```
machine Mach_IPC
//* ************************************************
//    The Event-B model of ARINC 653 Part 1
//    Created by Yongwang Zhao ( zhaoyongwang@gmail.com)
//    National Key Laboratory of Software Development Environment (NLSDE)
//    School of Computer and Engineering, Beihang University, Beijing, China
//    ************************************************/
refines Mach_IPC_Conds   sees Ctx_IPC

variables processes processes_of_partition  partition_mode process_state periodtype_of_process
          process_wait_type // mainproc_of_partition // the only one main proc of each partition
          locklevel_of_partition

          /* denotes the current lock level of the partition
              preemption_of_partitions */
          startcondition_of_partition
          /* denotes the reason the partition is started
              schedulable_of_partition //the scheduling of a partition is activated or disactivated? */
          basepriority_of_process  // Denotes the capability of the process to manipulate other processes.
          period_of_process // Identifies the period of activation for a periodic process. A distinct and unique
value should be specified to designate the process as aperiodic
```

timecapacity_of_process *// Defines the elapsed time within which the process should complete its execution.*

deadline_of_process *// Specifies the type of deadline relating to the process, and may be "hard" or "soft".*

currentpriority_of_process *// Defines the priority with which the process may access and receive resources. It is set to base priority at initialization time and is dynamic at runtime.*

deadlinetime_of_process *// The deadline time is periodically evaluated by the operating system to determine whether the process is satisfactorily completing its processing within the allotted time.*

releasepoint_of_process

/ the release point of processes*

*nextreleasepoint_of_process // the next release point of processes */*

delaytime_of_process *// if the proc is delayed started, the delaytime should be saved(used when parttion START --> NORMAL)*

current_partition *// the partition in which a thread is now running. at each time, only one thread is running*

current_process

current_partition_flag *// true:indicate that the current_partition is valid, false: indicate NULL (unavailable)*

current_process_flag *// same as current partition flag*

clock_tick *// system clock ticks*

need_reschedule *// indicate the flag to reschedule after some events, for example suspend a thread*

need_procresch

preempter_of_partition *// the process who execute the lock_preemption (increase the locklevel and disable scheduling), at most one preempter proc in a partition*

timeout_trigger *// all processes waiting for resources with a timeout, will be triggered after the timeout ellapsed.*

errorhandler_of_partition *// each partition has one error handler at most. other error handler can be created only after the previous handler is finished*

process_call_errorhandler

/ error handler is created by a process, then the process is preempted by the error handler
for inter-partition communication */*

ports *// the set of created ports*

RefreshPeriod_of_SamplingPorts

msgspace_of_samplingports

/ the only one msg space of sampling ports
lastwritetime_of_samplingports // */*

needtrans_of_sourcesamplingport *// indicate whether the msg in the source port has been transfered to dest ports?*

queue_of_queueingports quediscipline_of_queueingports

processes_waitingfor_queuingports *// for intra-partition communication*

buffers blackboards semaphores events_ buffers_of_partition blackboards_of_partition

semaphores_of_partition events_of_partition MaxMsgNum_of_Buffers queue_of_buffers
processes_waitingfor_buffers quediscipline_of_buffers msgspace_of_blackboards emptyindicator_of_blackboards
processes_waitingfor_blackboards MaxValue_of_Semaphores value_of_semaphores quediscipline_of_semaphores
processes_waitingfor_semaphores state_of_events processes_waitingfor_events used_messages

invariants

@inv_refreshprd_of_sampports RefreshPeriod_of_SamplingPorts ∈ **SamplingPorts** ⇸ ℕ1 *// partial function, the value will be assigned when created*

@inv_flag_sourcesampport needtrans_of_sourcesamplingport ∈ **SamplingPorts** ⇸ BOOL

@inv_flag_means_msg ∀*p*·(*p*∈**Source_SamplingPorts** ⇒ (*p*∈dom(needtrans_of_sourcesamplingport) ∧ needtrans_of_sourcesamplingport(*p*) = TRUE ⇒ *p*∈dom(msgspace_of_samplingports)))

@inv_noflag_means_nomsg ∀*p*·(*p*∈**Source_SamplingPorts** ⇒ (*p*∈dom(needtrans_of_sourcesamplingport) ∧ needtrans_of_sourcesamplingport(*p*) = FALSE ⇒ *p* ∉ dom(msgspace_of_samplingports)))

@inv_quediscipline_of_queueingports quediscipline_of_queueingports∈**QueuingPorts**∩ports→ **QUEUING_DISCIPLINE** *//partial function, the value will be assigned when created*

@inv_quediscipline_of_buffers quediscipline_of_buffers∈buffers→**QUEUING_DISCIPLINE**

@inv_quediscipline_of_semaphores quediscipline_of_semaphores∈semaphores→**QUEUING_DISCIPLINE**

@inv_procswfbuf_part ∀*buf*·(*buf*∈buffers ⇒ (∀p,tp,t,m·(p↦(tp↦t)↦m∈processes_waitingfor_buffers~[{*buf*}] ⇒ processes_of_partition(p)=buffers_of_partition(*buf*))))

@inv_procswfblkb_part ∀*bb*·(*bb*∈blackboards ⇒ (∀p·(p∈processes_waitingfor_blackboards~[{*bb*}] ⇒ processes_of_partition(p)=blackboards_of_partition(*bb*))))

@inv_procstate_waitfor_semophare_part ∀*sem*·(*sem*∈semaphores ⇒ (∀p,t·(p↦t∈ processes_waitingfor_semaphores~[{*sem*}] ⇒ processes_of_partition(p)=semaphores_of_partition(*sem*))))

@inv_procswfevts_part ∀*ev*·(*ev*∈events_ ⇒ (∀p·(p∈processes_waitingfor_events~[{*ev*}] ⇒ processes_of_partition(p)=events_of_partition(*ev*))))

@inv_procstate_waitfor_queport ∀*p*,*t*,*m*·((*p*↦*t*)↦*m*∈dom(processes_waitingfor_queuingports) ⇒ (process_state(*p*) = **PS_Waiting** ∨ process_state(*p*) = **PS_WaitandSuspend**)) //process waiting for queports is in WAITING or WAIT_SUSPEND state

@inv_procstate_waitfor_buffer ∀*p*,*t*,*n*,*m*,*b*·((*p*↦(*t*↦*n*)↦*m*)↦*b*∈processes_waitingfor_buffers ⇒ (process_state(*p*) = **PS_Waiting** ∨ process_state(*p*) = **PS_WaitandSuspend**)) //process waiting for queports is in WAITING or WAIT_SUSPEND state

@inv_procstate_waitfor_blkbrd ∀*p*·(*p*∈dom(processes_waitingfor_blackboards) ⇒ (process_state(*p*) = **PS_Waiting** ∨ process_state(*p*) = **PS_WaitandSuspend**))//process waiting for queports is in WAITING or WAIT_SUSPEND state

@inv_procstate_waitfor_semophare ∀*p*,*t*·(*p*↦*t*∈dom(processes_waitingfor_semaphores) ⇒ (process_state(*p*) = **PS_Waiting** ∨ process_state(*p*) = **PS_WaitandSuspend**)) //process waiting for queports is in WAITING or WAIT_SUSPEND state

@inv_procstate_waitfor_event ∀*p*·(*p*∈dom(processes_waitingfor_events) ⇒ (process_state(*p*) = **PS_Waiting** ∨ process_state(*p*) =

PS_WaitandSuspend))*//process waiting for queports is in WAITING or WAIT_SUSPEND state*

events

event INITIALISATION **extends** INITIALISATION

then

@act400 RefreshPeriod_of_SamplingPorts ≔ ∅

@act401 needtrans_of_sourcesamplingport ≔ ∅

@act402 quediscipline_of_queueingports ≔ ∅

@act407 quediscipline_of_buffers ≔ ∅

@act408 quediscipline_of_semaphores ≔ ∅

end

event create_sampling_port **refines** create_sampling_port

any *port refresh*

where

@grd01 current_partition_flag = TRUE ∧ (partition_mode(current_partition)=**PM_COLD_START** ∨ partition_mode(current_partition)=**PM_WARM_START**)

