

```

1 |----- MODULE service12 -----|
  | Follow up from "service11", with verification [V] conditions added |
  | Changes: Add P9 - P14 |
8 | EXTENDS Naturals, Sequences, TLC, FiniteSets
10 | CONSTANTS Node, NoNode,
11 |     UserTask, UserTaskL(-),
12 |     KernelTask, KernelTaskL(-),
13 |     TxTask, TxTaskL(-), HLinkTx(-),
14 |     RxTask, RxTaskL(-), RxHLink(-), HLinkRx(-),
15 |     HLink, NodeConnect(-),
16 |     RouterHL(-, -), RouterTx(-, -), RouterRx(-, -),
17 |     Port, PortL(-),
18 |     Message, NoMessage,
19 |     EmptyPacket,
20 |     RxAdr, RxAdrL(-)
21 |-----|
22 | DEFINITIONS
24 | L0ServiceID  $\triangleq$  {"start", "stop", "suspend", "resume", "acknw", "send", "receive"} \ *, "acksnd", "ackrcv" }
25 | L0Status  $\triangleq$  {"ok", "fail", "failTO", "blank"}
26 | TaskState  $\triangleq$  {"reserved", "inactive", "started"}
28 | Task  $\triangleq$  UserTask  $\cup$  KernelTask  $\cup$  TxTask  $\cup$  RxTask
30 | Packet  $\triangleq$  [service : L0ServiceID,
31 |     requesttask : UserTask,
32 |     destination : Port  $\cup$  UserTask,
33 |     status : L0Status,
34 |     data : Message  $\cup$  {NoMessage},
35 |     memoryaddress : UserTask  $\cup$  RxAdr The so-called "OWNER" field!!!
36 | ]
38 | TaskCB  $\triangleq$  [taskstate : TaskState,
39 |     issuspended : {0, 1} 0 = NotSuspended ; 1 = Suspended
40 | ]
42 | InitTaskCB  $\triangleq$  CHOOSE tcb  $\in$  TaskCB :  $\wedge$  tcb.taskstate = "inactive"
43 |      $\wedge$  tcb.issuspended = 0
45 | StartedTaskCB  $\triangleq$  CHOOSE tcb  $\in$  TaskCB :  $\wedge$  tcb.taskstate = "started"
46 |      $\wedge$  tcb.issuspended = 0
48 |-----|
49 | VARIABLES TaskController,
50 |     UserTaskMem,
51 |     KernelInputPort, TxInputPort,
52 |     ReadyList,
53 |     Error,
54 |     InitSetup,
55 |     RxMem, RxInputPort,
56 |     WireMem,
57 |     PortInputPort
59 | vars  $\triangleq$   $\langle$  TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort
60 |-----|
61 | TypeInvariant  $\triangleq$ 
62 |      $\wedge$  TaskController  $\in$  [Task  $\rightarrow$  TaskCB]
64 |      $\wedge$  UserTaskMem  $\in$  [UserTask  $\rightarrow$  Packet  $\cup$  {EmptyPacket}]
66 |      $\wedge$  KernelInputPort  $\in$  [KernelTask  $\rightarrow$  Seq(UserTask  $\cup$  RxAdr)]
68 |      $\wedge$  TxInputPort  $\in$  [TxTask  $\rightarrow$  Seq(UserTask  $\cup$  RxAdr)]

```

```

70  ∧ ReadyList ⊆ (Task)
72  ∧ Error ∈ {0, 1}
74  ∧ InitSetup ∈ {0, 1}
76  ∧ RxMem ∈ [RxAdr → Packet ∪ {EmptyPacket}]
78  ∧ RxInputPort ∈ [RxTask → Seq(RxAdr)]
80  ∧ WireMem ∈ [HLink → [Pac : Packet ∪ {EmptyPacket},
81                      Dest : Node ∪ {NoNode}]]
83  ∧ PortInputPort ∈ [Port → Seq(UserTask ∪ RxAdr)]
85  Init ≜
86  ∧ TaskController = [t ∈ Task ↦ InitTaskCB]
87  ∧ UserTaskMem = [ut ∈ UserTask ↦ EmptyPacket]
88  ∧ KernelInputPort = [kt ∈ KernelTask ↦ ⟨⟩]
89  ∧ TxInputPort = [txt ∈ TxTask ↦ ⟨⟩]
90  ∧ ReadyList = {}
91  ∧ Error = 0
92  ∧ InitSetup = 0
93  ∧ RxMem = [ra ∈ RxAdr ↦ EmptyPacket]
94  ∧ RxInputPort = [rxt ∈ RxTask ↦ ⟨⟩]
95  ∧ WireMem = [hl ∈ HLink ↦ [Pac ↦ EmptyPacket,
96                          Dest ↦ NoNode]]
97  ∧ PortInputPort = [p ∈ Port ↦ ⟨⟩]

```

---

99 **Functions**

```

101 NotEmpty(seq) ≜ IF Len(seq) ≥ 1 THEN TRUE ELSE FALSE
102 IsEmpty(seq) ≜ IF Len(seq) = 0 THEN TRUE ELSE FALSE

```

---

105 **“Services”**

```

107 ServeStartTask(requestPacket) ≜
108   LET
109     reqSource ≜ requestPacket.requestingtask
110     requestStatus ≜ requestPacket.status
111     destTask ≜ requestPacket.destination
112     destTaskState ≜ TaskController[destTask].taskstate
113
114     reqnode ≜ UserTaskL(reqSource)
115     destnode ≜ UserTaskL(destTask)
116
117   IN
118   IF reqnode = destnode THEN (local,local)
119
120     IF destTaskState = “inactive” THEN
121       activate the Task
122       ∧ TaskController' = [TaskController EXCEPT ![destTask].taskstate = “started”]
123       make task ready
124       ∧ ReadyList' = ReadyList ∪ {destTask} ∪ {reqSource}
125       “ok”'
126       ∧ UserTaskMem' = [UserTaskMem EXCEPT ![reqSource].status = “ok”]
127       ∧ UNCHANGED ⟨Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort⟩
128
129     ELSE packet was already “started”
130       ∧ UserTaskMem' = [UserTaskMem EXCEPT ![reqSource].status = “fail”]
131       ∧ ReadyList' = ReadyList ∪ {reqSource}
132       ∧ UNCHANGED ⟨TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort⟩
133
134     ELSE (remote, local)
135     LET

