MODULE *HiRTOS*

This spec describes the thread scheduler of a per-CPU *HiRTOS* instance.

EXTENDS *FiniteSets*, *Sequences*, *Naturals*, *TLC*

CONSTANTS
Invalid_Thread_Priority

$Num\_Thread\_Priorities \triangleq 3$
$Num\_Interrupt\_Priorities \triangleq 2$

$Valid\_Thread\_Priority\_Type \triangleq 0 \,..\, Num\_Thread\_Priorities - 1$
$Thread\_Priority\_Type \triangleq Valid\_Thread\_Priority\_Type \cup \{Invalid\_Thread\_Priority\}$

$Threads \triangleq \{\text{“Idle\_Thread”}, \text{“thread1”}, \text{“thread2”}, \text{“thread3”}\}$
$Mutexes \triangleq \{\text{“mutex1”}\}$
$Condvars \triangleq \{\text{“thread1\_condvar”}, \text{“thread2\_condvar”}, \text{“thread3\_condvar”}, \text{“condvar1”}\}$
$Timers \triangleq \{\text{“thread1\_timer”}, \text{“thread2\_timer”}, \text{“thread3\_timer”}\}$
$Interrupts \triangleq \{\text{“Timer\_Interrupt”}\}$

$Thread\_Id\_Type \triangleq Threads \cup \{\text{“Invalid\_Thread\_Id”}\}$
$Mutex\_Id\_Type \triangleq Mutexes \cup \{\text{“Invalid\_Mutex\_Id”}\}$
$Condvar\_Id\_Type \triangleq Condvars \cup \{\text{“Invalid\_Condvar\_Id”}\}$
$Timer\_Id\_Type \triangleq Timers \cup \{\text{“Invalid\_Timer\_Id”}\}$
$Interrupt\_Id\_Type \triangleq Interrupts \cup \{\text{“Invalid\_Interrupt\_Id”}\}$

$Thread\_State\_Type \triangleq \{\text{“Suspended”}, \text{“Runnable”}, \text{“Running”},$
$\text{“Blocked\_On\_Condvar”}, \text{“Blocked\_On\_Mutex”}\}$

$Timer\_State\_Type \triangleq \{\text{“Timer\_Stopped”}, \text{“Timer\_Running”}\}$

$No\_Duplicates\_List\_Type(S) \triangleq$
$\{x \in Seq(S) : Len(x) \neq 0 \Rightarrow \forall\, i, j \in 1 \,..\, Len(x) : i \neq j \Rightarrow x[i] \neq x[j]\}$

$Range(f) \triangleq \{f[x] : x \in \text{DOMAIN } f\}$

$Thread\_Queue\_Type \triangleq No\_Duplicates\_List\_Type(Thread\_Id\_Type)$

$Thread\_Priority\_Queue\_Type \triangleq [Valid\_Thread\_Priority\_Type \to Thread\_Queue\_Type]$

$HiRTOS\_Type \triangleq [$
$Current\_Thread\_Id : Thread\_Id\_Type,$
$Runnable\_Threads\_Queue : Thread\_Priority\_Queue\_Type,$
$Interrupts\_Enabled : \text{BOOLEAN}$
$]$

$$
\begin{array}{l}
HiRTOS\_Initializer \triangleq [\\
\quad Current\_Thread\_Id \mapsto \text{“Invalid\_Thread\_Id”}, \\
\quad Runnable\_Threads\_Queue \mapsto \\
\qquad [p \in Valid\_Thread\_Priority\_Type \mapsto \\
\qquad\quad \text{CASE } p = 0 \to \langle \text{“Idle\_Thread”} \rangle \\
\qquad\quad \Box p = 1 \to \langle \text{“thread1”} \rangle \\
\qquad\quad \Box p = 2 \to \langle \text{“thread2”}, \text{“thread3”} \rangle \\
\qquad], \\
\quad Interrupts\_Enabled \mapsto \text{TRUE} \\
]
\end{array}
$$

$$
\begin{array}{l}
Thread\_Object\_Type \triangleq [\\
\quad State : Thread\_State\_Type, \\
\quad Current\_Priority : Valid\_Thread\_Priority\_Type, \\
\quad Base\_Priority : Valid\_Thread\_Priority\_Type, \\
\quad Builtin\_Timer\_Id : Timer\_Id\_Type, \\
\quad Builtin\_Condvar\_Id : Condvar\_Id\_Type, \\
\quad Waiting\_On\_Condvar\_Id : Condvar\_Id\_Type, \\
\quad Waiting\_On\_Mutex\_Id : Mutex\_Id\_Type, \\
\quad Owned\_Mutexes : No\_Duplicates\_List\_Type(Mutex\_Id\_Type), \\
\quad ghost\_Time\_Slice\_Consumed : \text{BOOLEAN}, \\
\quad ghost\_Condvar\_Wait\_Mutex\_Id : Mutex\_Id\_Type \\
]
\end{array}
$$

$$
\begin{array}{l}
Thread\_Object\_Initializer(priority,\ timer\_id,\ condvar\_id) \triangleq [\\
\quad State \mapsto \text{“Runnable”}, \\
\quad Current\_Priority \mapsto priority, \\
\quad Base\_Priority \mapsto priority, \\
\quad Builtin\_Timer\_Id \mapsto timer\_id, \\
\quad Builtin\_Condvar\_Id \mapsto condvar\_id, \\
\quad Waiting\_On\_Condvar\_Id \mapsto \text{“Invalid\_Condvar\_Id”}, \\
\quad Waiting\_On\_Mutex\_Id \mapsto \text{“Invalid\_Mutex\_Id”}, \\
\quad Owned\_Mutexes \mapsto \langle\rangle, \\
\quad ghost\_Time\_Slice\_Consumed \mapsto \text{FALSE}, \\
\quad ghost\_Condvar\_Wait\_Mutex\_Id \mapsto \text{“Invalid\_Mutex\_Id”} \\
]
\end{array}
$$

$$
\begin{array}{l}
Mutex\_Object\_Type \triangleq [\\
\quad Owner\_Thread\_Id : Thread\_Id\_Type, \\
\quad Last\_Inherited\_Priority : Thread\_Priority\_Type, \\
\quad Waiting\_Threads\_Queue : Thread\_Priority\_Queue\_Type \\
]
\end{array}
$$

$$
\begin{array}{l}
Mutex\_Object\_Initializer \triangleq [\\
\quad Owner\_Thread\_Id \mapsto \text{“Invalid\_Thread\_Id”}, \\
\quad Last\_Inherited\_Priority \mapsto Invalid\_Thread\_Priority,
\end{array}
$$

```
    Waiting_Threads_Queue ↦
        [p ∈ Valid_Thread_Priority_Type ↦ ⟨⟩]
]

Condvar_Object_Type ≜ [
    Wakeup_Mutex_Id : Mutex_Id_Type,
    Waiting_Threads_Queue : Thread_Priority_Queue_Type
]

Condvar_Object_Initializer ≜ [
    Wakeup_Mutex_Id ↦ "Invalid_Mutex_Id",
    Waiting_Threads_Queue ↦
        [p ∈ Valid_Thread_Priority_Type ↦ ⟨⟩]
]

Timer_Object_Type ≜ [
    State : Timer_State_Type
]

Timer_Object_Initializer ≜ [
    State ↦ "Timer_Stopped"
]

Is_Thread_Priority_Queue_Empty(prio_queue) ≜
  ∀ p ∈ Valid_Thread_Priority_Type : prio_queue[p] = ⟨⟩

Is_Thread_In_Priority_Queue(prio_queue, thread_id) ≜
    ∃ p ∈ Valid_Thread_Priority_Type : thread_id ∈ Range(prio_queue[p])

Is_Thread_In_Priority_Queue_In_Only_One_Queue(prio_queue, thread_id) ≜
    Is_Thread_In_Priority_Queue(prio_queue, thread_id) ⇒
    Cardinality({p ∈ Valid_Thread_Priority_Type : thread_id ∈ Range(prio_queue[p])}) = 1

Get_Highest_Priority(prio_queue) ≜
    LET
        Non_Empty_Queues ≜ {p ∈ Valid_Thread_Priority_Type : prio_queue[p] ≠ ⟨⟩}
    IN
        CHOOSE p1 ∈ Non_Empty_Queues :
            ∀ p2 ∈ Non_Empty_Queues \ {p1} : p2 < p1

Get_Highest_Priority_Queue(prio_queue) ≜
      prio_queue[Get_Highest_Priority(prio_queue)]

 ****************************************************************************
--algorithm hirtos_threads_model
  variables
      HiRTOS = HiRTOS_Initializer,
      Thread_Objects = [
```

```
    Idle_Thread ↦
        Thread_Object_Initializer(0, "Invalid_Timer_Id", "Invalid_Condvar_Id"),
    thread1 ↦
        Thread_Object_Initializer(1, "thread1_timer", "thread1_condvar"),
    thread2 ↦
        Thread_Object_Initializer(2, "thread2_timer", "thread2_condvar"),
    thread3 ↦
        Thread_Object_Initializer(2, "thread3_timer", "thread3_condvar")
  ],
  Mutex_Objects = [m ∈ Mutexes ↦ Mutex_Object_Initializer],
  Condvar_Objects = [cv ∈ Condvars ↦ Condvar_Object_Initializer],
  Timer_Objects = [tm ∈ Timers ↦ Timer_Object_Initializer],
  Global_Resource_Available = FALSE ;

define
  Enqueue_Thread(priority_queue, thread_id) ≜
    LET
      priority ≜ Thread_Objects[thread_id].Current_Priority
    IN
      [priority_queue EXCEPT ![priority] = Append(@, thread_id)]

  Enqueue_Thread_As_Head(priority_queue, thread_id) ≜
    LET
      priority ≜ Thread_Objects[thread_id].Current_Priority
    IN
      [priority_queue EXCEPT ![priority] = ⟨thread_id⟩ ∘ @]

  Priority_Queue_Head(priority_queue) ≜
    Head(Get_Highest_Priority_Queue(priority_queue))

  Priority_Queue_Tail(priority_queue) ≜ [
      priority_queue EXCEPT ![Get_Highest_Priority(priority_queue)] = Tail(@)
  ]

end define ;

---------------------------------------------------------------------------

Macros

---------------------------------------------------------------------------

macro Move_Thread_To_Another_Queue(priority_queue, thread_id, old_prio, new_prio)
begin
  assert new_prio ≠ old_prio ;
    Remove thread from its current queue:
  priority_queue[old_prio] := SelectSeq(priority_queue[old_prio], LAMBDA x : x ≠ thread_id) ||
    Enqueue thread to new queue:
  priority_queue[new_prio] := Append(priority_queue[new_prio], thread_id) ||
```

```
   Thread_Objects[thread_id].Current_Priority := new_prio ;
end macro ;

macro Enter_Critical_Section(context_id)
begin
  await HiRTOS.Interrupts_Enabled ∧
        (context_id ∈ Threads ⇒
            Thread_Objects[context_id].State = "Running") ;
  HiRTOS.Interrupts_Enabled := FALSE ;
end macro ;

macro Exit_Critical_Section()
begin
  HiRTOS.Interrupts_Enabled := TRUE ;
end macro ;
```

