in *Connecting* state is received a *Sa_DISCONNECT.indication* from the *ER-SL*, the *SAI* moves to *Disconnected* state.

**R6:** When in *Connecting* state is received a *Sa_CONNECT.confirm* from the *ER-SL*, the *SAI* replies with a *Sa_ExecutionCycle(seqnum, ecnum)* to the *ER-SL* and moves to the *Initializing* state, waiting for a *Sa_ExecutionCycleStart* message from the *ER-SL* within a *maximum initialization delay*. The management of the *Sa_ExecutionCycleStart* parameters is done according to the requirements in the following *Sequence Numbers Management* and *Execution Cycle Counters Management* sections.

*When in Initializing State*

**R7:** When in *Initializing* state the *maximum initialization delay* expires, the *SAI* sends an *SAI_ERROR.report* to the *CSL* component, a *Sa_DISCONNECT.request* to the *ER-SL* and moves to *Disconnected* state.

**R9:** When in *Initializing* state is received a *Sa_DISCONNECT.indication* from the *ER-SL*, the *SAI* moves to *Disconnected* state.

**R11:** When in *Initializing* state is received a *Sa_ExecutionCycleStart(seqnum, ecnum)* from the *ER-SL*, the *SAI* sends a *SAI_CONNECT.confirm* to the *CSL* component and moves to *Connected* state. The received *seqnum* is accepted as initial remote sequence number and the *ecnum* is accepted as initial value of the remote execution cycle counter. The *execution_cycle_OFFSET* variable is set as the difference between the current execution cycle counter and the received execution cycle counter. While the *last_received_sequence_number* variable is set to the received sequence number.

*When in Connected State*

**R12:** When in *Connected* state is received a *SAI_DISCONNECT.request* from the *CSL* component, the *SAI* replies with a *SAI_DISCONNECT.indication* to the *CSL* component, sends a *Sa_DISCONNECT.request* to the *ER-SL*, and moves to *Disconnected* state.

**R13a:** When in *Connected* state is received a *SAI_DATA_request(msgtype, data)* from the *CSL* component, and yet no other data message has been sent in this cycle, the *SAI* sends a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* to the *ER-SL*.
The *ackreq* and *ackresp parameters are set according to* REQ_ACKs.
The *seqnum* parameter is set according to SEQ_NUMs and the *ecnum* parameter is set according to REQ_ECNUMs.

**R13b:** When in *Connected* state is received a *SAI_DATA.request(msgtype, data)* from the *CSL* component, but another data message has already been sent in this cycle, the *SAI_DATA.request is saved in* a *FIFO dataout buffer (see also REQ_OUTDATABUFF).*

**R14:** Each time that in *Connected* state the *set_ack_response* expires, the *SAI* sends a *SAI_ERROR.report* to the *CSL* component.

**R15:** When in *Connected* state is received a *Sa_DISCONNECT.indication* from the *ER-SL*, the *SAI* sends a *SAI_DISCONNECT.indication* to the *CSL* component and moves to *Disconnected* state.

**R16:** When in *Connected* state is received a *Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum)* from the *ER-SL* we can have four cases, depending on the received *seqnum* and *ecnum* values (see SEQ_NUMs and REC_ECNUMs Management).
* The *seqnum* is the one EXPECTED and *ecnum* is VALID: In this case the *SAI* sends *a SAI_DATA.indication(msgtype, data)* to the *CSL* component.
* The *seqnum* is ACCEPTABLE and the *ecnum* is VALID: in this case the *SAI* sends a *SAI_DATA.indication(msgtype, data)* and a *SAI_ERROR.report* to the *CSL* component.
* The *seqnum* is OLD or (the *seqnum* is ACCEPTABLE and the *ecnum* is VALID): In this case the *SAI* sends a *SAI_ERROR.report* to the *CSL* component and discards the *Sa_DATA.indication* message.
* The *seqnum* is NOT_ACCEPTABLE: In this case the *SAI* component sends a *Sa_DISCONNECT.request* to ER-SL and a *SAI_DISCONNECT.indication* to the *CSL* component, and then moves to *Disconnected* state.

*OUTDATA Buffer Management*

**REQ_OUTDATABUFF1:** At the beginning of each cycle, if the *dataout buffer* is not empty, the first *SAI_DATA.request(msgtype, data) in* the queue is removed and its data are used to send a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* to the *ER-SL.*
*The ackreq, ackresp, seqnum, ecnum* parameters are set according to *REQ_ECNUM, REQ_ACK*, and *REQ SEQNUM* requirements.

**REQ_OUTDATABUFF2:** When the *SAI* moves from the *Connected* state to the *Disconnected* state, the *dataout buffer is* emptied and the possibly waiting messages are discarded.

*Execution Cycle Counters Management*

**REQ_ECNUM1**: When entering in the *Initializing* state, the initial value of the *execution cycle counter* is set to 0.

**REQ_ECNUM2:** While in the *Initializing* or *Connected* state, the *execution cycle counter* is incremented modulo Mec at every cycle.

**REQ_ECNUM3:** When sending a *Sa_ExecutionCycleStart(seqnum,ecnum)* message or a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* the value of the *ecnum* parameter is set to the current value of the *execution cycle counter*.

**REQ_ECNUM4:** When receiving a *Sa_ExecutionCycleStart(seqnum,ecnum)* message from the *ER-SL,* the value of the *ecnum* parameter is used to compute the *EC_OFFSET* as difference between the current value of the *execution cycle counter and* the received *seqnum* value.

**REQ_ECNUM5:** When receiving a *Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message from *ER-SL*, the message in considered VALID if the message delay is less than *K*, where the message delay is computed as follows[3]:

```
message_delay = (execution_cycle_counter - EC_OFFSET) mod Mec[4]) - ecnum;
if message_delay < -Mec/2 then
        message_delay := message_delay + Mec;
elsif message_delay > Mec/2 then
        message_delay := message_delay - Mec;
end if
```

*Sequence Numbers Management*

**SEQ_NUM1**: When entering in the state *Connected*, the *sequence_number* is set to 0.

**SEQ_NUM2**: When in *Connecting* state a *Sa_ExecutionCycleStart(seqnum,ecnum)* message is sent to the *ER-SL,* the *seqnum* parameter is set to the current value of *sequence_number*.

[3] This is a simplification from what required by UNISIG-098 as we assume that the EC period is 1 cycle for both *SAI* sides.
[4] Also when applied to negative numbers, (N mod M) is assumed to be equal to ((N+M) mod M).

**SEQ_NUM3**: When in the *Initializing* or *Connected* state a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message is sent to the *ER-SL, the seqnum* parameter is set to the current value of *sequence_number*, and the *sequence_number* is incremented by *1 mod M*.

**SEQ_NUM4**: When in *Initializing* state is received a *Sa_ExecutionCycleStart(seqnum,ecnum)* message from *ER-SL*, the value of the *seqnum* parameter is saved as *last_received_sequence_number*.

**SEQ_NUM5:** When in the *Initializing* or *Connected* state is received a *Sa_DATA.indication (msgtype, data, ackreq, ackresp, seqnum, ecnum)* from the *ER-SL*, the distance of the current message from the last received one is computed as follows*:*

```
distance := seq_num - last_received_sequence_number;
if  (distance <  -M/2) then {distance :=  distance + M};
else if  (distance  >  M/2) then {distance := distance - M};
```

If the distance value is equal to 1, the *seqnum* is considered EXPECTED.
If the distance value is lower than 1, the *seqnum* is considered OLD.
If the distance value is greater than 1 and less or equals to *N*, the *seqnum* is considered ACCEPTABLE.
If the distance value is greater than *N*, the *seqnum* is considered NOT_ACCEPTABLE.

*ACK Management*

**REQ_ACK1**: When in *Connected* state, the *SAI* periodically (with a configurable *ack_request_period*) sets an *ackreq* flag to the first *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message to be forwarded to the *ER-SL* and starts an *ack_response_timer* with a *max_response_delay* limit.
The *ackreq* flag is not set and the timer is not started if the *SAI* is still waiting for the response to a previous ack request.

**REQ_ACK2**: When the *ack_response_timeout* expires, if a *Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message with an *ackresp* parameter set has not yet been received from the *ER-SL*, the *SAI* sends a *SAI_ERROR.report* to the *CSL* component and restarts the *ack request timer*.

**REQ _ACK3:** While in *Connected* or *Initializing* state, when it is received a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* from the *CSL* component, the *SAI* sets the *ackresp* parameter in next *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message to be sent to the *ER-SL*.

*Discarding of Messages*

**RD1**: When in *Disconnected* state the *SAI* discards any message except for,
- *SAI_CONNECT.request* and *SAI_DISCONNECT.request* from the *CSL* component.

**RD2:** When in *Connecting* state, the *SAI* discards any message except for,
- *Sa_DISCONNECT.indication*, and *Sa_CONNECT.confirm* from the *ER-SL*;

**RD3:** When in *Initializing* state, the *SAI* discards any message except for,
- *Sa_DISCONNECT.indication and Sa_ExecutionCycleStart* from the *ER-SL*;

**RD4:** When in *Connected* state, the *SAI* discards any message except for,

- *Sa_DISCONNECT.indication* and *Sa_DATA.indication* from the *ER-SL*;
- *SAI_DISCONNECT.request, SAI_DATA.request* from the *CSL* component.

## Requirements Specification for the ***Called CSL*** Component

**Configuration Parameters**
System parameters,

- *max delay between send operations:*
- *max delay between receive operations.*

**External Interactions**
The *Called CSL* can receive from the *Called RBC* component the following message:

- *RBC_User_Data.request(RBC_data_value);*

and can send to the *RBC* component the following messages:

- *RBC_User_Connect.indication;*
- *RBC_User_Disconnect.indication;*
- *RBC_User_Data.indication(RBC_data_value).*

The *CSL* can receive from the *Called SAI* component the following messages:

- *SAI_CONNECT.indication;*
- *SAI_DISCONNECT.indication;*
- *SAI_DATA.indication(message_type*[5]*, SAI_data_value);*
- *SAI_ERROR.report;*

and can send to the *SAI* component the following messages:

- *SAI_CONNECT.request;*
- *SAI_DISCONNECT.request;*
- *SAI_DATA.request(message_type, SAI_data_value).*

**States**
The *CSL* can be in the following two states:

- *Disconnected (NOCOMMS)*, when the communication is unactive;
- *Connected (COMMS)*, when the communication is active.

**External Guarantees**

- The frequency of messages being sent by *CSL* to *RBC* is limited by an upper bound;
- the frequency of messages being sent by *CSL* to *SAI* is limited by an upper bound;
- *CSL* sends *RBC_User_Data.indication* messages only after an *RBC_User_Connect.request* not followed by *RBC_User_Disconnect.indication*;
- *CSL* sends to the *RBC* component an *RBC_User_Disconnect.indication* message only after an *RBC_User_Connect.indication* message not already followed by *RBC_User_Disconnect.indication*;
- the first message (possibly) sent to the *RBC* component is an *RBC_User_Connect.indication* message;
- *CSL* sends to the *RBC* component an *RBC_User_Connect.indication* message only as first message or after an *RBC_User_Disconnect.indication* not already followed by *RBC_User_Connect.indication*;
- the called CSL, while in the *Connected* (*COMMS*) state periodically sends to the *SAI* component *SAI_DATA.request* messages;

[5] *message_type* may refer to either *life_sign* or *RBC_data.*

- if the called CSL, while in the *Connected* (*COMMS*) state, does not receive any *SAI_DATA.indication* message from the SAI for a certain specified amount of time, a *SAI_DISCONNECT.request* message is sent to the SAI;
- incoming messages are buffered and served with *FIFO* policy.

**Behavioral Requirements**

**R1:** At startup, the *CSL* is in *Disconnected* state.

*When in Disconnected State*

**R2:** When in *Disconnected* state is received a *SAI_CONNECT.indication* from the *SAI* component, the *CSL* sends an *RBC_User_Connect_indication* to the *RBC* component, starts both the *sendTimer* and the *recTimer*, and moves to *Connected* state. It is allowed to set the sendTimer so that an initial lifesign is sent without delay.

*When in Connected State*

**R4:** When in *Connected* state is received an *RBC_User_Data.request*(*userdata*) from the *RBC* component, the *CSL* sends a *SAI_DATA.request(RBC_data,userdata*) to the *SAI* component.

**R5:** Each time that in *Connected* state the *sendTimer* expires, the *CSL* sends a *SAI_DATA.request* with a *life_sign* to the *SAI* component.

**R6:** When in *Connected* state is received a *SAI_DATA.indication* with *a life_sign* from the *SAI* component, the *CSL* restarts the *recTimer*.

**R7:** When in *Connected* state is received a *SAI_DATA.indication* with *SAI_data* from the *SAI* component, the *CSL* sends an *RBC_User_Data.indication* with such *SAI_data* to the *RBC* component and restarts the *recTimer*.

**R8:** When in *Connected* state is received a *SAI_DISCONNECT.indication* from the *SAI* component, the *CSL* sends an *RBC_User_Disconnect.indication* to the *RBC* component and moves to *Disconnected* state.

**R9:** When in *Connected* state the *recTimer* expires, the *CSL* sends a *SAI_DISCONNECT.request* to the *SAI* component, an *RBC_User_Disconnect.indication* to the *RBC* component and moves to *Disconnected* state.

*Discarding of Messages*

**RD1**: When in *Disconnected* state the *CSL* does not accept any kind of message except for *SAI_CONNECT.indication* from the *SAI* component.

**RD2:** When in *Connected* state the *CSL* discards *SAI_CONNECT.indication* and *SAI_ERROR.report* messages from the *SAI* component.

## Requirements Specification for the ***Called SAI*** Component

**Configuration Parameters**
Initialization kind: *Execution Cycle* option.
System parameters,

- for *Execution Cycle* procedure:
  - *maximum initialization delay*
  - *Mec* (limit of the execution cycle counters)*;*
  - *K* (max acceptable transmission delay for a message);
- for *ACK* procedure:
  - *ack_request_period;*
  - *ack_response_timeout;*
- for *sequence number*:
  - *N* (limit of acceptable, consecutive message losses, *N* = *1* means no losses)*;*
  - *M* (limit of the sequence number values, which have range *0..M-1*).

**External Interactions**
The *Called SAI* can receive from the *Called CSL* component the following messages:

- *SAI_DISCONNECT.request*;
- *SAI_DATA.request(message_type*[6]*, RBC_data_value)*;

and can send to the *CSL* the following messages:

- *SAI_CONNECT.indication*;
- *SAI_DISCONNECT.indication*;
- *SAI_DATA.indication(message_type, RBC_data_value)*;
- *SAI_ERROR.report*.

The *SAI* can receive from the *EuroRadio* Safety Layer (henceforth *ER-SL*) the following messages:

- *Sa_CONNECT.indication*;
- *Sa_DISCONNECT.indication*;
- *Sa_DATA.indication(message_type, SAI_data_value, ack_request, ack_response, sequence_number, execution_cycle_number)*;
- *Sa_ExecutionCycleStart(sequence_number, execution_cycle_counter)*;

and can send to the *ER-SL* the following messages:

- *Sa_CONNECT.response*;
- *Sa_DISCONNECT.request*;
- *Sa_DATA.request(message_type, SAI_data_value, ack_request, ack_response, sequence_number, execution_cycle_number)*;
- *Sa_ExecutionCycle(sequence_number, execution_cycle_counter)*.

**Internal Variables**

- *sequence_number;*
- *execution_cycle_counter;*
- *last_received_sequence_number;*

[6] *message_type* may refer to either *life_sign* or *RBC_data.*

- *last_received_execution_cycle_counter;*
- *execution_cycle_OFFSET.*

**States**

The *SAI* can be in the following four main states:

- *Connected*, when the communication is active;
- *Connecting*, when the communication is in the establishment phase;
- *Initializing*, while performing the execution cycle start procedure;
- *Disconnected*, when the communication is unactive.