@grd02 *port*∈**PORTS**∖ports

@grd03 *port*∈**SamplingPorts**

@grd04 **Ports_of_Partition**(*port*) = current_partition
@grd05 *refresh*∈ℕ1
@grd06 partition_mode(current_partition)≠**PM_NORMAL**
then
@act01 RefreshPeriod_of_SamplingPorts(*port*)≔*refresh*
@act02 ports ≔ ports∪{*port*}
end

event write_sampling_message **refines** write_sampling_message
any *port msg*
where
@grd01 *port*∈ports
@grd00 current_partition_flag = TRUE ∧ **Ports_of_Partition**(*port*) = current_partition
@grd02 *port*∈**SamplingPorts**
@grd03 **Direction_of_Ports**(*port*)=**PORT_SOURCE**
@grd04 *msg*∈**MESSAGES**∖used_messages
then
@act02 msgspace_of_samplingports(*port*) ≔ *msg* ↦ clock_tick ∗ **ONE_TICK_TIME** *// @act03 lastwritetime_of_samplingports(port) ≔ clock_tick ∗ ONE_TICK_TIME*
@act04 needtrans_of_sourcesamplingport(*port*) ≔ TRUE
@act05 used_messages ≔ used_messages ∪ {*msg*}

```
  end

  event transfer_sampling_msg
    any p m t
    where
      @grd02 p∈SamplingPorts ∧ p∈ports
      @grd03 m ∈ MESSAGES ∧ p∈dom(msgspace_of_samplingports) ∧ m↠
t=msgspace_of_samplingports(p) // @grd04 p = SourcePort_of_Channels(c)
      @grd05 p∈dom(needtrans_of_sourcesamplingport) ∧ needtrans_of_sourcesamplingport(p) = TRUE
      @grd06 Sampling_Channels~[{p}] ⊆ ports // sampling ports has been created
    then
      @act01 needtrans_of_sourcesamplingport(p) ≔ FALSE
      @act02 msgspace_of_samplingports ≔ msgspace_of_samplingports  (Sampling_Channels~[{p}] × {m↠
t})
  end

  event read_sampling_message refines read_sampling_message
    any port m t
    where
      @grd01 port∈ports
      @grd02 port∈SamplingPorts
```

```
    @grd00 current_partition_flag = TRUE ∧ Ports_of_Partition(port) = current_partition
    @grd03 Direction_of_Ports(port)=PORT_DESTINATION
    @grd04 port∈dom(msgspace_of_samplingports) ∧ (m↦t)=msgspace_of_samplingports(port)
    @grd05 t+RefreshPeriod_of_SamplingPorts(port)≥clock_tick * ONE_TICK_TIME // the time from creating
to now should be in refresh period
  end

  event get_sampling_port_id
    any port
    where
      @grd01 port∈SamplingPorts ∧ port∈ports
      @grd00 current_partition_flag = TRUE ∧ Ports_of_Partition(port) = current_partition
  end

  event get_sampling_port_status
    any port
    where
      @grd01 port∈SamplingPorts ∧ port∈ports
      @grd00 current_partition_flag = TRUE ∧ Ports_of_Partition(port) = current_partition
  end
```

event create_queuing_port **refines** create_queuing_port

any *port discipline*

where

@grd02 *port*∈**PORTS**∖ports

@grd03 *port*∈**QueuingPorts**

@grd01 current_partition_flag = TRUE ∧ (partition_mode(current_partition)=**PM_COLD_START** ∨ partition_mode(current_partition)=**PM_WARM_START**)

@grd04 **Ports_of_Partition**(*port*) = current_partition

@grd05 *discipline*∈**QUEUING_DISCIPLINE**

@grd06 partition_mode(current_partition)≠**PM_NORMAL**

then

@act01 quediscipline_of_queueingports(*port*) ≔ *discipline*

@act02 ports ≔ ports∪{*port*}

end

event send_queuing_message **refines** send_queuing_message

any *port msg*

where

@grd01 *port*∈ports

@grd02 *port*∈**QueuingPorts**

@grd00 current_partition_flag = TRUE ∧ **Ports_of_Partition**(*port*) = current_partition

@grd03 **Direction_of_Ports**(*port*)=**PORT_SOURCE**

@grd04 *msg*∈**MESSAGES**∖used_messages

@grd05 card(queue_of_queueingports(*port*))<**MaxMsgNum_of_QueuingPorts**(*port*) *// there is sufficient space in the port's message queue to accept the message*

@grd06 processes_waitingfor_queuingports~[{*port*}] = ∅ *// no other process is waiting to send a message to that port*

then

@act01 queue_of_queueingports(*port*) ≔ queue_of_queueingports(*port*) ∪ {*msg*↦clock_tick ∗ **ONE_TICK_TIME**}

@act05 used_messages ≔ used_messages ∪ {*msg*}

end

event send_queuing_message_needwait *//extends req_busy_resource*

refines send_queuing_message_needwait

any *part proc newstate wt timeout tmout_trig port msg*

where

@grd40 current_partition_flag = TRUE ∧ current_process_flag = TRUE

@grd41 *part* = current_partition ∧ *proc* = current_process ∧ *newstate* = **PS_Waiting**

@grd43 *wt*∈**PROCESS_WAIT_TYPES** ∧ (*wt*= **PROC_WAIT_OBJ** ∨ *wt*=**PROC_WAIT_TIMEOUT**)

//@grd06 tmout >0 ∨ tmout = INFINITE_TIME_VALUE

//this line is correct, the next line is from ARINC653

@grd44 *timeout* ≠0

@grd45 *tmout_trig* ∈ processes ⇸ (**PROCESS_STATES** × ℕ1)

@grd46 (*timeout* = **INFINITE_TIME_VALUE** ⇒ *tmout_trig* = ∅)

∧ (*timeout* ≠**INFINITE_TIME_VALUE** ⇒ *tmout_trig* = {current_process↦(**PS_Ready**↦ (*timeout* +clock_tick ∗ **ONE_TICK_TIME**))})

@grd47 *timeout*≠**INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_TIMEOUT**

@grd48 *timeout* = **INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_OBJ**

@grd51 *port*∈ports

@grd52 *port*∈**QueuingPorts**

@grd50 **Ports_of_Partition**(*port*) = current_partition

@grd53 **Direction_of_Ports**(*port*)=**PORT_SOURCE**

@grd54 *msg*∈**MESSAGES**∖used_messages

@grd55 card(queue_of_queueingports(*port*))=**MaxMsgNum_of_QueuingPorts**(*port*) ∨ processes_waitingfor_queuingports~[{*port*}] ≠ ∅

@grd57 locklevel_of_partition(current_partition)=0 ∧ (current_partition∈dom(errorhandler_of_partition) ⇒ current_process≠errorhandler_of_partition(current_partition))

then

@act41 need_reschedule ≔ TRUE

@act42 current_process_flag ≔ FALSE

@act43 process_wait_type(current_process) ≔ *wt*

@act45 timeout_trigger ≔ timeout_trigger *tmout_trig*

@act52 processes_waitingfor_queuingports ≔ processes_waitingfor_queuingports ∪ {(*proc*↦clock_tick ∗ **ONE_TICK_TIME**↦*msg*)↦*port*}

@act55 used_messages ≔ used_messages ∪ {*msg*}

@act56 process_state(*proc*) ≔ *newstate*

end

event transfer_queuing_msg

any *p m t*

where

@grd00 current_partition_flag = TRUE ∧ partition_mode(current_partition)=**PM_NORMAL** *// @grd01 c ∈ Queuing_Channels*

@grd02 *p* ∈ **QueuingPorts** ∧ *p*∈ports *// @grd03 p = SourcePort_of_Channels(c)*

@grd04 *m* ∈ **MESSAGES**

@grd05 *m*↦*t* ∈ queue_of_queueingports(*p*) ∧ (∀m1,t1·(m1↦t1∈queue_of_queueingports(*p*) ⇒ *t*≤t1))