Procedures

```
procedure Run_Thread_Scheduler()
begin
  check_time_slice_step:
  assert ¬HiRTOS.Interrupts_Enabled ;
  if HiRTOS.Current_Thread_Id ≠ "Invalid_Thread_Id" then
    Thread_Objects[HiRTOS.Current_Thread_Id].State := "Runnable" ;
    if Thread_Objects[HiRTOS.Current_Thread_Id].ghost_Time_Slice_Consumed then
      HiRTOS.Runnable_Threads_Queue :=
          Enqueue_Thread(HiRTOS.Runnable_Threads_Queue, HiRTOS.Current_Thread_Id) ||
      HiRTOS.Current_Thread_Id := "Invalid_Thread_Id" ;
    else
      HiRTOS.Runnable_Threads_Queue :=
          Enqueue_Thread_As_Head(HiRTOS.Runnable_Threads_Queue,
                                 HiRTOS.Current_Thread_Id) ||
      HiRTOS.Current_Thread_Id := "Invalid_Thread_Id" ;
    end if ;
  end if ;

  choose_next_thread_step:
  HiRTOS.Current_Thread_Id := Priority_Queue_Head(HiRTOS.Runnable_Threads_Queue) ||
  HiRTOS.Runnable_Threads_Queue := Priority_Queue_Tail(HiRTOS.Runnable_Threads_Queue) ;
  assert HiRTOS.Current_Thread_Id ≠ "Invalid_Thread_Id" ;
  Thread_Objects[HiRTOS.Current_Thread_Id].ghost_Time_Slice_Consumed := FALSE ||
  Thread_Objects[HiRTOS.Current_Thread_Id].State := "Running" ;

  run_scheduler_return_step:
  return ;
```

```
end procedure ;

procedure Do_Acquire_Mutex(thread_id, mutex_id, waking_up_thread_after_condvar_wait)
  variable owner_thread_id = "Invalid_Thread_Id" ;
begin
  acquire_mutex_step:
  assert ¬HiRTOS.Interrupts_Enabled ;
  if Mutex_Objects[mutex_id].Owner_Thread_Id = "Invalid_Thread_Id" then
    acquire_mutex_acquire_step:
    Mutex_Objects[mutex_id].Owner_Thread_Id := thread_id ||
    Thread_Objects[thread_id].Owned_Mutexes  :=
        ⟨mutex_id⟩ ∘ Thread_Objects[thread_id].Owned_Mutexes ;
    if waking_up_thread_after_condvar_wait then
      acquire_mutex_make_condvar_wait_awoken_thread_runnable_step:
      assert thread_id ≠ HiRTOS.Current_Thread_Id ;
      assert Thread_Objects[thread_id].State = "Blocked_On_Condvar" ;
      assert thread_id ∉
          Range(HiRTOS.Runnable_Threads_Queue[Thread_Objects[thread_id].Current_Priority]) ;
      HiRTOS.Runnable_Threads_Queue :=
          Enqueue_Thread(HiRTOS.Runnable_Threads_Queue, thread_id) ||
      Thread_Objects[thread_id].State := "Runnable" ;
    else
      assert thread_id = HiRTOS.Current_Thread_Id ;
      assert Thread_Objects[thread_id].State = "Running" ;
    end if ;
  else
    acquire_mutex_wait_on_mutex_step:
    owner_thread_id := Mutex_Objects[mutex_id].Owner_Thread_Id ;
    assert owner_thread_id ≠ thread_id ;
    Mutex_Objects[mutex_id].Waiting_Threads_Queue :=
        Enqueue_Thread(Mutex_Objects[mutex_id].Waiting_Threads_Queue, thread_id) ;
    if waking_up_thread_after_condvar_wait then
      assert thread_id ≠ HiRTOS.Current_Thread_Id ;
      assert Thread_Objects[thread_id].State = "Blocked_On_Condvar" ;
    else
      assert thread_id = HiRTOS.Current_Thread_Id ;
      assert Thread_Objects[thread_id].State = "Running" ;
      HiRTOS.Current_Thread_Id := "Invalid_Thread_Id" ;
    end if ;

    Thread_Objects[thread_id].State := "Blocked_On_Mutex" ||
    Thread_Objects[thread_id].Waiting_On_Mutex_Id := mutex_id ;

    acquire_mutex_check_if_priority_inheritance_needed_step:
    if Thread_Objects[owner_thread_id].Current_Priority <
        Thread_Objects[thread_id].Current_Priority then
```

```
        acquire_mutex_priority_inheritance_step:
        Mutex_Objects[mutex_id].Last_Inherited_Priority :=
            Thread_Objects[thread_id].Current_Priority ;
        if Thread_Objects[owner_thread_id].State = "Runnable" then
          acquire_mutex_priority_inheritance_if_mutex_owner_runnable_step:
          Move_Thread_To_Another_Queue(
                HiRTOS.Runnable_Threads_Queue,
                owner_thread_id,
                Thread_Objects[owner_thread_id].Current_Priority,
                Thread_Objects[thread_id].Current_Priority) ;
         elsif Thread_Objects[owner_thread_id].State = "Blocked_On_Mutex" then
          acquire_mutex_priority_inheritance_if_mutex_owner_blocked_on_mutex_step:
          Move_Thread_To_Another_Queue(
            Mutex_Objects[Thread_Objects[owner_thread_id].Waiting_On_Mutex_Id].
              Waiting_Threads_Queue,
            owner_thread_id,
            Thread_Objects[owner_thread_id].Current_Priority,
            Thread_Objects[thread_id].Current_Priority) ;
         else
          assert Thread_Objects[owner_thread_id].State = "Blocked_On_Condvar" ;
          acquire_mutex_priority_inheritance_if_mutex_owner_blocked_on_condvar_step:
          Move_Thread_To_Another_Queue(
            Mutex_Objects[Thread_Objects[owner_thread_id].Waiting_On_Condvar_Id].
              Waiting_Threads_Queue,
            owner_thread_id,
            Thread_Objects[owner_thread_id].Current_Priority,
            Thread_Objects[thread_id].Current_Priority) ;
        end if ;

        acquire_mutex_priority_inheritance_update_prio_step:
        Thread_Objects[owner_thread_id].Current_Priority :=
            Thread_Objects[thread_id].Current_Priority ;
      end if ;

      acquire_mutex_check_if_synchronous_context_switch_needed_step:
      if ¬waking_up_thread_after_condvar_wait then
        acquire_mutex_synchronous_context_switch_step:
        call Run_Thread_Scheduler() ;
      end if ;
    end if ;

    do_acquire_mutex_return_step:
    return ;
end procedure ;

procedure Acquire_Mutex(thread_id, mutex_id)
```

```
   variable owner_thread_id = "Invalid_Thread_Id" ;
begin
   enter_critical_section_step:
   Enter_Critical_Section(thread_id) ;
   assert HiRTOS.Current_Thread_Id = thread_id ;
   call Do_Acquire_Mutex(thread_id, mutex_id, FALSE) ;
   exit_critical_section_step:
   Exit_Critical_Section() ;
   acquire_mutex_return_step:
   return ;
end procedure ;

procedure Do_Release_Mutex(thread_id, mutex_id, doing_condvar_wait)
   variable awoken_thread_id = "Invalid_Thread_Id" ;
begin
   release_mutex_step:
   assert ¬HiRTOS.Interrupts_Enabled ;
   assert Mutex_Objects[mutex_id].Owner_Thread_Id = thread_id ;
   assert Thread_Objects[thread_id].Owned_Mutexes  ≠ ⟨⟩ ;
   assert Head(Thread_Objects[thread_id].Owned_Mutexes) = mutex_id ;
   Thread_Objects[thread_id].Owned_Mutexes := Tail(Thread_Objects[thread_id].Owned_Mutexes) ;

   release_mutex_restore_priority_step:
   if Thread_Objects[thread_id].Owned_Mutexes ≠ ⟨⟩ ∧ ¬doing_condvar_wait then
     with prev_mutex_obj = Mutex_Objects[Head(Thread_Objects[thread_id].Owned_Mutexes)] do
       if prev_mutex_obj.Last_Inherited_Priority ≠ Invalid_Thread_Priority then
         Thread_Objects[thread_id].Current_Priority := prev_mutex_obj.Last_Inherited_Priority ;
       end if ;
     end with ;
    else
      Thread_Objects[thread_id].Current_Priority := Thread_Objects[thread_id].Base_Priority ;
   end if ;

   release_mutex_check_if_mutex_waiters_step:
   if Is_Thread_Priority_Queue_Empty(Mutex_Objects[mutex_id].Waiting_Threads_Queue) then
     Mutex_Objects[mutex_id].Owner_Thread_Id := "Invalid_Thread_Id" ;
    else
     release_mutex_wakeup_mutex_waiter_step:
     awoken_thread_id :=
       Priority_Queue_Head(Mutex_Objects[mutex_id].Waiting_Threads_Queue) ;
     assert Thread_Objects[awoken_thread_id].Waiting_On_Mutex_Id = mutex_id ;
     Mutex_Objects[mutex_id].Owner_Thread_Id := awoken_thread_id ||
     Mutex_Objects[mutex_id].Waiting_Threads_Queue :=
       Priority_Queue_Tail(Mutex_Objects[mutex_id].Waiting_Threads_Queue) ||
     HiRTOS.Runnable_Threads_Queue :=
       Enqueue_Thread(HiRTOS.Runnable_Threads_Queue, awoken_thread_id) ||
```

```
        Thread_Objects[awoken_thread_id].State := "Runnable" ||
        Thread_Objects[awoken_thread_id].Waiting_On_Mutex_Id := "Invalid_Mutex_Id" ||
        Thread_Objects[awoken_thread_id].Owned_Mutexes :=
          ⟨mutex_id⟩ ∘ Thread_Objects[awoken_thread_id].Owned_Mutexes ;

        if ¬doing_condvar_wait then
          release_mutex_synchronous_context_switch_step:
          call Run_Thread_Scheduler() ;
        end if ;
      end if ;

      do_release_mutex_return_step:
      return ;
    end procedure ;

  procedure Release_Mutex(thread_id, mutex_id)
    begin
      enter_critical_section_step:
      Enter_Critical_Section(thread_id) ;
      assert HiRTOS.Current_Thread_Id = thread_id ;
      call Do_Release_Mutex(thread_id, mutex_id, FALSE) ;
      exit_critical_section_step:
      Exit_Critical_Section() ;
      release_mutex_return_step:
      return ;
    end procedure ;

    procedure Do_Wait_On_Condvar(thread_id, condvar_id, mutex_id)
    begin
      wait_on_condvar_wait_step:
      assert ¬HiRTOS.Interrupts_Enabled ;
      Thread_Objects[thread_id].ghost_Condvar_Wait_Mutex_Id := mutex_id ||
      Condvar_Objects[condvar_id].Waiting_Threads_Queue :=
        Enqueue_Thread(Condvar_Objects[condvar_id].Waiting_Threads_Queue, thread_id) ||
      Thread_Objects[thread_id].State := "Blocked_On_Condvar" ||
      Thread_Objects[thread_id].Waiting_On_Condvar_Id := condvar_id ;
      HiRTOS.Current_Thread_Id := "Invalid_Thread_Id" ;

      wait_on_condvar_release_mutex_step:
      if mutex_id ≠ "Invalid_Mutex_Id" then
        call Do_Release_Mutex(thread_id, mutex_id, TRUE) ;
      end if ;

      wait_on_condvar_synchronous_context_switch_step:
      call Run_Thread_Scheduler() ;

      do_wait_on_condvar_return_step:
```

```
   return ;
end procedure ;

procedure Wait_On_Condvar(thread_id, condvar_id, mutex_id)
begin
   enter_critical_section_step:
   Enter_Critical_Section(thread_id) ;
   call Do_Wait_On_Condvar(thread_id, condvar_id, mutex_id) ;
   exit_critical_section_step:
   Exit_Critical_Section() ;
   wait_on_condvar_return_step:
   return ;
end procedure ;

procedure Do_Signal_Condvar(condvar_id, do_context_switch)
   variables awoken_thread_id = "Invalid_Thread_Id",
             to_reacquire_mutex_id = "Invalid_Mutex_Id" ;
begin
   signal_condvar_step:
   assert ¬HiRTOS.Interrupts_Enabled ;
   if ¬Is_Thread_Priority_Queue_Empty(Condvar_Objects[condvar_id].Waiting_Threads_Queue)
    then
      signal_condvar_wakeup_waiter_step:
      awoken_thread_id :=
         Priority_Queue_Head(Condvar_Objects[condvar_id].Waiting_Threads_Queue) ;
     Condvar_Objects[condvar_id].Waiting_Threads_Queue :=
         Priority_Queue_Tail(Condvar_Objects[condvar_id].Waiting_Threads_Queue) ;
      assert awoken_thread_id ≠ HiRTOS.Current_Thread_Id ;
      assert Thread_Objects[awoken_thread_id].Waiting_On_Condvar_Id = condvar_id ;
      assert Thread_Objects[awoken_thread_id].Waiting_On_Mutex_Id = "Invalid_Mutex_Id" ;

      to_reacquire_mutex_id := Thread_Objects[awoken_thread_id].ghost_Condvar_Wait_Mutex_Id ;
      Thread_Objects[awoken_thread_id].ghost_Condvar_Wait_Mutex_Id := "Invalid_Mutex_Id" ||
      Thread_Objects[awoken_thread_id].Waiting_On_Condvar_Id := "Invalid_Condvar_Id" ;

      signal_condvar_check_if_mutex_reacquire_needed_step:
      if to_reacquire_mutex_id ≠ "Invalid_Mutex_Id" then
         signal_condvar_reacquire_mutex_step:
         call Do_Acquire_Mutex(awoken_thread_id, to_reacquire_mutex_id, TRUE) ;
       else
         signal_condvar_awoken_thread_runnable_step:
         HiRTOS.Runnable_Threads_Queue :=
            Enqueue_Thread(HiRTOS.Runnable_Threads_Queue, awoken_thread_id) ||
         Thread_Objects[awoken_thread_id].State := "Runnable" ;
      end if ;
```

```
      signal_condvar_check_if_sync_context_switch_needed_step:
      if do_context_switch then
        signal_condvar_synchronous_context_switch_step:
        call Run_Thread_Scheduler() ;
      end if ;
    end if ;

    do_condvar_signal_return_step:
    return ;
end procedure ;

procedure Signal_Condvar(context_id, condvar_id)
begin
    enter_critical_section_step:
    Enter_Critical_Section(context_id) ;
    call Do_Signal_Condvar(condvar_id, TRUE) ;
    exit_critical_section_step:
    Exit_Critical_Section() ;
    condvar_signaled_step:
    return ;
end procedure ;

procedure Broadcast_Condvar(context_id, condvar_id)
    variable thread_was_awaken = FALSE ;
begin
    enter_critical_section_step:
    Enter_Critical_Section(context_id) ;

    broadcast_condvar_step:
    while ¬Is_Thread_Priority_Queue_Empty(Condvar_Objects[condvar_id].Waiting_Threads_Queue)
     do
       broadcast_condvar_wakeup_waiter_step:
       call Do_Signal_Condvar(condvar_id, FALSE) ;
      broadcast_condvar_after_waking_up_one_waiter_step:
      thread_was_awaken := TRUE ;
    end while ;

    broadcast_condvar_check_if_sync_context_switch_needed_step:
    if context_id ∈ Threads ∧ thread_was_awaken then
      broadcast_condvar_synchronous_context_switch_step:
      call Run_Thread_Scheduler() ;
    end if ;

    exit_critical_section_step:
    Exit_Critical_Section() ;
    condvar_broadcasted_step:
    return ;
```

```
end procedure ;

procedure Delay_Until(thread_id)
begin
  enter_critical_section_step:
  Enter_Critical_Section(thread_id) ;

  delay_until_step:
  Timer_Objects[Thread_Objects[thread_id].Builtin_Timer_Id].State := "Timer_Running" ;
  call Do_Wait_On_Condvar(thread_id, Thread_Objects[thread_id].Builtin_Condvar_Id,
                          "Invalid_Mutex_Id") ;

  exit_critical_section_step:
  Exit_Critical_Section() ;

  after_delay_until_step:
  return ;
end procedure ;
```