**External Guarantees**

- The data messages delivered to the *CSL* are valid (i.e., arrived with a limited delay), neither duplicated nor reordered;
- no more than one data message per execution cycle is sent to the *ER-SL*;
- incoming messages are buffered and served with *FIFO* policy.

**Behavioral Requirements**

**R1**: At startup, the *SAI* is in *Disconnected* state.

*When in Disconnected State*

**R2:** When in *Disconnected* state is received a *Sa_CONNECT.indication* from the *ER-SL*, the *SAI* replies with a *Sa_CONNECT.response* to the *ER-SL* and moves to *Connecting* state.

*When in Connecting State*

**R2b:** When in *Connecting* state is received a *Sa_CONNECT.indication* from the *ER-SL*, the *SAI* replies with a *Sa_CONNECT.response* to the *ER-SL* and remains in the *Connecting* state.

**R3:** When in *Connecting* state is received a *Sa_DISCONNECT.indication* from the *ER-SL*, the *SAI* moves to *Disconnected* state.

**R5:** When in *Connecting* state is received a *Sa_ExecutionCycleStart(seqnum, ecnum)* from the *ER-SL*, the *SAI* replies with a *Sa_ExecutionCycle(seqnum, ecnum)* to the *ER-SL*, starts an *initTimer* set to the *maximum initialization delay, and* moves to *Initializing* state. The management of the *Sa_ExecutionCycleStart* parameters are done according to the rules in the *Sequence Numbers Management* and *Execution Cycle Counters Management* sections.

*When in Initializing State*

**R2c:** When in *Initializing* state is received a *Sa_CONNECT.indication* from the *ER-SL*, the *SAI* replies with a *Sa_CONNECT.response* to the *ER-SL* and moves to *Connecting* state.

**R6:** When in *Initializing* state the *maximum initialization delay* expires, the *SAI* sends a *SAI_ERROR.report* to the *CSL* component and moves to *Disconnected* state.

**R8:** When in *Initializing* state is received a *Sa_DISCONNECT.indication* from the *ER-SL*, the *SAI* moves to *Disconnected* state.

**R9:** When in *Initializing* state is received a *Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum)* from the *ER-SL* may have four cases, depending on the received *seqnum* and *ecnum* values (see REQ SEQ_NUMs and REC_ECNUMs).
* The *seqnum* is the one EXPECTED and *ecnum* is VALID: In this case the *SAI* moves to *Connected* state and sends both a *SAI_CONNECT.indication* and a *SAI_DATA.indication(msgtype, data)* to the *CSL* component.
* The *seqnum* is ACCEPTABLE and the *ecnum* is VALID: in this case the *SAI* moves to *Connected* state and sends a *SAI_CONNECT.indication*, a *SAI_DATA.indication(msgtype, data) and* a *SAI_ERROR.report* to the *CSL* component.
* The *seqnum* is NOT_ACCEPTABLE: in this case the *SAI* component sends a *Sa_DISCONNECT.request* to *ER-SL* and moves to *Disconnected* state.
* The *seqnum* is OLD or (the *seqnum* is ACCEPTABLE and the *ecnum* is VALID): In this case the *SAI* sends a *SAI_ERROR.report* to the *CSL* component and discards the *Sa_DATA.indication* message.

*<u>When in Connected State</u>*

**R10:** When in *Connected* state is received a *SAI_DISCONNECT.request* from the *CSL* component, the *SAI* replies with a *SAI_DISCONNECT.indication* to the *CSL* component, sends a *Sa_DISCONNECT.request* to the *ER-SL*, and moves to *Disconnected* state.

**R11:** When in *Connected* state is received a *Sa_DISCONNECT.indication* from the *ER-SL*, the *SAI* sends a *SAI_DISCONNECT.indication* to the *CSL* component and moves to *Disconnected* state.

**R12:** When in *Connected* state is received a *Sa_CONNECT.indication* from the *ER-SL*, the *SAI* replies with a *Sa_CONNECT.response* to the *ER-SL*, sends a *SAI_DISCONNECT.indication* to the *CSL* component, and moves to *Connecting* state.

**R13a:** When in *Connected* state is received a *SAI_DATA_request(msgtype, data)* from the *CSL* component, and yet no other data message has been sent in this cycle, the *SAI* sends a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* to the *ER-SL*.
The *ackreq* and *ackresp parameters are set according to* REQ_ACKs.
The *seqnum* parameter is set according to SEQ_NUMs and the *ecnum* parameter is set according to REQ_ECNUMs.

**R13b:** When in *Connected* state is received a *SAI_DATA.request(msgtype, data)* from the *CSL* component, but another data message has already been sent in this cycle, the *SAI_DATA.request is saved in* a *FIFO dataout buffer (see also REQ_OUTDATABUFF).*

**R14:** When in *Connected* state is received a *Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum)* from the *ER-SL* we can have four cases, depending on the received *seqnum* and *ecnum* values (see SEQ_NUMs and REC_ECNUMs).

- * The *seqnum* is the one EXPECTED and *ecnum* is VALID: In this case the *SAI* sends a *SAI_DATA.indication(msgtype, data)* to the *CSL* component.
  Depending on the received values of the *ackreq* and *ackresp* parameters, appropriate actions are performed (see REQ_ACKs).
- * The *seqnum* is ACCEPTABLE and the *ecnum* is VALID: in this case the *SAI* sends a *SAI_DATA.indication(msgtype, data) and* a *SAI_ERROR.report* to the *CSL* component.
  Depending on the received values of the *ackreq* and *ackresp* parameters, appropriate actions are performed (see REQ_ACKs).

* The *seqnum* is OLD or (the *seqnum* is ACCEPTABLE and the *ecnum* is VALID): In this case the *SAI* sends a *SAI_ERROR.report* to the *CSL* component and discards the *Sa_DATA.indication* message.
* The *seqnum* is NOT_ACCEPTABLE: In this case the *SAI* component sends a *Sa_DISCONNECT.request* to ER-SL, a *SAI_DISCONNECT.indication* to the *CSL* component, and then moves to *Disconnected* state.

<u>*OUTDATA Buffer Management*</u>

**REQ_OUTDATABUFF1:** At the beginning of each cycle, if the *dataout buffer* is not empty, the first *SAI_DATA.request(msgtype, data)* in the queue is removed and its data are used to send a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* to the *ER-SL.*
*The ackreq, ackresp, seqnum, ecnum* parameters are set according to *REQ_ECNUM, REQ_ACK*, and *REQ SEQNUM* requirements.

**REQ_OUTDATABUFF2:** When the *SAI* moves from the *Connected* state to the *Disconnected* state, the *dataout buffer is* emptied and the possibly waiting messages are discarded.

<u>*Execution Cycle Counters Management*</u>

**REQ_ECNUM1**: When entering in the *Initializing* state, the initial value of the *execution cycle counter* is set to 0.

**REQ_ECNUM2:** While in the *Initializing* or *Connected* state, the *execution cycle counter* is incremented modulo Mec at every cycle.

**REQ_ECNUM3:** When sending a *Sa_ExecutionCycleStart(seqnum,ecnum)* message or a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* the value of the *ecnum* parameter is set to the current value of the *execution cycle counter*.

**REQ_ECNUM4:** When receiving a *Sa_ExecutionCycleStart(seqnum,ecnum)* message from the *ER-SL,* the value of the *ecnum* parameter is used to compute the *EC_OFFSET* as difference between the current value of the *execution cycle counter and* the received *seqnum* value.

**REQ_ECNUM5:** When receiving a *Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message from *ER-SL*, the message in considered VALID if the message delay is less than *K*, where the message delay is computed as follows[7]:

```
message_delay = (execution_cycle_counter - EC_OFFSET) mod Mec[8]) - ecnum;
if message_delay < -Mec/2 then
        message_delay := message_delay + Mec;
elsif message_delay > Mec/2 then
        message_delay := message_delay - Mec;
end if
```

<u>*Sequence Numbers Management*</u>

**SEQ_NUM1**: When entering in the state *Connected*, the *sequence_number* is set to 0.

[7] This is a simplification from what required by UNISIG-098 as we assume that the EC period is 1 cycle for both *SAI* sides.
[8] Also when applied to negative numbers, (N mod M) is assumed to be equal to ((N+M) mod M).

**SEQ_NUM2**: When in *Connecting* state a *Sa_ExecutionCycleStart(seqnum,ecnum)* message is sent to the *ER-SL,* the *seqnum* parameter is set to the current value of *sequence_number*.

**SEQ_NUM3**: When in the *Initializing* or *Connected* state a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message is sent to the *ER-SL, the seqnum* parameter is set to the current value of *sequence_number*, and the *sequence_number* is incremented by *1 mod M*.

**SEQ_NUM4**: When in *Initializing* state is received a *Sa_ExecutionCycleStart(seqnum,ecnum)* message from *ER-SL*, the value of the *seqnum* parameter is saved as *last_received_sequence_number*.

**SEQ_NUM5:** When in the *Initializing* or *Connected* state is received a *Sa_DATA.indication (msgtype, data, ackreq, ackresp, seqnum, ecnum)* from the *ER-SL*, the distance of the current message from the last received one is computed as follows*:*

```
distance ≔ last_received_sequence_number – seq_num;
if  (distance <  -M/2) then {distance :=  distance + M };
else if  (distance  >  M/2) then {distance := distance - M };
```

If the distance value is equal to 1, the *seqnum* is considered EXPECTED.
If the distance value is lower than 1, the *seqnum* is considered OLD.
If the distance value is greater than 1 and less or equals to *N*, the *seqnum* is considered ACCEPTABLE.
If the distance value is greater than *N*, the *seqnum* is considered NOT_ACCEPTABLE.

### *ACK Management*

**REQ_ACK1**: When in *Connected* state, the *SAI* periodically (with a configurable *ack_request_period*) sets an *ackreq* flag to the first *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message to be forwarded to the *ER-SL* and starts an *ack_response_timer* with a *max_response_delay* limit.
The *ackreq* flag is not set and the timer is not started if the *SAI* is still waiting for the response to a previous ack request.

**REQ_ACK2**: When the *ack_response_timeout* expires, if a *Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message with an *ackresp* parameter set has not yet been received from the *ER-SL*, the *SAI* sends a *SAI_ERROR.report* to the *CSL* component and restarts the *ack request timer*.

**REQ _ACK3:** While in *Connected* or *Initializing* state, when it is received a *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* from the *CSL* component, the *SAI* sets the *ackresp* parameter in next *Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)* message to be sent to the *ER-SL*.

### *Discarding of Messages*

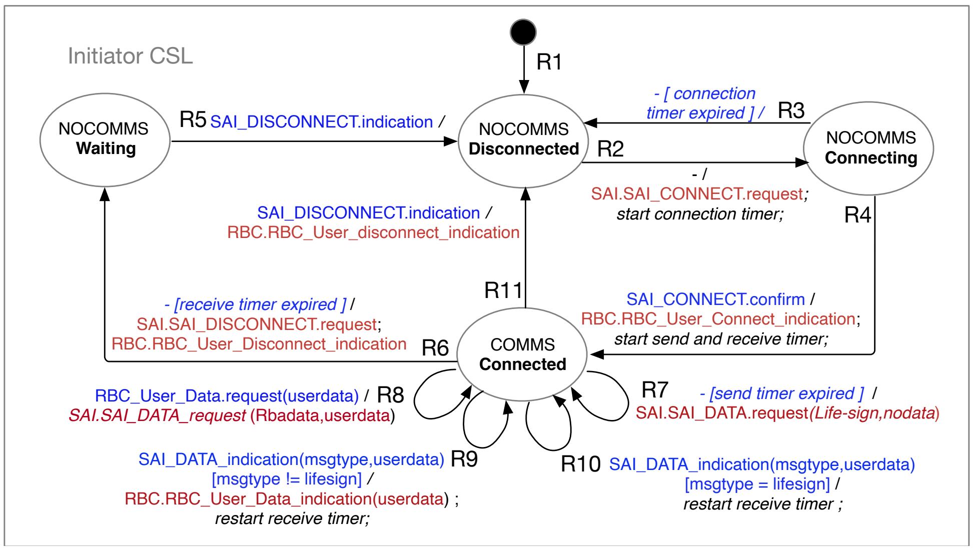
**RD1**: When in *Disconnected* state the *SAI* discards any message except for *Sa_CONNECT.indication* from the *ER-SL*.

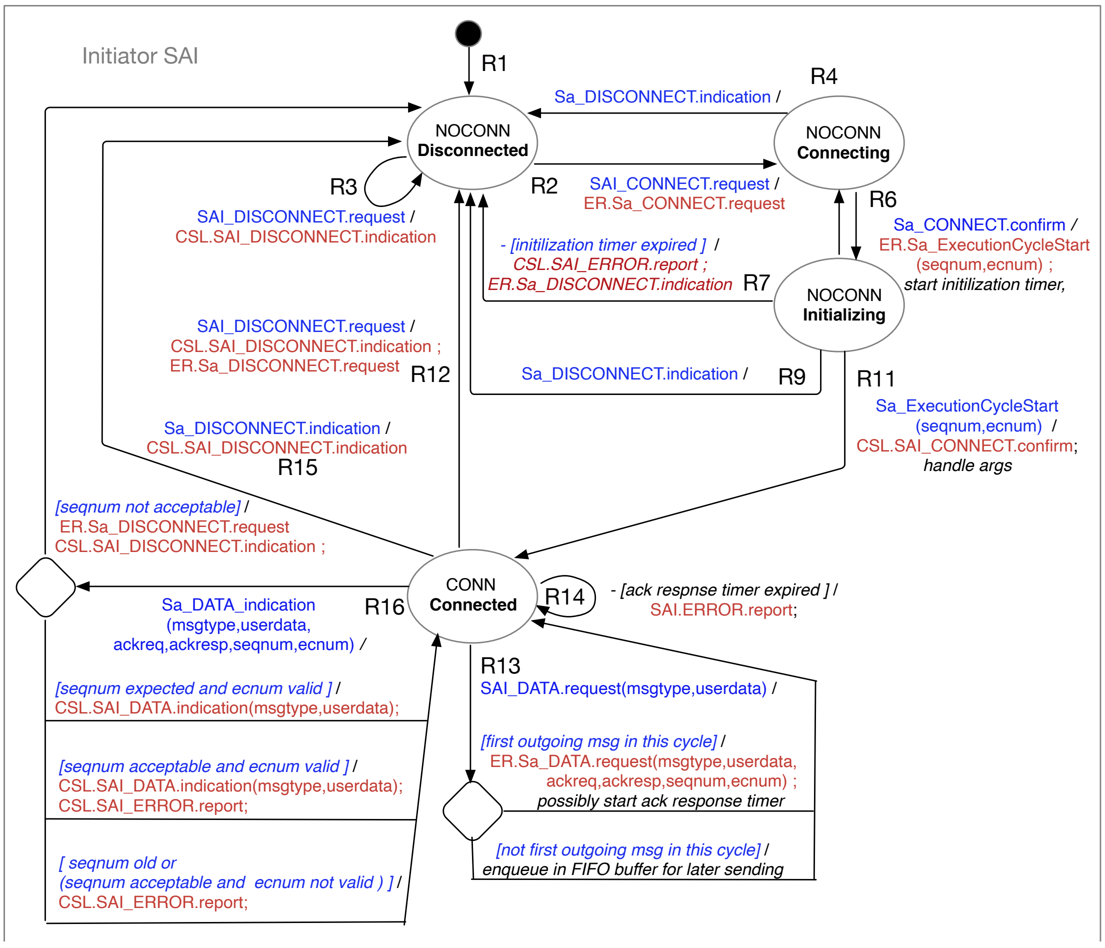
**RD2:** When in *Connecting* state, the *SAI* discards any message except for,
*Sa_DISCONNECT.indication* and *Sa_ExecutionCycleStart* from the *ER-SL*;

**RD3:** When in *Initializing* state, the *SAI* discards any message except for,
*Sa_DISCONNECT.indication* and *Sa_DATA.indication* from the *ER-SL*;

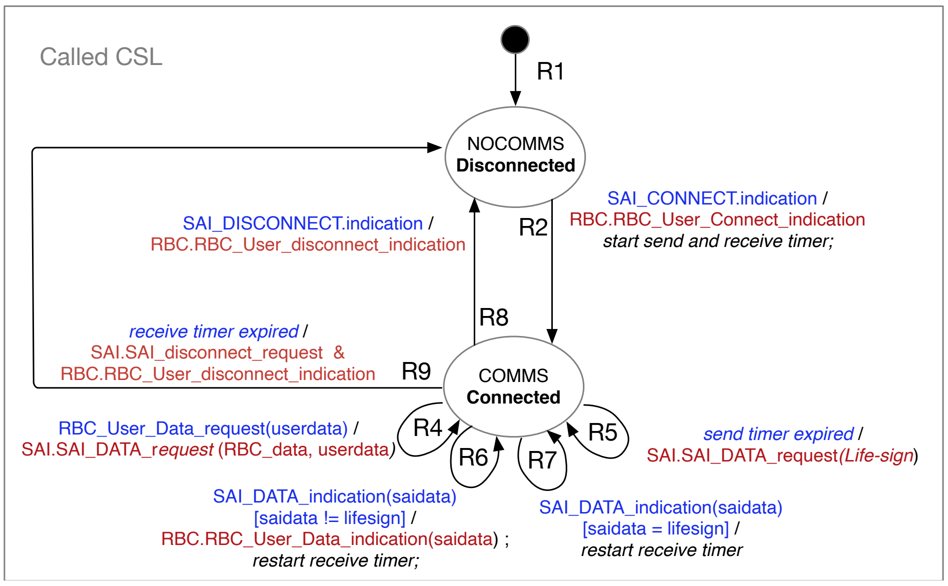
**RD4:** When in *Connected* state, the *SAI* discards any *Sa_ExecutionCycleStart* message from the *ER-SL*.