@grd06 card(queue_of_queueingports(*p*)) ≤ **MaxMsgNum_of_QueuingPorts**(*p*) ∧ card(queue_of_queueingports(*p*)) > 0 ∧

processes_waitingfor_queuingports~[{*p*}] =∅ // there is not waiting process

@grd07 ∀*pt*·(*pt*∈ **Queuing_Channels**~[{*p*}]⇒card(queue_of_queueingports(*pt*))<**MaxMsgNum_of_QueuingPorts**(*pt*)) // buffer of each destination port should not be full

@grd08 **Queuing_Channels**~[{*p*}] ⊆ ports

/* ports of the channel has been created
the next lines should not be commented */

then

@act01 queue_of_queueingports :| queue_of_queueingports'(*p*)=queue_of_queueingports(*p*)∖{*m*↦*t*} ∧
(∀pt·((pt∈**Queuing_Channels**~[{*p*}]⇒queue_of_queueingports'(pt)=queue_of_queueingports(pt)∪{*m*↦*t*})
∧ (pt ∉ **Queuing_Channels**~[{*p*}]⇒queue_of_queueingports'(pt)=queue_of_queueingports(pt))))

end

event wakeup_waitproc_on_srcqueports //extends resource_become_available

refines wakeup_waitproc_on_srcqueports

any *part proc newstate resch port msg t*

where

@grd500 current_partition_flag = TRUE ∧ partition_mode(current_partition)=**PM_NORMAL**

@grd509 *part* = current_partition // @grd501 ch ∈ CHANNELS

@grd02 *proc* ∈ processes

@grd03 *newstate* ∈ **PROCESS_STATES**

@grd06 processes_of_partition(*proc*) = *part*

@grd31 partition_mode(*part*) = **PM_NORMAL**

@grd32 process_state(*proc*) = **PS_Waiting** ∨ process_state(*proc*) = **PS_WaitandSuspend**

@grd33 process_state(*proc*) = **PS_Waiting** ⇒ *newstate* = **PS_Ready**

@grd34 process_state(*proc*) = **PS_WaitandSuspend** ⇒ *newstate* = **PS_Suspend**

@grd40 process_wait_type(*proc*)= **PROC_WAIT_OBJ**

@grd510 *resch* = TRUE

@grd502 *port* ∈ **Source_QueuingPorts** ∧ *port*∈ports

@grd504 card(queue_of_queueingports(*port*)) < **MaxMsgNum_of_QueuingPorts**(*port*)

@grd506 (*proc*↦*t*↦ *msg*) ∈processes_waitingfor_queuingports~[{*port*}]

@grd507 quediscipline_of_queueingports(*port*)=**QUEUE_FIFO** ⇒ (∀p1,t1,m·(p1↦t1↦m∈ processes_waitingfor_queuingports~[{*port*}] ⇒ *t*≤t1))

@grd508 quediscipline_of_queueingports(*port*)=**QUEUE_PRIORITY** ⇒ (∀p1,t1,m·(p1↦t1↦m∈ processes_waitingfor_queuingports~[{*port*}] ⇒ currentpriority_of_process(*proc*)≥currentpriority_of_process(p1)))

then

@act41 process_wait_type ≔ {*proc*}⩤process_wait_type

@act42 timeout_trigger ≔ {*proc*}⩤timeout_trigger

@act43 need_reschedule ≔ *resch*

@act501 processes_waitingfor_queuingports ≔ processes_waitingfor_queuingports ∖ {(*proc*↦*t*↦ *msg*)↦

```
port}

        //the next line is commented according to ARINC653 SENT_QUEUING_MSG operation. In face, it should not
be commented
        //@act506 queue_of_queueingports(port) ≔ queue_of_queueingports(port) ∪{msg ↦ clock_tick ∗
ONE_TICK_TIME}

        @act11 process_state(proc) ≔ newstate
    end

    event wakeup_waitproc_on_destqueports //extends resource_become_available
    refines wakeup_waitproc_on_destqueports
      any part proc newstate resch port msg t msg1 t1
      where
        @grd500 current_partition_flag = TRUE ∧ partition_mode(current_partition)=PM_NORMAL
        @grd503 part = current_partition // @grd501 ch ∈ CHANNELS
        @grd02 proc ∈ processes
        @grd03 newstate ∈ PROCESS_STATES
        @grd06 processes_of_partition(proc) = part
        @grd31 partition_mode(part) = PM_NORMAL
        @grd32 process_state(proc) = PS_Waiting ∨ process_state(proc) = PS_WaitandSuspend
```

@grd33 process_state(*proc*) = **PS_Waiting** ⇒ *newstate* = **PS_Ready**

@grd34 process_state(*proc*) = **PS_WaitandSuspend** ⇒ *newstate* = **PS_Suspend**

@grd40 process_wait_type(*proc*)= **PROC_WAIT_OBJ**

@grd501 *resch* = TRUE

@grd502 *port* ∈ **Dest_QueuingPorts** ∧ *port*∈ports

@grd504 card(queue_of_queueingports(*port*)) > 0

@grd506 *proc*↦*t*↦*msg* ∈processes_waitingfor_queuingports~[{*port*}]

@grd507 quediscipline_of_queueingports(*port*)=**QUEUE_FIFO** ⇒ (∀p1,tt,m·(p1↦tt↦m∈ processes_waitingfor_queuingports~[{*port*}] ⇒ *t*≤tt))

@grd508 quediscipline_of_queueingports(*port*)=**QUEUE_PRIORITY** ⇒ (∀p1,tt,m·(p1↦tt↦m∈ processes_waitingfor_queuingports~[{*port*}] ⇒ currentpriority_of_process(*proc*)≥currentpriority_of_process(p1)))

@grd509 *msg1*↦*t1*∈queue_of_queueingports(*port*)

@grd510 (∀*tt*,*mm*·(*mm*↦*tt*∈queue_of_queueingports(*port*) ⇒*t1*≤*tt*))

then

@act41 process_wait_type ≔ {*proc*}⩤process_wait_type

@act42 timeout_trigger ≔ {*proc*}⩤timeout_trigger

@act43 need_reschedule ≔ *resch*

@act501 processes_waitingfor_queuingports ≔ processes_waitingfor_queuingports ∖ {(*proc*↦*t*↦*msg*)↦*port*}

@act506 queue_of_queueingports(*port*) ≔ queue_of_queueingports(*port*)∖{*msg1* ↦ *t1*}

@act11 process_state(*proc*) ≔ *newstate*

end

event receive_queuing_message
refines receive_queuing_message
any *port msg t*
where
@grd01 *port*∈ports
@grd02 *port*∈**QueuingPorts**
@grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE ∧ **Ports_of_Partition**(*port*) = current_partition
@grd03 **Direction_of_Ports**(*port*)=**PORT_DESTINATION**
@grd04 *msg*∈**MESSAGES**
@grd06 card(queue_of_queueingports(*port*)) > 0
@grd05 (*msg* ↦*t*) ∈ queue_of_queueingports(*port*) ∧ (∀m,t1·(m↦t1∈queue_of_queueingports(*port*) ⇒ *t* ≤t1)) *// FIFO queue, read the first msg*
then
@act01 queue_of_queueingports(*port*) ≔ queue_of_queueingports(*port*) ∖ {*msg*↦*t*}
end

event receive_queuing_message_needwait *//extends req_busy_resource*
refines receive_queuing_message_needwait
any *part proc newstate port msg wt timeout tmout_trig*

where

@grd40 current_partition_flag = TRUE ∧ current_process_flag = TRUE

@grd41 *part* = current_partition ∧ *proc* = current_process

@grd42 *newstate* = **PS_Waiting**

@grd43 *wt*∈**PROCESS_WAIT_TYPES** ∧ (*wt*= **PROC_WAIT_OBJ** ∨ *wt*=**PROC_WAIT_TIMEOUT**)

//@grd06 tmout >0 ∨ tmout = INFINITE_TIME_VALUE

//this line is correct, the next line is from ARINC653

@grd44 *timeout* ≠0

@grd45 *tmout_trig* ∈ processes ⇸ (**PROCESS_STATES** × ℕ1)

@grd46 (*timeout* = **INFINITE_TIME_VALUE** ⇒ *tmout_trig* = ∅)

∧ (*timeout* ≠**INFINITE_TIME_VALUE** ⇒ *tmout_trig* = {*proc*↦(**PS_Ready**↦ (*timeout* +clock_tick ∗ **ONE_TICK_TIME**))})