Processes

```
fair process Thread_State_Machine ∈ Threads \ {"Idle_Thread"}
begin
  thread_state_machine_next_state_loop:
  while TRUE do
    await Thread_Objects[self].State = "Running" ∧ HiRTOS.Interrupts_Enabled ;
    context_switch0:
    either
      acquire_mutex_step:
      call Acquire_Mutex(self, "mutex1") ;
      context_switch1:
      await Thread_Objects[self].State = "Running" ∧ HiRTOS.Interrupts_Enabled ;
      assert (Mutex_Objects["mutex1"].Owner_Thread_Id = self) ;

      either
        waiting_for_resource_step:
        while ¬Global_Resource_Available do
          call Wait_On_Condvar(self, "condvar1", "mutex1") ;
          context_switch2:
          await Thread_Objects[self].State = "Running" ∧ HiRTOS.Interrupts_Enabled ;
          assert Mutex_Objects["mutex1"].Owner_Thread_Id = self ;
        end while ;

        Global_Resource_Available := FALSE ;
      or
```

```
            skip ;
          end either ;

          release_mutex_step:
          call Release_Mutex(self, "mutex1") ;
        or
          Global_Resource_Available := TRUE ;
          either
            call Signal_Condvar(self, "condvar1") ;
          or
            call Broadcast_Condvar(self, "condvar1") ;
          end either ;
        or
          call Delay_Until(self)
         end either ;

        thread_iteration_completed_step:
        skip ;
    end while ;
end process ;

fair process Idle_Thread = "Idle_Thread"
begin
    idle_thread_next_state_loop:
    while TRUE do
        await Thread_Objects["Idle_Thread"].State = "Running" ∧ HiRTOS.Interrupts_Enabled ;
    end while ;
end process ;

fair process Timer_Interrupt = "Timer_Interrupt"
    variable delayed_threads = {} ;
begin
    timer_interrupt_next_state_loop:
    while TRUE do
        enter_critical_section_step:
        Enter_Critical_Section("Timer_Interrupt") ;

        track_time_slice:
        if HiRTOS.Current_Thread_Id ≠ "Invalid_Thread_Id" then
          assert ¬Thread_Objects[HiRTOS.Current_Thread_Id].ghost_Time_Slice_Consumed ;
          Thread_Objects[HiRTOS.Current_Thread_Id].ghost_Time_Slice_Consumed := TRUE ;
        end if ;

        delayed_threads :=
          {t ∈ Threads \ {"Idle_Thread"} :
            Timer_Objects[Thread_Objects[t].Builtin_Timer_Id].State = "Timer_Running"} ;
```

```
      wakeup_delay_until_waiters:
      while delayed_threads ≠ {} do
        with t ∈ delayed_threads do
          delayed_threads := delayed_threads \ {t};
          Timer_Objects[Thread_Objects[t].Builtin_Timer_Id].State := "Timer_Stopped";
          call Do_Signal_Condvar(Thread_Objects[t].Builtin_Condvar_Id, FALSE);
        end with ;
      end while ;

      timer_interupt_asynchronous_context_switch_step:
      call Run_Thread_Scheduler();

      exit_critical_section_step:
      Exit_Critical_Section();
    end while ;
  end process ;

  fair process Other_Interrupt = "Other_Interrupt"
  begin
    other_interrupt_next_state_loop:
    while TRUE do
      enter_critical_section_step:
      Enter_Critical_Section("Other_Interrupt");

      other_interupt_asynchronous_context_switch_step:
      call Run_Thread_Scheduler();

      exit_critical_section_step:
      Exit_Critical_Section();
    end while ;
  end process ;

end algorithm ;
```

****

BEGIN TRANSLATION ($chksum(pcal) =$ "$ec6916c$" $\land$ $chksum(tla) =$ "$5dd4fa09$")

Label *acquire_mutex_step* of procedure *Do_Acquire_Mutex* at line 248 col 7 changed to *acquire_mutex_step_*

Label *enter_critical_section_step* of procedure *Acquire_Mutex* at line 200 col 7 changed to *enter_critical_section_step_*

Label *exit_critical_section_step* of procedure *Acquire_Mutex* at line 208 col 7 changed to *exit_critical_section_step_*

Label *release_mutex_step* of procedure *Do_Release_Mutex* at line 350 col 7 changed to *release_mutex_step_*

Label *enter_critical_section_step* of procedure *Release_Mutex* at line 200 col 7 changed to *enter_critical_section_step_R*

Label *exit_critical_section_step* of procedure *Release_Mutex* at line 208 col 7 changed to *exit_critical_section_step_R*

Label *enter_critical_section_step* of procedure *Wait_On_Condvar* at line 200 col 7 changed to *enter_critical_section_step_*

Label *exit_critical_section_step* of procedure *Wait_On_Condvar* at line 208 col 7 changed to *exit_critical_section_step_W*

Label *enter_critical_section_step* of procedure *Signal_Condvar* at line 200 col 7 changed to *enter_critical_section_step_S*

Label *exit_critical_section_step* of procedure *Signal_Condvar* at line 208 col 7 changed to *exit_critical_section_step_S*

Label *enter_critical_section_step* of procedure *Broadcast_Condvar* at line 200 col 7 changed to *enter_critical_section_step_*

Label *exit_critical_section_step* of procedure *Broadcast_Condvar* at line 208 col 7 changed to *exit_critical_section_step_B*
Label *enter_critical_section_step* of procedure *Delay_Until* at line 200 col 7 changed to *enter_critical_section_step_D*
Label *exit_critical_section_step* of procedure *Delay_Until* at line 208 col 7 changed to *exit_critical_section_step_D*
Label *enter_critical_section_step* of process *Timer_Interrupt* at line 200 col 7 changed to *enter_critical_section_step_T*
Label *exit_critical_section_step* of process *Timer_Interrupt* at line 208 col 7 changed to *exit_critical_section_step_T*
Procedure variable *owner_thread_id* of procedure *Do_Acquire_Mutex* at line 245 col 16 changed to *owner_thread_id_*
Procedure variable *awoken_thread_id* of procedure *Do_Release_Mutex* at line 347 col 16 changed to *awoken_thread_id_*
Parameter *thread_id* of procedure *Do_Acquire_Mutex* at line 244 col 31 changed to *thread_id_*
Parameter *mutex_id* of procedure *Do_Acquire_Mutex* at line 244 col 42 changed to *mutex_id_*
Parameter *thread_id* of procedure *Acquire_Mutex* at line 333 col 28 changed to *thread_id_A*
Parameter *mutex_id* of procedure *Acquire_Mutex* at line 333 col 39 changed to *mutex_id_A*
Parameter *thread_id* of procedure *Do_Release_Mutex* at line 346 col 31 changed to *thread_id_D*
Parameter *mutex_id* of procedure *Do_Release_Mutex* at line 346 col 42 changed to *mutex_id_D*
Parameter *thread_id* of procedure *Release_Mutex* at line 395 col 25 changed to *thread_id_R*
Parameter *mutex_id* of procedure *Release_Mutex* at line 395 col 36 changed to *mutex_id_R*
Parameter *thread_id* of procedure *Do_Wait_On_Condvar* at line 407 col 33 changed to *thread_id_Do*
Parameter *condvar_id* of procedure *Do_Wait_On_Condvar* at line 407 col 44 changed to *condvar_id_*
Parameter *mutex_id* of procedure *Do_Wait_On_Condvar* at line 407 col 56 changed to *mutex_id_Do*
Parameter *thread_id* of procedure *Wait_On_Condvar* at line 430 col 30 changed to *thread_id_W*
Parameter *condvar_id* of procedure *Wait_On_Condvar* at line 430 col 41 changed to *condvar_id_W*
Parameter *condvar_id* of procedure *Do_Signal_Condvar* at line 441 col 32 changed to *condvar_id_D*
Parameter *context_id* of procedure *Signal_Condvar* at line 484 col 29 changed to *context_id_*
Parameter *condvar_id* of procedure *Signal_Condvar* at line 484 col 41 changed to *condvar_id_S*

CONSTANT *defaultInitValue*

VARIABLES *HiRTOS*, *Thread_Objects*, *Mutex_Objects*, *Condvar_Objects*,
Timer_Objects, *Global_Resource_Available*, *pc*, *stack*

define statement

$$Enqueue\_Thread(priority\_queue, thread\_id) \triangleq$$
LET
$priority \triangleq Thread\_Objects[thread\_id].Current\_Priority$
IN
$[priority\_queue \text{ EXCEPT } ![priority] = Append(@, thread\_id)]$

$$Enqueue\_Thread\_As\_Head(priority\_queue, thread\_id) \triangleq$$
LET
$priority \triangleq Thread\_Objects[thread\_id].Current\_Priority$
IN
$[priority\_queue \text{ EXCEPT } ![priority] = \langle thread\_id \rangle \circ @]$

$$Priority\_Queue\_Head(priority\_queue) \triangleq$$
$Head(Get\_Highest\_Priority\_Queue(priority\_queue))$

$$Priority\_Queue\_Tail(priority\_queue) \triangleq [$$
$priority\_queue \text{ EXCEPT } ![Get\_Highest\_Priority(priority\_queue)] = Tail(@)$
]

VARIABLES *thread_id_*, *mutex_id_*, *waking_up_thread_after_condvar_wait*,
*owner_thread_id_*, *thread_id_A*, *mutex_id_A*, *owner_thread_id*,
thread_id_D, *mutex_id_D*, *doing_condvar_wait*, *awoken_thread_id_*,
thread_id_R, *mutex_id_R*, *thread_id_Do*, *condvar_id_*, *mutex_id_Do*,
thread_id_W, *condvar_id_W*, *mutex_id*, *condvar_id_D*,
do_context_switch, *awoken_thread_id*, *to_reacquire_mutex_id*,
*context_id_*, *condvar_id_S*, *context_id*, *condvar_id*,
thread_was_awaken, *thread_id*, *delayed_threads*

$vars \triangleq \langle HiRTOS,\ Thread\_Objects,\ Mutex\_Objects,\ Condvar\_Objects,$
$Timer\_Objects,\ Global\_Resource\_Available,\ pc,\ stack,\ thread\_id\_,$
$mutex\_id\_,\ waking\_up\_thread\_after\_condvar\_wait,\ owner\_thread\_id\_,$
$thread\_id\_A,\ mutex\_id\_A,\ owner\_thread\_id,\ thread\_id\_D,\ mutex\_id\_D,$
$doing\_condvar\_wait,\ awoken\_thread\_id\_,\ thread\_id\_R,\ mutex\_id\_R,$
$thread\_id\_Do,\ condvar\_id\_,\ mutex\_id\_Do,\ thread\_id\_W,\ condvar\_id\_W,$
$mutex\_id,\ condvar\_id\_D,\ do\_context\_switch,\ awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,\ context\_id\_,\ condvar\_id\_S,\ context\_id,$
$condvar\_id,\ thread\_was\_awaken,\ thread\_id,\ delayed\_threads\rangle$

$ProcSet \triangleq (Threads \setminus \{\text{“Idle\_Thread”}\}) \cup \{\text{“Idle\_Thread”}\} \cup \{\text{“Timer\_Interrupt”}\} \cup \{\text{“Other\_Interrupt”}\}$

$Init \triangleq$ Global variables

$\land\ HiRTOS = HiRTOS\_Initializer$

$\land\ Thread\_Objects = [$
$Idle\_Thread \mapsto Thread\_Object\_Initializer(0,\ \text{“Invalid\_Timer\_Id”},\ \text{“Invalid\_Condvar\_Id”}),$
$thread1 \mapsto Thread\_Object\_Initializer(1,\ \text{“thread1\_timer”},\ \text{“thread1\_condvar”}),$
$thread2 \mapsto Thread\_Object\_Initializer(2,\ \text{“thread2\_timer”},\ \text{“thread2\_condvar”}),$
$thread3 \mapsto Thread\_Object\_Initializer(2,\ \text{“thread3\_timer”},\ \text{“thread3\_condvar”})$
$]$

$\land\ Mutex\_Objects = [m \in Mutexes \mapsto Mutex\_Object\_Initializer]$

$\land\ Condvar\_Objects = [cv \in Condvars \mapsto Condvar\_Object\_Initializer]$

$\land\ Timer\_Objects = [tm \in Timers \mapsto Timer\_Object\_Initializer]$

$\land\ Global\_Resource\_Available = \text{FALSE}$

Procedure *Do_Acquire_Mutex*

$\land\ thread\_id\_ = [self \in ProcSet \mapsto defaultInitValue]$

$\land\ mutex\_id\_ = [self \in ProcSet \mapsto defaultInitValue]$

$\land\ waking\_up\_thread\_after\_condvar\_wait = [self \in ProcSet \mapsto defaultInitValue]$

$\land\ owner\_thread\_id\_ = [self \in ProcSet \mapsto \text{“Invalid\_Thread\_Id”}]$

Procedure *Acquire_Mutex*

$\land\ thread\_id\_A = [self \in ProcSet \mapsto defaultInitValue]$

$\land\ mutex\_id\_A = [self \in ProcSet \mapsto defaultInitValue]$

$\land\ owner\_thread\_id = [self \in ProcSet \mapsto \text{“Invalid\_Thread\_Id”}]$

Procedure *Do_Release_Mutex*
$\land\ thread\_id\_D = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ mutex\_id\_D = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ doing\_condvar\_wait = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ awoken\_thread\_id\_ = [self \in ProcSet \mapsto$ "Invalid_Thread_Id"$]$
Procedure *Release_Mutex*
$\land\ thread\_id\_R = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ mutex\_id\_R = [self \in ProcSet \mapsto defaultInitValue]$
Procedure *Do_Wait_On_Condvar*
$\land\ thread\_id\_Do = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ condvar\_id\_ \ = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ mutex\_id\_Do = [self \in ProcSet \mapsto defaultInitValue]$
Procedure *Wait_On_Condvar*
$\land\ thread\_id\_W = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ condvar\_id\_W = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ mutex\_id = [self \in ProcSet \mapsto defaultInitValue]$
Procedure *Do_Signal_Condvar*
$\land\ condvar\_id\_D = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ do\_context\_switch = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ awoken\_thread\_id = [self \in ProcSet \mapsto$ "Invalid_Thread_Id"$]$
$\land\ to\_reacquire\_mutex\_id = [self \in ProcSet \mapsto$ "Invalid_Mutex_Id"$]$
Procedure *Signal_Condvar*
$\land\ context\_id\_ = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ condvar\_id\_S = [self \in ProcSet \mapsto defaultInitValue]$
Procedure *Broadcast_Condvar*
$\land\ context\_id \ = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ condvar\_id = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ thread\_was\_awaken = [self \in ProcSet \mapsto$ FALSE$]$
Procedure *Delay_Until*
$\land\ thread\_id = [self \in ProcSet \mapsto defaultInitValue]$
Process *Timer_Interrupt*
$\land\ delayed\_threads = \{\}$
$\land\ stack = [self \in ProcSet \mapsto \langle\rangle]$
$\land\ pc = [self \in ProcSet \mapsto$ CASE $self \in Threads \setminus \{$"Idle_Thread"$\} \to$ "thread_state_machine_next_st
$\square$ $self =$ "Idle_Thread" $\to$ "idle_thread_next_state_loop"
$\square$ $self =$ "Timer_Interrupt" $\to$ "timer_interrupt_next_state_loop"
$\square$ $self =$ "Other_Interrupt" $\to$ "other_interrupt_next_state_loop"$]$