## 3 Graphical semiformal UML state machine designs

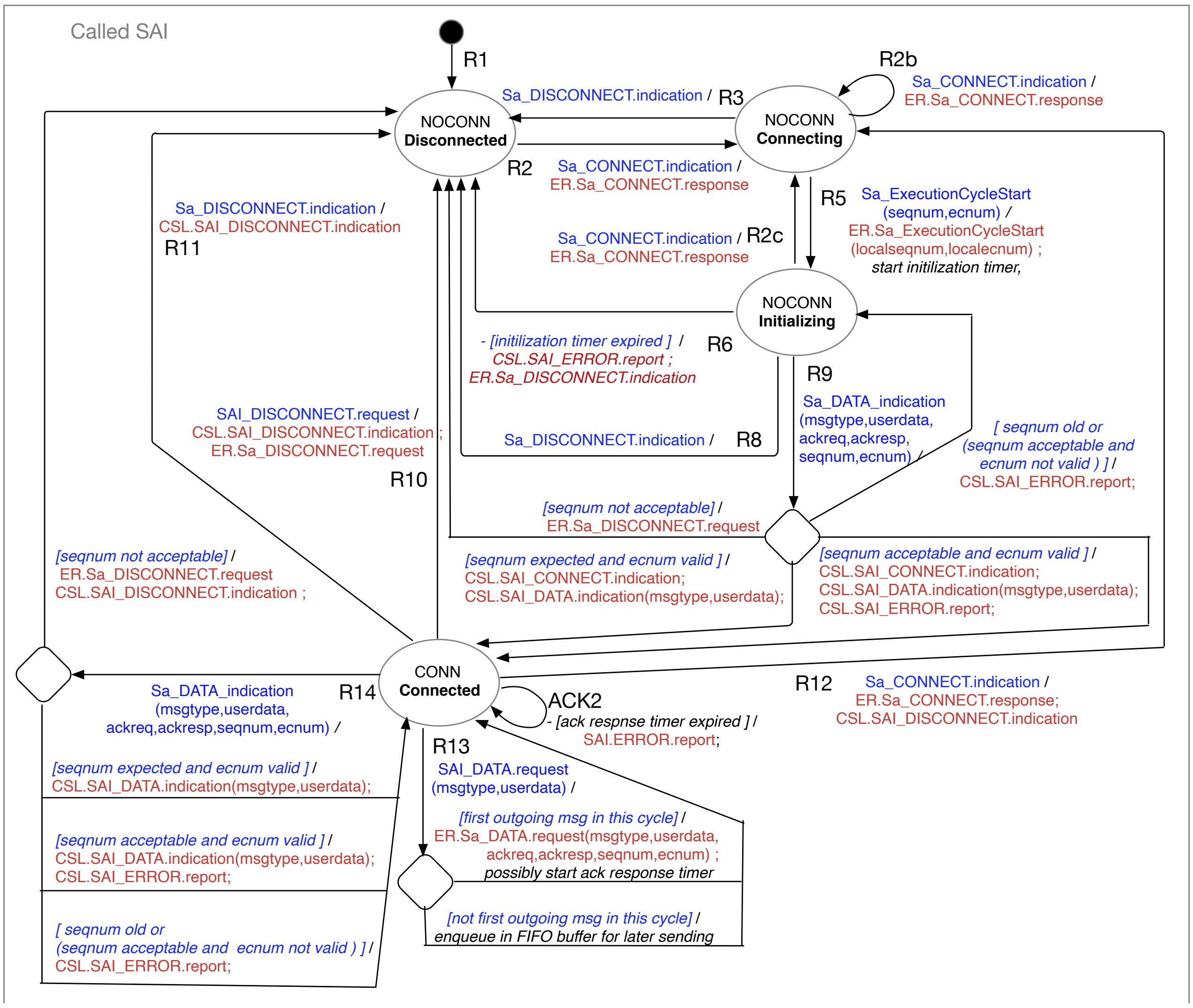

Initiator CSL
R1
NOCOMMS Waiting
R5 SAI_DISCONNECT.indication /
NOCOMMS Disconnected
- [ connection timer expired ] / R3
NOCOMMS Connecting
R2
- / SAI.SAI_CONNECT.request; start connection timer;
R4
SAI_DISCONNECT.indication / RBC.RBC_User_disconnect_indication
R11
- [receive timer expired ] / SAI.SAI_DISCONNECT.request; RBC.RBC_User_Disconnect_indication
R6
COMMS Connected
SAI_CONNECT.confirm / RBC.RBC_User_Connect_indication; start send and receive timer;
RBC_User_Data.request(userdata) / R8 SAI.SAI_DATA_request (Rbadata,userdata)
R7 - [send timer expired ] / SAI.SAI_DATA.request(Life-sign,nodata)
SAI_DATA_indication(msgtype,userdata) R9 [msgtype != lifesign] / RBC.RBC_User_Data_indication(userdata) ; restart receive timer;
R10 SAI_DATA_indication(msgtype,userdata) [msgtype = lifesign] / restart receive timer ;



Initiator SAI
R1
NOCONN Disconnected
R4
Sa_DISCONNECT.indication /
NOCONN Connecting
R3
R2
SAI_CONNECT.request /
ER.Sa_CONNECT.request
R6
SAI_DISCONNECT.request /
CSL.SAI_DISCONNECT.indication
Sa_CONNECT.confirm /
ER.Sa_ExecutionCycleStart
(seqnum,ecnum) ;
start initilization timer,
- [initilization timer expired ] /
CSL.SAI_ERROR.report ;
ER.Sa_DISCONNECT.indication
R7
NOCONN Initializing
SAI_DISCONNECT.request /
CSL.SAI_DISCONNECT.indication ;
ER.Sa_DISCONNECT.request
R12
Sa_DISCONNECT.indication /
R9
R11
Sa_ExecutionCycleStart
(seqnum,ecnum) /
CSL.SAI_CONNECT.confirm;
handle args
Sa_DISCONNECT.indication /
CSL.SAI_DISCONNECT.indication
R15
[seqnum not acceptable] /
ER.Sa_DISCONNECT.request
CSL.SAI_DISCONNECT.indication ;
CONN Connected
R14
- [ack respnse timer expired ] /
SAI.ERROR.report;
Sa_DATA_indication
(msgtype,userdata,
ackreq,ackresp,seqnum,ecnum) /
R16
R13
SAI_DATA.request(msgtype,userdata) /
[seqnum expected and ecnum valid ] /
CSL.SAI_DATA.indication(msgtype,userdata);
[first outgoing msg in this cycle] /
ER.Sa_DATA.request(msgtype,userdata,
ackreq,ackresp,seqnum,ecnum) ;
possibly start ack response timer
[seqnum acceptable and ecnum valid ] /
CSL.SAI_DATA.indication(msgtype,userdata);
CSL.SAI_ERROR.report;
[not first outgoing msg in this cycle] /
enqueue in FIFO buffer for later sending
[ seqnum old or
(seqnum acceptable and ecnum not valid ) ] /
CSL.SAI_ERROR.report;



Called CSL
R1
NOCOMMS Disconnected
SAI_CONNECT.indication /
RBC.RBC_User_Connect_indication
start send and receive timer;
SAI_DISCONNECT.indication /
RBC.RBC_User_disconnect_indication
R2
R8
receive timer expired /
SAI.SAI_disconnect_request &
RBC.RBC_User_disconnect_indication
R9
COMMS Connected
RBC_User_Data_request(userdata) /
SAI.SAI_DATA_request (RBC_data, userdata)
R4
R6
R7
R5
send timer expired /
SAI.SAI_DATA_request(Life-sign)
SAI_DATA_indication(saidata)
[saidata != lifesign] /
RBC.RBC_User_Data_indication(saidata) ;
restart receive timer;
SAI_DATA_indication(saidata)
[saidata = lifesign] /
restart receive timer



Called SAI
R1
R2b
Sa_CONNECT.indication /
ER.Sa_CONNECT.response
Sa_DISCONNECT.indication / R3
NOCONN
Disconnected
NOCONN
Connecting
R2
Sa_CONNECT.indication /
ER.Sa_CONNECT.response
R5
Sa_ExecutionCycleStart
(seqnum,ecnum) /
ER.Sa_ExecutionCycleStart
(localseqnum,localecnum) ;
start initilization timer,
Sa_DISCONNECT.indication /
CSL.SAI_DISCONNECT.indication
R11
Sa_CONNECT.indication / R2c
ER.Sa_CONNECT.response
NOCONN
Initializing
- [initilization timer expired ] /
CSL.SAI_ERROR.report ;
ER.Sa_DISCONNECT.indication
R6
R9
Sa_DATA_indication
(msgtype,userdata,
ackreq,ackresp,
seqnum,ecnum) /
[ seqnum old or
(seqnum acceptable and
ecnum not valid ) ] /
CSL.SAI_ERROR.report;
SAI_DISCONNECT.request /
CSL.SAI_DISCONNECT.indication ;
ER.Sa_DISCONNECT.request
R10
Sa_DISCONNECT.indication / R8
[seqnum not acceptable] /
ER.Sa_DISCONNECT.request
[seqnum not acceptable] /
ER.Sa_DISCONNECT.request
CSL.SAI_DISCONNECT.indication ;
[seqnum expected and ecnum valid ] /
CSL.SAI_CONNECT.indication;
CSL.SAI_DATA.indication(msgtype,userdata);
[seqnum acceptable and ecnum valid ] /
CSL.SAI_CONNECT.indication;
CSL.SAI_DATA.indication(msgtype,userdata);
CSL.SAI_ERROR.report;
CONN
Connected
Sa_DATA_indication
(msgtype,userdata,
ackreq,ackresp,seqnum,ecnum) /
R14
ACK2
- [ack respnse timer expired ] /
SAI.ERROR.report;
R12
Sa_CONNECT.indication /
ER.Sa_CONNECT.response;
CSL.SAI_DISCONNECT.indication
R13
SAI_DATA.request
(msgtype,userdata) /
[seqnum expected and ecnum valid ] /
CSL.SAI_DATA.indication(msgtype,userdata);
[seqnum acceptable and ecnum valid ] /
CSL.SAI_DATA.indication(msgtype,userdata);
CSL.SAI_ERROR.report;
[ seqnum old or
(seqnum acceptable and ecnum not valid ) ] /
CSL.SAI_ERROR.report;
[first outgoing msg in this cycle] /
ER.Sa_DATA.request(msgtype,userdata,
ackreq,ackresp,seqnum,ecnum) ;
possibly start ack response timer
[not first outgoing msg in this cycle] /
enqueue in FIFO buffer for later sending