@grd47 *timeout*≠**INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_TIMEOUT**

@grd48 *timeout* = **INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_OBJ**

@grd502 *port*∈ports

@grd503 *port*∈**QueuingPorts**

@grd500 current_partition_flag = TRUE ∧ current_process_flag=TRUE ∧ **Ports_of_Partition**(*port*) = current_partition

@grd501 *part* = current_partition ∧ *proc* = current_process

@grd504 **Direction_of_Ports**(*port*)=**PORT_DESTINATION**

@grd505 card(queue_of_queueingports(*port*)) = 0

@grd506 *msg*∈**MESSAGES**

@grd507 locklevel_of_partition(current_partition)=0

@grd508 current_partition∈dom(errorhandler_of_partition) ⇒ current_process ≠ errorhandler_of_partition(current_partition)

then

@act41 need_reschedule ≔ TRUE

@act42 current_process_flag ≔ FALSE

@act43 process_wait_type(*proc*) ≔ *wt*

@act05 timeout_trigger ≔ timeout_trigger *tmout_trig*

@act501 processes_waitingfor_queuingports ≔ processes_waitingfor_queuingports ∪ {(current_process↦clock_tick ∗ **ONE_TICK_TIME**↦*msg*)↦*port*}

@act11 process_state(*proc*) ≔ *newstate*

end

event get_queuing_port_id

any *port*

where

@grd01 *port*∈**QueuingPorts** ∧*port*∈ports

@grd00 current_partition_flag = TRUE ∧ **Ports_of_Partition**(*port*) = current_partition

```
end

event get_queuing_port_status
  any port
  where
    @grd01 port∈QueuingPorts ∧ port ∈ports
    @grd00 current_partition_flag = TRUE ∧ Ports_of_Partition(port) = current_partition
end

event clear_queuing_port
  any port
  where
    @grd01 port∈QueuingPorts ∧ port∈ports
    @grd00 current_partition_flag = TRUE ∧ Ports_of_Partition(port) = current_partition
    @grd02 Direction_of_Ports(port)=PORT_DESTINATION
  then
    @act01 queue_of_queueingports(port) ≔ ∅
end

event create_buffer refines create_buffer
  any buf max_msg_size quediscip
```

```
    where
      @grd00 current_partition_flag = TRUE ∧ (partition_mode(current_partition)=PM_COLD_START ∨
partition_mode(current_partition)=PM_WARM_START)
      @grd01 buf∈BUFFERS∖buffers // @grd02 Buffers_of_Partition(buf) = current_partition
      @grd03 max_msg_size∈ℕ1
      @grd04 quediscip∈QUEUING_DISCIPLINE
      @grd06 partition_mode(current_partition)≠PM_NORMAL
    then
      @act01 MaxMsgNum_of_Buffers(buf)≔max_msg_size
      @act02 buffers ≔ buffers∪{buf}
      @act03 quediscipline_of_buffers(buf) ≔ quediscip
      @act04 buffers_of_partition(buf) ≔ current_partition
      @act05 queue_of_buffers(buf) ≔ ∅
  end

  event send_buffer refines send_buffer
    any buf msg
    where
      @grd01 buf ∈ buffers
      @grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE ∧ buffers_of_partition(buf) =
current_partition
```

@grd02 *msg*∈**MESSAGES**∖used_messages

@grd05 card(queue_of_buffers(*buf*))<MaxMsgNum_of_Buffers(*buf*) // *buffer is not full*

@grd06 ¬ (∃p,t,m·(p∈processes ∧t∈ℕ1∧m∈**MESSAGES**∧ (p↦(**WAITING_R** ↦t)↦m) ∈ processes_waitingfor_buffers~[{*buf*}])) // *there is no waiting proc to receive the buffer*

then

@act01 queue_of_buffers(*buf*) ≔ queue_of_buffers(*buf*) ∪ {*msg*↦clock_tick ∗ **ONE_TICK_TIME**}

@act05 used_messages ≔ used_messages ∪ {*msg*}

end

event send_buffer_needwakeuprecvproc //*extends resource_become_available*

refines send_buffer_needwakeuprecvproc

any *part proc newstate resch buf msg t m*

where

@grd500 current_partition_flag = TRUE ∧ current_process_flag=TRUE ∧ buffers_of_partition(*buf*) = current_partition

@grd501 *part* = current_partition

@grd02 *proc* ∈ processes

@grd32 process_state(*proc*) = **PS_Waiting** ∨ process_state(*proc*) = **PS_WaitandSuspend**

@grd33 process_state(*proc*) = **PS_Waiting** ⇒ *newstate* = **PS_Ready**

@grd34 process_state(*proc*) = **PS_WaitandSuspend** ⇒ *newstate* = **PS_Suspend**

@grd40 process_wait_type(*proc*)= **PROC_WAIT_OBJ**

@grd41 *resch* ∈BOOL

@grd508 (locklevel_of_partition(current_partition)=0 ⇒ *resch*=TRUE) ∧ (locklevel_of_partition(current_partition)>0 ⇒ *resch*=need_reschedule)

@grd502 *buf* ∈ buffers

@grd503 *msg*∈**MESSAGES**∖used_messages

@grd504 card(queue_of_buffers(*buf*))<MaxMsgNum_of_Buffers(*buf*) *// buffer is not full*

@grd505 card(processes_waitingfor_buffers~[{*buf*}])>0 ∧ (*proc*↦(**WAITING_R** ↦*t*)↦*m*) ∈ processes_waitingfor_buffers~[{*buf*}]

@grd506 quediscipline_of_buffers(*buf*)=**QUEUE_FIFO** ⇒ (∀p1,m1,t1·(p1↦(**WAITING_R**↦t1)↦m1∈ processes_waitingfor_buffers~[{*buf*}] ⇒ *t*≤t1))

@grd507 quediscipline_of_buffers(*buf*)=**QUEUE_PRIORITY** ⇒ (∀p1,m1,t1·(p1↦(**WAITING_R**↦t1)↦m1∈ processes_waitingfor_buffers~[{*buf*}] ⇒ currentpriority_of_process(*proc*)≥currentpriority_of_process(p1)))

then

@act41 process_wait_type ≔ {*proc*}⩤process_wait_type

@act42 timeout_trigger ≔ {*proc*}⩤timeout_trigger

@act43 need_reschedule ≔ *resch*

@act501 processes_waitingfor_buffers ≔ processes_waitingfor_buffers ∖ {(*proc*↦(**WAITING_R**↦*t*)↦*m*) ↦ *buf*}

@act502 used_messages ≔ used_messages ∪ {*msg*}

@act11 process_state(*proc*) ≔ *newstate*

end

event send_buffer_withfull *//extends req_busy_resource*
refines send_buffer_withfull
any *part proc newstate wt timeout tmout_trig buf msg*
where
@grd40 current_partition_flag = TRUE ∧ current_process_flag = TRUE
@grd41 *part* = current_partition
@grd42 *proc* = current_process
@grd34 *newstate* = **PS_Waiting**
@grd43 *wt*∈**PROCESS_WAIT_TYPES** ∧ (*wt*= **PROC_WAIT_OBJ** ∨ *wt*=**PROC_WAIT_TIMEOUT**)
//@grd06 tmout >0 ∨ tmout = INFINITE_TIME_VALUE
//this line is correct, the next line is from ARINC653
@grd44 *timeout* ≠0
@grd45 *tmout_trig* ∈ processes ⇸ (**PROCESS_STATES** × ℕ1)
@grd46 (*timeout* = **INFINITE_TIME_VALUE** ⇒ *tmout_trig* = ∅)
∧ (*timeout* ≠**INFINITE_TIME_VALUE** ⇒ *tmout_trig* = {*proc*↦(**PS_Ready**↦ (*timeout* +clock_tick ∗ **ONE_TICK_TIME**))})
@grd47 *timeout*≠**INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_TIMEOUT**
@grd48 *timeout* = **INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_OBJ**