$check\_time\_slice\_step(self) \triangleq$ $\land\ pc[self] =$ "check_time_slice_step"
$\land\ Assert(\neg HiRTOS.Interrupts\_Enabled,$
"Failure of assertion at line 218, column 7.")
$\land$ IF $HiRTOS.Current\_Thread\_Id \neq$ "Invalid_Thread_Id"
THEN $\land\ Thread\_Objects' = [Thread\_Objects$ EXCEPT $![HiRTOS.Cu$
$\land$ IF $Thread\_Objects'[HiRTOS.Current\_Thread\_Id].ghost\_Ti$
THEN $\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Runnable\_Th$

```
                                                                !.Current_Thre
                                  ELSE  ∧ HiRTOS' = [HiRTOS EXCEPT !.Runnable_Th

                                                                !.Current_Thre
                          ELSE  ∧ TRUE
                                ∧ UNCHANGED ⟨HiRTOS,
                                             Thread_Objects⟩
                        ∧ pc' = [pc EXCEPT ![self] = "choose_next_thread_step"]
                        ∧ UNCHANGED ⟨Mutex_Objects, Condvar_Objects,
                                     Timer_Objects,
                                     Global_Resource_Available,
                                     stack, thread_id_, mutex_id_,
                                     waking_up_thread_after_condvar_wait,
                                     owner_thread_id_, thread_id_A,
                                     mutex_id_A, owner_thread_id,
                                     thread_id_D, mutex_id_D,
                                     doing_condvar_wait,
                                     awoken_thread_id_, thread_id_R,
                                     mutex_id_R, thread_id_Do,
                                     condvar_id_, mutex_id_Do,
                                     thread_id_W, condvar_id_W,
                                     mutex_id, condvar_id_D,
                                     do_context_switch,
                                     awoken_thread_id,
                                     to_reacquire_mutex_id,
                                     context_id_, condvar_id_S,
                                     context_id, condvar_id,
                                     thread_was_awaken, thread_id,
                                     delayed_threads⟩

choose_next_thread_step(self) ≜  ∧ pc[self] = "choose_next_thread_step"
                                 ∧ HiRTOS' = [HiRTOS EXCEPT !.Current_Thread_Id = Priority_Que
                                                            !.Runnable_Threads_Queue = Priorit
                                 ∧ Assert(HiRTOS'.Current_Thread_Id ≠ "Invalid_Thread_Id",
                                          "Failure of assertion at line 236, column 7.")
                                 ∧ Thread_Objects' = [Thread_Objects EXCEPT ![HiRTOS'.Current_Thr
                                                                           ![HiRTOS'.Current_Thr
                                 ∧ pc' = [pc EXCEPT ![self] = "run_scheduler_return_step"]
                                 ∧ UNCHANGED ⟨Mutex_Objects,
                                              Condvar_Objects,
                                              Timer_Objects,
                                              Global_Resource_Available,
                                              stack, thread_id_, mutex_id_,
                                              waking_up_thread_after_condvar_wait,
                                              owner_thread_id_, thread_id_A,
```

$mutex\_id\_A$, $owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$run\_scheduler\_return\_step(self) \triangleq$ $\land\ pc[self] =$ "run_scheduler_return_step"
$\land\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
$\land\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$, $delayed\_threads\rangle$

$Run\_Thread\_Scheduler(self) \triangleq\ check\_time\_slice\_step(self)$
$\quad \lor\ choose\_next\_thread\_step(self)$
$\quad \lor\ run\_scheduler\_return\_step(self)$

$acquire\_mutex\_step\_(self) \triangleq\ \land\ pc[self] =$ "acquire_mutex_step_"
$\quad \land\ Assert(\neg HiRTOS.Interrupts\_Enabled,$
$\quad\quad$ "Failure of assertion at line 248, column 7.")
$\quad \land$ IF $Mutex\_Objects[mutex\_id\_[self]].Owner\_Thread\_Id =$ "Invalid_Thread_I
$\quad\quad$ THEN $\land\ pc' = [pc$ EXCEPT $![self] =$ "acquire_mutex_acquire_step"$]$
$\quad\quad$ ELSE $\land\ pc' = [pc$ EXCEPT $![self] =$ "acquire_mutex_wait_on_mutex_st
$\quad \land$ UNCHANGED $\langle HiRTOS,\ Thread\_Objects,$
$\quad\quad Mutex\_Objects,\ Condvar\_Objects,$
$\quad\quad Timer\_Objects,$
$\quad\quad Global\_Resource\_Available,\ stack,$
$\quad\quad thread\_id\_,\ mutex\_id\_,$
$\quad\quad waking\_up\_thread\_after\_condvar\_wait,$
$\quad\quad owner\_thread\_id\_,\ thread\_id\_A,$
$\quad\quad mutex\_id\_A,\ owner\_thread\_id,$
$\quad\quad thread\_id\_D,\ mutex\_id\_D,$
$\quad\quad doing\_condvar\_wait,$
$\quad\quad awoken\_thread\_id\_,\ thread\_id\_R,$
$\quad\quad mutex\_id\_R,\ thread\_id\_Do,$
$\quad\quad condvar\_id\_,\ mutex\_id\_Do,$
$\quad\quad thread\_id\_W,\ condvar\_id\_W,$
$\quad\quad mutex\_id,\ condvar\_id\_D,$
$\quad\quad do\_context\_switch,$
$\quad\quad awoken\_thread\_id,$
$\quad\quad to\_reacquire\_mutex\_id,$
$\quad\quad context\_id\_,\ condvar\_id\_S,$
$\quad\quad context\_id,\ condvar\_id,$
$\quad\quad thread\_was\_awaken,\ thread\_id,$
$\quad\quad delayed\_threads\rangle$

$acquire\_mutex\_acquire\_step(self) \triangleq\ \land\ pc[self] =$ "acquire_mutex_acquire_step"
$\quad \land\ \land\ Mutex\_Objects' = [Mutex\_Objects$ EXCEPT $![mutex\_id\_[self]].C$
$\quad\quad \land\ Thread\_Objects' = [Thread\_Objects$ EXCEPT $![thread\_id\_[self]].$
$\quad \land$ IF $waking\_up\_thread\_after\_condvar\_wait[self]$
$\quad\quad$ THEN $\land\ pc' = [pc$ EXCEPT $![self] =$ "acquire_mutex_make_cond
$\quad\quad$ ELSE $\land\ Assert(thread\_id\_[self] = HiRTOS.Current\_Thread\_I$
$\quad\quad\quad\quad$ "Failure of assertion at line 264, column 13.")
$\quad\quad\quad\quad \land\ Assert(Thread\_Objects'[thread\_id\_[self]].State =$ "Ru
$\quad\quad\quad\quad\quad\quad$ "Failure of assertion at line 265, column 13.")
$\quad\quad\quad\quad \land\ pc' = [pc$ EXCEPT $![self] =$ "do_acquire_mutex_return_
$\quad \land$ UNCHANGED $\langle HiRTOS,\ Condvar\_Objects,$
$\quad\quad Timer\_Objects,$

$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$, $delayed\_threads\rangle$

$acquire\_mutex\_make\_condvar\_wait\_awoken\_thread\_runnable\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_m
$\land\ Assert(thread\_id\_[self] \neq H$
"Failure of assertion
$\land\ Assert(Thread\_Objects[thre$
"Failure of assertion
$\land\ Assert(\ \ thread\_id\_[self] \notin$
$Range(HiRTOS.Ru$
"Failure of assertion
$\land\ \land\ HiRTOS' = [HiRTOS$ EX
$\land\ Thread\_Objects' = [Thre$
$\land\ pc' = [pc$ EXCEPT $![self] =$
$\land$ UNCHANGED $\langle Mutex\_Object$
$Condvar\_Obj$
$Timer\_Object$
$Global\_Resou$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_th$
$owner\_thread$
$thread\_id\_A$,
$mutex\_id\_A$,

$owner\_thread$
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condva$
$awoken\_threa$
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$
$condvar\_id\_$,
$mutex\_id\_Do$
$thread\_id\_W$
$condvar\_id\_W$
$mutex\_id$,
$condvar\_id\_D$
$do\_context\_su$
$awoken\_threa$
$to\_reacquire\_$
$context\_id\_$,
$condvar\_id\_S$
$context\_id$,
$condvar\_id$,
$thread\_was\_a$
$thread\_id$,
$delayed\_threa$

$acquire\_mutex\_wait\_on\_mutex\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_wait_on_mutex_step"
$\land\ owner\_thread\_id\_' = [owner\_thread\_id\_$ EXCEPT $![self] =$
$\land\ Assert(owner\_thread\_id\_'[self] \neq thread\_id\_[self],$
"Failure of assertion at line 270, column 10.")
$\land\ Mutex\_Objects' = [Mutex\_Objects$ EXCEPT $![mutex\_id\_[s$
$\land$ IF $waking\_up\_thread\_after\_condvar\_wait[self]$
THEN $\land\ Assert(thread\_id\_[self] \neq HiRTOS.Current\_T$
"Failure of assertion at line 274, column
$\land\ Assert(Thread\_Objects[thread\_id\_[self]].State$
"Failure of assertion at line 275, column
$\land$ UNCHANGED $HiRTOS$
ELSE $\land\ Assert(thread\_id\_[self] = HiRTOS.Current\_T$
"Failure of assertion at line 277, column
$\land\ Assert(Thread\_Objects[thread\_id\_[self]].State$
"Failure of assertion at line 278, column
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Current\_Th$
$\land\ Thread\_Objects' = [Thread\_Objects$ EXCEPT $![thread\_id\_[$
$![thread\_id\_[$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "acquire_mutex_check_if_priorit
$\land$ UNCHANGED $\langle Condvar\_Objects,$

$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$acquire\_mutex\_check\_if\_priority\_inheritance\_needed\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_check_if_p
$\land$ IF $Thread\_Objects[owner\_thread\_id$
$Thread\_Objects[thread\_id\_[self$
THEN $\land\ pc' = [pc$ EXCEPT $![self$
ELSE $\land\ pc' = [pc$ EXCEPT $![self$
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Ava$
$stack$,
$thread\_id\_$,

$mutex\_id\_$,
$waking\_up\_thread\_aft$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_i$
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$acquire\_mutex\_priority\_inheritance\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_priority_inheritance_step"
$\land\ Mutex\_Objects' = [Mutex\_Objects$ EXCEPT $![mutex\_$
$\land$ IF $Thread\_Objects[owner\_thread\_id\_[self]].State =$ "F
THEN $\land\ pc' = [pc$ EXCEPT $![self] =$ "acquire_mut
ELSE $\land$ IF $Thread\_Objects[owner\_thread\_id\_[self$
THEN $\land\ pc' = [pc$ EXCEPT $![self] =$
ELSE $\land\ Assert(Thread\_Objects[own$
"Failure of assertion a
$\land\ pc' = [pc$ EXCEPT $![self] =$
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$,

$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$acquire\_mutex\_priority\_inheritance\_if\_mutex\_owner\_runnable\_step(self) \triangleq$ $\land pc[self] =$ "acquire_mutex
$\land Assert((Thread\_Objects[t$
"Failure of assertio
$\land \land HiRTOS' = [HiRTOS$

$\land Thread\_Objects' = [Th$
$\land pc' = [pc$ EXCEPT $![self] =$
$\land$ UNCHANGED $\langle Mutex\_Obj$
$Condvar\_O$
$Timer\_Obj$
$Global\_Res$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,

$waking\_up\_$
$owner\_thre$
$thread\_id\_A$
$mutex\_id\_A$
$owner\_thre$
$thread\_id\_I$
$mutex\_id\_I$
$doing\_cond$
$awoken\_thr$
$thread\_id\_I$
$mutex\_id\_I$
$thread\_id\_I$
$condvar\_id$
$mutex\_id\_I$
$thread\_id\_W$
$condvar\_id$
$mutex\_id,$
$condvar\_id$
$do\_context\_$
$awoken\_thr$
$to\_reacquire$
$context\_id\_$
$condvar\_id$
$context\_id,$
$condvar\_id$
$thread\_was\_$
$thread\_id,$
$delayed\_thr$

$acquire\_mutex\_priority\_inheritance\_if\_mutex\_owner\_blocked\_on\_mutex\_step(self) \triangleq$ $\wedge pc[self] =$ "acqui
$\wedge Assert((Thread\_$
"Failure o
$\wedge \wedge Mutex\_Object$

$\wedge Thread\_Objec$
$\wedge pc' = [pc$ EXCEP
$\wedge$ UNCHANGED $\langle Ha$
Co
Th
Gl
sta

th
m
wa
ou
th
m
ou
th
m
do
au
th
m
th
co
m
th
co
m
co
do
au
to
co
co
co
co
th
th
de

$acquire\_mutex\_priority\_inheritance\_if\_mutex\_owner\_blocked\_on\_condvar\_step(self) \triangleq$ $\wedge\ pc[self] =$ "acq
$\wedge\ Assert((Thread$
"Failure
$\wedge\ \wedge\ \wedge Mutex\_Obje$

$\wedge\ Thread\_Obj$
$\wedge\ pc' = [pc$ EXCE
$\wedge$ UNCHANGED $\langle$

$$
\begin{aligned}
acquire\_mutex\_priority\_inheritance\_update\_prio\_step(self) \triangleq\ & \land\ pc[self] = \text{“acquire\_mutex\_priority\_inherit} \\
& \land\ Thread\_Objects' = [Thread\_Objects \text{ EXC} \\
& \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“acquire\_mut} \\
& \land \text{ UNCHANGED } \langle HiRTOS, \\
& \qquad\qquad Mutex\_Objects, \\
& \qquad\qquad Condvar\_Objects, \\
& \qquad\qquad Timer\_Objects, \\
& \qquad\qquad Global\_Resource\_Available \\
& \qquad\qquad stack, \\
& \qquad\qquad thread\_id\_, \\
& \qquad\qquad mutex\_id\_, \\
& \qquad\qquad waking\_up\_thread\_after\_c
\end{aligned}
$$

$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$acquire\_mutex\_check\_if\_synchronous\_context\_switch\_needed\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_ch
$\land$ IF $\neg waking\_up\_thread\_after$
THEN $\land\ pc' = [pc$ EXCEI
ELSE $\land\ pc' = [pc$ EXCEI
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objec$
$Mutex\_Object$
$Condvar\_Obj$
$Timer\_Object$
$Global\_Resou$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_th$
$owner\_thread$
$thread\_id\_A$,
$mutex\_id\_A$,

