

## Initiator SAI

```

K: int;
max_initTimer: int;
N: int; -- msg loss limit
M: int;
max_ack_requestTimer: int;
max_ack_responseTimer: int;
Mec: int;
    
```

```

/
R1
initTimer := 0;
OFFSET := 0;
EC_expected := 0;
DELTA := 0;
currentEC := 0;
last_in := 0;
next_out := 0;
dist := 0;
ack_requestTimer := 0;
ack_responseTimer := 0;
ack_reply := 0;
ack:request := 0;
outdatabuff := [];
waitnextcycle := False;
    
```

```

R3
SAI_DISCONNECT.request /
CSL.SAI_DISCONNECT.indication
    
```

```

R4
Sa_DISCONNECT.indication /
    
```

```

R12
SAI_DISCONNECT.request /
CSL.SAI_DISCONNECT.indication;
ER.Sa_DISCONNECT.request;
ack_requestTimer :=
    max_ack_requestTimer;
ack_responseTimer := 0;
ack_reply := 0;
currentEC := 0;
outdatabuff := [];
ack_request := 0;
waitnextcycle := False
    
```

```

R2
SAI_CONNECT.request /
ER.Sa_CONNECT.request
    
```

```

R7
[initTimer = max_initTimer] /
CSL.SAI_ERROR.report;
ER.Sa_DISCONNECT.request;
initTimer := 0;
currentEC := 0;
    
```

```

R6
Sa_CONNECT.confirm /
ER.Sa_ExecutionCycle
    (next_out,currentEC);
next_out := (next_out + 1) mod M;
initTimer := 0;
    
```

```

isai_tick /
Timer.ok_isai;
initTimer := initTimer + 1;
currentEC :=
    (currentEC + 1) mod Mec;
[initTimer < max_initTimer] /
    
```

```

R11
Sa_ExecutionCycleStart
(seqnum,ecnum) /
CSL.SAI_CONNECT.confirm;
OFFSET := currentEC - ecnum;
initTimer := 0;
ack_requestTimer := 0;
ack_responseTimer :=
    max_ack_responseTimer + 1;
ack_reply := 0;
last_in := seqnum
    
```

```

R16d
[dist > N] /
ER.Sa_DISCONNECT.request
CSL.SAI_DISCONNECT.indication;
ack_responseTimer := 0;
next_out := 0;
dist := 0;
DELTA := 0;
outdatabuff := [];
ack_request := 0;
waitnextcycle := False
    
```

```

R15
Sa_DISCONNECT.indication /
CSL.SAI_DISCONNECT.indication;
ack_requestTimer :=
    max_ack_requestTimer;
ack_responseTimer := 0;
ack_reply := 0;
outdatabuff := [];
ack_request := 0;
waitnextcycle := False;
    
```

```

R9
Sa_DISCONNECT.indication /
initTimer := 0;
currentEC := 0;
    
```

### R16a

```

[dist = 1 and DELTA < K] /
CSL.SAI_DATA.indication(msgtype,userdata);
if ( ackresp = 1 and
    ack_responseTimer <
        max_ack_responseTimer )
{ ack_responseTimer :=
    max_ack_responseTimer + 1 };
dist := 0;
DELTA := 0
    
```

### R16b

```

[dist > 1 and dist <= N and DELTA < K] /
CSL.SAI_DATA.indication(msgtype,userdata);
if (ackresp = 1 and
    ack_responseTimer < max_ack_responseTimer)
{ ack_responseTimer :=
    max_ack_responseTimer + 1 };
CSL.SAI_ERROR.report;
dist := 0;
DELTA := 0
    
```

```

R16
Sa_DATA_indication
(msgtype,userdata,
ackreq,ackresp,
seqnum,ecnum) /
dist := seqnum - 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 := seqnum};
ack_reply :=
    ack_reply + ackreq - ack_reply*ackreq;
EC_expected :=
    (currentEC + Mec - OFFSET) mod Mec;
DELTA := EC_expected - ecnum;
if (DELTA < -Mec/2) {DELTA := DELTA + Mec};
if (DELTA > Mec/2) {DELTA := DELTA - Mec};
EC_expected := 0
    
```

### R16c

```

[dist < 1 or (dist <= N and DELTA >= K)] /
CSL.SAI_ERROR.report;
dist := 0;
DELTA := 0
    
```

```

R13b
SAI_DATA.request(msgtype,userdata)
[waitnextcycle = True] /
outdatabuff :=
    outdatabuff + [msgtype,userdata];
    
```

### R13a

```

SAI_DATA.request(msgtype,userdata)
[waitnextcycle = False] /
ER.Sa_DATA.request(msgtype,userdata,
    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;
    
```

```

isai_tick /
Timer.ok_isai;
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};
currentEC := (currentEC + 1) mod Mec;
waitnextcycle := False
    
```

### R14

```

if ( ack_responseTimer =
    max_ack_responseTimer ) {
    CSL.SAI_ERROR.report;
    ack_responseTimer :=
        max_ack_responseTimer + 1; }

if ( outdatabuff != [] ) {
    ER.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; }
    
```

CONN  
Connected

NOCONN  
Disconnected

NOCONN  
Connecting

NOCONN  
Initializing