@grd503 $buf \in$ buffers

@grd500 buffers_of_partition(buf) = current_partition

@grd502 $msg \in$ **MESSAGES** $\setminus$ used_messages

@grd504 buffers_of_partition(buf) = current_partition

@grd505 card(queue_of_buffers(buf))=MaxMsgNum_of_Buffers(buf) // buffer is full

@grd509 locklevel_of_partition(current_partition) = 0

@grd510 current_partition $\in$ dom(errorhandler_of_partition) $\Rightarrow$ current_process $\neq$ errorhandler_of_partition(current_partition)

then

@act41 need_reschedule $:=$ TRUE

@act42 current_process_flag $:=$ FALSE

@act43 process_wait_type($proc$) $:=$ wt

@act05 timeout_trigger $:=$ timeout_trigger $+$ $tmout\_trig$

@act501 processes_waitingfor_buffers $:=$ processes_waitingfor_buffers $\cup$ {(current_process $\mapsto$ (**WAITING_W** $\mapsto$ clock_tick $*$ **ONE_TICK_TIME**) $\mapsto$ msg) $\mapsto$ buf}

@act502 used_messages $:=$ used_messages $\cup$ {msg}

@act11 process_state($proc$) $:=$ $newstate$

end

```
event receive_buffer refines receive_buffer
  any buf msg t
  where
    @grd01 buf ∈ buffers
    @grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE ∧ buffers_of_partition(buf) =
current_partition
    @grd02 msg∈MESSAGES
    @grd03 card(queue_of_buffers(buf))>0 // buffer is not empty
    @grd04 msg↦t∈queue_of_buffers(buf) ∧ (∀m1,t1·(m1↦t1∈queue_of_buffers(buf)⇒t≤t1)) // FIFO queue
    @grd05 ¬ (∃p,t1,m·(p∈processes ∧t1∈ℕ1∧ (p↦(WAITING_W ↦t1)↦m) ∈
processes_waitingfor_buffers~[{buf}])) // there is no waiting proc to send the buffer
    //the next two lines are correct, but commented according to arinc 653
    //then
    //  @act01 queue_of_buffers(buf) ≔ queue_of_buffers(buf) ∖ {msg↦t}
end

event receive_buffer_needwakeupsendproc //extends resource_become_available
refines receive_buffer_needwakeupsendproc
  any part proc newstate resch buf msg t m t_
  where
    @grd500 current_partition_flag = TRUE ∧ current_process_flag=TRUE
```

@grd501 *part* = current_partition

@grd02 *proc* ∈ processes

@grd03 *newstate* ∈ **PROCESS_STATES**

@grd06 processes_of_partition(*proc*) = *part*

@grd31 partition_mode(*part*) = **PM_NORMAL**

@grd32 process_state(*proc*) = **PS_Waiting** ∨ process_state(*proc*) = **PS_WaitandSuspend**

@grd33 process_state(*proc*) = **PS_Waiting** ⇒ *newstate* = **PS_Ready**

@grd34 process_state(*proc*) = **PS_WaitandSuspend** ⇒ *newstate* = **PS_Suspend**

@grd40 process_wait_type(*proc*)= **PROC_WAIT_OBJ**

@grd41 *resch*∈BOOL

@grd509 (locklevel_of_partition(current_partition)=0 ⇒ *resch*=TRUE) ∧ (locklevel_of_partition(current_partition)>0 ⇒ *resch*=need_reschedule)

@grd506 *buf* ∈ buffers

@grd05 buffers_of_partition(*buf*) = current_partition

@grd502 *msg*∈**MESSAGES**

@grd503 card(queue_of_buffers(*buf*))>0 *// buffer is not empty*

@grd504 *msg*↦*t*∈queue_of_buffers(*buf*) ∧ (∀m1,t1·(m1↦t1∈queue_of_buffers(*buf*)⇒*t*≤t1)) *// FIFO queue*

@grd505 card(processes_waitingfor_buffers~[{*buf*}])>0 ∧ (*proc*↦(**WAITING_W** ↦*t_*)↦*m*) ∈ processes_waitingfor_buffers~[{*buf*}]

@grd507 quediscipline_of_buffers(*buf*)=**QUEUE_FIFO** ⇒ (∀p1,m1,t1·(p1↦(**WAITING_W**↦t1)↦m1∈ processes_waitingfor_buffers~[{*buf*}] ⇒ *t_*≤t1))

@grd508 quediscipline_of_buffers(*buf*)=**QUEUE_PRIORITY** ⇒ (∀p1,m1,t1·(p1↦(**WAITING_W**↦t1)↦m1∈processes_waitingfor_buffers~[{*buf*}] ⇒ currentpriority_of_process(*proc*)≥currentpriority_of_process(p1)))

then

@act41 process_wait_type ≔ {*proc*}⩤process_wait_type

@act42 timeout_trigger ≔ {*proc*}⩤timeout_trigger

@act43 need_reschedule ≔ *resch*

//@act501 queue_of_buffers(buf) ≔ (queue_of_buffers(buf) ∖ {msg↦t})∪{m↦t_}

//this line is correct, the next line is according to ARINC653

@act501 queue_of_buffers(*buf*) ≔ queue_of_buffers(*buf*)∪{*m*↦*t_*}

@act502 processes_waitingfor_buffers ≔ processes_waitingfor_buffers ∖ {(*proc*↦(**WAITING_W**↦*t_*)↦*m*) ↦ *buf*}

@act11 process_state(*proc*) ≔ *newstate*

end

event receive_buffer_whenempty *//extends req_busy_resource*

refines receive_buffer_whenempty

any *part proc newstate wt timeout tmout_trig buf msg*

where

@grd40 current_partition_flag = TRUE ∧ current_process_flag = TRUE

@grd41 *part* = current_partition

@grd42 *proc* = current_process

@grd34 *newstate* = **PS_Waiting**

@grd43 *wt*∈**PROCESS_WAIT_TYPES** ∧ (*wt*= **PROC_WAIT_OBJ** ∨ *wt*=**PROC_WAIT_TIMEOUT**)

//@grd06 tmout >0 ∨ tmout = INFINITE_TIME_VALUE

//this line is correct, the next line is from ARINC653

@grd44 *timeout* ≠0

@grd45 *tmout_trig* ∈ processes ⇸ (**PROCESS_STATES** × ℕ1)

@grd46 (*timeout* = **INFINITE_TIME_VALUE** ⇒ *tmout_trig* = ∅)

∧ (*timeout* ≠**INFINITE_TIME_VALUE** ⇒ *tmout_trig* = {*proc*↦(**PS_Ready**↦ (*timeout* +clock_tick ∗ **ONE_TICK_TIME**))})

@grd47 *timeout*≠**INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_TIMEOUT**

@grd48 *timeout* = **INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_OBJ**

@grd504 *buf* ∈ buffers

@grd500 buffers_of_partition(*buf*) = current_partition

@grd502 card(queue_of_buffers(*buf*))=0 *// buffer is empty*

@grd503 *msg*∈**MESSAGES**

@grd509 locklevel_of_partition(current_partition) = 0

@grd510 current_partition∈dom(errorhandler_of_partition) ⇒ current_process ≠ errorhandler_of_partition(current_partition)

```
    then
      @act41 need_reschedule ≔ TRUE
      @act42 current_process_flag ≔ FALSE
      @act43 process_wait_type(proc) ≔ wt
      @act05 timeout_trigger ≔ timeout_trigger        tmout_trig
      @act11 process_state(proc) ≔ newstate
      @act501 processes_waitingfor_buffers ≔ processes_waitingfor_buffers ∪ {(current_process ↦
(WAITING_R↦clock_tick ∗ ONE_TICK_TIME)↦msg) ↦ buf}
  end

  event get_buffer_id
    any buf
    where
      @grd01 buf ∈ buffers
      @grd00 current_partition_flag = TRUE ∧ buffers_of_partition(buf) = current_partition
  end

  event get_buffer_status
    any buf
    where
      @grd01 buf ∈ buffers
```

@grd00 current_partition_flag = TRUE ∧ buffers_of_partition(*buf*) = current_partition

end

event create_blackboard **refines** create_blackboard

any *bb*

where

@grd00 current_partition_flag = TRUE ∧ (partition_mode(current_partition)=**PM_COLD_START** ∨ partition_mode(current_partition)=**PM_WARM_START**)