```

```

136     RxEntry  $\triangleq$  requestPacket.memoryaddress
137     TxToUse  $\triangleq$  RouterTx(destnode, reqnode)
138 IN
139     IF destTaskState = "inactive" THEN
141         activate the Task
142          $\wedge$  TaskController' = [TaskController EXCEPT ![destTask].taskstate = "started"]
143         make task ready
144          $\wedge$  ReadyList' = ReadyList  $\cup$  {destTask}  $\cup$  {TxToUse}
146         Change the packet header to send acknowledgment!
148          $\wedge$  RxMem' = [RxMem EXCEPT ![RxEntry].service = "acknw",
149                 ![RxEntry].destination = reqSource,
150                 ![RxEntry].status = "ok"]
152          $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToUse] = Append(@, RxEntry)]
153          $\wedge$  UNCHANGED  $\langle$ UserTaskMem, Error, InitSetup, RxInputPort, WireMem, PortInputPort $\rangle$ 
155     ELSE     packet was already "started"
157          $\wedge$  ReadyList' = ReadyList  $\cup$  {TxToUse}
159          $\wedge$  RxMem' = [RxMem EXCEPT ![RxEntry].service = "acknw",
160                 ![RxEntry].destination = reqSource,
161                 ![RxEntry].status = "fail"]
163          $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToUse] = Append(@, RxEntry)]
165          $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem, PortInputPort $\rangle$ 
168 |-----|
171 ServeAcknowledgment(requestPacket)  $\triangleq$ 
172     LET
173     reqSource  $\triangleq$  requestPacket.requestingtask
174     destTask  $\triangleq$  requestPacket.destination
175     reqowner  $\triangleq$  requestPacket.memoryaddress
176     IN
177     IF reqSource = destTask THEN     This is just a sanity check! Remove afterwards!
179     |
180     IF reqowner  $\in$  UserTask THEN     (local,local)
182         Set the task ready and 'clean' the packet allocated to the usertask
183          $\wedge$  UserTaskMem' = [UserTaskMem EXCEPT ![reqSource] = EmptyPacket]
184          $\wedge$  ReadyList' = ReadyList  $\cup$  {reqSource}
185          $\wedge$  Error' = 2
186          $\wedge$  UNCHANGED  $\langle$ TaskController, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort $\rangle$ 
188     ELSE     (remote,local)
190         Set the task ready, Clean the packet allocated to the usertask, and clean RxMem
192          $\wedge$  UserTaskMem' = [UserTaskMem EXCEPT ![reqSource] = EmptyPacket]
193          $\wedge$  ReadyList' = ReadyList  $\cup$  {reqSource}
195          $\wedge$  RxMem' = [RxMem EXCEPT ![reqowner] = EmptyPacket]
197          $\wedge$  UNCHANGED  $\langle$ TaskController, Error, InitSetup, TxInputPort, RxInputPort, WireMem, PortInputPort $\rangle$ 
201     ELSE     (remote, local)
202          $\wedge$  Error' = 2
203          $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, ReadyList, InitSetup, TxInputPort, RxMem, RxInputPort, WireM

```

207  $ServeSend(requestPacket) \triangleq$

208 LET

209  $reqSource \triangleq requestPacket.memoryaddress \quad requestingtask$

!!!! TRADEOFF DECISION HERE!!!!: If  $reqSource$  is read from the "owner field", then there's no difference between (local,local) and (remote,local). If 'owner field' wouldn't exist then we would have to make an IF  $reqnode = destnode$  THEN ELSE and split the code. !!!!BTW!!!!: How would you know the  $RxAdr$ ??? *ok, i know, the distinction would have to made in the  $RxLoop$ .*

!!Same question for  $SERVERECEIVE()$ !!

215  $destin \triangleq requestPacket.destination$

216 IN

218 IF  $Len(PortInputPort[destin]) > 0$  THEN **Port is not empty**

219 LET

220  $storedReq \triangleq Head(PortInputPort[destin])$

221  $storedPac \triangleq$  IF  $storedReq \in UserTask$  THEN  $UserTaskMem[storedReq]$  ELSE  $RxMem[storedReq]$

222  $storedReqType \triangleq storedPac.service$

223 IN

224 IF  $storedReqType = "receive"$  THEN **AND There's a complementary request waiting  $\rightarrow SYNCHRONIZE!$**

227 CASE  $((reqSource \in UserTask) \wedge (storedReq \in UserTask)) \rightarrow$  **(local,local)**

1. Set the status of both requests to "ok" and COPY THE DATA FROM ONE PACKET TO THE OTHER!

2. Add both tasks to the  $ReadyList$

3. Remove the stored request from the  $PortInputPort$

232  $\wedge UserTaskMem' = [UserTaskMem \text{ EXCEPT } ![reqSource].status = "ok",$

233  $![storedReq].status = "ok",$

234  $![storedReq].data = UserTaskMem[reqSource].data]$

235  $\wedge ReadyList' = ReadyList \cup \{reqSource\} \cup \{storedReq\}$

236  $\wedge PortInputPort' = [PortInputPort \text{ EXCEPT } ![destin] = Tail(@)]$

237  $\wedge$  UNCHANGED  $\langle TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem \rangle$

240  $\square ((reqSource \in UserTask) \wedge (storedReq \in RxAdr)) \rightarrow$  **(local,remote)**

1. The requesting task is local  $\rightarrow$  simply set its status "ok".

2. The stored task is remote  $\rightarrow$  Update the packet (With the Data COPIED) and send the Acknowledgment through  $Tx$

3. Add the requesting task and  $Tx$  to the  $ReadyList$

4. Append the request in  $Tx$

5. Remove the stored request from the  $PortInputPort$

247 LET

248  $localnode \triangleq PortL(destin)$

249  $remnode \triangleq UserTaskL(RxMem[storedReq].requestingtask)$

250  $TxToUse \triangleq RouterTx(localnode, remnode)$

251 IN

252  $\wedge UserTaskMem' = [UserTaskMem \text{ EXCEPT } ![reqSource].status = "ok"]$