$owner\_thread$
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condva$
$awoken\_threa$
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$
$condvar\_id\_$,
$mutex\_id\_Do$
$thread\_id\_W$,
$condvar\_id\_V$
$mutex\_id$,
$condvar\_id\_D$
$do\_context\_sw$
$awoken\_threa$
$to\_reacquire\_$
$context\_id\_$,
$condvar\_id\_S$
$context\_id$,
$condvar\_id$,
$thread\_was\_a$
$thread\_id$,
$delayed\_threa$

$acquire\_mutex\_synchronous\_context\_switch\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_synchronous_contex
$\land\ stack' = [stack$ EXCEPT $![self] = \langle [procedure$
pc
$\circ\ stack[sel$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "check_time_slice_
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condv$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,

$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$do\_acquire\_mutex\_return\_step(self) \triangleq$ $\land\ pc[self] =$ "do_acquire_mutex_return_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ owner\_thread\_id\_' = [owner\_thread\_id\_$ EXCEPT $![self] = Head$
$\land\ thread\_id\_' = [thread\_id\_$ EXCEPT $![self] = Head(stack[self]).th$
$\land\ mutex\_id\_' = [mutex\_id\_$ EXCEPT $![self] = Head(stack[self]).m$
$\land\ waking\_up\_thread\_after\_condvar\_wait' = [waking\_up\_thread\_af$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS, Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_A, mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D, mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R, mutex\_id\_R,$
$thread\_id\_Do,$
$condvar\_id\_, mutex\_id\_Do,$
$thread\_id\_W,$
$condvar\_id\_W, mutex\_id,$
$condvar\_id\_D,$

$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,$
$condvar\_id\_S,\ context\_id,$
$condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,$
$delayed\_threads\rangle$

$Do\_Acquire\_Mutex(self) \triangleq\ acquire\_mutex\_step\_(self)$
$\lor\ acquire\_mutex\_acquire\_step(self)$
$\lor\ acquire\_mutex\_make\_condvar\_wait\_awoken\_thread\_runnable\_step(self)$
$\lor\ acquire\_mutex\_wait\_on\_mutex\_step(self)$
$\lor\ acquire\_mutex\_check\_if\_priority\_inheritance\_needed\_step(self)$
$\lor\ acquire\_mutex\_priority\_inheritance\_step(self)$
$\lor\ acquire\_mutex\_priority\_inheritance\_if\_mutex\_owner\_runnable\_step(sel$
$\lor\ acquire\_mutex\_priority\_inheritance\_if\_mutex\_owner\_blocked\_on\_mutex$
$\lor\ acquire\_mutex\_priority\_inheritance\_if\_mutex\_owner\_blocked\_on\_condv$
$\lor\ acquire\_mutex\_priority\_inheritance\_update\_prio\_step(self)$
$\lor\ acquire\_mutex\_check\_if\_synchronous\_context\_switch\_needed\_step(self)$
$\lor\ acquire\_mutex\_synchronous\_context\_switch\_step(self)$
$\lor\ do\_acquire\_mutex\_return\_step(self)$

$enter\_critical\_section\_step\_(self) \triangleq\ \land\ pc[self] = \text{“enter\_critical\_section\_step\_”}$
$\land\ HiRTOS.Interrupts\_Enabled \land$
$(thread\_id\_A[self] \in Threads \Rightarrow$
$Thread\_Objects[thread\_id\_A[self]].State = \text{“Running”})$
$\land\ HiRTOS' = [HiRTOS\ \text{EXCEPT}\ !.Interrupts\_Enabled = \text{FALSE}]$
$\land\ Assert(HiRTOS'.Current\_Thread\_Id = thread\_id\_A[self],$
$\text{“Failure of assertion at line 338, column 7.”})$
$\land\ \land\ mutex\_id\_' = [mutex\_id\_\ \text{EXCEPT}\ ![self] = mutex\_id\_A[self]]$
$\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = \langle[procedure \mapsto \text{“Do\_Acquire\_M}$
$pc \mapsto \text{“exit\_critical\_se}$
$owner\_thread\_id\_ \mapsto owner$
$thread\_id\_ \mapsto thread\_id\_[se$
$mutex\_id\_ \mapsto mutex\_id\_[se$
$waking\_up\_thread\_after\_con$
$\circ\ stack[self]]$
$\land\ thread\_id\_' = [thread\_id\_\ \text{EXCEPT}\ ![self] = thread\_id\_A[self]]$
$\land\ waking\_up\_thread\_after\_condvar\_wait' = [waking\_up\_thread\_af$
$\land\ owner\_thread\_id\_' = [owner\_thread\_id\_\ \text{EXCEPT}\ ![self] = \text{“Invalid}$
$\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“acquire\_mutex\_step\_”}]$
$\land\ \text{UNCHANGED}\ \langle Thread\_Objects,$
$Mutex\_Objects,$

$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$exit\_critical\_section\_step\_(self) \triangleq$ $\land\ pc[self] =$ "exit_critical_section_step_"
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ TRUE$]$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "acquire_mutex_return_step"$]$
$\land$ UNCHANGED $\langle Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,

$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$acquire\_mutex\_return\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_return_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ owner\_thread\_id' = [owner\_thread\_id$ EXCEPT $![self] = Head(stac$
$\land\ thread\_id\_A' = [thread\_id\_A$ EXCEPT $![self] = Head(stack[self]).t$
$\land\ mutex\_id\_A' = [mutex\_id\_A$ EXCEPT $![self] = Head(stack[self]).m$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$, $delayed\_threads\rangle$

$Acquire\_Mutex(self) \triangleq$ $enter\_critical\_section\_step\_(self)$
$\lor\ exit\_critical\_section\_step\_(self)$
$\lor\ acquire\_mutex\_return\_step(self)$

$release\_mutex\_step\_(self) \triangleq$ $\land\ pc[self] =$ "release_mutex_step_"
$\land\ Assert(\neg HiRTOS.Interrupts\_Enabled,$

"Failure of assertion at line 350, column 7.")
$\land\ Assert(Mutex\_Objects[mutex\_id\_D[self]].Owner\_Thread\_Id = thread\_id\_$
"Failure of assertion at line 351, column 7.")
$\land\ Assert(Thread\_Objects[thread\_id\_D[self]].Owned\_Mutexes \neq \langle\rangle,$
"Failure of assertion at line 352, column 7.")
$\land\ Assert(Head(Thread\_Objects[thread\_id\_D[self]].Owned\_Mutexes) = mute$
"Failure of assertion at line 353, column 7.")
$\land\ Thread\_Objects' = [Thread\_Objects$ EXCEPT $![thread\_id\_D[self]].Owned\_$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "release_mutex_restore_priority_step"]
$\land$ UNCHANGED $\langle HiRTOS,\ Mutex\_Objects,$
$Condvar\_Objects,\ Timer\_Objects,$
$Global\_Resource\_Available,\ stack,$
$thread\_id\_,\ mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,\ thread\_id\_A,$
$mutex\_id\_A,\ owner\_thread\_id,$
$thread\_id\_D,\ mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,\ thread\_id\_R,$
$mutex\_id\_R,\ thread\_id\_Do,$
$condvar\_id\_,\ mutex\_id\_Do,$
$thread\_id\_W,\ condvar\_id\_W,$
$mutex\_id,\ condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,\ condvar\_id\_S,$
$context\_id,\ condvar\_id,$
$thread\_was\_awaken,\ thread\_id,$
$delayed\_threads\rangle$

$release\_mutex\_restore\_priority\_step(self) \triangleq$ $\land\ pc[self] =$ "release_mutex_restore_priority_step"
$\land$ IF $Thread\_Objects[thread\_id\_D[self]].Owned\_Mutexes \neq \langle$
THEN $\land$ LET $prev\_mutex\_obj \triangleq Mutex\_Objects[Head$
IF $prev\_mutex\_obj.Last\_Inherited\_Priority$
THEN $\land\ Thread\_Objects' = [Thread\_Ob$
ELSE $\land$ TRUE
$\land$ UNCHANGED $Thread\_Objects$
ELSE $\land\ Thread\_Objects' = [Thread\_Objects$ EXCEPT
$\land\ pc' = [pc$ EXCEPT $![self] =$ "release_mutex_check_if_mutex_
$\land$ UNCHANGED $\langle HiRTOS,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$

$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$release\_mutex\_check\_if\_mutex\_waiters\_step(self) \triangleq$ $\land\ pc[self] =$ "release_mutex_check_if_mutex_waiters_s
$\land$ IF $Is\_Thread\_Priority\_Queue\_Empty(Mutex\_Obj$
THEN $\land\ Mutex\_Objects' = [Mutex\_Objects$ EX
$\land\ pc' = [pc$ EXCEPT $![self] =$ "do_releas
ELSE $\land\ pc' = [pc$ EXCEPT $![self] =$ "release_m
$\land$ UNCHANGED $Mutex\_Objects$
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$,
$thread\_id\_$,
$mutex\_id\_$,

$waking\_up\_thread\_after\_condvar\_u$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$release\_mutex\_wakeup\_mutex\_waiter\_step(self) \triangleq$ $\land\ pc[self] =$ "release_mutex_wakeup_mutex_waiter_step
$\land\ awoken\_thread\_id\_' = [awoken\_thread\_id\_$ EXCEPT
$\land\ Assert(Thread\_Objects[awoken\_thread\_id\_'[self]].W$
"Failure of assertion at line 374, column 10.")
$\land\ \land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Runnable\_Thre$
$\land\ Mutex\_Objects' = [Mutex\_Objects$ EXCEPT $![mu$
$![mu$
$\land\ Thread\_Objects' = [Thread\_Objects$ EXCEPT $![a$
$![a$
$![a$
$\land$ IF $\neg doing\_condvar\_wait[self]$
THEN $\land\ pc' = [pc$ EXCEPT $![self] =$ "release_mu
ELSE $\land\ pc' = [pc$ EXCEPT $![self] =$ "do_release_
$\land$ UNCHANGED $\langle Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$

$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wai$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$release\_mutex\_synchronous\_context\_switch\_step(self) \triangleq$ $\land\ pc[self] =$ "release_mutex_synchronous_context
$\land\ stack' = [stack$ EXCEPT $![self] = \langle[procedure \mapsto$
$pc \mapsto$
$\circ\ stack[self$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "check_time_slice_
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condva$

$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$do\_release\_mutex\_return\_step(self) \triangleq$ $\land\ pc[self] =$ "do_release_mutex_return_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ awoken\_thread\_id\_' = [awoken\_thread\_id\_$ EXCEPT $![self] = He$
$\land\ thread\_id\_D' = [thread\_id\_D$ EXCEPT $![self] = Head(stack[self$
$\land\ mutex\_id\_D' = [mutex\_id\_D$ EXCEPT $![self] = Head(stack[self$
$\land\ doing\_condvar\_wait' = [doing\_condvar\_wait$ EXCEPT $![self] =$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS,\ Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_,\ mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,\ mutex\_id\_A,$
$owner\_thread\_id,$

$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$Do\_Release\_Mutex(self) \triangleq release\_mutex\_step\_(self)$
$\lor release\_mutex\_restore\_priority\_step(self)$
$\lor release\_mutex\_check\_if\_mutex\_waiters\_step(self)$
$\lor release\_mutex\_wakeup\_mutex\_waiter\_step(self)$
$\lor release\_mutex\_synchronous\_context\_switch\_step(self)$
$\lor do\_release\_mutex\_return\_step(self)$

$enter\_critical\_section\_step\_R(self) \triangleq \land pc[self] =$ "enter_critical_section_step_R"
$\land HiRTOS.Interrupts\_Enabled \land$
$(thread\_id\_R[self] \in Threads \Rightarrow$
$Thread\_Objects[thread\_id\_R[self]].State =$ "Running")
$\land HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ FALSE$]$
$\land Assert(HiRTOS'.Current\_Thread\_Id = thread\_id\_R[self],$
"Failure of assertion at line 399, column 7.")
$\land \land doing\_condvar\_wait' = [doing\_condvar\_wait$ EXCEPT $![self] =$
$\land mutex\_id\_D' = [mutex\_id\_D$ EXCEPT $![self] = mutex\_id\_R[s$
$\land stack' = [stack$ EXCEPT $![self] = \langle[procedure \mapsto$ "Do_Release_
$pc \mapsto$ "exit_critical_
$awoken\_thread\_id\_ \mapsto awo$
$thread\_id\_D \mapsto thread\_id$
$mutex\_id\_D \mapsto mutex\_id$
$doing\_condvar\_wait \mapsto do$
$\circ stack[self]]$
$\land thread\_id\_D' = [thread\_id\_D$ EXCEPT $![self] = thread\_id\_R[s$
$\land awoken\_thread\_id\_' = [awoken\_thread\_id\_$ EXCEPT $![self] =$ "Inv
$\land pc' = [pc$ EXCEPT $![self] =$ "release_mutex_step_"$]$
$\land$ UNCHANGED $\langle Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$

$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$exit\_critical\_section\_step\_R(self) \triangleq$ $\land\ pc[self] =$ "exit_critical_section_step_R"
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ TRUE$]$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "release_mutex_return_step"$]$
$\land$ UNCHANGED $\langle Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$,

$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$release\_mutex\_return\_step(self) \triangleq$ $\land\ pc[self] =$ "release_mutex_return_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ thread\_id\_R' = [thread\_id\_R$ EXCEPT $![self] = Head(stack[self]).th$
$\land\ mutex\_id\_R' = [mutex\_id\_R$ EXCEPT $![self] = Head(stack[self]).m$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$, $delayed\_threads\rangle$

$Release\_Mutex(self) \triangleq$ $enter\_critical\_section\_step\_R(self)$
$\lor\ exit\_critical\_section\_step\_R(self)$
$\lor\ release\_mutex\_return\_step(self)$