@grd01 *bb*∈**BLACKBOARDS**∖blackboards

@grd06 partition_mode(current_partition)≠**PM_NORMAL**

then

@act02 blackboards ≔ blackboards∪{*bb*}

@act03 blackboards_of_partition(*bb*)≔ current_partition

@act04 emptyindicator_of_blackboards(*bb*)≔**BB_EMPTY**

end

event display_blackboard **refines** display_blackboard

any *bb msg*

where

@grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE

@grd01 *bb*∈blackboards ∧ blackboards_of_partition(*bb*) = current_partition

@grd02 $msg \in \mathbf{MESSAGES} \setminus$ used_messages
@grd03 processes_waitingfor_blackboards~[{bb}] = $\varnothing$

then

@act01 msgspace_of_blackboards(bb) $\coloneqq$ msg
@act02 emptyindicator_of_blackboards(bb) $\coloneqq$ **BB_OCCUPIED**
@act03 used_messages $\coloneqq$ used_messages $\cup$ {msg}

end

event display_blackboard_needwakeuprdprocs *//extends resource_become_available2*
refines display_blackboard_needwakeuprdprocs

any *part procs newstates resch bb msg*

where

@grd500 current_partition_flag = TRUE $\wedge$ current_process_flag=TRUE
@grd502 $part$ = current_partition
@grd06 $procs \subseteq$ processes_of_partition~[{$part$}]
@grd40 $\forall proc \cdot (proc \in procs \Rightarrow$ process_wait_type($proc$)= **PROC_WAIT_OBJ**)
@grd506 $procs$ = processes_waitingfor_blackboards~[{bb}]
@grd03 $newstates \in procs \rightarrow$ **PROCESS_STATES**

@grd31 partition_mode($part$) = **PM_NORMAL**
@grd32 $\forall proc \cdot (proc \in procs \Rightarrow$ process_state($proc$) = **PS_Waiting** $\vee$ process_state($proc$) =

PS_WaitandSuspend)

@grd33 ∀*proc*·(*proc*∈*procs* ∧ process_state(*proc*) = **PS_Waiting** ⇒ *newstates*(*proc*) = **PS_Ready**)

@grd34 ∀*proc*·(*proc*∈*procs* ∧ process_state(*proc*) = **PS_WaitandSuspend** ⇒ *newstates*(*proc*) = **PS_Suspend**)

@grd41 *resch*∈BOOL

@grd507 (locklevel_of_partition(current_partition)=0 ⇒ *resch*=TRUE) ∧ (locklevel_of_partition(current_partition)>0 ⇒ *resch*=need_reschedule)

@grd503 *bb*∈blackboards ∧ blackboards_of_partition(*bb*) = current_partition

@grd504 *msg*∈**MESSAGES**∖used_messages

@grd505 processes_waitingfor_blackboards~[{*bb*}] ≠ ∅

then

@act41 process_wait_type ≔ *procs*⩤process_wait_type

@act42 timeout_trigger ≔ *procs*⩤timeout_trigger

@act43 need_reschedule ≔ *resch*

@act11 process_state ≔ process_state *newstates*

@act501 msgspace_of_blackboards(*bb*) ≔ *msg*

@act502 emptyindicator_of_blackboards(*bb*) ≔ **BB_OCCUPIED**

@act503 processes_waitingfor_blackboards ≔ *procs*⩤processes_waitingfor_blackboards

@act504 used_messages ≔ used_messages ∪ {*msg*}

```
end

event read_blackboard refines read_blackboard
  any bb msg
  where
    @grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE
    @grd01 bb∈blackboards ∧ blackboards_of_partition(bb) = current_partition
    @grd02 msg∈MESSAGES
    @grd03 bb∈dom(msgspace_of_blackboards) ∧ msg = msgspace_of_blackboards(bb)
    @grd04 emptyindicator_of_blackboards(bb) = BB_OCCUPIED
end

event read_blackboard_whenempty //extends req_busy_resource
refines read_blackboard_whenempty
  any part proc newstate wt timeout tmout_trig bb
  where
    @grd40 current_partition_flag = TRUE ∧ current_process_flag = TRUE
    @grd41 part = current_partition
    @grd42 proc = current_process
    @grd43 wt∈PROCESS_WAIT_TYPES ∧ (wt= PROC_WAIT_OBJ ∨ wt=PROC_WAIT_TIMEOUT)
```

@grd34 *newstate* = **PS_Waiting**

//@grd06 tmout >0 ∨ tmout = INFINITE_TIME_VALUE

//this line is correct, the next line is from ARINC653

@grd44 *timeout* ≠0

@grd45 *tmout_trig* ∈ processes ⇸ (**PROCESS_STATES** × ℕ1)

@grd46 (*timeout* = **INFINITE_TIME_VALUE** ⇒ *tmout_trig* = ∅)

∧ (*timeout* ≠**INFINITE_TIME_VALUE** ⇒ *tmout_trig* = {*proc*↦(**PS_Ready**↦ (*timeout* +clock_tick ∗ **ONE_TICK_TIME**))})

@grd47 *timeout*≠**INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_TIMEOUT**

@grd48 *timeout* = **INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_OBJ**

@grd501 *bb*∈blackboards ∧ blackboards_of_partition(*bb*) = current_partition

@grd503 emptyindicator_of_blackboards(*bb*) = **BB_EMPTY**

@grd504 locklevel_of_partition(current_partition) = 0

@grd515 current_partition∈dom(errorhandler_of_partition) ⇒ current_process ≠ errorhandler_of_partition(current_partition)

then

@act41 need_reschedule ≔ TRUE

@act42 current_process_flag ≔ FALSE

@act43 process_wait_type(*proc*) ≔ *wt*

```
    @act05 timeout_trigger ≔ timeout_trigger        tmout_trig

    @act501 processes_waitingfor_blackboards ≔ processes_waitingfor_blackboards ∪ {current_process ↦
bb}
    @act11 process_state(proc) ≔ newstate

  end

  event clear_blackboard refines clear_blackboard
    any bb
    where
      @grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE
      @grd01 bb∈blackboards ∧ blackboards_of_partition(bb) = current_partition
    then
      @act01 emptyindicator_of_blackboards(bb) ≔ BB_EMPTY
  end

  event get_blackboard_id
    any bb
    where
      @grd01 bb∈blackboards
```

@grd00 current_partition_flag = TRUE ∧ blackboards_of_partition(*bb*) = current_partition

end

event get_blackboard_status

any *bb*

where

@grd01 *bb*∈blackboards

@grd00 current_partition_flag = TRUE ∧ blackboards_of_partition(*bb*) = current_partition

end

event create_semaphore **refines** create_semaphore

any *sem maxval currentval quediscip*

where

@grd01 current_partition_flag = TRUE ∧ (partition_mode(current_partition)=**PM_COLD_START** ∨ partition_mode(current_partition)=**PM_WARM_START**)

@grd02 *sem*∈**SEMAPHORES**∖semaphores

@grd04 semaphores_of_partition(*sem*) = current_partition

@grd05 *quediscip*∈**QUEUING_DISCIPLINE**

@grd06 partition_mode(current_partition)≠**PM_NORMAL**

@grd07 *maxval*∈ℕ1

@grd08 *currentval*∈ℕ ∧ *currentval*≤*maxval*

```
  then
    @act01 quediscipline_of_semaphores(sem)≔quediscip
    @act02 semaphores ≔ semaphores∪{sem}
    @act03 value_of_semaphores(sem)≔currentval
    @act04 MaxValue_of_Semaphores(sem)≔maxval
    @act05 semaphores_of_partition(sem) ≔current_partition
end

event wait_semaphore refines wait_semaphore
  any sem
  where
    @grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE
    @grd01 sem∈semaphores ∧ semaphores_of_partition(sem) = current_partition
    @grd02 value_of_semaphores(sem) > 0
  then
    @act01 value_of_semaphores(sem) ≔ value_of_semaphores(sem) −1
end

event wait_semahpore_whenzero //extends req_busy_resource
refines wait_semahpore_whenzero
any part proc newstate wt timeout tmout_trig sem
```

where

@grd40 current_partition_flag = TRUE ∧ current_process_flag = TRUE

@grd41 *part* = current_partition

@grd42 *proc* = current_process

@grd34 *newstate* = **PS_Waiting**

@grd43 *wt*∈**PROCESS_WAIT_TYPES** ∧ (*wt*= **PROC_WAIT_OBJ** ∨ *wt*=**PROC_WAIT_TIMEOUT**)