253  $\wedge RxMem' = [RxMem \text{ EXCEPT } ![storedReq].service = "acknw",$

254  $![storedReq].destination = RxMem[storedReq].requestingtask,$

255  $![storedReq].status = "ok",$

256  $![storedReq].data = UserTaskMem[reqSource].data]$

257  $\wedge ReadyList' = ReadyList \cup \{reqSource\} \cup \{TxToUse\}$

258  $\wedge TxInputPort' = [TxInputPort \text{ EXCEPT } ![TxToUse] = Append(@, storedReq)]$

259  $\wedge PortInputPort' = [PortInputPort \text{ EXCEPT } ![destin] = Tail(@)]$

260  $\wedge$  UNCHANGED  $\langle TaskController, Error, InitSetup, RxInputPort, WireMem \rangle$

263  $\square ((reqSource \in RxAdr) \wedge (storedReq \in UserTask)) \rightarrow$  **(remote,local)**

1. The requesting task is remote  $\rightarrow$  Update the packet (With the Data ERASED!!) and send the Acknowledgment through  $Tx$

2. The stored task is local  $\rightarrow$  set its status "ok" and COPY the DATA

3. Add the stored task and  $Tx$  to the  $ReadyList$

4. Append the request in  $Tx$

5. Remove the stored request from the  $PortInputPort$

270 LET

271  $localnode \triangleq PortL(destin)$

272  $remnode \triangleq UserTaskL(RxMem[reqSource].requestingtask)$

```

273     TxToUse  $\triangleq$  RouterTx(localnode, remnode)
274 IN
275      $\wedge$  RxMem' = [RxMem EXCEPT ![reqSource].service = "acknw",
276                 ![reqSource].destination = RxMem[reqSource].requestingtask,
277                 ![reqSource].status = "ok",
278                 ![reqSource].data = NoMessage]
280      $\wedge$  UserTaskMem' = [UserTaskMem EXCEPT ![storedReq].status = "ok",
281                       ![storedReq].data = RxMem[reqSource].data]
283      $\wedge$  ReadyList' = ReadyList  $\cup$  {storedReq}  $\cup$  {TxToUse}
284      $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToUse] = Append(@, reqSource)]
285      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Tail(@)]
286      $\wedge$  UNCHANGED  $\langle$ TaskController, Error, InitSetup, RxInputPort, WireMem $\rangle$ 
289      $\square$  ((reqSource  $\in$  RxAdr)  $\wedge$  (storedReq  $\in$  RxAdr))  $\rightarrow$  (remote,remote)

```

```

1. The requesting task is remote  $\rightarrow$  Update the packet (With the Data ERASED!!), send the Acknowledgment through Tx1
2. The stored task is remote  $\rightarrow$  Update the packet(With the Data COPIED!!), send the Acknowledgment through Tx2
3. Remove the stored request from the PortInputPort
4. Add Tx1 and Tx2 to the ReadyList (They may or not be the same!!!)
5. Append the requests in Tx1 and Tx2

```

```

296     LET
297     localnode  $\triangleq$  PortL(destin)
298     remnodeReq  $\triangleq$  UserTaskL(RxMem[reqSource].requestingtask)
299     remnodeSto  $\triangleq$  UserTaskL(RxMem[storedReq].requestingtask)
300     TxToReq  $\triangleq$  RouterTx(localnode, remnodeReq)
301     TxToSto  $\triangleq$  RouterTx(localnode, remnodeSto)
302 IN
303      $\wedge$  RxMem' = [RxMem EXCEPT ![reqSource].service = "acknw",
304                 ![reqSource].destination = RxMem[reqSource].requestingtask,
305                 ![reqSource].status = "ok",
306                 ![reqSource].data = NoMessage,
308                 ![storedReq].service = "acknw",
309                 ![storedReq].destination = RxMem[storedReq].requestingtask,
310                 ![storedReq].status = "ok",
311                 ![storedReq].data = RxMem[reqSource].data]
313      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Tail(@)]
315      $\wedge$  IF TxToReq = TxToSto THEN
316          $\wedge$  ReadyList' = ReadyList  $\cup$  {TxToReq}
317          $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToReq] = Append(Append(@, reqSource), storedReq)]
318          $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem $\rangle$ 
319     ELSE
320          $\wedge$  ReadyList' = ReadyList  $\cup$  {TxToReq}  $\cup$  {TxToSto}
321          $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToReq] = Append(@, reqSource),
322                           ![TxToSto] = Append(@, storedReq)]
323          $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem $\rangle$ 
325     ————— End of SYNCHRONIZATION —————
327     ELSE There are requests in the queue, but NOT complementary  $\rightarrow$  Store the request in the queue
329     (local  $\vee$  remote , local  $\vee$  remote) (There's no difference)
330      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
331      $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem $\rangle$ 
333     ELSE The Port is Empty  $\rightarrow$  Store the request in the queue (No difference from the previous!)
335     (local  $\vee$  remote , local  $\vee$  remote) (There's no difference)
336      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
337      $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem $\rangle$ 