$wait\_on\_condvar\_wait\_step(self) \triangleq$ $\land pc[self] =$ "wait_on_condvar_wait_step"
$\land Assert(\neg HiRTOS.Interrupts\_Enabled,$
"Failure of assertion at line 410, column 7.")
$\land \land Condvar\_Objects' = [Condvar\_Objects$ EXCEPT $![condvar\_id\_[s$
$\land Thread\_Objects' = [Thread\_Objects$ EXCEPT $![thread\_id\_Do[sel$
$![thread\_id\_Do[sel$
$![thread\_id\_Do[sel$
$\land HiRTOS' = [HiRTOS$ EXCEPT $!.Current\_Thread\_Id =$ "Invalid_T
$\land pc' = [pc$ EXCEPT $![self] =$ "wait_on_condvar_release_mutex_step"]
$\land$ UNCHANGED $\langle Mutex\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$stack,\ thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,\ mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,\ mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,\ mutex\_id\_R,$
$thread\_id\_Do,\ condvar\_id\_,$
$mutex\_id\_Do,\ thread\_id\_W,$
$condvar\_id\_W,\ mutex\_id,$
$condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,\ condvar\_id\_S,$
$context\_id,\ condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,\ delayed\_threads\rangle$

$wait\_on\_condvar\_release\_mutex\_step(self) \triangleq$ $\land pc[self] =$ "wait_on_condvar_release_mutex_step"
$\land$ IF $mutex\_id\_Do[self] \neq$ "Invalid_Mutex_Id"
THEN $\land \land doing\_condvar\_wait' = [doing\_condvar\_u$
$\land mutex\_id\_D' = [mutex\_id\_D$ EXCEPT $![s$
$\land stack' = [stack$ EXCEPT $![self] = \langle [proced$
pc
$awoke$
$thread$
$mutex$
$doing\_$
$\circ stack$

$\land\ thread\_id\_D' = [thread\_id\_D \text{ EXCEPT } ![s$
$\land\ awoken\_thread\_id\_' = [awoken\_thread\_id\_ \text{ E}$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“release\_mutex\_s}$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“wait\_on\_condva}$
$\land$ UNCHANGED $\langle stack,$
$thread\_id\_D,$
$mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_\rangle$

$\land$ UNCHANGED $\langle HiRTOS,$
$Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,$
$mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_R,$
$mutex\_id\_R,$
$thread\_id\_Do,$
$condvar\_id\_,$
$mutex\_id\_Do,$
$thread\_id\_W,$
$condvar\_id\_W,$
$mutex\_id,$
$condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,$
$condvar\_id\_S,$
$context\_id,$
$condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,$
$delayed\_threads\rangle$

$wait\_on\_condvar\_synchronous\_context\_switch\_step(self) \triangleq$ $\land\ pc[self] = \text{“wait\_on\_condvar\_synchronous\_c}$
$\land\ stack' = [stack \text{ EXCEPT } ![self] = \langle [procedu$
pc

$\circ\ stack[$
$\wedge\ pc' = [pc \text{ EXCEPT } ![self] = \text{“check\_time\_sli}$
$\wedge\ \text{UNCHANGED } \langle HiRTOS,$
$Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_con$
$owner\_thread\_id\_,$
$thread\_id\_A,$
$mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,$
$mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,$
$mutex\_id\_R,$
$thread\_id\_Do,$
$condvar\_id\_,$
$mutex\_id\_Do,$
$thread\_id\_W,$
$condvar\_id\_W,$
$mutex\_id,$
$condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,$
$condvar\_id\_S,$
$context\_id,$
$condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,$
$delayed\_threads\rangle$

$do\_wait\_on\_condvar\_return\_step(self) \triangleq$
$\wedge\ pc[self] = \text{“do\_wait\_on\_condvar\_return\_step”}$
$\wedge\ pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
$\wedge\ thread\_id\_Do' = [thread\_id\_Do \text{ EXCEPT } ![self] = Head(stac$
$\wedge\ condvar\_id\_' = [condvar\_id\_ \text{ EXCEPT } ![self] = Head(stac$
$\wedge\ mutex\_id\_Do' = [mutex\_id\_Do \text{ EXCEPT } ![self] = Head(stac$
$\wedge\ stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$

$\land$ UNCHANGED $\langle HiRTOS,\ Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_,\ mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,$
$mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,$
$mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,$
$mutex\_id\_R,$
$thread\_id\_W,$
$condvar\_id\_W,\ mutex\_id,$
$condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,$
$condvar\_id\_S,$
$context\_id,\ condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,$
$delayed\_threads\rangle$

$Do\_Wait\_On\_Condvar(self) \triangleq\ wait\_on\_condvar\_wait\_step(self)$
$\lor\ wait\_on\_condvar\_release\_mutex\_step(self)$
$\lor\ wait\_on\_condvar\_synchronous\_context\_switch\_step(self)$
$\lor\ do\_wait\_on\_condvar\_return\_step(self)$

$enter\_critical\_section\_step\_W(self) \triangleq\ \land\ pc[self] =$ "enter_critical_section_step_W"
$\land\ HiRTOS.Interrupts\_Enabled \land$
$(thread\_id\_W[self] \in Threads \Rightarrow$
$Thread\_Objects[thread\_id\_W[self]].State =$ "Running")
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ FALSE]
$\land\ \land\ condvar\_id\_'\ = [condvar\_id\_$ EXCEPT $![self] = condvar$
$\land\ mutex\_id\_Do' = [mutex\_id\_Do$ EXCEPT $![self] = mutex\_$
$\land\ stack' = [stack$ EXCEPT $![self] = \langle[procedure \mapsto$ "Do_Wait_C
$pc \mapsto$ "exit_critical
$thread\_id\_Do \mapsto thread\_$

$condvar\_id\_ \mapsto condvar$
$mutex\_id\_Do \mapsto mutex\_$
$\circ\ stack[self]]$
$\land\ thread\_id\_Do' = [thread\_id\_Do \text{ EXCEPT } ![self] = thread\_id\_$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“wait\_on\_condvar\_wait\_step”}]$
$\land \text{ UNCHANGED } \langle Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_,\ mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,\ mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,\ mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,\ mutex\_id\_R,$
$thread\_id\_W,$
$condvar\_id\_W,\ mutex\_id,$
$condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,$
$condvar\_id\_S,$
$context\_id,\ condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,$
$delayed\_threads\rangle$

$exit\_critical\_section\_step\_W(self) \triangleq \land\ pc[self] = \text{“exit\_critical\_section\_step\_W”}$
$\land\ HiRTOS' = [HiRTOS \text{ EXCEPT } !.Interrupts\_Enabled = \text{TRUE}]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“wait\_on\_condvar\_return\_step”}]$
$\land \text{ UNCHANGED } \langle Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$stack,\ thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,\ mutex\_id\_A,$

$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$wait\_on\_condvar\_return\_step(self) \triangleq$ $\land\ pc[self] =$ "wait_on_condvar_return_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ thread\_id\_W' = [thread\_id\_W$ EXCEPT $![self] = Head(stack[sel$
$\land\ condvar\_id\_W' = [condvar\_id\_W$ EXCEPT $![self] = Head(stack$
$\land\ mutex\_id' = [mutex\_id$ EXCEPT $![self] = Head(stack[self]).mute$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,

$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$Wait\_On\_Condvar(self) \triangleq enter\_critical\_section\_step\_W(self)$
$\lor exit\_critical\_section\_step\_W(self)$
$\lor wait\_on\_condvar\_return\_step(self)$

$signal\_condvar\_step(self) \triangleq \land pc[self] =$ "signal_condvar_step"
$\land Assert(\neg HiRTOS.Interrupts\_Enabled,$
"Failure of assertion at line 446, column 7.")
$\land$ IF $\neg Is\_Thread\_Priority\_Queue\_Empty(Condvar\_Objects[condvar\_id\_D[s$
THEN $\land pc' = [pc$ EXCEPT $![self] =$ "signal_condvar_wakeup_waiter_ste
ELSE $\land pc' = [pc$ EXCEPT $![self] =$ "do_condvar_signal_return_step"]
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$, $Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$, $stack$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$, $thread\_id\_A$,
$mutex\_id\_A$, $owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$, $thread\_id\_R$,
$mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$, $condvar\_id\_W$,
$mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$signal\_condvar\_wakeup\_waiter\_step(self) \triangleq \land pc[self] =$ "signal_condvar_wakeup_waiter_step"
$\land awoken\_thread\_id' = [awoken\_thread\_id$ EXCEPT $![self] =$
$\land Condvar\_Objects' = [Condvar\_Objects$ EXCEPT $![condvar$
$\land Assert(awoken\_thread\_id'[self] \neq HiRTOS.Current\_Thre$
"Failure of assertion at line 454, column 10.")
$\land Assert(Thread\_Objects[awoken\_thread\_id'[self]].Waiting\_$
"Failure of assertion at line 455, column 10.")

$\land\ Assert(Thread\_Objects[awoken\_thread\_id'[self]].Waiting\_$
"Failure of assertion at line 456, column 10.")
$\land\ to\_reacquire\_mutex\_id' = [to\_reacquire\_mutex\_id$ EXCEPT
$\land\ Thread\_Objects' = [Thread\_Objects$ EXCEPT $![awoken\_thr$
$![awoken\_thr$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "signal_condvar_check_if_mutex
$\land$ UNCHANGED $\langle HiRTOS,$
$Mutex\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$stack,\ thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,$
$mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,$
$mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,$
$mutex\_id\_R,$
$thread\_id\_Do,$
$condvar\_id\_,$
$mutex\_id\_Do,$
$thread\_id\_W,$
$condvar\_id\_W,$
$mutex\_id,$
$condvar\_id\_D,$
$do\_context\_switch,$
$context\_id\_,$
$condvar\_id\_S,$
$context\_id,$
$condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,$
$delayed\_threads\rangle$

$signal\_condvar\_check\_if\_mutex\_reacquire\_needed\_step(self) \triangleq$ $\land\ pc[self] =$ "signal_condvar_check_if_mute
$\land$ IF $to\_reacquire\_mutex\_id[self] \neq$ "Inval
THEN $\land\ pc' = [pc$ EXCEPT $![self] =$
ELSE $\land\ pc' = [pc$ EXCEPT $![self] =$
$\land$ UNCHANGED $\langle HiRTOS,$
$Thread\_Objects,$

$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Availab$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$signal\_condvar\_reacquire\_mutex\_step(self) \triangleq \land pc[self] =$ "signal_condvar_reacquire_mutex_step"
$\land \land mutex\_id\_' = [mutex\_id\_$ EXCEPT $![self] = to\_reacqu$
$\land stack' = [stack$ EXCEPT $![self] = \langle [procedure \mapsto$ "Do_
$pc \mapsto$ "sign
$owner\_thread\_id\_ \mapsto$
$thread\_id\_ \mapsto thre$
$mutex\_id\_ \mapsto mut$
$waking\_up\_thread\_$
$\circ stack[self]]$

$\land$ $thread\_id\_' = [thread\_id\_$ EXCEPT $![self] = awoken\_th$
$\land$ $waking\_up\_thread\_after\_condvar\_wait' = [waking\_up\_$
$\land$ $owner\_thread\_id\_' = [owner\_thread\_id\_$ EXCEPT $![self]$
$\land$ $pc' = [pc$ EXCEPT $![self] =$ "acquire_mutex_step_"]
$\land$ UNCHANGED $\langle HiRTOS,$
$Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_A,$
$mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,$
$mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,$
$mutex\_id\_R,$
$thread\_id\_Do,$
$condvar\_id\_,$
$mutex\_id\_Do,$
$thread\_id\_W,$
$condvar\_id\_W,$
$mutex\_id,$
$condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,$
$condvar\_id\_S,$
$context\_id,$
$condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,$
$delayed\_threads\rangle$

$signal\_condvar\_awoken\_thread\_runnable\_step(self) \triangleq$ $\land$ $pc[self] =$ "signal_condvar_awoken_thread_runnab
$\land$ $\land$ $HiRTOS' = [HiRTOS$ EXCEPT $!.Runnable\_T$
$\land$ $Thread\_Objects' = [Thread\_Objects$ EXCEPT
$\land$ $pc' = [pc$ EXCEPT $![self] =$ "signal_condvar_chec
$\land$ UNCHANGED $\langle Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$

$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$signal\_condvar\_check\_if\_sync\_context\_switch\_needed\_step(self) \triangleq$ $\land pc[self] =$ "signal_condvar_check_if_s
$\land$ IF $do\_context\_switch[self]$
THEN $\land pc' = [pc$ EXCEPT $![sel$
ELSE $\land pc' = [pc$ EXCEPT $![sel$
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Ava$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,

$waking\_up\_thread\_af$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_$
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$signal\_condvar\_synchronous\_context\_switch\_step(self) \triangleq$ $\land\ pc[self] =$ "signal_condvar_synchronous_contex
$\land\ stack' = [stack$ EXCEPT $![self] = \langle [procedure$
pc
$\circ\ stack[se$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "check_time_slice
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condv$
$owner\_thread\_id\_$,
$thread\_id\_A$,

$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$do\_condvar\_signal\_return\_step(self) \triangleq$ $\land\ pc[self] =$ "do_condvar_signal_return_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ awoken\_thread\_id' = [awoken\_thread\_id$ EXCEPT $![self] = Hea$
$\land\ to\_reacquire\_mutex\_id' = [to\_reacquire\_mutex\_id$ EXCEPT $![se$
$\land\ condvar\_id\_D' = [condvar\_id\_D$ EXCEPT $![self] = Head(stack$
$\land\ do\_context\_switch' = [do\_context\_switch$ EXCEPT $![self] = Hea$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,

$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$Do\_Signal\_Condvar(self) \triangleq signal\_condvar\_step(self)$
$\lor signal\_condvar\_wakeup\_waiter\_step(self)$
$\lor signal\_condvar\_check\_if\_mutex\_reacquire\_needed\_step(self)$
$\lor signal\_condvar\_reacquire\_mutex\_step(self)$
$\lor signal\_condvar\_awoken\_thread\_runnable\_step(self)$
$\lor signal\_condvar\_check\_if\_sync\_context\_switch\_needed\_step(self)$
$\lor signal\_condvar\_synchronous\_context\_switch\_step(self)$
$\lor do\_condvar\_signal\_return\_step(self)$

