```
machine Mach_HM
//* ***************************************************
//     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   sees Ctx_HM

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

        //Health monitor
        module_shutdown //TRUE: shutdown, FALSE: normal. When it is shutdown, all events are disabled

invariants
      @inv_module_shutdown module_shutdown∈BOOL

events
  event INITIALISATION extends INITIALISATION
  then
    @act701 module_shutdown ≔ FALSE
  end

  event create_error_handler extends create_process
  when
    @grd700 module_shutdown = FALSE
    @grd701 basepriority=MAX_PRIORITY_VALUE
    @grd702 part∉dom(errorhandler_of_partition)
  end

  event report_application_message
  when
    @grd700 module_shutdown = FALSE

  end

  event get_error_status
  when
    @grd700 module_shutdown = FALSE
    @grd01 current_partition_flag = TRUE ∧ current_process_flag = TRUE
    @grd02 current_partition∈dom(errorhandler_of_partition) ∧ current_process =
errorhandler_of_partition(current_partition)
    @grd03 current_process ∈ dom(process_call_errorhandler)
  end

  event hm_recoveryaction_shutdown_module
  any errcode part
  where
    @grd700 module_shutdown = FALSE
    @grd701 errcode∈SYSTEM_ERRORS
    @grd702 errcode∈dom(MultiPart_HM_Table(part))
    @grd703 errcode ↦ MLA_SHUTDOWN ∈ MultiPart_HM_Table(part)
  then
    @act701 module_shutdown≔TRUE
  end

  event hm_recoveryaction_reset_module
  any errcode part
  where
    @grd700 module_shutdown = FALSE
     @grd701 errcode∈SYSTEM_ERRORS
    @grd702 errcode∈dom(MultiPart_HM_Table(part))
    @grd703 errcode ↦ MLA_RESET ∈ MultiPart_HM_Table(part)

  end

  event hm_recoveryaction_ignore_module
  any errcode part
  where
    @grd700 module_shutdown = FALSE
```

```
    @grd701 errcode∈SYSTEM_ERRORS
    @grd702 errcode∈dom(MultiPart_HM_Table(part))
    @grd703 errcode ↦ MLA_IGNORE ∈ MultiPart_HM_Table(part)

  end

  event hm_recoveryaction_idle_partition extends set_partition_mode_to_idle
  any errcode
  where
    @grd700 module_shutdown = FALSE
    @grd701 errcode∈SYSTEM_ERRORS ∧ part∈PARTITIONS
    //@grd702 errcode∉dom(MultiPart_HM_Table(part))
    @grd703 (errcode∈dom(Partition_HM_Table(part)) ∧ ERROR_LEVEL_PARTITION2↦PLA_IDLE∈
dom(Partition_HM_Table(part)(errcode)))
                  ∨ (part∉dom(errorhandler_of_partition)) ∨ (current_process = errorhandler_of_partition(part))

  end

  event hm_recoveryaction_coldstart_partition extends set_partition_mode_to_coldstart
  any errcode
  where
```

```
    @grd700 module_shutdown = FALSE
    @grd701 errcode∈SYSTEM_ERRORS ∧ part∈PARTITIONS
    //@grd702 errcode∉dom(MultiPart_HM_Table(part))
    @grd703 (errcode∈dom(Partition_HM_Table(part)) ∧ ERROR_LEVEL_PARTITION2↦PLA_COLD_START∈
dom(Partition_HM_Table(part)(errcode)))
                ∨ (part∉dom(errorhandler_of_partition)) ∨ (current_process = errorhandler_of_partition(part))

  end

  event hm_recoveryaction_warmstart_partition extends set_partition_mode_to_warmstart
  any errcode
  when
    @grd700 module_shutdown = FALSE
    @grd701 errcode∈SYSTEM_ERRORS // ∧ errcode∉dom(MultiPart_HM_Table(part))
    @grd703 (errcode∈dom(Partition_HM_Table(part)) ∧ ERROR_LEVEL_PARTITION2↦PLA_WARM_START∈
dom(Partition_HM_Table(part)(errcode)))
                ∨ (part∉dom(errorhandler_of_partition)) ∨ (current_process = errorhandler_of_partition(part))

  end

  event hm_recoveryaction_ignore_partition
  any errcode part
  where
    @grd700 module_shutdown = FALSE
    @grd701 errcode∈SYSTEM_ERRORS  ∧ part∈PARTITIONS
    //@grd702 errcode∉dom(MultiPart_HM_Table(part))
    @grd703 (errcode∈dom(Partition_HM_Table(part)) ∧ ERROR_LEVEL_PARTITION2↦PLA_IGNORE∈
dom(Partition_HM_Table(part)(errcode)))
                 ∨ (part∉dom(errorhandler_of_partition)) ∨ (current_process = errorhandler_of_partition(part))

  end

  event hm_recoveryaction_errorhandler extends start_aperiodprocess_innormal
  any errcode
  where
    @grd700 module_shutdown = FALSE
    @grd701 errcode∈SYSTEM_ERRORS
     @grd702 (errcode∈dom(Partition_HM_Table(part)) ∧ ∃a·(a∈PARTITION_RECOVERY_ACTIONS ∧