```

```

340 |-----|
342 ServeReceive(requestPacket)  $\triangleq$ 
344 LET
345   reqSource  $\triangleq$  requestPacket.memoryaddress
346   destin  $\triangleq$  requestPacket.destination      ID of the port being used
347 IN
349 IF Len(PortInputPort[destin]) > 0 THEN  Port is not empty
350 LET
351   storedReq  $\triangleq$  Head(PortInputPort[destin])
352   storedPac  $\triangleq$  IF storedReq  $\in$  UserTask THEN UserTaskMem[storedReq] ELSE RxMem[storedReq]
353   storedReqType  $\triangleq$  storedPac.service
354 IN
355   IF storedReqType = "send" THEN  THERE's a complementary request waiting  $\rightarrow$  SYNCHRONIZE
358
359   CASE ((reqSource  $\in$  UserTask)  $\wedge$  (storedReq  $\in$  UserTask))  $\rightarrow$  (local,local)
360   1. Set the status of both requests to "ok" and COPY THE DATA FROM ONE PACKET TO THE OTHER!
361   2. Add both tasks to the ReadyList
362   3. Remove the stored request from the PortInputPort
363        $\wedge$  UserTaskMem' = [UserTaskMem EXCEPT ![reqSource].status = "ok",
364                               ![reqSource].data = UserTaskMem[storedReq].data,
365                               ![storedReq].status = "ok"]
366        $\wedge$  ReadyList' = ReadyList  $\cup$  {reqSource}  $\cup$  {storedReq}
367        $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Tail(@)]
368        $\wedge$  UNCHANGED  $\langle$ TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem $\rangle$ 
371
372    $\square$  ((reqSource  $\in$  UserTask)  $\wedge$  (storedReq  $\in$  RxAdr))  $\rightarrow$  (local,remote)
373   1. The requesting task is local  $\rightarrow$  set its status "ok" and COPY THE DATA FROM the PACKET !!!IN RxMem!!!
374   2. The stored task is remote  $\rightarrow$  Update the packet and send the Acknowledgment through Tx
375   3. Add the requesting task and Tx to the ReadyList
376   4. Append the request in Tx
377   5. Remove the stored request from the PortInputPort
378 LET
379   localnode  $\triangleq$  PortL(destin)
380   remnode  $\triangleq$  UserTaskL(RxMem[storedReq].requestingtask)
381   TxToUse  $\triangleq$  RouterTx(localnode, remnode)
382 IN
383    $\wedge$  UserTaskMem' = [UserTaskMem EXCEPT ![reqSource].status = "ok",
384                               ![reqSource].data = RxMem[storedReq].data]
385    $\wedge$  RxMem' = [RxMem EXCEPT ![storedReq].service = "acknw",
386                               ![storedReq].destination = RxMem[storedReq].requestingtask,
387                               ![storedReq].status = "ok",
388                               ![storedReq].data = NoMessage]
389   !!! Delete the data so that it doesn't have to be sent back!!!IS this worth??? It could also be
390   used to make a distinction between acknowledgments: WHENEVER There is Data Copy IT!
391    $\wedge$  ReadyList' = ReadyList  $\cup$  {reqSource}  $\cup$  {TxToUse}
392    $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToUse] = Append(@, storedReq)]
393    $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Tail(@)]
394    $\wedge$  UNCHANGED  $\langle$ TaskController, Error, InitSetup, RxInputPort, WireMem $\rangle$ 
396
397    $\square$  ((reqSource  $\in$  RxAdr)  $\wedge$  (storedReq  $\in$  UserTask))  $\rightarrow$  (remote,local)
398   1. The requesting task is remote  $\rightarrow$  Update the packet (WITH THE DATA COPIED!!) and send the Acknowledgment through
399   Tx
400   2. The stored task is local  $\rightarrow$  simply set its status "ok"
401   3. Add the stored task and Tx to the ReadyList
402   4. Append the request in Tx
403   5. Remove the stored request from the PortInputPort
404 LET
405   localnode  $\triangleq$  PortL(destin)

```



```

405     remnode  $\triangleq$  UserTaskL(RxMem[reqSource].requestingtask)
406     TxToUse  $\triangleq$  RouterTx(localnode, remnode)
407
408     IN
409      $\wedge$  RxMem' = [RxMem EXCEPT ![reqSource].service = "acknw",
410                ![reqSource].destination = RxMem[reqSource].requestingtask,
411                ![reqSource].status = "ok",
412                ![reqSource].data = UserTaskMem[storedReq].data]
413      $\wedge$  UserTaskMem' = [UserTaskMem EXCEPT ![storedReq].status = "ok"]
414      $\wedge$  ReadyList' = ReadyList  $\cup$  {storedReq}  $\cup$  {TxToUse}
415      $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToUse] = Append(@, reqSource)]
416      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Tail(@)]
417      $\wedge$  UNCHANGED  $\langle$ TaskController, Error, InitSetup, RxInputPort, WireMem $\rangle$ 
418
419      $\square$  ((reqSource  $\in$  RxAdr)  $\wedge$  (storedReq  $\in$  RxAdr))  $\rightarrow$  (remote,remote)
420
421     1. The requesting task is remote  $\rightarrow$  Update the packet (WITH THE DATA COPIED!!), send the Acknowledgment through Tx1
422     2. The stored task is remote  $\rightarrow$  Update the packet(With the Data Erased(worth ??)), send the Acknowledgment through Tx2
423     3. Remove the stored request from the PortInputPort
424     4. Add Tx1 and Tx2 to the ReadyList (They may or not be the same!!!)
425     5. Append the requests in Tx1 and Tx2
426
427     LET
428     localnode  $\triangleq$  PortL(destin)
429     remnodeReq  $\triangleq$  UserTaskL(RxMem[reqSource].requestingtask)
430     remnodeSto  $\triangleq$  UserTaskL(RxMem[storedReq].requestingtask)
431     TxToReq  $\triangleq$  RouterTx(localnode, remnodeReq)
432     TxToSto  $\triangleq$  RouterTx(localnode, remnodeSto)
433
434     IN
435      $\wedge$  RxMem' = [RxMem EXCEPT ![reqSource].service = "acknw",
436                ![reqSource].destination = RxMem[reqSource].requestingtask,
437                ![reqSource].status = "ok",
438                ![reqSource].data = RxMem[storedReq].data,
439                ![storedReq].service = "acknw",
440                ![storedReq].destination = RxMem[storedReq].requestingtask,
441                ![storedReq].status = "ok",
442                ![storedReq].data = NoMessage]
443
444      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Tail(@)]
445
446      $\wedge$  IF TxToReq = TxToSto THEN
447          $\wedge$  ReadyList' = ReadyList  $\cup$  {TxToReq}
448          $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToReq] = Append(Append(@, reqSource), storedReq)]
449          $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem $\rangle$ 
450     ELSE
451          $\wedge$  ReadyList' = ReadyList  $\cup$  {TxToReq}  $\cup$  {TxToSto}
452          $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![TxToReq] = Append(@, reqSource),
453                           ![TxToSto] = Append(@, storedReq)]
454          $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, Error, InitSetup, RxInputPort, WireMem $\rangle$ 
455
456     ————— End of SYNCHRONIZATION —————
457
458     ELSE There are requests in the queue, but NOT complementary  $\rightarrow$  Store the request in the queue
459
460     (local  $\vee$  remote , local  $\vee$  remote) (There's no difference)
461      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
462      $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem $\rangle$ 
463
464     ELSE The Port is Empty  $\rightarrow$  Store the request in the queue (No difference from the previous!)
465
466     (local  $\vee$  remote , local  $\vee$  remote) (There's no difference)
467      $\wedge$  PortInputPort' = [PortInputPort EXCEPT ![destin] = Append(@, reqSource)]
468      $\wedge$  UNCHANGED  $\langle$ TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem $\rangle$ 
469