# 4 Executable UML Model

```
-------------------------------------------------------------------------------
-- VERSION 53  26-07-2021
-------------------------------------------------------------------------------
--
-- ERnice_irbcdata_V53.umc
--
--------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Class I_CSL is
-------------------------------------------------------------------------------

Signals
       -- from RBC
       IRBC_User_Data_request(arg1: int);

       -- from I_SAI
       ISAI_CONNECT_confirm;
       ISAI_DISCONNECT_indication;
       ISAI_Error_report;
       ISAI_DATA_indication(arg1: Token, arg2: int);

       -- from Timer
       icsl_tick;

--     outgoing to RBC
--     IRBC_User_Connect_indication;
--     IRBC_User_Disconnect_indication;
--     IRBC_User_Data_indication(arg1);

--     outgoig to SAI
--     ISAI_CONNECT_request;
--     ISAI_DISCONNECT_request;
--     ISAI_DATA_request(arg1,arg2);

--     outgoig to Timer
--     ok_icsl

Vars
       -------------  PORTS
       RBC_User: I_RBC;
       SAI: I_SAI;
       -------------

       -------------  CONFIGURATION PARAMS
       max_receiveTimer: int;  -- CONFIGURATION PARAM
       max_sendTimer:    int;  -- CONFIGURATION PARAM
       max_connectTimer: int;  -- CONFIGURATION PARAM
       -------------

       -------------  LOCAL VARS
       receiveTimer: int := 0;
       sendTimer:          int := 0;
       connectTimer: int := 0;
       -------------

Behaviour
```

```
---------------------------------------------------
-- R1: At startup, the CSL is in Disconnected state
---------------------------------------------------

R1_ICSL:
initial -> NOCOMMSready

--------------------------------------------------------
-- R2: When in Disconnected state, the CSL immediately
--     sends a SAI_CONNECT.request to the SAI component,
--     starts a connTimer, and moves to Connecting state
--------------------------------------------------------

R2_ICSL_connecting:
       NOCOMMSready -> NOCOMMSconnecting
              {- /
              SAI.ISAI_CONNECT_request;
              connectTimer := 0;}

----------------------------------------------------------
-- R3: When in Connecting state the connTimer expires, the
--     CSL moves to Disconnected state
----------------------------------------------------------

RTa_ICSL_okicsl_incr:
       NOCOMMSconnecting -> NOCOMMSconnecting
              {icsl_tick [connectTimer < max_connectTimer ] /
       Timer.ok_icsl;
        connectTimer := connectTimer +1}

-- while connecting in case of timeout become ready to retry
R3_ICSL_okicsl_connect:
       NOCOMMSconnecting -> NOCOMMSready
              {icsl_tick [connectTimer = max_connectTimer ] /
              Timer.ok_icsl}
------------------------------------------------------------------------
-- R4: When in Connecting state is received a SAI_CONNECT.confirm from
--     the SAI component, the CSL sends a RBC_User_Connect.indication
--     to the RBC component, starts both the sendTimer and the recTimer,
--     and moves to Connected state
------------------------------------------------------------------------

R4_ICSL_userconnind:
       NOCOMMSconnecting -> COMMS
              {ISAI_CONNECT_confirm /
              RBC_User.IRBC_User_Connect_indication;
        connectTimer := max_connectTimer;
        receiveTimer := 0;
              sendTimer := max_sendTimer}

--------------------------------------------------------------------
-- R5: When in Waiting state is received a SAI_DISCONNECT.indication
--     from the SAI component, the CSL moves to Disconnected state
--------------------------------------------------------------------

RTb_ICSL_okicsl_incr:
       NOCOMMSwait -> NOCOMMSwait
              {icsl_tick /
              Timer.ok_icsl}

R5_ICSL_becomeready:
       NOCOMMSwait -> NOCOMMSready
              {ISAI_DISCONNECT_indication}
```

```
---------------------------------------------------------------------
-- R6: When in Connected state the recTimer expires, the CSL sends a
--   SAI_DISCONNECT.request to the SAI component, a
--   RBC_User_Disconnect.indication to the RBC and moves to  Waiting state
---------------------------------------------------------------------

RTc_ICSL_okicsl:
      COMMS -> COMMS
            {icsl_tick [(receiveTimer < max_receiveTimer)
                  and (sendTimer < max_sendTimer)] /
        Timer.ok_icsl;
        sendTimer := sendTimer +1;
        receiveTimer := receiveTimer+1}

R6_ICSL_okicsl:
      COMMS -> tbcr6
            {icsl_tick [receiveTimer = max_receiveTimer] /
       Timer.ok_icsl}

R6_ICSL_saidisconnreq:
      tbcr6 -> tbcr6a
            { - /
      SAI.ISAI_DISCONNECT_request;
      receiveTimer := 0;
            sendTimer := 0}

R6_ICSL_userdisconnind:
      tbcr6a -> NOCOMMSwait
            { - /
            RBC_User.IRBC_User_Disconnect_indication}

----------------------------------------------------------------
-- R7: Each time that in Connected state the sendTimer expires,
--     the CSL sends a SAI_DATA.request with a life_sign to the
--     SAI component
----------------------------------------------------------------

R7_ICSL_okicsl:
      COMMS -> tbcr7
            {icsl_tick [(receiveTimer < max_receiveTimer)
                  and (sendTimer = max_sendTimer)] /
            Timer.ok_icsl;
            sendTimer := 0;
            receiveTimer := receiveTimer+1}

R7_ICSL_saidatareq:
      tbcr7 -> COMMS
      { - /
        SAI.ISAI_DATA_request(LifeSign,0)}

------------------------------------------------------------------
-- R8: When in Connected state is received a RBC_User_Data.request
--     with RBC_data from the RBC component, the CSL sends a
--     SAI_DATA.request with such RBC_data to the SAI component
------------------------------------------------------------------

R8_ICSL_saidatareq:
      COMMS -> COMMS
            {IRBC_User_Data_request(arg1) /
       SAI.ISAI_DATA_request(RBCdata, arg1);
       sendTimer := 0}

----------------------------------------------------------------
-- R9: When in Connected state is received a SAI_DATA.indication with SAI_data
--     from the SAI component, the CSL sends a RBC_User_Data.indication with
--     such SAI_data to the RBC component and restarts the recTimer
----------------------------------------------------------------

R9_ICSL_userdataind:
       COMMS -> COMMS
              {ISAI_DATA_indication(arg1, arg2)
                     [arg1  /= LifeSign] /
        RBC_User.IRBC_User_Data_indication(arg2);
        receiveTimer := 0}

-----------------------------------------------------------------
-- R10: When in Connected state is received a SAI_DATA.indication with a
--      life_sign from the SAI component, the CSL restarts the recTimer
-----------------------------------------------------------------

R10_ICSL_handlelifesign:
       COMMS -> COMMS
              {ISAI_DATA_indication(arg1, arg2)
                     [arg1 = LifeSign] /
              receiveTimer := 0}

-----------------------------------------------------------------
-- R11: When in Connected state is received a SAI_DISCONNECT.indication from
--      the SAI component, the  CSL sends a RBC_User_Disconnect.indication
--      to the RBC component and moves to Disconnected state
-----------------------------------------------------------------

R11_ICSL_userdisconnind:
       COMMS -> NOCOMMSready
              {ISAI_DISCONNECT_indication /
        RBC_User.IRBC_User_Disconnect_indication;
        receiveTimer := 0;
        sendTimer := 0}

------------------------------------------------------------------
-- RD1: When in Disconnected state the CSL does not accept any
--      kind of message
--
-- RD2: When in Connecting state, the CSL discards any message
--      except for SAI_CONNECT.confirm from the SAI component
--
-- RD3: When in Waiting state, the CSL discards any message except
--      for SAI_DISCONNECT.indication from the SAI component
--
-- RD4: When in Connected state, the CSL component discards only
--      SAI_CONNECT.confirm and SAI_ERROR.report messages from
--      the SAI component
------------------------------------------------------------------

RD2a_ICSL_discuserdata:
       NOCOMMSconnecting -> NOCOMMSconnecting
              {IRBC_User_Data_request(arg1)}

RD2b_ICSL_discdisconnind:
       NOCOMMSconnecting -> NOCOMMSconnecting
              {ISAI_DISCONNECT_indication}

RD2c_ICSL_discerrorreport:
       NOCOMMSconnecting -> NOCOMMSconnecting
              {ISAI_Error_report}

RD2d_ICSL_discdataind:
       NOCOMMSconnecting -> NOCOMMSconnecting
```

```
             {ISAI_DATA_indication(arg1, arg2)}

RD3a_ICSL_discuserdata:
       NOCOMMSwait -> NOCOMMSwait
             {IRBC_User_Data_request(arg1)}

RD3b_ICSL_discerrorreport:
       NOCOMMSwait -> NOCOMMSwait
             {ISAI_Error_report}

RD3c_ICSL_discdataind:
       NOCOMMSwait -> NOCOMMSwait
             {ISAI_DATA_indication(arg1, arg2)}

RD3d_ICSL_discconfirm:
       NOCOMMSwait -> NOCOMMSwait
             {ISAI_CONNECT_confirm}

RD4a_ICSL_disccommconfirm:
  COMMS -> COMMS
     {ISAI_CONNECT_confirm}

RD4b_ICSL_usererror:
       COMMS -> COMMS
             {ISAI_Error_report}

end I_CSL;
```

```
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
Class I_SAI is
------------------------------------------------------------------------------
------------------------------------------------------------------------------

Signals

       -- from CSL
       ISAI_CONNECT_request;
       ISAI_DISCONNECT_request;
       ISAI_DATA_request(arg1:int,arg2:int);
       -- arg1 = message type, arg2 = RBC data value

       -- outgoing to CSL
--     ISAI_DISCONNECT_indication;
--     ISAI_CONNECT_confirm;
--     ISAI_DATA_indication(arg1:int,arg2:int);
--     arg1 = message type, arg2 = RBC data value
--     ISAI_Error_report;

       -- from ER
       ISAI_SA_Connect_confirm;
       ISAI_SA_Disconnect_indication;
       ISAI_SA_Data_indication
              (arg1:int,arg2:int,arg3:int,arg4:int,arg5:int,arg6:int);
       -- arg1 = message type,    arg2 = data value,
       -- arg3 = ack request,     arg4 = ack response,
       -- arg5 = sequence number, arg6 = EC number
       ISAI_SA_Execution_Cycle_Start(arg1:int,arg2:int);

       -- outgoing to ER
--     ISAI_SA_Connect_request;
--     ISAI_SA_Disconnect_request;
--     ISAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6);
--       arg1 = message type,      arg2 = data value,
--       arg3 = ack request,       arg4 = ack response,
--       arg5 = sequence number,   arg6 = EC number
--     ISAI_SA_Execution_Cycle(arg1:int,arg2:int);
--       arg1 = sequence number,   arg2 = EC counter

       -- from Timer
       isai_tick;

--     outgoing to Timer
--  ok_isai;

Vars

       -------------  PORTS
       CSL: I_CSL;
       ER: EuroRadio;
       -------------

       ------------- CONFIGURATION PARAMS
       K:                         int;   -- ec delay limit
       max_initTimer:             int;   -- init timeout
       N:                         int;   -- msg loss limit
       M:                         int;   -- seq num limit
       max_ack_requestTimer:      int;   -- ack request period
       max_ack_responseTimer:     int;   -- ack response timeout
       Mec:                       int;   -- ec counter limit
       -------------
```

```
        ------------- LOCAL VARS
        sarg1:                    int := 0;
        sarg2:                    int := 0;
        sarg4:                    int := 0;
        -----------------------
        -- for ECS procedure --
        -----------------------
        initTimer:          int := 0;
        OFFSET:             int := 0;
        EC_expected:  int := 0;
        DELTA:                    int := 0;
        currentEC:          int := 0;
        -------------------------
        -- for sequence number --
        -------------------------
        last_in: int := 0;
        next_out: int := 0;
        dist: int := 0;
        -----------------------
        -- for ACK procedure --
        -----------------------
        ack_requestTimer: int := 0;
        ack_responseTimer: int := 0;
        ack_reply: int := 0;
        ack_request: int := 0;
        -------------
        -- for DATA storing
     outdatabuff:  int[]  := [];
     waitnextcycle: bool := False;

Behaviour

-------------------------------------------------------------------
-- R1: At startup the SAI is in Disconnected state
-------------------------------------------------------------------

R1_ISAI:
       initial -> Disconnected

-------------------------------------------------------------------
-- R2: When in Disconnected state is received a SAI_CONNECT.request
--     from the CSL component, the SAI sends a Sa_CONNECT.request
--     to the ER-SL and moves to Connecting state
-------------------------------------------------------------------

R2_ISAI_connreq:
       Disconnected -> Connecting
              {ISAI_CONNECT_request /
              ER.ISAI_SA_Connect_request}

----------------------------------------------------------------------
-- R3: When in Disconnected state is received a SAI_DISCONNECT.request
--     from the CSL component, the SAI replies with a
--     SAI_DISCONNECT.indication to the CSL component.
--     This is necessary to avoid deadlocks in the CSL component, when the CSL is
--     NOCOMM_waiting state.
----------------------------------------------------------------------

R3_ISAI_disconndiscard:
       Disconnected -> Disconnected
              {ISAI_DISCONNECT_request /
                CSL.ISAI_DISCONNECT_indication}
```

```
----------------------------------------------------------------------
-- R4: When in Connecting state is received a Sa_DISCONNECT.indication
--     from the ER-SL, the SAI moves to Disconnected state
----------------------------------------------------------------------

RTg_ISAI_okisai:
       Connecting -> Connecting
              {isai_tick /
              Timer.ok_isai}

R4_ISAI_abortconn:
       Connecting -> Disconnected
              {ISAI_SA_Disconnect_indication}

--------------------------------------------------------------------
-- R5: When in Connecting state is received a SAI_DISCONNECT.request
--     the message is discarded (it should never happen)
--------------------------------------------------------------------

R5_ISAI_CSLdisconnind:
       Connecting -> Connecting
              {ISAI_DISCONNECT_request}

-------------------------------------------------------------------
-- R6: When in Connecting state is received a Sa_CONNECT.confirm from the ER-SL,
--     the SAI replies with a Sa_ExecutionCycle(seqnum, ecnum) to the ER-SL and
--     moves to the Initializing state, waiting for a Sa_ExecutionCycleStart
--     message from the ER-SL within a maximum initialization delay.
--     For the management of the Sa_ExecutionCycleStart parameters see the
--     SEQ_NUMs and REQ_ECNUMs Management sections
-------------------------------------------------------------------

R6_ISAI_confinitproc:
       Connecting -> Initializing
              {ISAI_SA_Connect_confirm /
              ER.ISAI_SA_Execution_Cycle(next_out,currentEC);
              next_out := (next_out + 1) mod M;
              initTimer := 0}

-------------------------------------------------------------------
-- R7: When in Initializing state the maximum initialization delay expires,
--     the SAI sends an SAI_ERROR.report to the CSL component, a
--     Sa_DISCONNECT.request to the ER-SL and moves to Disconnected state
-------------------------------------------------------------------

RTa_ISAI_confinitprocwait:
       Initializing -> tbcrt
              {isai_tick /
              Timer.ok_isai;
              initTimer := initTimer + 1;
              currentEC := (currentEC + 1) mod Mec}

R7_ISAI_confinitprocwaiterrorr:
       tbcrt -> tbcr7a
              { - [initTimer = max_initTimer] /
              CSL.ISAI_Error_report}

R7_ISAI_confinitprocwaitdisconn:
       tbcr7a -> Disconnected
              { - / ER.ISAI_SA_Disconnect_request;
              initTimer := 0;
              currentEC := 0}

RTb_ISAI_connreqdiscard:
      tbcrt -> Initializing
            { - [initTimer < max_initTimer]}

-------------------------------------------------------------------
-- R8: When in Initializing state is received a SAI_DISCONNECT.request from the
--     CSL component, the message is discarded (it should never happen).
-------------------------------------------------------------------

R8_ISAI_CSLdisconnind:
      Initializing -> Initializing
            {ISAI_DISCONNECT_request / }

-------------------------------------------------------------------
-- R9: When in Initializing state is received a Sa_DISCONNECT.indication
--     from the ER-SL the SAI moves to Disconnected state
-------------------------------------------------------------------

R9_ISAI_abortconn:
      Initializing -> Disconnected
            {ISAI_SA_Disconnect_indication /
            initTimer := 0;
            currentEC := 0}

-------------------------------------------------------------------
-- R10: When in Initializing state is received a SAI_CONNECT.request from
--      the CSL component, the message is discarded
-------------------------------------------------------------------

R10_ISAI_connreqdiscard:
      Initializing -> Initializing
            {ISAI_CONNECT_request}

-------------------------------------------------------------------
-- R11: When in Initializing state is received a
--   Sa_ExecutionCycleStart(seqnum, ecnum) from the ER-SL, the SAI sends
--   a SAI_CONNECT.confirm to the CSL component and moves to Connected state.
--   The received seqnum is accepted as initial remote sequence number and the
--   ecnum is accepted as initial value of the remote execution cycle counter.
--   The execution_cycle_OFFSET variable is set as the difference between the
--   current execution cycle counter and the received execution cycle counter.
--   While the last_received_sequence_number variable is set to the received
--   sequence number
-------------------------------------------------------------------

R11_ISAI_confinitproc:
      Initializing -> Connected
            {ISAI_SA_Execution_Cycle_Start(arg1,arg2) /
            CSL.ISAI_CONNECT_confirm;
            OFFSET := currentEC - arg2;
            initTimer := 0;
            ack_requestTimer := 0;
            ack_responseTimer := max_ack_responseTimer + 1;
            ack_reply := 0;
            last_in := arg1}

-------------------------------------------------------------------
-- R12: When in Connected state is received a SAI_DISCONNECT.request from the
--     CSL component,the SAI replies with a SAI_DISCONNECT.indication to the
--    CSL component, sends a Sa_DISCONNECT.request to the ER-SL, and moves
--    to Disconnected state
-------------------------------------------------------------------

R12_ISAI_CSLdisconnind:
      Connected -> tbcr12
```

```
                {ISAI_DISCONNECT_request /
                CSL.ISAI_DISCONNECT_indication}

R12_ISAI_CSLdisconnreq:
        tbcr12 -> Disconnected
                { - /
                ER.ISAI_SA_Disconnect_request;
                ack_requestTimer := max_ack_requestTimer;
                ack_responseTimer := 0;
                ack_reply := 0;
                currentEC := 0;
                outdatabuff := [];
                ack_request := 0;
                waitnextcycle := False}

-------------------------------------------------------------------
-- R13a: When in Connected state is received a SAI_DATA_request(msgtype, data)
--    from the CSL component, and yet no other data message has been sent in
--    this cycle, the SAI sends a
--    Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)
--    to the ER-SL. The ackreq  and  ackresp parameters are set  according to
--    REQ_ACKs.
--    The seqnum parameter isset according to SEQ_NUMs and the ecnum parameter
--     is set according to REQ_ECNUMs Management
-------------------------------------------------------------------

R13a_ISAI_datareqforward:
        Connected -> Connected
                {ISAI_DATA_request(arg1,arg2)
                        [waitnextcycle = False]/

        ER.ISAI_SA_Data_request(arg1,arg2,ack_request,ack_reply,next_out,currentEC);
                next_out := (next_out + 1) mod M;
                if (ack_request = 1)
                   {ack_request := 0;
                        ack_requestTimer := 0;
                        ack_responseTimer := 0};
                ack_reply := 0;
                waitnextcycle := True}

-------------------------------------------------------------------
-- R13b: When in Connected state is received a SAI_DATA.request(msgtype, data)
--       from the CSL component,but another data message has already been
--       sent in this cycle, the SAI_DATA.request is  saved in a FIFO dataout
--       buffer (see also the REQ_OUTDATABUFFs Management)
-------------------------------------------------------------------

R13b_ISAI_datareqstore:
        Connected -> Connected
                {ISAI_DATA_request(arg1,arg2)
                        [waitnextcycle = True]/
                        outdatabuff := outdatabuff + [arg1,arg2]}

-------------------------------------------------------------------
-- R14: When in Connected state is received a Sa_DISCONNECT.indication from
--      the ER-SL, the SAI sends a SAI_DISCONNECT.indication to the CSL
--      component and moves to Disconnected state
-------------------------------------------------------------------

R14_ISAI_abortconn:
        Connected -> Disconnected
                {ISAI_SA_Disconnect_indication /
                CSL.ISAI_DISCONNECT_indication;
                ack_requestTimer := max_ack_requestTimer;
```

```
             ack_responseTimer := 0;
             ack_reply := 0;
             outdatabuff := [];
             ack_request := 0;
             waitnextcycle := False}

--------------------------------------------------------------------
-- R15: When in Connected state is received a
--     Sa_DATA.indication(msgtype, data, ackreq, ackresp, seqnum, ecnum) from
--     the ER-SL we can have four cases, depending on the received seqnum
--     and ecnum values (see SEQ_NUMs and REC_ECNUMs Management): *
--------------------------------------------------------------------

R15_ISAI_datareceive:
       Connected -> tbcr15
             {ISAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6) /
             dist := arg5 - last_in;
             if  (dist < -M/2) then {dist := dist + M };
             if  (dist >  M/2) then {dist := dist - M };
             if ((dist >  0) and (dist <= N)) then
                {last_in := arg5};
             sarg1 := arg1;
             sarg2 := arg2;
             ack_reply := ack_reply + arg3 - ack_reply*arg3;
             sarg4 := arg4;
             EC_expected := (currentEC + Mec - OFFSET) mod Mec;
             DELTA := EC_expected - arg6;
             if  (DELTA < -Mec/2) then {DELTA := DELTA + Mec };
             if  (DELTA >  Mec/2) then {DELTA := DELTA - Mec };
             EC_expected := 0}

--------------------------------------------------------------------
--    Case 1
-- *  The seqnum is the one EXPECTED and ecnum is VALID: In this case the
--    SAI sends a SAI_DATA.indication(msgtype, data) to the CSL component
--------------------------------------------------------------------

R15a_ISAI_dataindforward:
       tbcr15 -> Connected
             { - [dist = 1 and DELTA < K] /
             CSL.ISAI_DATA_indication(sarg1,sarg2);
             if((sarg4 = 1) and (ack_responseTimer < max_ack_responseTimer))
                    {ack_responseTimer := max_ack_responseTimer + 1};
             sarg1 := 0;
             sarg2 := 0;
             sarg4 := 0;
             dist  := 0;
             DELTA := 0}

--------------------------------------------------------------------
--    Case 2
-- *  The seqnum is ACCEPTABLE and the ecnum is VALID: in this case the
--    SAI sends a SAI_DATA.indication(msgtype, data) and a SAI_ERROR.report
--    to the CSL component
--------------------------------------------------------------------

R15b_ISAI_dataindforward:
       tbcr15 -> tbcr15b
             { - [dist > 1 and dist <= N and DELTA < K] /
             CSL.ISAI_DATA_indication(sarg1,sarg2);
             if((sarg4 = 1) and (ack_responseTimer < max_ack_responseTimer))
                    {ack_responseTimer := max_ack_responseTimer + 1};
             sarg1 := 0;
             sarg2 := 0;
```

```
                sarg4 := 0;
                dist  := 0;
                DELTA := 0}

R15b_ISAI_datainderror:
        tbcr15b -> Connected
                { - /
                CSL.ISAI_Error_report;
                sarg1 := 0;
                sarg2 := 0;
                sarg4 := 0;
                dist  := 0;
                DELTA := 0}

--------------------------------------------------------------------
--    Case 3
-- *  The seqnum is OLD or (the seqnum is ACCEPTABLE and the ecnum is VALID):
--     In this case the SAI sends a SAI_ERROR.report to the CSL component and
--    discards the Sa_DATA.indication message
--------------------------------------------------------------------

R15c_ISAI_datainvalid_or_old:
        tbcr15 -> Connected
                { - [dist < 1 or (dist <= N and DELTA >= K)] /
                CSL.ISAI_Error_report;
                sarg1 := 0;
                sarg2 := 0;
                sarg4 := 0;
                dist  := 0;
                DELTA := 0}

--------------------------------------------------------------------
--    Case 4
-- *  The seqnum is NOT_ACCEPTABLE: In this case the SAI component sends a
--    Sa_DISCONNECT.request to ER-SL and a SAI_DISCONNECT.indication to the CSL
--    component, and then moves to Disconnected state
--------------------------------------------------------------------

R15d_ISAI_abortconn:
        tbcr15 -> tbcr15d
                { - [dist > N] /
                ER.ISAI_SA_Disconnect_request}

R15d_ISAI_notifyfatal:
        tbcr15d -> Disconnected
                { - /
                CSL.ISAI_DISCONNECT_indication;
                ack_responseTimer := 0;
                sarg1 := 0;
                sarg2 := 0;
                sarg4 := 0;
                next_out :=0;
                dist  := 0;
                DELTA := 0;
                outdatabuff := [];
                ack_request := 0;
                waitnextcycle := False}

RTf_ISAI_okisai:
        Disconnected -> Disconnected
                { isai_tick /
                Timer.ok_isai}

RD1a_ISAI_datadiscard:
        Disconnected -> Disconnected
                {ISAI_DATA_request(arg1,arg2)}

RD1b_ISAI_connconfdiscard:
        Disconnected -> Disconnected
                {ISAI_SA_Connect_confirm}

RD1c_ISAI_discinddiscard:
        Disconnected -> Disconnected
                {ISAI_SA_Disconnect_indication}

RD1d_ISAI_datainddiscard:
        Disconnected -> Disconnected
                {ISAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6)}

RD1e_ISAI_disconndiscard:
        Disconnected -> Disconnected
                {ISAI_SA_Execution_Cycle_Start(arg1,arg2)}

RD2a_ISAI_datainddiscard:
        Connecting -> Connecting
                {ISAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6)}

RD2b_ISAI_datareqdiscard:
        Connecting -> Connecting
                {ISAI_DATA_request(arg1,arg2)}

RD2c_ISAI_connreqdiscard:
        Connecting -> Connecting
                {ISAI_CONNECT_request}

RD2d_ISAI_ecsdiscard:
        Connecting -> Connecting
                {ISAI_SA_Execution_Cycle_Start(arg1,arg2)}

RD3a_ISAI_datainddiscard:
        Initializing -> Initializing
                {ISAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6)}

RD3b_ISAI_datareqdiscard:
        Initializing -> Initializing
                {ISAI_DATA_request(arg1,arg2)}

RD3c_ISAI_connconfdiscard:
        Initializing -> Initializing
                {ISAI_SA_Connect_confirm}

RD4a_ISAI_connconfdiscard:
        Connected -> Connected
                {ISAI_SA_Connect_confirm}

RD4b_ISAI_connreqdiscard:
        Connected -> Connected
                {ISAI_CONNECT_request}

RD4c_ISAI_ecsdiscard:
        Connected -> Connected
                {ISAI_SA_Execution_Cycle_Start(arg1,arg2)}

-------------------------------------------------------------------
--  Each time that in Connected state the set_ack_response expires,
--  the SAI sends a SAI_ERROR.report to the CSL component
--  (for further details see REQ_ACKs)
-------------------------------------------------------------------
```

```
RTc_ISAI_okisai:
       Connected -> tbcrta
              {isai_tick /
              Timer.ok_isai;
              ------ ack timing management
              if (ack_responseTimer < max_ack_responseTimer)
                     {ack_responseTimer := ack_responseTimer + 1};
              if (ack_requestTimer < max_ack_requestTimer)
                     {ack_requestTimer := ack_requestTimer + 1};
              if (ack_requestTimer = max_ack_requestTimer and
                            ack_responseTimer >= max_ack_responseTimer)
                  {ack_request := 1};
           ------ ec timing management
              currentEC := (currentEC + 1) mod Mec;
              ------
              waitnextcycle := False}

RACK2_ISAI_ackresponserror:
       tbcrta -> tbcrack
              { - [ack_responseTimer = max_ack_responseTimer] /
                     CSL.ISAI_Error_report;
                     ack_responseTimer := max_ack_responseTimer + 1}

ROUTDATABUFF_ISAI_sendbuffered:
       tbcrack -> Connected
              { - [ outdatabuff /= [] ] /
              ER.ISAI_SA_Data_request(
                 outdatabuff.head, outdatabuff.tail.head,
                 ack_request,ack_reply,next_out,currentEC);
              outdatabuff :=  outdatabuff.tail.tail;
              waitnextcycle := True;
              next_out := (next_out + 1) mod M;
              if (ack_request = 1)
                 {ack_request := 0;
                     ack_requestTimer := 0;
                     ack_responseTimer := 0};
              ack_reply := 0;
              }

RTd_ISAI_continue:
       tbcrta -> tbcrack
              { - [ack_responseTimer /= max_ack_responseTimer] }

RTe_ISAI_okisai:
       tbcrack -> Connected
         { - [ outdatabuff = [] ]}

end I_SAI;
```