//@grd06 tmout >0 ∨ tmout = INFINITE_TIME_VALUE

//this line is correct, the next line is from ARINC653

@grd44 *timeout* ≠0

@grd45 *tmout_trig* ∈ processes ⇸ (**PROCESS_STATES** × ℕ1)

@grd46 (*timeout* = **INFINITE_TIME_VALUE** ⇒ *tmout_trig* = ∅)

∧ (*timeout* ≠**INFINITE_TIME_VALUE** ⇒ *tmout_trig* = {*proc*↦(**PS_Ready**↦ (*timeout* +clock_tick ∗ **ONE_TICK_TIME**))})

@grd47 *timeout*≠**INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_TIMEOUT**

@grd48 *timeout* = **INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_OBJ**

@grd502 *sem*∈semaphores ∧ semaphores_of_partition(*sem*) = current_partition

@grd504 value_of_semaphores(*sem*) = 0

@grd505 locklevel_of_partition(current_partition) = 0

@grd506 current_partition∈dom(errorhandler_of_partition) ⇒ current_process ≠ errorhandler_of_partition(current_partition)

then

@act41 need_reschedule ≔ TRUE

@act42 current_process_flag ≔ FALSE

@act43 process_wait_type(*proc*) ≔ *wt*

@act05 timeout_trigger ≔ timeout_trigger *tmout_trig*

@act501 processes_waitingfor_semaphores ≔ processes_waitingfor_semaphores ∪ {(current_process ↦ clock_tick ∗ **ONE_TICK_TIME**)↦ *sem*}

@act11 process_state(*proc*) ≔ *newstate*

end

event signal_semaphore **refines** signal_semaphore

any *sem*

where

@grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE

@grd01 *sem*∈semaphores ∧ semaphores_of_partition(*sem*) = current_partition

@grd02 value_of_semaphores(*sem*) ≠ MaxValue_of_Semaphores(*sem*)

@grd03 processes_waitingfor_semaphores~[{*sem*}] = ∅

then

@act01 value_of_semaphores(*sem*) ≔ value_of_semaphores(*sem*) + 1

end

event signal_semaphore_needwakeupproc *//extends resource_become_available*

refines signal_semaphore_needwakeupproc

any *part proc newstate resch sem t*

where

@grd500 current_partition_flag = TRUE ∧ current_process_flag=TRUE

@grd501 *part* = current_partition

@grd02 *proc* ∈ processes

@grd34 *newstate* = **PS_Waiting**

@grd40 process_wait_type(*proc*)= **PROC_WAIT_OBJ**

@grd41 *resch*∈BOOL

@grd509 (locklevel_of_partition(current_partition)=0 ⇒ *resch*=TRUE) ∧ (locklevel_of_partition(current_partition)>0 ⇒ *resch*=need_reschedule)

@grd502 *sem*∈semaphores ∧ semaphores_of_partition(*sem*) = current_partition

@grd503 value_of_semaphores(*sem*) ≠ MaxValue_of_Semaphores(*sem*)

@grd506 card(processes_waitingfor_semaphores~[{*sem*}])>0 ∧ (*proc*↦*t*) ∈ processes_waitingfor_semaphores~[{*sem*}]

@grd507 quediscipline_of_semaphores(*sem*)=**QUEUE_FIFO** ⇒ (∀p1,t1·(p1↦t1∈ processes_waitingfor_semaphores~[{*sem*}] ⇒ *t*≤t1))

@grd508 quediscipline_of_semaphores(*sem*)=**QUEUE_PRIORITY** ⇒ (∀p1,t1·(p1↦t1∈

processes_waitingfor_semaphores~[{*sem*}] ⇒ currentpriority_of_process(*proc*)≥currentpriority_of_process(p1)))

then

@act41 process_wait_type ≔ {*proc*}⩤process_wait_type

@act42 timeout_trigger ≔ {*proc*}⩤timeout_trigger

@act43 need_reschedule ≔ *resch*

@act501 processes_waitingfor_semaphores ≔ processes_waitingfor_semaphores ∖ {*proc* ↦ *t*↦*sem*}

@act11 process_state(*proc*) ≔ *newstate*

end

event get_semaphore_id

any *sem*

where

@grd01 *sem*∈semaphores

@grd00 current_partition_flag = TRUE ∧ semaphores_of_partition(*sem*) = current_partition

end

event get_semaphore_status

any *sem*

where

@grd01 *sem*∈semaphores

@grd00 current_partition_flag = TRUE ∧ semaphores_of_partition(*sem*) = current_partition

end

event create_event **refines** create_event

any *ev*

where

@grd01 current_partition_flag = TRUE ∧ (partition_mode(current_partition)=**PM_COLD_START** ∨ partition_mode(current_partition)=**PM_WARM_START**)

@grd02 *ev*∈**EVENTS**∖events_

@grd04 events_of_partition(*ev*) = current_partition

@grd06 partition_mode(current_partition)≠**PM_NORMAL**

then

@act01 events_ ≔ events_∪{*ev*}

@act02 state_of_events(*ev*) ≔ **EVENT_DOWN**

@act03 events_of_partition(*ev*) ≔ current_partition

end

event set_event **refines** set_event

any *ev*

where

@grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE

@grd01 $ev \in events\_ \land events\_of\_partition(ev) = current\_partition$

@grd03 $processes\_waitingfor\_events[\{ev\}] = \varnothing$

then

@act01 $state\_of\_events(ev) := \mathbf{EVENT\_UP}$

end

event set_event_needwakeupprocs *//extends resource_become_available2*

refines set_event_needwakeupprocs

any *part procs newstates resch ev*

where

@grd500 $current\_partition\_flag = TRUE \land current\_process\_flag = TRUE$

@grd501 $part = current\_partition$

@grd02 $procs \subseteq processes$

@grd06 $procs \subseteq processes\_of\_partition[\{part\}]$

@grd504 $procs = processes\_waitingfor\_events[\{ev\}]$

@grd40 $\forall proc \cdot (proc \in procs \Rightarrow process\_wait\_type(proc) = \mathbf{PROC\_WAIT\_OBJ})$

@grd03 $newstates \in procs \rightarrow \mathbf{PROCESS\_STATES}$

@grd32 $\forall proc \cdot (proc \in procs \Rightarrow process\_state(proc) = \mathbf{PS\_Waiting} \lor process\_state(proc) = \mathbf{PS\_WaitandSuspend})$

@grd33 $\forall proc \cdot (proc \in procs \land process\_state(proc) = \mathbf{PS\_Waiting} \Rightarrow newstates(proc) = \mathbf{PS\_Ready})$

@grd34 $\forall proc \cdot (proc \in procs \land process\_state(proc) = \mathbf{PS\_WaitandSuspend} \Rightarrow newstates(proc) =$

PS_Suspend)

@grd41 *resch*∈BOOL

@grd507 (locklevel_of_partition(current_partition)=0 ⇒ *resch*=TRUE) ∧ (locklevel_of_partition(current_partition)>0 ⇒ *resch*=need_reschedule)