```

471 **Actions**473  $StartSystem \triangleq$ 474  $\wedge$  Wakes up the *KernelTask*, *TxTask*, *RxTask* and (in this example!!!) a *UserTask*475  $\wedge$  *InitSetup* = 0476  $\wedge$  LET477  $firsttask \triangleq$  CHOOSE  $ut \in UserTask : TRUE$ 479  $AllTasksToStart \triangleq KernelTask \cup TxTask \cup RxTask \cup \{firsttask\}$ 481  $SetToSeq(set) \triangleq$ 482 LET  $TC[S \in SUBSET set] \triangleq$  LET  $elt \triangleq$  CHOOSE  $e \in S : TRUE$ 483 IN IF  $S = \{\}$  THEN  $\langle \rangle$ 484 ELSE  $Append(TC[S \setminus \{elt\}], elt)$ 485 IN  $TC[set]$ 487  $SeqTasks \triangleq SetToSeq(AllTasksToStart)$ 

489 IN

491  $\wedge TaskController' = [t \in Task \mapsto \text{IF } t \in AllTasksToStart \text{ THEN } StartedTaskCB \text{ ELSE } InitTaskCB]$ 493 **!!!TxTask is not put in the ReadyList, the KernelTask will do it when necessary !!!**494  $\wedge ReadyList' = KernelTask \cup RxTask \cup \{firsttask\}$ 495  $\wedge InitSetup' = 1$ 496  $\wedge$  UNCHANGED  $\langle UserTaskMem, KernelInputPort, Error, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort \rangle$ 

499 |-----|

501  $CreateStartTaskRequest(reqtask, desttask) \triangleq$ 502  $\wedge InitSetup = 1$ 503  $\wedge TaskController[reqtask].taskstate = \text{"started"}$  **!!! Pay attention to this! In this case the kernel could indefinitely accumulate requests, over**504  $\wedge reqtask \in ReadyList$ 505  $\wedge$  LET506  $requestPacket \triangleq$  CHOOSE  $p \in Packet : \wedge p.service = \text{"start"}$ 507  $\wedge p.requestingtask = reqtask$ 508  $\wedge p.destination = desttask$ 509  $\wedge p.status = \text{"blank"}$ 510  $\wedge p.data = NoMessage$ 511  $\wedge p.memoryaddress = reqtask$ 513  $localkernel \triangleq$  CHOOSE  $k \in KernelTask : KernelTaskL(k) = UserTaskL(reqtask)$ 

514 IN

515  $\wedge UserTaskMem' = [UserTaskMem \text{ EXCEPT } ![reqtask] = requestPacket]$ 516  $\wedge KernelInputPort' = [KernelInputPort \text{ EXCEPT } ![localkernel] = Append(@, reqtask)]$ 517  $\wedge ReadyList' = ReadyList \setminus \{reqtask\}$ 518  $\wedge$  UNCHANGED  $\langle TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort \rangle$ 