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Class C_SAI is
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Signals

   -- from ER
   CSAI_SA_Connect_indication;
   CSAI_SA_Disconnect_indication;
   CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6:int);
   --  arg1 = message type,    arg2 = data value,
   --  arg3 = ack request,     arg4 = ack response,
   --  arg5 = sequence number, arg6 = EC number

   CSAI_SA_Execution_Cycle_Start(arg1:int,arg2:int);
   --  arg1 = sequence number,   arg2 = EC counter

   -- from CSL
   CSAI_DISCONNECT_request;
   CSAI_DATA_request(arg1:int,arg2:int);
   -- arg1 = message type, arg2 = RBC data value

   -- from Timer
   csai_tick;

-- outgoing to ER
-- CSAI_SA_Connect_response;
-- CSAI_SA_Disconnect_request;
-- CSAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6);
--     arg1 = message type,    arg2 = data value,
--     arg3 = ack request,     arg4 = ack response,
--     arg5 = sequence number, arg6 = EC number
-- CSAI_SA_Execution_Cycle(arg1,arg2);
--     arg1 = sequence number,   arg2 = EC counter

   -- outgoing to CSL
-- CSAI_CONNECT_indication;
-- CSAI_DISCONNECT_indication;
-- CSAI_DATA_indication(arg1:int,arg2:int);
--             arg1 = message type, arg2 = RBC data value
-- CSAI_Error_report;

-- outgoing to Timer
-- ok_csai

Vars
   ------------- PORTS
   CSL: C_CSL;
   ER: EuroRadio;
   -------------

   ------------- CONFIGURATION PARAMS
   K:                             int;   -- ec delay limit
   max_initTimer:                 int;   -- init timeout
   N:                             int;   -- msg loss limit
   M:                             int;   -- seq num limit
   max_ack_requestTimer:   int;   -- ack request period
   max_ack_responseTimer:  int;   -- ack response timeout
   Mec:                           int;   -- ec counter limit
   -------------

   ------------- LOCAL VARS
   sarg1:                  int := 0;
   sarg2:                   int := 0;
   sarg3:                   int := 0;
   sarg4:                   int := 0;
   -----------------------
   -- for ECS procedure --
   -----------------------
   initTimer:               int := 0;
   OFFSET:                  int := 0;
   EC_expected:             int := 0;
   DELTA:                   int := 0;
   currentEC:               int := 0;
   -------------------------
   -- for sequence number --
   -------------------------
   last_in:                 int := 0;
   next_out:         int := 0;
   dist:                    int := 0;
   -----------------------
   -- for ACK procedure --
   -----------------------
   ack_requestTimer:        int := 0;
   ack_responseTimer:       int := 0;
   ack_reply:               int := 0;
   ack_request:             int := 0;
   -------------
   -- for DATA storing
     outdatabuff:  int[]  := [];
     waitnextcycle: bool := False;

Behaviour

--------------------------------------------------------------------
-- R1: At startup, the SAI is in Disconnected state
--------------------------------------------------------------------
R1_CSAI_startup:
initial -> Disconnected

--------------------------------------------------------------------
-- R2: When in Disconnected state is received a Sa_CONNECT.indication from
--      the ER-SL, the SAI replies with a Sa_CONNECT.response to the ER-SL
--      and moves to Connecting state
--------------------------------------------------------------------

RTf_CSAI_okcsai:
   Disconnected -> Disconnected
       { csai_tick /
       Timer.ok_csai}

R2_CSAI_connreq:
   Disconnected -> Connecting
       {CSAI_SA_Connect_indication /
       ER.CSAI_SA_Connect_response}

--------------------------------------------------------------------
-- R3: When in Connecting state is received a Sa_DISCONNECT.indication from
--      the ER-SL, the SAI moves to Disconnected state
--------------------------------------------------------------------

RTg_CSAI_okcsai:
   Connecting -> Connecting
       {csai_tick /
       Timer.ok_csai}

R3_CSAI_abortconn:
   Connecting -> Disconnected
       {CSAI_SA_Disconnect_indication }

--------------------------------------------------------------------
-- R4: When in Connecting state is received a SAI_DISCONNECT.request from the
--     CSL component, the message is discarded (it should never happen).
--------------------------------------------------------------------

R4_CSAI_CSLdisconnind:
   Connecting -> Connecting
       {CSAI_DISCONNECT_request}

--------------------------------------------------------------------
-- R2b: When in Connecting state is received a SAI_SA_Connect.request from the
--     ER sublayer, the connection process is restarted.
--------------------------------------------------------------------
R2b_CSAI_restartconnect:
   Connecting -> Connecting
       {CSAI_SA_Connect_indication /
       ER.CSAI_SA_Connect_response}

--------------------------------------------------------------------
-- R5: When in Connecting state is received a
--     Sa_ExecutionCycleStart(seqnum, ecnum)
--     from the ER-SL, the SAI replies with a Sa_ExecutionCycle(seqnum, ecnum)
--     to the ER-SL, starts an initTimer set to the maximum initialization
--     delay, and moves to Initializing state. For the management of the
--     Sa_ExecutionCycleStart parameters see the following Sequence Numbers
--     Management and Execution Cycle Counters Management sections
--------------------------------------------------------------------

R5_CSAI_confinitproc:
   Connecting -> Initializing
       {CSAI_SA_Execution_Cycle_Start(arg1,arg2) /
       ER.CSAI_SA_Execution_Cycle(next_out,currentEC);
       OFFSET := currentEC - arg2;
       initTimer := 0;
       last_in := arg1;
       next_out := (next_out +1) mod M}

--------------------------------------------------------------------
-- R6: When in Initializing state the maximum initialization delay expires, the
--     SAI sends a SAI_ERROR.report to the CSL component, Sa_Disconnect_request
--     to ER, and moves to Disconnected state
--------------------------------------------------------------------