@grd502 *ev*∈events_ ∧ events_of_partition(*ev*) = current_partition

@grd503 processes_waitingfor_events~[{*ev*}] ≠ ∅

then

@act41 process_wait_type ≔ *procs*⩤process_wait_type

@act42 timeout_trigger ≔ *procs*⩤timeout_trigger

@act43 need_reschedule ≔ *resch*

@act11 process_state ≔ process_state *newstates*

@act501 state_of_events(*ev*)≔**EVENT_UP**

@act503 processes_waitingfor_events ≔ *procs*⩤processes_waitingfor_events

end

event reset_event **refines** reset_event

any *ev*

where

@grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE

@grd01 *ev*∈events_ ∧ events_of_partition(*ev*) = current_partition

then

```
    @act01 state_of_events(ev) := EVENT_DOWN
end

event wait_event refines wait_event
  any ev
  where
    @grd00 current_partition_flag = TRUE ∧ current_process_flag=TRUE
    @grd01 ev∈events_ ∧ events_of_partition(ev) = current_partition
    @grd02 state_of_events(ev)=EVENT_UP
end

 event wait_event_whendown //extends req_busy_resource
 refines wait_event_whendown
  any part proc newstate wt timeout tmout_trig ev
  where
    @grd40 current_partition_flag = TRUE ∧ current_process_flag = TRUE
    @grd41 part = current_partition
    @grd42 proc = current_process
    @grd34 newstate = PS_Waiting
    @grd43 wt∈PROCESS_WAIT_TYPES ∧ (wt= PROC_WAIT_OBJ ∨ wt=PROC_WAIT_TIMEOUT)
    //@grd06 tmout >0 ∨ tmout = INFINITE_TIME_VALUE
```

//this line is correct, the next line is from ARINC653

@grd44 *timeout* ≠0

@grd45 *tmout_trig* ∈ processes ⇸ (**PROCESS_STATES** × ℕ1)

@grd46 (*timeout* = **INFINITE_TIME_VALUE** ⇒ *tmout_trig* = ∅)

∧ (*timeout* ≠**INFINITE_TIME_VALUE** ⇒ *tmout_trig* = {*proc*↦(**PS_Ready**↦ (*timeout* +clock_tick ∗ **ONE_TICK_TIME**))})

@grd47 *timeout*≠**INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_TIMEOUT**

@grd48 *timeout* = **INFINITE_TIME_VALUE** ⇒ *wt* = **PROC_WAIT_OBJ**

@grd503 *ev*∈events_ ∧ events_of_partition(*ev*) = current_partition

@grd504 state_of_events(*ev*)=**EVENT_DOWN**

@grd509 locklevel_of_partition(current_partition) = 0

@grd510 current_partition∈dom(errorhandler_of_partition) ⇒ current_process ≠ errorhandler_of_partition(current_partition)

then

@act41 need_reschedule ≔ TRUE

@act42 current_process_flag ≔ FALSE

@act43 process_wait_type(*proc*) ≔ *wt*

@act05 timeout_trigger ≔ timeout_trigger *tmout_trig*

@act501 processes_waitingfor_events ≔ processes_waitingfor_events ∪ {current_process ↦ *ev*}

```
    @act11 process_state(proc) ≔ newstate
end

event get_event_id
  any ev
  where
    @grd01 ev∈events_
    @grd00 current_partition_flag = TRUE ∧ events_of_partition(ev) = current_partition
end

event get_event_status
  any ev
  where
    @grd01 ev∈events_
    @grd00 current_partition_flag = TRUE ∧ events_of_partition(ev) = current_partition
end

event ticktock // timer interrupt event, triggered by the timer in hardware. one tick in each ONE_TICK_TIME
extends ticktock
end
```

```
  event partition_schedule extends partition_schedule
  end

  event process_schedule // if there is not error handler and preempter in this partition
  extends process_schedule
  end

  event run_errorhandler_preempter // if there is the error handler, it is executed, otherwise the preempter is
executed
  extends run_errorhandler_preempter
  end

  event get_partition_status extends get_partition_status
  end

  event set_partition_mode_to_idle // shutdown the partition
  extends set_partition_mode_to_idle
    then
      @act601 RefreshPeriod_of_SamplingPorts ≔ Ports_of_Partition~[{part}] ⩤
RefreshPeriod_of_SamplingPorts
      @act602 needtrans_of_sourcesamplingport ≔ Ports_of_Partition~[{part}] ⩤
```

needtrans_of_sourcesamplingport

@act603 quediscipline_of_queueingports ≔ **Ports_of_Partition**~[{part}] ⩤ quediscipline_of_queueingports

@act604 quediscipline_of_buffers ≔ buffers_of_partition~[{part}]⩤quediscipline_of_buffers

@act605 quediscipline_of_semaphores ≔ semaphores_of_partition~[{part}] ⩤ quediscipline_of_semaphores

end

event set_partition_mode_to_normal **extends** set_partition_mode_to_normal

end

event set_partition_mode_to_coldstart **extends** set_partition_mode_to_coldstart

then

@act601 RefreshPeriod_of_SamplingPorts ≔ **Ports_of_Partition**~[{part}] ⩤ RefreshPeriod_of_SamplingPorts

@act602 needtrans_of_sourcesamplingport ≔ **Ports_of_Partition**~[{part}] ⩤ needtrans_of_sourcesamplingport

@act603 quediscipline_of_queueingports ≔ **Ports_of_Partition**~[{part}] ⩤ quediscipline_of_queueingports

@act604 quediscipline_of_buffers ≔ buffers_of_partition~[{part}]⩤quediscipline_of_buffers

@act605 quediscipline_of_semaphores ≔ semaphores_of_partition~[{part}] ⩤ quediscipline_of_semaphores

end

```
  event set_partition_mode_to_warmstart extends set_partition_mode_to_warmstart
    then
      @act601 RefreshPeriod_of_SamplingPorts ≔ Ports_of_Partition~[{part}] ⩤
RefreshPeriod_of_SamplingPorts
      @act602 needtrans_of_sourcesamplingport ≔ Ports_of_Partition~[{part}] ⩤
needtrans_of_sourcesamplingport
      @act603 quediscipline_of_queueingports ≔ Ports_of_Partition~[{part}] ⩤ quediscipline_of_queueingports
      @act604 quediscipline_of_buffers ≔ buffers_of_partition~[{part}]⩤quediscipline_of_buffers
      @act605 quediscipline_of_semaphores ≔ semaphores_of_partition~[{part}] ⩤
quediscipline_of_semaphores
  end

  event get_process_id extends get_process_id
  end

  event get_process_status extends get_process_status
  end

  event create_process extends create_process
  end
```

```
event set_priority extends set_priority
end

event suspend_self
/* extends suspend_self
    any timeout timeouttrig waittype */
extends suspend_self
end

event suspend // extends suspend
extends suspend
end

event resume // extends resume
extends resume
end

event stop_self extends stop_self
end
```

```
event stop extends stop

end

event start_aperiodprocess_instart
/* start an aperiodic process in COLD_START or WARM_START mode
    extends start */
extends start_aperiodprocess_instart
end

event start_aperiodprocess_innormal
/* start an aperiodic process in NORMAL mode
    extends start */
extends start_aperiodprocess_innormal
end

event start_periodprocess_instart
/* start a periodic process in COLD_START or WARM_START mode
    extends start */
extends start_periodprocess_instart
end
```

```
event start_periodprocess_innormal
/* start a periodic process in NORMAL mode
    extends start */
extends start_periodprocess_innormal
end

event delaystart_aperiodprocess_instart  // extends delayed_start
extends delaystart_aperiodprocess_instart
end

event delaystart_aperiodprocess_innormal
/* if delaytime=0, then immediately transit to READY, this is modelled in start_aperiod_process_whennormal
    extends delayed_start
    any delaytime */
extends delaystart_aperiodprocess_innormal
end

event delaystart_periodprocess_instart  // extends delayed_start
extends delaystart_periodprocess_instart
end
```

```
event delaystart_periodprocess_innormal  // extends delayed_start
extends delaystart_periodprocess_innormal
end

event lock_preemption extends lock_preemption
end

event unlock_preemption extends unlock_preemption
end

event get_my_id extends get_my_id
end

event timed_wait extends timed_wait
end

event period_wait extends period_wait
end

event get_time extends get_time
```

```
end

event replenish extends replenish
end

event aperiodicprocess_finished extends aperiodicprocess_finished
end

event periodicprocess_finished extends periodicprocess_finished
end

event time_out // should refined to support remove process on waiting queue of comm resources
extends time_out

end

event periodicproc_reach_releasepoint extends periodicproc_reach_releasepoint
end

event coldstart_partition_fromidle extends coldstart_partition_fromidle
end
```

```
    event warmstart_partition_fromidle extends warmstart_partition_fromidle
    end


end
```