ERROR_LEVEL_PROCESS↦a∈dom(Partition_HM_Table(part)(errcode)))   )
    @grd703 DEADLINE_MISSED∈ran(Partition_HM_Table(part)(errcode)) ⇒ (∃proc·(proc∈
processes_of_partition∼[{part}] ∧ clock_tick∗ONE_TICK_TIME > deadlinetime_of_process(proc)))
    @grd704 part∈dom(errorhandler_of_partition)
    @grd705 current_process  ≠  errorhandler_of_partition(part)
     @grd706 proc = errorhandler_of_partition(part)
  end




  event create_sampling_port extends create_sampling_port
  when
    @grd700 module_shutdown  = FALSE
  end

  event write_sampling_message  extends write_sampling_message
  when
    @grd700 module_shutdown  = FALSE
  end

  event transfer_sampling_msg extends transfer_sampling_msg
  when
    @grd700 module_shutdown  = FALSE
  end
```

```
event read_sampling_message extends read_sampling_message
when
  @grd700 module_shutdown = FALSE
end

event get_sampling_port_id extends get_sampling_port_id
when
  @grd700 module_shutdown = FALSE
end

event get_sampling_port_status extends get_sampling_port_status
when
  @grd700 module_shutdown = FALSE
end

event create_queuing_port extends create_queuing_port
when
  @grd700 module_shutdown = FALSE
end
```

```
event send_queuing_message extends send_queuing_message
when
  @grd700 module_shutdown = FALSE
end

event send_queuing_message_needwait // extends req_busy_resource
extends send_queuing_message_needwait
when
  @grd700 module_shutdown = FALSE
end

event transfer_queuing_msg extends transfer_queuing_msg
when
  @grd700 module_shutdown = FALSE
end

event wakeup_waitproc_on_srcqueports // extends resource_become_available
extends wakeup_waitproc_on_srcqueports
when
  @grd700 module_shutdown = FALSE
end
```

```
event wakeup_waitproc_on_destqueports // extends resource_become_available
extends wakeup_waitproc_on_destqueports
when
  @grd700 module_shutdown = FALSE
end

event receive_queuing_message extends receive_queuing_message
when
  @grd700 module_shutdown = FALSE
end

event receive_queuing_message_needwait // extends req_busy_resource
extends receive_queuing_message_needwait
when
  @grd700 module_shutdown = FALSE
end

event get_queuing_port_id extends get_queuing_port_id
when
  @grd700 module_shutdown = FALSE
```

```
end

event get_queuing_port_status extends get_queuing_port_status
when
  @grd700 module_shutdown = FALSE
end

event clear_queuing_port extends clear_queuing_port
when
  @grd700 module_shutdown = FALSE
end

event create_buffer extends create_buffer
when
  @grd700 module_shutdown = FALSE
end

event send_buffer extends send_buffer
when
  @grd700 module_shutdown = FALSE
end
```

```
event send_buffer_needwakeuprecvproc // extends resource_become_available
extends send_buffer_needwakeuprecvproc
when
  @grd700 module_shutdown = FALSE
end

event send_buffer_withfull // extends req_busy_resource
extends send_buffer_withfull
when
  @grd700 module_shutdown = FALSE
end

event receive_buffer extends receive_buffer
when
  @grd700 module_shutdown = FALSE
end

event receive_buffer_needwakeupsendproc // extends resource_become_available
extends receive_buffer_needwakeupsendproc
when
```

```
  @grd700 module_shutdown = FALSE
end

event receive_buffer_whenempty // extends req_busy_resource
extends receive_buffer_whenempty
when
  @grd700 module_shutdown = FALSE
end

event get_buffer_id extends get_buffer_id
when
  @grd700 module_shutdown = FALSE
end

event get_buffer_status extends get_buffer_status
when
  @grd700 module_shutdown = FALSE
end

event create_blackboard extends create_blackboard
when
```

```
  @grd700 module_shutdown = FALSE
end

event display_blackboard extends display_blackboard
when
  @grd700 module_shutdown = FALSE
end

event display_blackboard_needwakeuprdprocs // extends resource_become_available2
extends display_blackboard_needwakeuprdprocs
when
  @grd700 module_shutdown = FALSE
end

event read_blackboard extends read_blackboard
when
  @grd700 module_shutdown = FALSE
end

event read_blackboard_whenempty // extends req_busy_resource
extends read_blackboard_whenempty
```

```
when
  @grd700 module_shutdown = FALSE
end

event clear_blackboard extends clear_blackboard
when
  @grd700 module_shutdown = FALSE
end