RTa_CSAI_confinitprocwait:
   Initializing -> tbcrta
       {csai_tick /
       Timer.ok_csai;
       initTimer := initTimer + 1;
       currentEC := (currentEC + 1) mod Mec}

R6_CSAI_confinitfail:
   tbcrta -> tbcr6
       { - [initTimer = max_initTimer] /
       ER.CSAI_SA_Disconnect_request }

R6_CSAI_confinitprocwait:
   tbcr6  -> Disconnected
       { - /
        CSL.CSAI_Error_report;
       ack_requestTimer := max_ack_requestTimer;
       ack_responseTimer := 0;
```

```
        ack_reply := 0;
        ack_request := 0;
        currentEC := 0;
        initTimer  := 0;
        OFFSET := 0;
        -- EC_expected := 0;
        DELTA := 0;
        dist := 0;
        last_in := 0;
        next_out := 0;
        outdatabuff := [];
        waitnextcycle := False}

-- when initializing wait first data
RTb_CSAI_waitdata:
   tbcrta -> Initializing
        { - [initTimer < max_initTimer] }


-------------------------------------------------------------------
-- R2c: When in Connecting state is received a SAI_SA_Connect.request from the
--     ER sublayer, the connection process is restarted.
-------------------------------------------------------------------
R2c_CSAI_restartconnect:
   Initializing -> Connecting
        {CSAI_SA_Connect_indication /
        ER.CSAI_SA_Connect_response;
        ack_requestTimer := max_ack_requestTimer;
        ack_responseTimer := 0;
        ack_reply := 0;
        ack_request := 0;
        currentEC := 0;
        initTimer  := 0;
        OFFSET := 0;
        -- EC_expected := 0;
        DELTA := 0;
        dist := 0;
        last_in := 0;
        next_out := 0}


-------------------------------------------------------------------
-- R7: When in Initializing state is received a SAI_DISCONNECT.request from the
--     CSL component, the message is discarded (it should never happen)
-------------------------------------------------------------------

R7_CSAI_CSLdisconnind:
   Initializing -> Initializing
        {CSAI_DISCONNECT_request }

-------------------------------------------------------------------
-- R8: When in Initializing state is received a Sa_DISCONNECT.indication from
--   the ER-SL, the SAI  moves to the Disconnected state
-------------------------------------------------------------------

R8_CSAI_abortconn:
   Initializing -> Disconnected
        {CSAI_SA_Disconnect_indication /
        ack_requestTimer := max_ack_requestTimer;
        ack_responseTimer := 0;
        ack_reply := 0;
        ack_request := 0;
        currentEC := 0;
        initTimer  := 0;
        OFFSET := 0;
        -- EC_expected := 0;
        DELTA := 0;
        dist := 0;
        last_in := 0;
        next_out := 0;
        outdatabuff := [];
        waitnextcycle := False}

------------------------------------------------------------------
-- R9: When in Initializing state is received a Sa_DATA.indication
--     from the ER-SL may have four cases, depending on the received seqnum
--    and ecnum values (see REQ SEQ_NUMs and REC_ECNUMs): *
------------------------------------------------------------------

R9_CSAI_confinitproc:
   Initializing -> tbcr9
       {CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6) /
        dist := arg5 - last_in;
        if (dist < -M/2) then {dist := dist + M };
        if (dist >  M/2) then {dist := dist - M };
        if ((dist > 0) and (dist <= N)) then
               {last_in := arg5};
        sarg1 := arg1;
        sarg2 := arg2;
        sarg3 := arg3;
        EC_expected := (currentEC + Mec - OFFSET) mod Mec;
        DELTA := EC_expected - arg6;
        if  (DELTA < -Mec/2) then {DELTA := DELTA + Mec };
        if  (DELTA >  Mec/2) then {DELTA := DELTA - Mec };
        EC_expected := 0}


------------------------------------------------------------------
--    Case 1
--  The seqnum is the one EXPECTED and ecnum is VALID : In this case the SAI
--   moves to Connected state and sends both a SAI_CONNECT.indication and a
--   SAI_DATA.indication(msgtype, data) to the CSL component
------------------------------------------------------------------

R9a_CSAI_connecting:
     tbcr9 -> tbcr9a
       { - [dist = 1 and DELTA < K] /
        CSL.CSAI_CONNECT_indication;
        ack_reply := sarg3;
        sarg3 := 0;
        initTimer := 0 ;
        dist  := 0;
        DELTA := 0}

R9a_CSAI_forwarding:
     tbcr9a  -> Connected
       { - /
        CSL.CSAI_DATA_indication(sarg1,sarg2);
        ack_requestTimer := 0;
        ack_responseTimer := max_ack_responseTimer + 1;
        sarg1 := 0;
        sarg2 := 0}

------------------------------------------------------------------
--   Case 2
-- The seqnum is ACCEPTABLE and the ecnum is VALID: in this case the SAI moves
--  to Connected state and sends a SAI_CONNECT.indication, a
--   SAI_DATA.indication
--   and a SAI_ERROR.report to the CSL component
-------------------------------------------------------------------

R9b_CSAI_connecting:
     tbcr9 -> tbcr9b
       { - [dist > 1 and dist <= N and DELTA < K] /
      CSL.CSAI_CONNECT_indication;
      ack_reply := sarg3;
      sarg3 := 0;
      initTimer := 0 ;
      dist  := 0;
      DELTA := 0}

R9b_CSAI_forwarding:
     tbcr9b -> tbcr9ba
       { - /
      CSL.CSAI_DATA_indication(sarg1,sarg2);
      sarg1 := 0;
      sarg2 := 0}

R9b_CSAI_reportingding:
     tbcr9ba -> Connected
       { - /
      CSL.CSAI_Error_report;
      ack_requestTimer := 0;
      ack_responseTimer := max_ack_responseTimer + 1}

-------------------------------------------------------------------
--   Case 3
-- The seqnum is OLD or (the seqnum is ACCEPTABLE and the ecnum is VALID):
--  In this case the SAI sends a SAI_ERROR.report to the CSL component and
--  discards the Sa_DATA.indication message
-------------------------------------------------------------------

R9c_CSAI_discarding:
     tbcr9 -> Initializing
      { - [dist < 1 or (dist <= N and DELTA >= K)] /
      CSL.CSAI_Error_report;
      sarg1 := 0;
      sarg2 := 0;
      sarg3 := 0;
      dist  := 0;
      DELTA := 0}

-------------------------------------------------------------------
--   Case 4
-- * The seqnum is NOT_ACCEPTABLE: in this case the SAI component sends a
--   Sa_DISCONNECT.request  to ER-SL and  moves to Disconnected state
-------------------------------------------------------------------

R9d_CSAI_disconnecting:
     tbcr9 -> Disconnected
      { - [dist > N] /
      ER.CSAI_SA_Disconnect_request;
      sarg1 := 0;
      sarg2 := 0;
      sarg3 := 0;
      ack_requestTimer := max_ack_requestTimer;
      ack_responseTimer := 0;
      ack_reply := 0;
      ack_request := 0;
      currentEC := 0;
      initTimer  := 0;
      OFFSET := 0;
```

```
        -- EC_expected := 0;
        DELTA := 0;
        dist := 0;
        last_in := 0;
        next_out := 0;
        outdatabuff := [];
        waitnextcycle := False}

------------------------------------------------------------------
-- R10: When in Connected state is received a SAI_DISCONNECT.request from the
--   CSL component, the SAI replies with   a SAI_DISCONNECT.indication to the
--   CSL component, sends a Sa_DISCONNECT.request to the ER-SL, and moves to
--   Disconnected state
------------------------------------------------------------------

R10_CSAI_CSLdisconnind:
   Connected -> tbcr10
        {CSAI_DISCONNECT_request /
        CSL.CSAI_DISCONNECT_indication}

R10_CSAI_CSLdisconnreq:
   tbcr10 -> Disconnected
        { - /
        ER.CSAI_SA_Disconnect_request;
        ack_requestTimer := max_ack_requestTimer;
        ack_responseTimer := 0;
        ack_reply := 0;
        ack_request := 0;
        currentEC := 0;
        initTimer  := 0;
        OFFSET := 0;
        -- EC_expected := 0;
        DELTA := 0;
        dist := 0;
        last_in := 0;
        next_out := 0;
        outdatabuff := [];
        waitnextcycle := False}

------------------------------------------------------------------
-- R11: When in Connected state is received a Sa_DISCONNECT.indication from the
--     ER-SL, the SAI sends a SAI_DISCONNECT.indication to the CSL component
--     and moves to Disconnected state
------------------------------------------------------------------

R11_CSAI_abortconn:
   Connected -> Disconnected
        {CSAI_SA_Disconnect_indication /
        CSL.CSAI_DISCONNECT_indication;
        ack_requestTimer := max_ack_requestTimer;
        ack_responseTimer := 0;
        ack_reply := 0;
        ack_request := 0;
        currentEC := 0;
        initTimer  := 0;
        OFFSET := 0;
        -- EC_expected := 0;
        DELTA := 0;
        dist := 0;
        last_in := 0;
        next_out :=0;
        outdatabuff := [];
        waitnextcycle := False}
```

```
-------------------------------------------------------------------
-- R12: When in Connected state is received a Sa_CONNECT.indication from the
--    ER-SL, the SAI replies with a Sa_CONNECT.response to the ER-SL, sends a
--   SAI_DISCONNECT.indication to the CSL component, and moves to Connecting
--   state
-------------------------------------------------------------------

R12_CSAI_connconfresp:
   Connected -> tbcr12
       {CSAI_SA_Connect_indication /
       ER.CSAI_SA_Connect_response}

R12_CSAI_connconfdiscard:
   tbcr12 -> Connecting
       { - /
       CSL.CSAI_DISCONNECT_indication;
       ack_requestTimer := max_ack_requestTimer;
       ack_responseTimer := 0;
       ack_reply := 0;
       ack_request := 0;
       currentEC := 0;
       initTimer  := 0;
       OFFSET := 0;
       -- EC_expected := 0;
       DELTA := 0;
       last_in := 0;
       next_out := 0;
       dist := 0;
       outdatabuff := [];
       waitnextcycle := False;}

-------------------------------------------------------------------
-- R13a: When in Connected state is received a SAI_DATA_request(msgtype, data)
--    from the CSL component, and yet no other data message has been sent in
--    this cycle, the SAI sends a
--       Sa_DATA.request(msgtype, data, ackreq, ackresp, seqnum, ecnum)
--  to the ER-SL.
--  The ackreq  and  ackresp parameters are set  according to REQ_ACKs.
--  The seqnum parameter is set according to SEQ_NUMs and the ecnum parameter
--  is set according to REQ_ECNUMs
-------------------------------------------------------------------

R13a_CSAI_datareqforward:
   Connected -> Connected
       {CSAI_DATA_request(arg1,arg2)
              [waitnextcycle = False]/
       ER.CSAI_SA_Data_request(arg1,arg2,1,ack_reply,next_out,currentEC);
       next_out := (next_out + 1) mod M;
       if (ack_request = 1)
          {ack_request := 0;
              ack_requestTimer := 0;
              ack_responseTimer := 0 };
       ack_reply := 0;
       waitnextcycle := True}

-------------------------------------------------------------------
-- R13b: When in Connected state is received a SAI_DATA.request(msgtype, data)
--    from the CSL component, but another data message has already been sent in
--    this cycle, the SAI_DATA.request is  saved in a FIFO dataout buffer
--    (see also REQ_OUTDATABUFF)
-------------------------------------------------------------------

R13b_CSAI_datareqstore:
   Connected -> Connected
```

```
        {CSAI_DATA_request(arg1,arg2)
               [waitnextcycle = True]/
               outdatabuff := outdatabuff +[arg1,arg2]}

-----------------------------------------------------------------
-- R14: When in Connected state is received a Sa_DATA.indication from the ER-SL
--    we can have four cases, depending on the received seqnum and ecnum values
--      (see SEQ_NUMs and REC_ECNUMs):
-----------------------------------------------------------------

R14_CSAI_datareceive:
   Connected -> tbcr14
        {CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6) /
        dist := arg5 - last_in;
        if (dist < -M/2) then {dist := dist + M };
        if (dist >  M/2) then {dist := dist - M };
        if ((dist > 0) and (dist <= N)) then
               {last_in := arg5};
        sarg1 := arg1;
        sarg2 := arg2;
        ack_reply := ack_reply + arg3 - ack_reply*arg3;
        sarg4 := arg4;
        EC_expected := (currentEC + Mec - OFFSET) mod Mec;
        DELTA := EC_expected - arg6;
        if  (DELTA < -Mec/2) then {DELTA := DELTA + Mec };
        if  (DELTA >  Mec/2) then {DELTA := DELTA - Mec };
        EC_expected := 0}

-----------------------------------------------------------------
--   Case 1
-- * The seqnum is the one EXPECTED and ecnum is VALID: In this case the SAI
--   sends a SAI_DATA.indication(msgtype, data) to the CSL component.
--   Depending on the received values of the ackreq and ackresp parameters,
--   appropriate actions are performed (see REQ_ACKs)
-----------------------------------------------------------------

R14a_CSAI_dataindforward:
   tbcr14     -> Connected
        { - [dist = 1 and DELTA < K] /
        CSL.CSAI_DATA_indication(sarg1,sarg2);
        if((sarg4 = 1) and (ack_responseTimer < max_ack_responseTimer))
               {ack_responseTimer := max_ack_responseTimer + 1};
        sarg1 := 0;
        sarg2 := 0;
        sarg4 := 0;
        dist  := 0;
        DELTA := 0}

-----------------------------------------------------------------
--   Case 2
-- * The seqnum is ACCEPTABLE and the ecnum is VALID: in this case the SAI
--   sends a SAI_DATA.indication(msgtype, data) and a SAI_ERROR.report to the
--   CSL component.
--   Depending on the received values of the ackreq and ackresp parameters,
--   appropriate actions are performed (see REQ_ACKs)
-----------------------------------------------------------------

R14b_CSAI_dataindforward:
   tbcr14     -> tbcr14b
        { - [dist > 1 and dist <= N and DELTA < K] /
        CSL.CSAI_DATA_indication(sarg1,sarg2);
        if((sarg4 = 1) and (ack_responseTimer < max_ack_responseTimer))
               {ack_responseTimer := max_ack_responseTimer + 1};
        sarg1 := 0;
```

```
       sarg2 := 0;
       sarg4 := 0;
       dist  := 0;
       DELTA := 0}

R14b_CSAI_dataindreport:
   tbcr14b -> Connected
       { - /
       CSL.CSAI_Error_report;
       sarg1 := 0;
       sarg2 := 0;
       sarg4 := 0;
       dist  := 0;
       DELTA := 0}

--------------------------------------------------------------------
--   Case 3
-- * The seqnum is OLD or (the seqnum is ACCEPTABLE and the ecnum is VALID):
-- In this case the SAI sends a SAI_ERROR.report to the CSL component and
--  discards the Sa_DATA.indication message
--------------------------------------------------------------------

R14c_CSAI_datainvalid_or_old:
   tbcr14     -> Connected
       { - [dist < 1 or (dist <= N and DELTA >= K)] /
       CSL.CSAI_Error_report;
       sarg1 := 0;
       sarg2 := 0;
       sarg4 := 0;
       dist  := 0;
       DELTA := 0}

--------------------------------------------------------------------
--   Case 4
-- * The seqnum is NOT_ACCEPTABLE: In this case the SAI component sends a
--   Sa_DISCONNECT.request to ER-SL, a SAI_DISCONNECT.indication to the CSL
--   component,and then moves to Disconnected state
--------------------------------------------------------------------

R14d_CSAI_abortconn:
   tbcr14 -> tbcr14d
       { - [dist > N] /
       ER.CSAI_SA_Disconnect_request;}