520 |-----|

522  $AllStartedUserTasks \triangleq \{ut \in UserTask : TaskController[ut].taskstate = \text{"started"}\}$ 523  $AllReadyUserTasks \triangleq \{ut \in UserTask : ut \in ReadyList\}$ 526  $CreateSendRequest(reqtask, port, message) \triangleq$ 527  $\wedge InitSetup = 1$ **!!!NEED TO CHECK THIS!!!THERE CAN NOT BE ONLY A STARTED TASK, OTHERWISE NO COMPLEMENTARY REQUEST WILL EVER BE CREATED AND THE TASK IS BLOCKED FOR EVER!**530  $\wedge Cardinality(AllStartedUserTasks) > 1$ 531  $\wedge Cardinality(AllReadyUserTasks) > 1$  **!!!NOT NECESSARILY!! If there would be complementary requests already generated we wo**533  $\wedge reqtask \in ReadyList$ 534  $\wedge$  LET535  $requestPacket \triangleq$  CHOOSE  $p \in Packet : \wedge p.service = \text{"send"}$ 536  $\wedge p.requestingtask = reqtask$ 537  $\wedge p.destination = port$



```

538          $\wedge p.status = \text{"blank"}$ 
539          $\wedge p.data = message$ 
540          $\wedge p.memoryaddress = reqtask$ 
542      $localkernel \triangleq \text{CHOOSE } k \in KernelTask : KernelTaskL(k) = UserTaskL(reqtask)$ 
543     IN
544      $\wedge UserTaskMem' = [UserTaskMem \text{ EXCEPT } ![reqtask] = requestPacket]$ 
545      $\wedge KernelInputPort' = [KernelInputPort \text{ EXCEPT } ![localkernel] = Append(@, reqtask)]$ 
546      $\wedge ReadyList' = ReadyList \setminus \{reqtask\}$ 
547      $\wedge \text{UNCHANGED } \langle TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort \rangle$ 
551  $CreateReceiveRequest(reqtask, port) \triangleq$ 
552      $\wedge InitSetup = 1$ 
553     !!!NEED TO CHECK THIS!!!THERE CAN NOT BE ONLY A STARTED TASK, OTHERWISE NO COMPLEMENTARY
554     REQUEST WILL EVER BE CREATED AND THE TASK IS BLOCKED FOR EVER!
555      $\wedge Cardinality(AllStartedUserTasks) > 1$ 
556      $\wedge Cardinality(AllReadyUserTasks) > 1$  !!!NOT NECESSARILY!! If there would be complementary requests already generated we would
557      $\wedge reqtask \in ReadyList$ 
558      $\wedge \text{LET}$ 
559      $requestPacket \triangleq \text{CHOOSE } p \in Packet : \wedge p.service = \text{"receive"}$ 
560      $\wedge p.requestingtask = reqtask$ 
561      $\wedge p.destination = port$ 
562      $\wedge p.status = \text{"blank"}$ 
563      $\wedge p.data = NoMessage$ 
564      $\wedge p.memoryaddress = reqtask$ 
565
566      $localkernel \triangleq \text{CHOOSE } k \in KernelTask : KernelTaskL(k) = UserTaskL(reqtask)$ 
567     IN
568      $\wedge UserTaskMem' = [UserTaskMem \text{ EXCEPT } ![reqtask] = requestPacket]$ 
569      $\wedge KernelInputPort' = [KernelInputPort \text{ EXCEPT } ![localkernel] = Append(@, reqtask)]$ 
570      $\wedge ReadyList' = ReadyList \setminus \{reqtask\}$ 
571      $\wedge \text{UNCHANGED } \langle TaskController, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortInputPort \rangle$ 
572
573 |-----|
576  $KernelLoop(localnode) \triangleq$ 
577      $\wedge InitSetup = 1$ 
578      $\wedge \text{IF } IsEmpty(KernelInputPort) \text{ THEN}$ 
579     UNCHANGED } \langle TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem \rangle
580     ELSE \setminus * Serve the pending request
581      $\wedge \text{LET}$ 
582      $localkernel \triangleq \text{CHOOSE } k \in KernelTask : KernelTaskL(k) = localnode$ 
583     IN
584      $\wedge Len(KernelInputPort[localkernel]) > 0$ 
585
586      $\wedge KernelInputPort' = [KernelInputPort \text{ EXCEPT } ![localkernel] = Tail(@)]$ 
587      $\wedge \text{LET}$ 
588      $requestEntry \triangleq Head(KernelInputPort[localkernel])$ 
589      $requestPacket \triangleq \text{IF } requestEntry \in UserTask \text{ THEN } UserTaskMem[requestEntry] \text{ ELSE } RxMem[requestEntry]$ 
590      $destin \triangleq requestPacket.destination$ 
591      $destnode \triangleq \text{IF } destin \in UserTask \text{ THEN } UserTaskL(destin) \text{ ELSE } PortL(destin)$ 
592     IN
593      $\text{IF } destnode \neq localnode \text{ THEN } \text{The request is for a remote usertask}$ 
594     LET
595      $TxToUse \triangleq RouterTx(localnode, destnode)$ 
596     IN
597      $\wedge ReadyList' = ReadyList \cup \{TxToUse\}$ 
598      $\wedge TxInputPort' = [TxInputPort \text{ EXCEPT } ![TxToUse] = Append(@, requestEntry)]$ 
599      $\wedge \text{UNCHANGED } \langle TaskController, UserTaskMem, Error, InitSetup, RxMem, RxInputPort, WireMem \rangle$ 
600
601     ELSE The request is for a local usertask

```

```

603         LET
604             requestService  $\triangleq$  requestPacket.service
605         IN
606              $\wedge$  CASE requestService = "start"  $\rightarrow$  ServeStartTask(requestPacket)
607                  $\square$  requestService = "acknw"  $\rightarrow$  ServeAcknowledgment(requestPacket)
608                  $\square$  requestService = "send"  $\rightarrow$  ServeSend(requestPacket)
609                  $\square$  requestService = "receive"  $\rightarrow$  ServeReceive(requestPacket)
610             Add the other services...

614 TxSendsPacketAway(txtask)  $\triangleq$ 
615      $\wedge$  txtask  $\in$  ReadyList
616      $\wedge$  LET
617         reqSource  $\triangleq$  Head(TxInputPort[txtask])
618         HLinkToUse  $\triangleq$  HLinkTx(txtask)
619     IN
620
621      $\wedge$  WireMem[HLinkToUse].Pac = EmptyPacket !!! See how to REMOVE THIS!!!!
622
623      $\wedge$  Read packet "through the pointer" and copy it directly to the Wire.
624
625      $\wedge$  TxInputPort' = [TxInputPort EXCEPT ![txtask] = Tail(@)]
626
627      $\wedge$  IF reqSource  $\in$  UserTask THEN The request is from a local task: simply put in on wire!
628         LET
629             !!!!Isto devia SAIR daqui!Sabemos Tx  $\Rightarrow$  Sabemos Destination!REVER 'Modelo de Rede'!!!!
630             reqPacket  $\triangleq$  UserTaskMem[reqSource]
631             destin  $\triangleq$  reqPacket.destination
632             destnode  $\triangleq$  IF destin  $\in$  UserTask THEN UserTaskL(destin) ELSE PortL(destin)
633         IN
634              $\wedge$  WireMem' = [WireMem EXCEPT ![HLinkToUse].Pac = reqPacket,
635                 ![HLinkToUse].Dest = destnode]
636
637          $\wedge$  IF IsEmpty (TxInputPort[txtask]') THEN
638              $\wedge$  IF Len(TxInputPort[txtask]) = 1 THEN
639                  $\wedge$  ReadyList' = ReadyList \ {txtask}
640                  $\wedge$  UNCHANGED  $\langle$  TaskController, UserTaskMem, KernelInputPort, Error, InitSetup, RxMem, RxInputPort, P
641             ELSE
642                 UNCHANGED  $\langle$  TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, RxMem
643
644         ELSE reqSource  $\in$  RxAdr  $\Rightarrow$  The request is remote: !!!!!!!Make RxMem free AND Update Request Packet!!!!!!
645
646         LET
647             reqPacket  $\triangleq$  RxMem[reqSource]
648             destin  $\triangleq$  reqPacket.destination
649             destnode  $\triangleq$  IF destin  $\in$  UserTask THEN UserTaskL(destin) ELSE PortL(destin)
650         IN
651              $\wedge$  WireMem' = [WireMem EXCEPT ![HLinkToUse].Pac = reqPacket,
652                 ![HLinkToUse].Dest = destnode]
653              $\wedge$  RxMem' = [RxMem EXCEPT ![reqSource] = EmptyPacket]
654          $\wedge$  IF IsEmpty (TxInputPort'[txtask]) THEN
655              $\wedge$  IF Len(TxInputPort[txtask]) = 1 THEN
656                  $\wedge$  ReadyList' = ReadyList \ {txtask}
657                  $\wedge$  UNCHANGED  $\langle$  TaskController, UserTaskMem, KernelInputPort, Error, InitSetup, RxInputPort, P
658             ELSE
659                 UNCHANGED  $\langle$  TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, RxInputPort, P

664 HLinkPutsPacketInRx(hlink)  $\triangleq$ 
665      $\wedge$  InitSetup = 1
666
667      $\wedge$  WireMem[hlink].Pac  $\in$  Packet Not an EmptyPacket
668
669      $\wedge$  LET