event get_blackboard_id extends get_blackboard_id
when
  @grd700 module_shutdown = FALSE
end

event get_blackboard_status extends get_blackboard_status
when
  @grd700 module_shutdown = FALSE
end

event create_semaphore extends create_semaphore
when
```

```
  @grd700 module_shutdown = FALSE
end

event wait_semaphore extends wait_semaphore
when
  @grd700 module_shutdown = FALSE
end

event wait_semahpore_whenzero // extends req_busy_resource
extends wait_semahpore_whenzero
when
  @grd700 module_shutdown = FALSE
end

event signal_semaphore extends signal_semaphore
when
  @grd700 module_shutdown = FALSE
end

event signal_semaphore_needwakeupproc // extends resource_become_available
extends signal_semaphore_needwakeupproc
```

```
when
  @grd700 module_shutdown = FALSE
end

event get_semaphore_id extends get_semaphore_id
when
  @grd700 module_shutdown = FALSE
end

event get_semaphore_status extends get_semaphore_status
when
  @grd700 module_shutdown = FALSE
end

event create_event extends create_event
when
  @grd700 module_shutdown = FALSE
end

event set_event extends set_event
when
```

```
  @grd700 module_shutdown  = FALSE
end

event set_event_needwakeupprocs // extends resource_become_available2
extends set_event_needwakeupprocs
when
  @grd700 module_shutdown  = FALSE
end

event reset_event extends reset_event
when
  @grd700 module_shutdown  = FALSE
end

event wait_event extends wait_event
when
  @grd700 module_shutdown  = FALSE
end

event wait_event_whendown // extends req_busy_resource
extends wait_event_whendown
```

```
when
  @grd700 module_shutdown = FALSE
end

event get_event_id extends get_event_id
when
  @grd700 module_shutdown = FALSE
end

event get_event_status extends get_event_status
when
  @grd700 module_shutdown = FALSE
end

event ticktock // timer interrupt event, triggered by the timer in hardware. one tick in each ONE_TICK_TIME
extends ticktock
when
  @grd700 module_shutdown = FALSE
end

event partition_schedule extends partition_schedule
```

```
  when
    @grd700 module_shutdown = FALSE
  end

  event process_schedule // if there is not error handler and preempter in this partition
  extends process_schedule
  when
    @grd700 module_shutdown = FALSE
  end

  event run_errorhandler_preempter // if there is the error handler, it is executed, otherwise the preempter is
executed
  extends run_errorhandler_preempter
  when
    @grd700 module_shutdown = FALSE
  end

  event get_partition_status extends get_partition_status
  when
    @grd700 module_shutdown = FALSE
  end
```

```
event set_partition_mode_to_idle // shutdown the partition
extends set_partition_mode_to_idle
when
  @grd700 module_shutdown = FALSE
end

event set_partition_mode_to_normal extends set_partition_mode_to_normal
when
  @grd700 module_shutdown = FALSE
end

event set_partition_mode_to_coldstart extends set_partition_mode_to_coldstart
when
  @grd700 module_shutdown = FALSE
end

event set_partition_mode_to_warmstart extends set_partition_mode_to_warmstart
when
  @grd700 module_shutdown = FALSE
end
```

```
event get_process_id extends get_process_id
when
  @grd700 module_shutdown = FALSE
end

event get_process_status extends get_process_status
when
  @grd700 module_shutdown = FALSE
end

event create_process extends create_process
when
  @grd700 module_shutdown = FALSE
end

event set_priority extends set_priority
when
  @grd700 module_shutdown = FALSE
end
```

```
event suspend_self
/* extends suspend_self
    any timeout timeouttrig waittype */
extends suspend_self
when
  @grd700 module_shutdown = FALSE
end

event suspend // extends suspend
extends suspend
when
  @grd700 module_shutdown = FALSE
end

event resume // extends resume
extends resume
when
  @grd700 module_shutdown = FALSE
end

event stop_self extends stop_self
```

```
when
  @grd700 module_shutdown = FALSE
end

event stop extends stop
when
  @grd700 module_shutdown = FALSE
end

event start_aperiodprocess_instart
/* start an aperiodic process in COLD_START or WARM_START mode
    extends start */
extends start_aperiodprocess_instart
when
  @grd700 module_shutdown = FALSE
end

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