R14d_CSAI_notifydisconn:
   tbcr14d -> Disconnected
       { - /
       CSL.CSAI_DISCONNECT_indication;
       sarg1 := 0;
       sarg2 := 0;
       sarg4 := 0;
       ack_requestTimer := max_ack_requestTimer;
       ack_responseTimer := 0;
       ack_reply := 0;
       ack_request := 0;
       currentEC := 0;
       initTimer  := 0;
       OFFSET := 0;
       -- EC_expected := 0;
       DELTA := 0;
       last_in := 0;
       next_out := 0;
       dist := 0;
       outdatabuff := [];
       waitnextcycle := False}


RD1a_CSAI_disconndiscard:
   Disconnected -> Disconnected
       {CSAI_DISCONNECT_request}

RD1b_CSAI_datadiscard:
   Disconnected -> Disconnected
       {CSAI_DATA_request(arg1,arg2)}

RD1c_CSAI_discinddiscard:
   Disconnected -> Disconnected
       {CSAI_SA_Disconnect_indication}

RD1d_CSAI_datainddiscard:
   Disconnected -> Disconnected
       {CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6)}

RD1e_CSAI_disconndiscard:
   Disconnected -> Disconnected
       {CSAI_SA_Execution_Cycle_Start(arg1,arg2)}

RD2a_CSAI_datainddiscard:
   Connecting -> Connecting
       {CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6)}

RD2b_CSAI_datareqdiscard:
   Connecting -> Connecting
       {CSAI_DATA_request(arg1,arg2)}

RD3a_CSAI_datareqdiscard:
   Initializing -> Initializing
       {CSAI_DATA_request(arg1,arg2)}

RD3c_CSAI_execycstardiscard:
   Initializing -> Initializing
       {CSAI_SA_Execution_Cycle_Start(arg1,arg2)}

RD4a_CSAI_ecsdiscard:
   Connected -> Connected
       {CSAI_SA_Execution_Cycle_Start(arg1,arg2)}

-------------------------------------------------------------------
--  Each time that in Connected state the set_ack_response expires,
--  the SAI sends a SAI_ERROR.report to the CSL component
--  (for further details see REQ_ACKs)
-------------------------------------------------------------------

RTc_CSAI_okcsai:
   Connected -> tbcrtc
       {csai_tick /
       Timer.ok_csai;
       ------ ack timing management
       if (ack_responseTimer < max_ack_responseTimer)
              {ack_responseTimer := ack_responseTimer + 1};
       if (ack_requestTimer < max_ack_requestTimer)
              {ack_requestTimer := ack_requestTimer + 1};
       if (ack_requestTimer = max_ack_requestTimer and
                     ack_responseTimer >= max_ack_responseTimer)
           {ack_request := 1};
       ------ ec timing management
       currentEC := (currentEC + 1) mod Mec;
       ------
       waitnextcycle := False}

RACK2_CSAI_ackresponserror:
   tbcrtc -> tbcrack
       { - [ack_responseTimer = max_ack_responseTimer] /
              CSL.CSAI_Error_report;
              ack_responseTimer  := max_ack_responseTimer + 1}

ROUTDATABUFF_CSAI_sendbuffered:
   tbcrack -> Connected
       { - [ outdatabuff /= [] ] /
       ER.CSAI_SA_Data_request(
          outdatabuff.head, outdatabuff.tail.head,
          ack_request,ack_reply,next_out,currentEC);
       outdatabuff :=  outdatabuff.tail.tail;
       waitnextcycle := True;
       next_out := (next_out + 1) mod M;
       if (ack_request = 1)
          {ack_request := 0;
              ack_requestTimer := 0;
              ack_responseTimer  := 0};
       ack_reply := 0;
       }

RTd_CSAI_continue:
   tbcrtc -> tbcrack
       { - [ack_responseTimer /= max_ack_responseTimer]}

RTe_CSAI_okisai:
   tbcrack -> Connected
     { - [ outdatabuff = [] ]}


end C_SAI;
```

```
------------------------------------------------------------------------------
------------------------------------------------------------------------------
Class C_CSL is
------------------------------------------------------------------------------
------------------------------------------------------------------------------
Signals

   -- from C_RBC
   CRBC_User_Data_request(arg1: int);

   -- from C_SAI
   CSAI_CONNECT_indication;
   CSAI_DISCONNECT_indication;
   CSAI_DATA_indication(arg1: Token, arg2: int);
   CSAI_Error_report;

   -- from Timer
   ccsl_tick;

-- outgoing to RBC
--     RBC_User_Connect_indication;
-- RBC_User_Disconnect_indication;
-- RBC_User_Data_indication(arg1);

-- outgoing to C_SAI
-- CSAI_CONNECT_request;
-- CSAI_DISCONNECT_request;
-- CSAI_DATA_request(arg1,arg2);

-- outgoing to Timer
-- ok_ccsl

Vars
   -------------  PORTS
   RBC_User: C_RBC;
   SAI: C_SAI;
   -------------

   -------------  CONFIGURATION PARAMS
   max_receiveTimer: int;  -- CONFIGURATION PARAM
   max_sendTimer:    int;  -- CONFIGURATION PARAM
   -------------

   -------------  LOCAL VARS
   receiveTimer:    int := 0;
   sendTimer:              int := 0;
   -------------

Behaviour

-------------------------------------------------------------------
-- R1: At startup, the CSL is in Disconnected state
-------------------------------------------------------------------

R1_CCSL:
   initial -> NOCOMMS

-------------------------------------------------------------------
-- R2: When in Disconnected state is received a SAI_CONNECT.indication
--      from the SAI component, the CSL sends a RBC_User_Connect_indication
--      to the RBC component, starts both the sendTimer and the recTimer,
--      and moves to Connected state
-------------------------------------------------------------------
```

```
RTa_CCSL_okccsl:
   NOCOMMS -> NOCOMMS
       {ccsl_tick /
       Timer.ok_ccsl}

R2_CCSL_userconnind:
   NOCOMMS  -> COMMS
       {CSAI_CONNECT_indication /
       RBC_User.CRBC_User_Connect_indication;
       receiveTimer := 0;
       sendTimer :=max_sendTimer}

-------------------------------------------------------------------
-- R3: When in Connected state is received a SAI_CONNECT.indication from
--     the SAI component (this is not supposed to occur), the message is
--    discarded.
-------------------------------------------------------------------

RTb_CCSL_okccsl:
   COMMS -> COMMS
       {ccsl_tick [(receiveTimer < max_receiveTimer)
              and (sendTimer < max_sendTimer)] /
       Timer.ok_ccsl;
       sendTimer := sendTimer +1;
       receiveTimer := receiveTimer+1;  }

R3_CCSL_userconnind:
   COMMS -> COMMS
       {CSAI_CONNECT_indication}

-------------------------------------------------------------------
-- R4: When in Connected state is received a RBC_User_Data.request with
--     RBC_data from the RBC component, the CSL sends a SAI_DATA.request
--     with such RBC_data to the SAI component
-------------------------------------------------------------------

R4_CCSL_saidatareq:
   COMMS -> COMMS
       {CRBC_User_Data_request(arg1) /
       SAI.CSAI_DATA_request(RBCdata,arg1);
       sendTimer := 0}

-------------------------------------------------------------------
-- R5: Each time that in Connected state the sendTimer expires, the CSL
--     sends a SAI_DATA.request with a life_sign to the SAI component
-------------------------------------------------------------------

R5_CCSL_okccsl:
   COMMS -> tbcr5
       {ccsl_tick [(receiveTimer < max_receiveTimer)
              and (sendTimer = max_sendTimer)] /
       Timer.ok_ccsl;
       sendTimer := 0;
       receiveTimer := receiveTimer+1}

R5_CCSL_saidatareq:
   tbcr5 -> COMMS
       {-/
       SAI.CSAI_DATA_request(LifeSign,0)}

-------------------------------------------------------------------
-- R6: When in Connected state is received a SAI_DATA.indication with a
--     life_sign from the SAI component, the CSL restarts the recTimer
-------------------------------------------------------------------
```

```
R6_CCSL_handlelifesign:
   COMMS -> COMMS
        {CSAI_DATA_indication(arg1,arg2)
                [arg1 = LifeSign] /
        receiveTimer := 0}

-------------------------------------------------------------------
-- R7: When in Connected state is received a SAI_DATA.indication with
--     SAI_data from the SAI component, the CSL sends a
--     RBC_User_Data.indication with such SAI_data to the RBC component
--     and restarts the recTimer
-------------------------------------------------------------------

R7_CCSL_userdataind:
   COMMS -> COMMS
        {CSAI_DATA_indication(arg1,arg2)
                [arg1 /= LifeSign] /
        RBC_User.CRBC_User_Data_indication(arg2);
        receiveTimer := 0}

-------------------------------------------------------------------
-- R8: When in Connected state is received a SAI_DISCONNECT.indication
--     from the SAI component, the CSL sends a RBC_User_Disconnect.indication
--     to the RBC component and moves to Disconnected state
-------------------------------------------------------------------

R8_CCSL_userdisconnind:
   COMMS -> NOCOMMS
        {CSAI_DISCONNECT_indication /
        RBC_User.CRBC_User_Disconnect_indication;
        receiveTimer := 0;
        sendTimer := 0}

-------------------------------------------------------------------
-- R9: When in Connected state the recTimer expires, the CSL sends a
--     SAI_DISCONNECT.request to the SAI component, a RBC_User_Disconnect.indication
--     to the RBC component and moves to Disconnected state
-------------------------------------------------------------------

R9_CCSL_okccsl:
   COMMS -> tbcr9
        {ccsl_tick [receiveTimer = max_receiveTimer] /
        Timer.ok_ccsl}

R9_CCSL_saidisconnreq:
   tbcr9 -> tbcr9a
        {- /
        SAI.CSAI_DISCONNECT_request;
        receiveTimer := 0;
        sendTimer := 0}

R9_CCSL_userdisconnind:
   tbcr9a -> NOCOMMS
        {- /
        RBC_User.CRBC_User_Disconnect_indication}

-------------------------------------------------------------------
-- RD1: When in Disconnected state the CSL does not accept any kind of message
--      except for SAI_CONNECT.indication from the SAI component
--
-- RD2: When in Connected state the CSL discards only SAI_ERROR.report messages
--      from the SAI component
-------------------------------------------------------------------
```

```
RD1a_CCSL_discuserdata:
   NOCOMMS -> NOCOMMS
        {CRBC_User_Data_request(arg1) }

RD1b_CCSL_discdisconnind:
   NOCOMMS -> NOCOMMS
        {CSAI_DISCONNECT_indication }

RD1c_CCSL_discerrorreport:
   NOCOMMS -> NOCOMMS
        {CSAI_Error_report }

RD1d_CCSL_discdataind:
   NOCOMMS -> NOCOMMS
        {CSAI_DATA_indication(arg1,arg2) }

RD2a_CCSL_usererror:
   COMMS -> COMMS
        {CSAI_Error_report }

end C_CSL;
------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
Class I_RBC is
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Signals
   --   incoming from CSL
   IRBC_User_Connect_indication;
   IRBC_User_Disconnect_indication;
   IRBC_User_Data_indication(arg1: int);
   -- incoming from Timer
   irbc_tick
   --   outgoing to CSL
   -- IRBC_User_Data_request(arg1)
   --   outgoing to timer
   -- ok_irbc
Vars
  CSL:obj;    -- CSL port
  n: int := 1;
  nmax: int;  -- initialized at configuration time
Behaviour

R1a_IRBCS_tick:
 wait -> wait
   {irbc_tick /
      Timer.ok_irbc;}

R1d_IRBC:
 wait -> connected
   {IRBC_User_Connect_indication}

R2a_IRBC:
 connected  -> sending
   {irbc_tick /
      Timer.ok_irbc;}

R2b_IRBC:
 connected  -> connected
   { IRBC_User_Data_indication(arg1)}

R2c_IRBC:
 sending  -> connected
   { - [n <= nmax] /
     CSL.IRBC_User_Data_request(n);
     n := n+1; }

R2f_IRBC:
 sending  -> connected
   { - [n > nmax] }

R3_IRBC:
 connected  -> wait
    {IRBC_User_Disconnect_indication }

end I_RBC;


--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Class C_RBC is
--------------------------------------------------------------------------------
-- Senario SC1
-- CRBC not sending NRBC messages, just receiving indications
--------------------------------------------------------------------------------
Signals
```

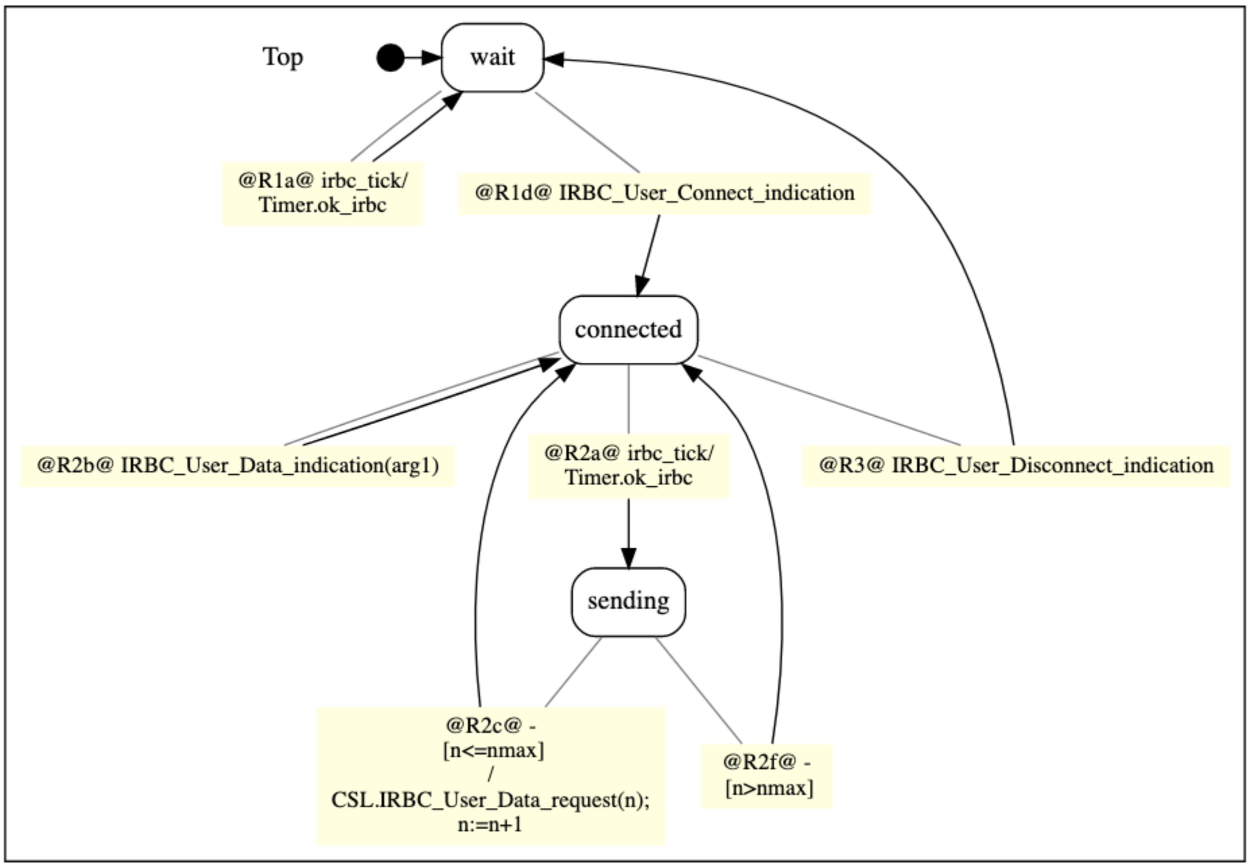
```
     -- incoming from CSL
   CRBC_User_Connect_indication;
   CRBC_User_Disconnect_indication;
   CRBC_User_Data_indication(arg1: int);
   crbc_tick
   --
   -- outgoing to CSL
   -- CRBC_User_Data_request(arg1)
Vars
   CSL: C_CSL;  -- CSL port