```

```

670     reqPacket  $\triangleq$  WireMem[hlink].Pac
671     destnode  $\triangleq$  WireMem[hlink].Dest
672     RxToUse  $\triangleq$  CHOOSE  $rx \in RxTask : (rx \in RxHLink(hlink) \wedge RxTaskL(rx) = destnode)$ 
673     IN
674     !!! Like this it's always guaranteed that a packet will only arrive when there's memory available for it!!! What to do otherwise???
675      $\wedge \exists ra \in RxAdr : (RxMem[ra] = EmptyPacket \wedge RxAdrL(ra) = destnode)$ 
676      $\wedge$  LET
677         RxAdrToFill  $\triangleq$  CHOOSE  $ra \in RxAdr : (RxMem[ra] = EmptyPacket \wedge RxAdrL(ra) = destnode)$ 
678     IN
679      $\wedge RxMem' = [RxMem \text{ EXCEPT } ![RxAdrToFill] = reqPacket]$ 
680      $\wedge RxInputPort' = [RxInputPort \text{ EXCEPT } ![RxToUse] = Append(@, RxAdrToFill)]$ 
681      $\wedge WireMem' = [WireMem \text{ EXCEPT } ![hlink].Pac = EmptyPacket,$ 
682          $![hlink].Dest = NoNode]$ 
683      $\wedge$  UNCHANGED  $\langle TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, TxInputPort, WireMem, PortList \rangle$ 

```

```

687 RxLoop(rxtask)  $\triangleq$ 
688      $\wedge InitSetup = 1$ 
689      $\wedge$  IF IsEmpty(RxInputPort) THEN
690         UNCHANGED  $\langle TaskController, UserTaskMem, KernelInputPort, ReadyList, Error, InitSetup, TxInputPort, RxMem, RxInputPort, WireMem, PortList \rangle$ 
691     ELSE \* Send the requests to the KernelInputPort
692      $\wedge$  LET
693         localnode  $\triangleq$  RxTaskL(rxtask)
694         localkernel  $\triangleq$  CHOOSE  $k \in KernelTask : KernelTaskL(k) = localnode$ 
695     IN
696      $\wedge Len(RxInputPort[rxtask]) > 0$ 
697      $\wedge$  LET
698         RxMemEntry  $\triangleq$  Head(RxInputPort[rxtask])
699         reqPacket  $\triangleq$  RxMem[RxMemEntry]
700         UpdatedReqPacket  $\triangleq$  [reqPacket EXCEPT !.memoryaddress = RxMemEntry]
701     IN
702     !!!!!!!!!!!!!Update the Packet in RxMem!!!!!!!!!!!!
703      $\wedge RxMem' = [RxMem \text{ EXCEPT } ![RxMemEntry] = UpdatedReqPacket]$ 
704     The pointer to the packet is moved from the RxInputPort to the KernelInputport
705      $\wedge KernelInputPort' = [KernelInputPort \text{ EXCEPT } ![localkernel] = Append(@, RxMemEntry)]$ 
706      $\wedge RxInputPort' = [RxInputPort \text{ EXCEPT } ![rxtask] = Tail(@)]$ 
707      $\wedge$  UNCHANGED  $\langle TaskController, UserTaskMem, ReadyList, Error, InitSetup, TxInputPort, WireMem, PortList \rangle$ 

```

---

```

711 |
712 | Summary
713 |
714 | Next  $\triangleq$ 
715 |      $\vee StartSystem$ 
716 |      $\vee \exists n \in Node : KernelLoop(n)$ 
717 |      $\vee \exists txt \in TxTask : TxSendsPacketAway(txt)$ 
718 |      $\vee \exists rxt \in RxTask : RxLoop(rxt)$ 
719 |      $\vee \exists hl \in HLink : HLinkPutsPacketInRx(hl)$ 
720 |      $\vee \exists uts \in UserTask :$ 
721 |          $\vee \exists utd \in UserTask :$ 
722 |             CreateStartTaskRequest(uts, utd)
723 |          $\vee \exists p \in Port :$ 
724 |              $\vee \exists d \in Message :$ 
725 |                 CreateSendRequest(uts, p, d)
726 |              $\vee CreateReceiveRequest(uts, p)$ 
727 |
728 |
729 |
730 |
731 |
732 |
733 |
734 |

```

---

738  $LivenessW \triangleq$   
739  $\wedge WF_{\langle vars \rangle}(StartSystem)$   
740  $\wedge \forall n \in Node : WF_{\langle vars \rangle}(KernelLoop(n))$   
741  $\wedge \forall txt \in TxTask : WF_{\langle vars \rangle}(TxSendsPacketAway(txt))$   
742  $\wedge \forall rxt \in RxTask : WF_{\langle vars \rangle}(RxLoop(rxt))$   
743  $\wedge \forall hl \in HLink : WF_{\langle vars \rangle}(HLinkPutsPacketInRx(hl))$   
744  $\wedge \forall uts \in UserTask :$   
745  $\quad \wedge \forall utd \in UserTask :$   
746  $\quad \quad WF_{\langle vars \rangle}(CreateStartTaskRequest(uts, utd))$   
747  $\quad \wedge \forall p \in Port :$   
748  $\quad \quad \wedge \forall d \in Message :$   
749  $\quad \quad \quad WF_{\langle vars \rangle}(CreateSendRequest(uts, p, d))$   
750  $\quad \quad \wedge WF_{\langle vars \rangle}(CreateReceiveRequest(uts, p))$