$enter\_critical\_section\_step\_S(self) \triangleq \land pc[self] =$ "enter_critical_section_step_S"
$\land HiRTOS.Interrupts\_Enabled \land$
$(context\_id\_[self] \in Threads \Rightarrow$
$Thread\_Objects[context\_id\_[self]].State =$ "Running")
$\land HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ FALSE]
$\land \land condvar\_id\_D' = [condvar\_id\_D$ EXCEPT $![self] = condvar\_id$
$\land do\_context\_switch' = [do\_context\_switch$ EXCEPT $![self] =$ TR
$\land stack' = [stack$ EXCEPT $![self] = \langle [procedure \mapsto$ "Do_Signal_C
$pc \mapsto$ "exit_critical_
$awoken\_thread\_id \mapsto awok$
$to\_reacquire\_mutex\_id \mapsto$
$condvar\_id\_D \mapsto condvar$
$do\_context\_switch \mapsto do\_c$
$\circ stack[self]]$
$\land awoken\_thread\_id' = [awoken\_thread\_id$ EXCEPT $![self] =$ "Inval
$\land to\_reacquire\_mutex\_id' = [to\_reacquire\_mutex\_id$ EXCEPT $![self$
$\land pc' = [pc$ EXCEPT $![self] =$ "signal_condvar_step"]
$\land$ UNCHANGED $\langle Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$

$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$exit\_critical\_section\_step\_S(self) \triangleq$ $\land\ pc[self] =$ "exit_critical_section_step_S"
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ TRUE$]$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "condvar_signaled_step"$]$
$\land$ UNCHANGED $\langle Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,

$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$condvar\_signaled\_step(self) \triangleq$ $\land\ pc[self] =$ "condvar_signaled_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ context\_id\_' = [context\_id\_$ EXCEPT $![self] = Head(stack[self]).context\_$
$\land\ condvar\_id\_S' = [condvar\_id\_S$ EXCEPT $![self] = Head(stack[self]).con$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$, $Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$, $thread\_id\_A$,
$mutex\_id\_A$, $owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$, $thread\_id\_R$,
$mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$, $condvar\_id\_W$,
$mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$Signal\_Condvar(self) \triangleq$ $enter\_critical\_section\_step\_S(self)$
$\lor\ exit\_critical\_section\_step\_S(self)$
$\lor\ condvar\_signaled\_step(self)$

$enter\_critical\_section\_step\_B(self) \triangleq$ $\land\ pc[self] =$ "enter_critical_section_step_B"
$\land\ HiRTOS.Interrupts\_Enabled \land$
$(context\_id[self] \in Threads \Rightarrow$
$Thread\_Objects[context\_id[self]].State =$ "Running")
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ FALSE$]$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "broadcast_condvar_step"$]$

$\land$ UNCHANGED $\langle$ *Thread_Objects*,
Mutex_Objects,
Condvar_Objects,
Timer_Objects,
Global_Resource_Available,
stack, *thread_id_*,
*mutex_id_*,
waking_up_thread_after_condvar_wait,
*owner_thread_id_*,
thread_id_A, *mutex_id_A*,
owner_thread_id,
thread_id_D, *mutex_id_D*,
doing_condvar_wait,
*awoken_thread_id_*,
thread_id_R, *mutex_id_R*,
thread_id_Do,
*condvar_id_*,
mutex_id_Do,
thread_id_W,
condvar_id_W, *mutex_id*,
condvar_id_D,
do_context_switch,
awoken_thread_id,
to_reacquire_mutex_id,
*context_id_*,
condvar_id_S,
context_id, *condvar_id*,
thread_was_awaken,
thread_id,
delayed_threads$\rangle$

broadcast_condvar_step(*self*) $\triangleq$ $\land$ *pc*[*self*] = "broadcast_condvar_step"
$\land$ IF $\neg$*Is_Thread_Priority_Queue_Empty*(*Condvar_Objects*[*condvar_id*[*s*
THEN $\land$ *pc*′ = [*pc* EXCEPT ![*self*] = "broadcast_condvar_wakeup_wa
ELSE $\land$ *pc*′ = [*pc* EXCEPT ![*self*] = "broadcast_condvar_check_if_sy
$\land$ UNCHANGED $\langle$*HiRTOS*, *Thread_Objects*,
Mutex_Objects, *Condvar_Objects*,
Timer_Objects,
Global_Resource_Available,
stack, *thread_id_*, *mutex_id_*,
waking_up_thread_after_condvar_wait,
*owner_thread_id_*, *thread_id_A*,
mutex_id_A, *owner_thread_id*,
thread_id_D, *mutex_id_D*,
doing_condvar_wait,

$awoken\_thread\_id\_$, $thread\_id\_R$,
$mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$, $condvar\_id\_W$,
$mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$broadcast\_condvar\_wakeup\_waiter\_step(self) \triangleq$ $\land\ pc[self] =$ "broadcast_condvar_wakeup_waiter_step"
$\land\ \land\ condvar\_id\_D' = [condvar\_id\_D$ EXCEPT $![self] =$
$\land\ do\_context\_switch' = [do\_context\_switch$ EXCEPT $![$
$\land\ stack' = [stack$ EXCEPT $![self] = \langle[procedure \mapsto$ "D
$pc \mapsto$ "br
$awoken\_thread\_i$
$to\_reacquire\_mut$
$condvar\_id\_D \mapsto$
$do\_context\_switc$
$\circ\ stack[self]]$
$\land\ awoken\_thread\_id' = [awoken\_thread\_id$ EXCEPT $![sel$
$\land\ to\_reacquire\_mutex\_id' = [to\_reacquire\_mutex\_id$ EXC
$\land\ pc' = [pc$ EXCEPT $![self] =$ "signal_condvar_step"]
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,

$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$broadcast\_condvar\_after\_waking\_up\_one\_waiter\_step(self) \triangleq$ $\land\ pc[self] =$ "broadcast_condvar_after_wakin
$\land\ thread\_was\_awaken' = [thread\_was\_awak$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "broadcast_co
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_c$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,

$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_id$,
$delayed\_threads\rangle$

$broadcast\_condvar\_check\_if\_sync\_context\_switch\_needed\_step(self) \triangleq$ $\land\ pc[self] =$ "broadcast_condvar_ch
$\land$ IF $context\_id[self] \in Threads \land$
THEN $\land\ pc' = [pc$ EXCEPT !
ELSE $\land\ pc' = [pc$ EXCEPT !
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread$
$owner\_thread\_id\_$
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_w$
$awoken\_thread\_id$
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$
$awoken\_thread\_id$
$to\_reacquire\_mute$
$context\_id\_$,
$condvar\_id\_S$,

$context\_id$,
$condvar\_id$,
$thread\_was\_awake$
$thread\_id$,
$delayed\_threads\rangle$

$broadcast\_condvar\_synchronous\_context\_switch\_step(self) \triangleq$ $\land\ pc[self] =$ "broadcast_condvar_synchronous
$\land\ stack' = [stack$ EXCEPT $![self] = \langle [proced$
pc
$\circ\ stack$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "check_time_s
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_co$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,

$thread\_id$,
$delayed\_threads\rangle$

$exit\_critical\_section\_step\_B(self) \triangleq$ $\land\ pc[self] =$ "exit_critical_section_step_B"
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ TRUE$]$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "condvar_broadcasted_step"$]$
$\land$ UNCHANGED $\langle Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$condvar\_broadcasted\_step(self) \triangleq$ $\land\ pc[self] =$ "condvar_broadcasted_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ thread\_was\_awaken' = [thread\_was\_awaken$ EXCEPT $![self] = Head($
$\land\ context\_id' = [context\_id$ EXCEPT $![self] = Head(stack[self]).conte$
$\land\ condvar\_id' = [condvar\_id$ EXCEPT $![self] = Head(stack[self]).cond$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,

$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$, $thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$thread\_id$, $delayed\_threads\rangle$

$$
\begin{aligned}
Broadcast\_Condvar(self) \triangleq\ & enter\_critical\_section\_step\_B(self) \\
& \lor broadcast\_condvar\_step(self) \\
& \lor broadcast\_condvar\_wakeup\_waiter\_step(self) \\
& \lor broadcast\_condvar\_after\_waking\_up\_one\_waiter\_step(self) \\
& \lor broadcast\_condvar\_check\_if\_sync\_context\_switch\_needed\_step(self) \\
& \lor broadcast\_condvar\_synchronous\_context\_switch\_step(self) \\
& \lor exit\_critical\_section\_step\_B(self) \\
& \lor condvar\_broadcasted\_step(self)
\end{aligned}
$$

$$
\begin{aligned}
enter\_critical\_section\_step\_D(self) \triangleq\ & \land pc[self] = \text{“enter\_critical\_section\_step\_D”} \\
& \land HiRTOS.Interrupts\_Enabled \land \\
& \quad (thread\_id[self] \in Threads \Rightarrow \\
& \qquad Thread\_Objects[thread\_id[self]].State = \text{“Running”}) \\
& \land HiRTOS' = [HiRTOS \text{ EXCEPT } !.Interrupts\_Enabled = \text{FALSE}] \\
& \land pc' = [pc \text{ EXCEPT } ![self] = \text{“delay\_until\_step”}] \\
& \land \text{UNCHANGED } \langle Thread\_Objects, \\
& \qquad Mutex\_Objects, \\
& \qquad Condvar\_Objects, \\
& \qquad Timer\_Objects, \\
& \qquad Global\_Resource\_Available, \\
& \qquad stack, thread\_id\_, \\
& \qquad mutex\_id\_, \\
& \qquad waking\_up\_thread\_after\_condvar\_wait, \\
& \qquad owner\_thread\_id\_,
\end{aligned}
$$

$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$delay\_until\_step(self) \triangleq \land pc[self] = \text{“delay\_until\_step”}$
$\land Timer\_Objects' = [Timer\_Objects \text{ EXCEPT } ![Thread\_Objects[thread\_id[self]].$
$\land \land condvar\_id\_' = [condvar\_id\_ \text{ EXCEPT } ![self] = Thread\_Objects[thread\_id$
$\land mutex\_id\_Do' = [mutex\_id\_Do \text{ EXCEPT } ![self] = \text{“Invalid\_Mutex\_Id”}]$
$\land stack' = [stack \text{ EXCEPT } ![self] = \langle [procedure \mapsto \text{“Do\_Wait\_On\_Condvar”},$
$pc \mapsto \text{“exit\_critical\_section\_step\_}$
$thread\_id\_Do \mapsto thread\_id\_Do[self],$
$condvar\_id\_ \mapsto condvar\_id\_[self],$
$mutex\_id\_Do \mapsto mutex\_id\_Do[self]]\rangle$
$\circ stack[self]]$
$\land thread\_id\_Do' = [thread\_id\_Do \text{ EXCEPT } ![self] = thread\_id[self]]$
$\land pc' = [pc \text{ EXCEPT } ![self] = \text{“wait\_on\_condvar\_wait\_step”}]$
$\land \text{UNCHANGED } \langle HiRTOS, Thread\_Objects,$
$Mutex\_Objects, Condvar\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_, mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_, thread\_id\_A,$
$mutex\_id\_A, owner\_thread\_id,$
$thread\_id\_D, mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_, thread\_id\_R,$
$mutex\_id\_R, thread\_id\_W,$

$condvar\_id\_W$, $mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$, $awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$, $context\_id\_$,
$condvar\_id\_S$, $context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$exit\_critical\_section\_step\_D(self) \triangleq$ $\land\ pc[self] =$ "exit_critical_section_step_D"
$\land\ HiRTOS' = [HiRTOS$ EXCEPT $!.Interrupts\_Enabled =$ TRUE$]$
$\land\ pc' = [pc$ EXCEPT $![self] =$ "after_delay_until_step"$]$
$\land$ UNCHANGED $\langle Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$, $mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$after\_delay\_until\_step(self) \triangleq$ $\land\ pc[self] =$ "after_delay_until_step"
$\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
$\land\ thread\_id' = [thread\_id$ EXCEPT $![self] = Head(stack[self]).thread\_id]$
$\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,

$Mutex\_Objects$, $Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$, $thread\_id\_A$,
$mutex\_id\_A$, $owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$, $thread\_id\_R$,
$mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$, $condvar\_id\_W$,
$mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$delayed\_threads\rangle$

$Delay\_Until(self) \triangleq enter\_critical\_section\_step\_D(self)$
$\lor delay\_until\_step(self)$
$\lor exit\_critical\_section\_step\_D(self)$
$\lor after\_delay\_until\_step(self)$

$thread\_state\_machine\_next\_state\_loop(self) \triangleq \land pc[self] =$ "thread_state_machine_next_state_loop"
$\land Thread\_Objects[self].State =$ "Running" $\land HiRTOS.Inte$
$\land pc' = [pc \text{ EXCEPT } ![self] =$ "context_switch0"$]$
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,