Behaviour

-- increment S counter at every time slop
R1_CRBCS_tick:
   wait -> wait
        {crbc_tick /
        Timer.ok_crbc;}

-- do not handle connect indications
R4_CRBC_discard_CI:
    wait -> wait
        {CRBC_User_Connect_indication }

-- do not handle disconnect indications
R5_CRBC_discard_DI:
   wait -> wait
        {CRBC_User_Disconnect_indication }

-- do not handle data indications
R6a_CRBC_discard_UD:
   wait -> wait
        {CRBC_User_Data_indication(arg1)}

-- never triggered , used just for CADP compatibility.
R7_CRBC_justforsync_DR:
   sending -> wait
        {- /
        CSL.CRBC_User_Data_request(0); }

end C_RBC;
--------------------------------------------------------------------------------



--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Class Clock is
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Signals
   ok_irbc, ok_icsl, ok_isai, ok_eur, ok_csai, ok_ccsl, ok_crbc

Vars
-- Priority := 0;

Behaviour

R0_Timer_IRBC:
   s0 -> s1
        { - /
        IRBC.irbc_tick;}

R1_Timer_ICSL:
   s1 -> s2
       { ok_irbc /
       ICSL.icsl_tick;}

R2_Timer_ISAI:
   s2 -> s3
       {ok_icsl /
       ISAI.isai_tick; }

R3_Timer_EUR:
   s3 -> s4
       { ok_isai /
       EUR.EUR_tick; }

R4_Timer_ISAI:
   s4 -> s5
       {ok_eur /
       CSAI.csai_tick; }

R5_Timer_ISAI:
   s5 -> s6
       {ok_csai /
       CCSL.ccsl_tick; }

R6_Timer_ISAI:
   s6 -> s7
       {ok_ccsl /
       CRBC.crbc_tick; }

R7_Timer_ISAI:
   s7 -> s1
       {ok_crbc /
       IRBC.irbc_tick; }

end Clock;
--------------------------------------------------------------------------------



--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Class EuroRadio is
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

Signals

   -- incoming from ISAI
   ISAI_SA_Connect_request;
   ISAI_SA_Disconnect_request;
   ISAI_SA_Data_request(arg1:int,arg2:int,arg3:int,arg4:int,arg5:int,arg6:int);
   -- arg1 = message type,    arg2 = data value,
   -- arg3 = ack request,     arg4 = ack response,
   -- arg5 = sequence number, arg6 = EC number
   ISAI_SA_Execution_Cycle(arg1:int,arg2:int);
   -- arg1 = sequence number,   arg2 = EC counter

   -- incoming from CSAI
   CSAI_SA_Disconnect_request;
   CSAI_SA_Data_request(arg1:int,arg2:int,arg3:int,arg4:int,arg5:int,arg6:int);
   -- arg1 = message type,    arg2 = data value,
   -- arg3 = ack request,     arg4 = ack response,
   -- arg5 = sequence number, arg6 = EC number
   CSAI_SA_Connect_response;
   CSAI_SA_Execution_Cycle(arg1:int,arg2:int);
   -- arg1 = sequence number,   arg2 = EC counter

   -- incoming from Timer
   EUR_tick;

-- outgoing to ISAI
--  ISAI_SA_Disconnect_indication;
--  ISAI_SA_Connect_confirm;
--  ISAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6);
--     arg1 = message type,    arg2 = data value,
--     arg3 = ack request,     arg4 = ack response,
--     arg5 = sequence number, arg6 = EC number
-- ISAI_SA_Execution_Cycle_Start(arg1,arg2);
--     arg1 = sequence number,   arg2 = EC counter

--  outgoing to CSAI
--  CSAI_SA_Disconnect_indication;
--  CSAI_SA_Connect_indication;
--  CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6);
--     arg1 = message type,    arg2 = data value,
--     arg3 = ack request,     arg4 = ack response,
--     arg5 = sequence number, arg6 = EC number
-- CSAI_SA_Execution_Cycle_Start(arg1,arg2);
--     arg1 = sequence number,   arg2 = EC counter

--  outgoing to Timer
--  ok_eur

Vars
--    Priority := 0;
   Iside: I_SAI;
   Cside: C_SAI;

Behaviour

--------------------------
-- handle time slots
--------------------------

R1_EUR_okier:
   s0 -> s0
       { EUR_tick /
       Timer.ok_eur}

--------------------------
-- handle inoming ISAI requests
--------------------------

R2i_EUR_newconnection:
   s0 -> s0
       {ISAI_SA_Connect_request /
       Cside.CSAI_SA_Connect_indication;
       }

R3i_EUR_disconnection:
   s0-> s0
       {ISAI_SA_Disconnect_request /
        Cside.CSAI_SA_Disconnect_indication;
       }

R4i_EUR_acceptdata:
   s0 -> s0
```

```
        { ISAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6) /
        Cside.CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6);
        }

-- R5i_EUR_losedata:
-- s0 -> s0
-- { ISAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6) }

--------------------------------
-- handle incoming CSAI requests
--------------------------------
--
R2c_EUR_confirm:
   s0 -> s0
        {CSAI_SA_Connect_response  /
        Iside.ISAI_SA_Connect_confirm;
        }
--
R3c_EUR_disconnection:
   s0-> s0
        {CSAI_SA_Disconnect_request /
        Iside.ISAI_SA_Disconnect_indication;
        }

--
R4c_EUR_acceptdata:
   s0 -> s0
        { CSAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6) /
        Iside.ISAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6);
        }

--      R5c_EUR_losedata:
--      s0 -> s0
-- { CSAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6) }


-- when ISAI responding with execution cycle start
R6a_EUR_execcycstart_call:
   s0 -> s0
        {ISAI_SA_Execution_Cycle(arg1,arg2) /
        Cside.CSAI_SA_Execution_Cycle_Start(arg1,arg2)}

-- when CSAI performing execution cycle start
R6b_EUR_execcycstart_response:
   s0 -> s0
        {CSAI_SA_Execution_Cycle(arg1,arg2) /
        Iside.ISAI_SA_Execution_Cycle_Start(arg1,arg2)}

end EuroRadio;
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----     SYSTEM CONFIGURATION
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Objects:

LifeSign, RBCdata: Token;

Timer: Clock;

IRBC: I_RBC
   (CSL -> ICSL, nmax -> 5);
```

```
ICSL: I_CSL
   (RBC_User -> IRBC, SAI -> ISAI,
   max_receiveTimer        -> 20,   -- timeout for receiving data
   max_sendTimer                  -> 10,   -- timeout for sending data
   max_connectTimer        -> 20)  -- timeout for connection phase

ISAI: I_SAI
   (CSL -> ICSL, ER -> EUR,
   -- EC numbers management
   max_initTimer            -> 20,  -- initialiation phase timeout
   Mec                      -> 7,   -- limit for EC numbers
   K                        -> 3,  -- delay limit for incoming messages
   --acks management
   max_ack_requestTimer    -> 20,  -- ack request period
   max_ack_responseTimer   -> 20,  -- max delay before ack response
   -- sequence numbers management
   N                                      -> 1,  -- limit for consecutive loss of
messages
   M                       -> 3)  -- limit for sequence numbers

EUR: EuroRadio
   (Iside -> ISAI, Cside -> CSAI);

CCSL: C_CSL
   (RBC_User ->CRBC, SAI -> CSAI,
   max_receiveTimer        -> 20, -- timeout for receiving data
   max_sendTimer                  -> 10);      -- timeout for sending data

CSAI: C_SAI
   (CSL -> CCSL, ER -> EUR,
   -- EC numbers management
   max_initTimer           -> 10, -- initialiation phase timeout
   Mec                      -> 7,  -- limit for EC numbers
   K                        -> 3,  -- delay limit for incoming messages
   --acks management
   max_ack_requestTimer     -> 20, -- ack request period
   max_ack_responseTimer    -> 20, -- ack response timeout
   -- sequence numbers management
   N                       -> 1,   -- limit for consecutive loss of messages
   M                       -> 3)        -- limit for sequence numbers

CRBC: C_RBC (CSL -> CCSL);



Abstractions {
TLABELS
-- DEBUG
Action lostevent($1) -> lostevent($1)
Action $1($*)  -> $1($*)
}


--------------------------------------------------------------------
-- UMC ON THE FLY VERIFICATION:
--------------------------------------------------------------------
-- umc -300 -x  ERnice_irbcdata_V53_Mec7K3.umc
-- **  REC=20, SEND=10, CONN=20, INIT=20, MEC=7, K=3, AckRq=20, AckRs=20, N=1, M=3 **
--
-- A[ {not CRBC_User_Data_indication} U { CRBC_User_Connect_indication}]
--
-- AF {CRBC_User_Connect_indication}
--    A[ {not CRBC_User_Data_indication} U { CRBC_User_Data_indication(1)}]
(2.473719)
--
-- A[true {not CSAI_Error_report} U {CRBC_User_Data_indication(1) } true]
(2.473719)
--
-- A[true {not CSAI_Error_report} U {CRBC_User_Data_indication(2) } ]
(7.334053)
--
-- AF { CRBC_User_Data_indication(1)}
--    A[ {not CRBC_User_Data_indication} U { CRBC_User_Data_indication(2)}]
(7.334053)
--
-- A[true {not CRBC_User_Disconnect_indication} U {CRBC_User_Data_indication(3)}]
(11.990690)
--
-- AF {CRBC_User_Data_indication(2)}
--   A[true {not CRBC_User_Data_indication} U {CRBC_User_Data_indication(3)}]
(11.990690)
--
-- AF {CRBC_User_Data_indication(3)}
--   A[true {not CRBC_User_Data_indication} U {CRBC_User_Data_indication(4)}]
(15.109362)
--
-- AF {CRBC_User_Data_indication(4)}
--   A[true {not CRBC_User_Data_indication} U {CRBC_User_Data_indication(5)}]
(16.522994,)

--------------------------------------------------------------------
```

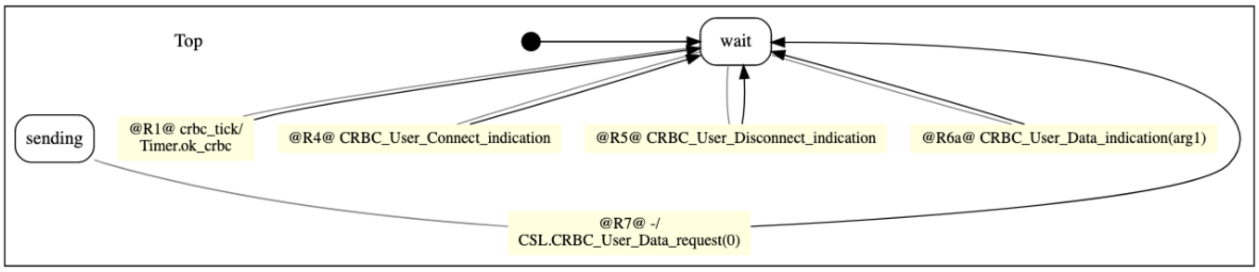
Statechart for Class I_RBC

Zoom Out Zoom In


Top
wait
@R1a@ irbc_tick/
Timer.ok_irbc
@R1d@ IRBC_User_Connect_indication
connected
@R2b@ IRBC_User_Data_indication(arg1)
@R2a@ irbc_tick/
Timer.ok_irbc
@R3@ IRBC_User_Disconnect_indication
sending
@R2c@ -
[n<=nmax]
/
CSL.IRBC_User_Data_request(n);
n:=n+1
@R2f@ -
[n>nmax]

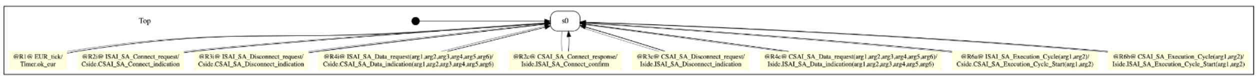
Statechart for Class C_RBC

Zoom Out Zoom In

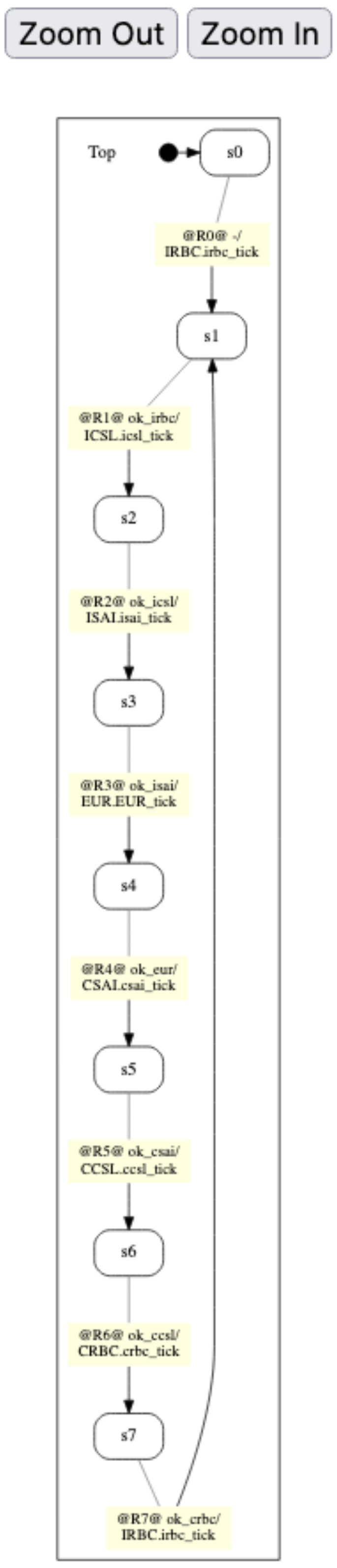

Top
wait
sending
@R1@ crbc_tick/
Timer.ok_crbc
@R4@ CRBC_User_Connect_indication
@R5@ CRBC_User_Disconnect_indication
@R6a@ CRBC_User_Data_indication(arg1)
@R7@ -/
CSL.CRBC_User_Data_request(0)

Statechart for Class EuroRadio

Zoom Out Zoom In


Top
s0
@R1@ EUR_tick/
Timer.ok_eur
@R2i@ ISAI_SA_Connect_request/
Cside.CSAI_SA_Connect_indication
@R3i@ ISAI_SA_Disconnect_request/
Cside.CSAI_SA_Disconnect_indication
@R4i@ ISAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6)/
Cside.CSAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6)
@R2c@ CSAI_SA_Connect_response/
Iside.ISAI_SA_Connect_confirm
@R3c@ CSAI_SA_Disconnect_request/
Iside.ISAI_SA_Disconnect_indication
@R4c@ CSAI_SA_Data_request(arg1,arg2,arg3,arg4,arg5,arg6)/
Iside.ISAI_SA_Data_indication(arg1,arg2,arg3,arg4,arg5,arg6)
@R6a@ ISAI_SA_Execution_Cycle(arg1,arg2)/
Cside.CSAI_SA_Execution_Cycle_Start(arg1,arg2)
@R6b@ CSAI_SA_Execution_Cycle(arg1,arg2)/
Iside.ISAI_SA_Execution_Cycle_Start(arg1,arg2)

Statechart for Class Clock


Zoom Out
Zoom In
Top
s0
@R0@ -/
IRBC.irbc_tick
s1
@R1@ ok_irbc/
ICSL.icsl_tick
s2
@R2@ ok_icsl/
ISAI.isai_tick
s3
@R3@ ok_isai/
EUR.EUR_tick
s4
@R4@ ok_eur/
CSAI.csai_tick
s5
@R5@ ok_csai/
CCSL.ccsl_tick
s6
@R6@ ok_ccsl/
CRBC.crbc_tick
s7
@R7@ ok_crbc/
IRBC.irbc_tick