754  $Spec \triangleq Init \wedge \square[Next]_{vars} \wedge LivenessW$

756 |-----|  
757 **Correctness conditions**

760 **All tasks in the ReadyList have to be "started"**  
761  $P2 \triangleq \forall t \in Task : t \in ReadyList \Rightarrow TaskController[t].taskstate = \text{"started"}$

764 **TxTask is in the ReadyList if and only if it has requests to serve**  
765  $P4 \triangleq \forall txt \in TxTask : txt \in ReadyList \equiv Len(TxInputPort[txt]) > 0$

767 **RxTask is always in the ReadyList (after initialization)**  
768  $P5 \triangleq InitSetup = 1 \Rightarrow \forall rxt \in RxTask : rxt \in ReadyList$

771 **A PortInputPort never has complementary requests**  
772  $P9 \triangleq LET$   
773  $\quad pacsto(x) \triangleq IF x \in UserTask THEN UserTaskMem[x] ELSE RxMem[x]$   
774  $\quad IN$   
775  $\quad \forall p \in Port : Len(PortInputPort[p]) > 1 \Rightarrow \forall i, j \in 1 .. Len(PortInputPort[p]) :$   
776  $\quad \quad pacsto(PortInputPort[p][i]).service = pacsto(PortInputPort[p][j]).service$

778 **A Port only gets "send" and "receive" requests**  
779  $P10 \triangleq LET$   
780  $\quad pacsto(x) \triangleq IF x \in UserTask THEN UserTaskMem[x] ELSE RxMem[x]$   
781  $\quad IN$   
782  $\quad \forall p \in Port : Len(PortInputPort[p]) > 0 \Rightarrow \forall i \in 1 .. Len(PortInputPort[p]) :$   
783  $\quad \quad pacsto(PortInputPort[p][i]).service \in \{\text{"send"}, \text{"receive"}\}$

787 **Queues only have distinct elements**  
789  $P11 \triangleq \wedge \forall p \in Port : Len(PortInputPort[p]) > 1 \Rightarrow \forall i, j \in 1 .. Len(PortInputPort[p]) :$   
790  $\quad PortInputPort[p][i] = PortInputPort[p][j] \Rightarrow i = j$

793  $P12 \triangleq \wedge \forall rx \in RxTask : Len(RxInputPort[rx]) > 1 \Rightarrow \forall i, j \in 1 .. Len(RxInputPort[rx]) :$   
794  $\quad RxInputPort[rx][i] = RxInputPort[rx][j] \Rightarrow i = j$

796 **Queues only have distinct elements — A request is never repeated in any two places (InputPorts) at the same time**

797  $P13 \triangleq LET$   
798  $\quad SepSeq(seq) \triangleq$   
799  $\quad \quad LET YY[F \in SUBSET DOMAIN seq] \triangleq$   
800  $\quad \quad \quad LET ind \triangleq CHOOSE nr \in F : TRUEIN$   
801  $\quad \quad \quad \quad IF F = \{\} THEN \langle \rangle$   
802  $\quad \quad \quad \quad ELSE Append(YY[F \setminus \{ind\}], seq[ind])$   
803  $\quad \quad IN YY[DOMAIN seq]$   
805  $\quad TotalQ(set, inputport) \triangleq$

```

806     LET  $XX[S \in \text{SUBSET } set] \triangleq$ 
807     LET  $elt \triangleq \text{CHOOSE } e \in S : \text{TRUE}$  IN
808     IF  $S = \{\}$  THEN  $\langle \rangle$ 
809     ELSE
810         IF  $\text{Len}(\text{inputport}[elt]) > 0$  THEN
811              $\text{Append}(XX[S \setminus \{elt\}], \text{SepSeq}(\text{inputport}[elt]))$ 
812         ELSE  $XX[S \setminus \{elt\}]$ 
813     IN  $XX[set]$ 

815      $AllKerQ \triangleq \text{TotalQ}(\text{KernelTask}, \text{KernelInputPort})$ 
817      $AllRxQ \triangleq \text{TotalQ}(\text{RxTask}, \text{RxInputPort})$ 
819      $AllTxQ \triangleq \text{TotalQ}(\text{TxTask}, \text{TxInputPort})$ 
821      $AllPortQ \triangleq \text{TotalQ}(\text{Port}, \text{PortInputPort})$ 
823      $GeneralQ \triangleq AllKerQ \circ AllRxQ \circ AllTxQ \circ AllPortQ$ 

825     IN
826      $\text{Len}(GeneralQ) > 1 \Rightarrow \forall i, j \in 1 .. \text{Len}(GeneralQ) :$ 
827      $GeneralQ[i] = GeneralQ[j] \Rightarrow i = j$ 

831     "Every  $XInputPort$  can contain only local pointers"
832      $P14 \triangleq \text{LET}$ 
833      $location(a) \triangleq \text{IF } a \in \text{UserTask} \text{ THEN } \text{UserTaskL}(a) \text{ ELSE } \text{RxAdrL}(a)$ 
834     IN
835      $\wedge \forall p \in \text{Port} : \text{Len}(\text{PortInputPort}[p]) > 0 \Rightarrow \forall i \in 1 .. \text{Len}(\text{PortInputPort}[p]) :$ 
836      $location(\text{PortInputPort}[p][i]) = \text{PortL}(p)$ 

838      $\wedge \forall k \in \text{KernelTask} : \text{Len}(\text{KernelInputPort}[k]) > 0 \Rightarrow \forall i \in 1 .. \text{Len}(\text{KernelInputPort}[k]) :$ 
839      $location(\text{KernelInputPort}[k][i]) = \text{KernelTaskL}(k)$ 

841      $\wedge \forall txt \in \text{TxTask} : \text{Len}(\text{TxInputPort}[txt]) > 0 \Rightarrow \forall i \in 1 .. \text{Len}(\text{TxInputPort}[txt]) :$ 
842      $location(\text{TxInputPort}[txt][i]) = \text{TxTaskL}(txt)$ 

844      $\wedge \forall rxt \in \text{RxTask} : \text{Len}(\text{RxInputPort}[rxt]) > 0 \Rightarrow \forall i \in 1 .. \text{Len}(\text{RxInputPort}[rxt]) :$ 
845      $location(\text{RxInputPort}[rxt][i]) = \text{RxTaskL}(rxt)$ 

848 |

```