```
when
  @grd700 module_shutdown = FALSE
end

event start_periodprocess_instart
/* start a periodic process in COLD_START or WARM_START mode
    extends start */
extends start_periodprocess_instart
when
  @grd700 module_shutdown = FALSE
end

event start_periodprocess_innormal
/* start a periodic process in NORMAL mode
    extends start */
extends start_periodprocess_innormal
when
  @grd700 module_shutdown = FALSE
end

event delaystart_aperiodprocess_instart  // extends delayed_start
```

```
extends delaystart_aperiodprocess_instart
when
  @grd700 module_shutdown = FALSE
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
when
  @grd700 module_shutdown = FALSE
end

event delaystart_periodprocess_instart  // extends delayed_start
extends delaystart_periodprocess_instart
when
  @grd700 module_shutdown = FALSE
end

event delaystart_periodprocess_innormal  // extends delayed_start
```

```
extends delaystart_periodprocess_innormal
when
  @grd700 module_shutdown = FALSE
end

event lock_preemption extends lock_preemption
when
  @grd700 module_shutdown = FALSE
end

event unlock_preemption extends unlock_preemption
when
  @grd700 module_shutdown = FALSE
end

event get_my_id extends get_my_id
when
  @grd700 module_shutdown = FALSE
end

event timed_wait extends timed_wait
```

```
when
  @grd700 module_shutdown = FALSE
end

event period_wait extends period_wait
when
  @grd700 module_shutdown = FALSE
end

event get_time extends get_time
when
  @grd700 module_shutdown = FALSE
end

event replenish extends replenish
when
  @grd700 module_shutdown = FALSE
end

event aperiodicprocess_finished extends aperiodicprocess_finished
when
```

```
  @grd700 module_shutdown = FALSE
end

event periodicprocess_finished extends periodicprocess_finished
when
  @grd700 module_shutdown = FALSE
end

event time_out // should refined to support remove process on waiting queue of comm resources
extends time_out
when
  @grd700 module_shutdown = FALSE
end

event periodicproc_reach_releasepoint extends periodicproc_reach_releasepoint
when
  @grd700 module_shutdown = FALSE
end

event coldstart_partition_fromidle extends coldstart_partition_fromidle
when
```

```
    @grd700 module_shutdown = FALSE
  end

  event warmstart_partition_fromidle extends warmstart_partition_fromidle
  when
    @grd700 module_shutdown = FALSE
  end
end
```