$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$context\_switch0(self) \triangleq \land pc[self] = \text{“context\_switch0”}$
$\land \lor \land pc' = [pc \text{ EXCEPT } ![self] = \text{“acquire\_mutex\_step”}]$
$\land \text{UNCHANGED } \langle Global\_Resource\_Available, stack, context\_id\_, condvar\_id$
$\lor \land Global\_Resource\_Available' = \text{TRUE}$
$\land \lor \land \land condvar\_id\_S' = [condvar\_id\_S \text{ EXCEPT } ![self] = \text{“condvar1”}]$
$\land context\_id\_' = [context\_id\_ \text{ EXCEPT } ![self] = self]$
$\land stack' = [stack \text{ EXCEPT } ![self] = \langle [procedure \mapsto \text{“Signal\_Condvar”}$
$pc \mapsto \text{“thread\_iteration}$
$context\_id\_ \mapsto context\_id\_[se$
$condvar\_id\_S \mapsto condvar\_id\_$
$\circ stack[self]]$
$\land pc' = [pc \text{ EXCEPT } ![self] = \text{“enter\_critical\_section\_step\_S”}]$
$\land \text{UNCHANGED } \langle context\_id, condvar\_id, thread\_was\_awaken\rangle$
$\lor \land \land condvar\_id' = [condvar\_id \text{ EXCEPT } ![self] = \text{“condvar1”}]$
$\land context\_id' = [context\_id \text{ EXCEPT } ![self] = self]$
$\land stack' = [stack \text{ EXCEPT } ![self] = \langle [procedure \mapsto \text{“Broadcast\_Cond}$
$pc \mapsto \text{“thread\_iteration}$
$thread\_was\_awaken \mapsto thread$
$context\_id \mapsto context\_id[self$
$condvar\_id \mapsto condvar\_id[sel$
$\circ stack[self]]$
$\land thread\_was\_awaken' = [thread\_was\_awaken \text{ EXCEPT } ![self] = \text{FALSE}$

```
                    ∧ pc' = [pc EXCEPT ![self] = "enter_critical_section_step_B"]
                    ∧ UNCHANGED ⟨context_id_, condvar_id_S⟩
                ∧ UNCHANGED thread_id
              ∨ ∧ ∧ stack' = [stack EXCEPT ![self] = ⟨[procedure ↦  "Delay_Until",
                                                        pc        ↦  "thread_iteration_com
                                                        thread_id ↦  thread_id[self]]⟩
                                                        ∘ stack[self]]
                  ∧ thread_id' = [thread_id EXCEPT ![self] = self]
                ∧ pc' = [pc EXCEPT ![self] = "enter_critical_section_step_D"]
                ∧ UNCHANGED ⟨Global_Resource_Available, context_id_, condvar_id_S, co
            ∧ UNCHANGED ⟨HiRTOS, Thread_Objects, Mutex_Objects,
                          Condvar_Objects, Timer_Objects,
                          thread_id_, mutex_id_,
                          waking_up_thread_after_condvar_wait,
                          owner_thread_id_, thread_id_A,
                          mutex_id_A, owner_thread_id,
                          thread_id_D, mutex_id_D,
                          doing_condvar_wait, awoken_thread_id_,
                          thread_id_R, mutex_id_R, thread_id_Do,
                          condvar_id_, mutex_id_Do, thread_id_W,
                          condvar_id_W, mutex_id, condvar_id_D,
                          do_context_switch, awoken_thread_id,
                          to_reacquire_mutex_id,
                          delayed_threads⟩

acquire_mutex_step(self) ≜ ∧ pc[self] = "acquire_mutex_step"
                           ∧ ∧ mutex_id_A' = [mutex_id_A EXCEPT ![self] = "mutex1"]
                             ∧ stack' = [stack EXCEPT ![self] = ⟨[procedure ↦  "Acquire_Mutex",
                                                                  pc        ↦  "context_switch1",
                                                                  owner_thread_id ↦  owner_thread_i
                                                                  thread_id_A ↦  thread_id_A[self],
                                                                  mutex_id_A ↦  mutex_id_A[self]]⟩
                                                                  ∘ stack[self]]
                             ∧ thread_id_A' = [thread_id_A EXCEPT ![self] = self]
                           ∧ owner_thread_id' = [owner_thread_id EXCEPT ![self] = "Invalid_Thread_Ic
                           ∧ pc' = [pc EXCEPT ![self] = "enter_critical_section_step_"]
                           ∧ UNCHANGED ⟨HiRTOS, Thread_Objects,
                                        Mutex_Objects, Condvar_Objects,
                                        Timer_Objects,
                                        Global_Resource_Available,
                                        thread_id_, mutex_id_,
                                        waking_up_thread_after_condvar_wait,
                                        owner_thread_id_, thread_id_D,
                                        mutex_id_D, doing_condvar_wait,
                                        awoken_thread_id_, thread_id_R,
```

$mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$, $condvar\_id\_W$,
$mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$, $context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$, $thread\_was\_awaken$,
$thread\_id$, $delayed\_threads\rangle$

$context\_switch1(self) \triangleq$ $\land\ pc[self] =$ "context_switch1"
$\land\ Thread\_Objects[self].State =$ "Running" $\land\ HiRTOS.Interrupts\_Enabled$
$\land\ Assert((Mutex\_Objects[$"mutex1"$].Owner\_Thread\_Id = self),$
"Failure of assertion at line 554, column 13.")
$\land\ \lor\ \land\ pc' = [pc$ EXCEPT $![self] =$ "waiting_for_resource_step"$]$
$\lor\ \land$ TRUE
$\land\ pc' = [pc$ EXCEPT $![self] =$ "release_mutex_step"$]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$, $Mutex\_Objects$,
$Condvar\_Objects$, $Timer\_Objects$,
$Global\_Resource\_Available$, $stack$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$, $thread\_id\_A$,
$mutex\_id\_A$, $owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$, $awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$, $awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$, $context\_id\_$,
$condvar\_id\_S$, $context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$waiting\_for\_resource\_step(self) \triangleq$ $\land\ pc[self] =$ "waiting_for_resource_step"
$\land$ IF $\neg Global\_Resource\_Available$
THEN $\land\ \land\ condvar\_id\_W' = [condvar\_id\_W$ EXCEPT $![self] =$
$\land\ mutex\_id' = [mutex\_id$ EXCEPT $![self] =$ "mutex1"$]$
$\land\ stack' = [stack$ EXCEPT $![self] = \langle[procedure \mapsto$ "Wa
$pc \mapsto$ "co
$thread\_id\_W \mapsto$
$condvar\_id\_W \mapsto$
$mutex\_id \mapsto mu$

$\circ\ stack[self]]$
$\wedge\ thread\_id\_W' = [thread\_id\_W \text{ EXCEPT } ![self] = self$
$\wedge\ pc' = [pc \text{ EXCEPT } ![self] = \text{“enter\_critical\_section\_step\_V}$
$\wedge \text{ UNCHANGED } Global\_Resource\_Available$
ELSE $\wedge\ Global\_Resource\_Available' = \text{FALSE}$
$\wedge\ pc' = [pc \text{ EXCEPT } ![self] = \text{“release\_mutex\_step”}]$
$\wedge \text{ UNCHANGED } \langle stack,$
$thread\_id\_W,$
$condvar\_id\_W,$
$mutex\_id\rangle$
$\wedge \text{ UNCHANGED } \langle HiRTOS,\ Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,\ thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A,\ mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,\ mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,\ mutex\_id\_R,$
$thread\_id\_Do,\ condvar\_id\_,$
$mutex\_id\_Do,\ condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_,\ condvar\_id\_S,$
$context\_id,\ condvar\_id,$
$thread\_was\_awaken,$
$thread\_id,\ delayed\_threads\rangle$

$context\_switch2(self) \triangleq \wedge\ pc[self] = \text{“context\_switch2”}$
$\wedge\ Thread\_Objects[self].State = \text{“Running”} \wedge HiRTOS.Interrupts\_Enabled$
$\wedge\ Assert(Mutex\_Objects[\text{“mutex1”}].Owner\_Thread\_Id = self,$
“Failure of assertion at line 562, column 21.”)
$\wedge\ pc' = [pc \text{ EXCEPT } ![self] = \text{“waiting\_for\_resource\_step”}]$
$\wedge \text{ UNCHANGED } \langle HiRTOS,\ Thread\_Objects,\ Mutex\_Objects,$
$Condvar\_Objects,\ Timer\_Objects,$
$Global\_Resource\_Available,\ stack,$
$thread\_id\_,\ mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,\ thread\_id\_A,$
$mutex\_id\_A,\ owner\_thread\_id,$

$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$, $awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$, $awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$, $context\_id\_$,
$condvar\_id\_S$, $context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$release\_mutex\_step(self) \triangleq \land pc[self] =$ "release_mutex_step"
$\land \land mutex\_id\_R' = [mutex\_id\_R$ EXCEPT $![self] =$ "mutex1"$]$
$\land stack' = [stack$ EXCEPT $![self] = \langle [procedure \mapsto$ "Release_Mutex",
$pc \mapsto$ "thread_iteration_compl
$thread\_id\_R \mapsto thread\_id\_R[self]$,
$mutex\_id\_R \mapsto mutex\_id\_R[self]]\rangle$
$\circ stack[self]]$
$\land thread\_id\_R' = [thread\_id\_R$ EXCEPT $![self] = self]$
$\land pc' = [pc$ EXCEPT $![self] =$ "enter_critical_section_step_R"$]$
$\land$ UNCHANGED $\langle HiRTOS$, $Thread\_Objects$,
$Mutex\_Objects$, $Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$, $thread\_id\_A$,
$mutex\_id\_A$, $owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$, $condvar\_id\_W$,
$mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$, $context\_id\_$,
$condvar\_id\_S$, $context\_id$,
$condvar\_id$, $thread\_was\_awaken$,
$thread\_id$, $delayed\_threads\rangle$

$thread\_iteration\_completed\_step(self) \triangleq \land pc[self] =$ "thread_iteration_completed_step"
$\land$ TRUE
$\land pc' = [pc$ EXCEPT $![self] =$ "thread_state_machine_next_state_l
$\land$ UNCHANGED $\langle HiRTOS$,

$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$$
\begin{aligned}
Thread\_State\_Machine(self) \triangleq\ & thread\_state\_machine\_next\_state\_loop(self) \\
& \lor context\_switch0(self) \\
& \lor acquire\_mutex\_step(self) \\
& \lor context\_switch1(self) \\
& \lor waiting\_for\_resource\_step(self) \\
& \lor context\_switch2(self) \\
& \lor release\_mutex\_step(self) \\
& \lor thread\_iteration\_completed\_step(self)
\end{aligned}
$$

$idle\_thread\_next\_state\_loop \triangleq$ $\land pc[\text{“Idle\_Thread”}] = \text{“idle\_thread\_next\_state\_loop”}$
$\land Thread\_Objects[\text{“Idle\_Thread”}].State = \text{“Running”} \land HiRTOS.Interrupts$
$\land pc' = [pc \text{ EXCEPT } ![\text{“Idle\_Thread”}] = \text{“idle\_thread\_next\_state\_loop”}]$
$\land \text{UNCHANGED } \langle HiRTOS, Thread\_Objects,$
$Mutex\_Objects, Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$stack, thread\_id\_, mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_, thread\_id\_A,$
$mutex\_id\_A, owner\_thread\_id,$
$thread\_id\_D, mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_, thread\_id\_R,$
$mutex\_id\_R, thread\_id\_Do,$
$condvar\_id\_, mutex\_id\_Do,$
$thread\_id\_W, condvar\_id\_W,$
$mutex\_id, condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_, condvar\_id\_S,$
$context\_id, condvar\_id,$
$thread\_was\_awaken, thread\_id,$
$delayed\_threads\rangle$

$Idle\_Thread \triangleq idle\_thread\_next\_state\_loop$

$timer\_interrupt\_next\_state\_loop \triangleq$ $\land pc[\text{“Timer\_Interrupt”}] = \text{“timer\_interrupt\_next\_state\_loop”}$
$\land pc' = [pc \text{ EXCEPT } ![\text{“Timer\_Interrupt”}] = \text{“enter\_critical\_section\_step}$
$\land \text{UNCHANGED } \langle HiRTOS, Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$stack, thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,$
$thread\_id\_A, mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D, mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R, mutex\_id\_R,$

$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$, $delayed\_threads\rangle$

$enter\_critical\_section\_step\_T \triangleq$ $\land\ pc[\text{“Timer\_Interrupt”}] = \text{“enter\_critical\_section\_step\_T”}$
$\land\ HiRTOS.Interrupts\_Enabled \land$
$(\text{“Timer\_Interrupt”} \in Threads \Rightarrow$
$Thread\_Objects[\text{“Timer\_Interrupt”}].State = \text{“Running”})$
$\land\ HiRTOS' = [HiRTOS \text{ EXCEPT } !.Interrupts\_Enabled = \text{FALSE}]$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Timer\_Interrupt”}] = \text{“track\_time\_slice”}]$
$\land$ UNCHANGED $\langle Thread\_Objects$, $Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$stack$, $thread\_id\_$, $mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_wait$,
$owner\_thread\_id\_$, $thread\_id\_A$,
$mutex\_id\_A$, $owner\_thread\_id$,
$thread\_id\_D$, $mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$, $mutex\_id\_R$,
$thread\_id\_Do$, $condvar\_id\_$,
$mutex\_id\_Do$, $thread\_id\_W$,
$condvar\_id\_W$, $mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$track\_time\_slice \triangleq$ $\land\ pc[\text{“Timer\_Interrupt”}] = \text{“track\_time\_slice”}$
$\land$ IF $HiRTOS.Current\_Thread\_Id \neq \text{“Invalid\_Thread\_Id”}$
THEN $\land\ Assert(\neg Thread\_Objects[HiRTOS.Current\_Thread\_Id].ghost\_Time\_Sl$

```
                                "Failure of assertion at line 606, column 13.")
                        ∧ Thread_Objects' = [Thread_Objects EXCEPT ![HiRTOS.Current_Thread
                ELSE    ∧ TRUE
                        ∧ UNCHANGED Thread_Objects
        ∧ delayed_threads' = {t ∈ Threads \ {"Idle_Thread"} :
                              Timer_Objects[Thread_Objects'[t].Builtin_Timer_Id].State =
        ∧ pc' = [pc EXCEPT !["Timer_Interrupt"] = "wakeup_delay_until_waiters"]
        ∧ UNCHANGED ⟨HiRTOS, Mutex_Objects, Condvar_Objects,
                     Timer_Objects, Global_Resource_Available,
                     stack, thread_id_, mutex_id_,
                     waking_up_thread_after_condvar_wait,
                     owner_thread_id_, thread_id_A, mutex_id_A,
                     owner_thread_id, thread_id_D, mutex_id_D,
                     doing_condvar_wait, awoken_thread_id_,
                     thread_id_R, mutex_id_R, thread_id_Do,
                     condvar_id_, mutex_id_Do, thread_id_W,
                     condvar_id_W, mutex_id, condvar_id_D,
                     do_context_switch, awoken_thread_id,
                     to_reacquire_mutex_id, context_id_,
                     condvar_id_S, context_id, condvar_id,
                     thread_was_awaken, thread_id⟩

wakeup_delay_until_waiters ≜ ∧ pc["Timer_Interrupt"] = "wakeup_delay_until_waiters"
                             ∧ IF delayed_threads ≠ {}
                                  THEN ∧ ∃ t ∈ delayed_threads :
                                          ∧ delayed_threads' = delayed_threads \ {t}
                                          ∧ Timer_Objects' = [Timer_Objects EXCEPT ![Thread_O
                                          ∧ ∧ condvar_id_D' = [condvar_id_D EXCEPT !["Timer_
                                            ∧ do_context_switch' = [do_context_switch EXCEPT ![
                                            ∧ stack' = [stack EXCEPT !["Timer_Interrupt"] = ⟨[pro
                                                                                              pc
                                                                                              awo
                                                                                              to_
                                                                                              con
                                                                                              do_
                                                                                              ∘ st
                                          ∧ awoken_thread_id' = [awoken_thread_id EXCEPT !["Ti
                                          ∧ to_reacquire_mutex_id' = [to_reacquire_mutex_id EXCE
                                          ∧ pc' = [pc EXCEPT !["Timer_Interrupt"] = "signal_condv
                                  ELSE ∧ pc' = [pc EXCEPT !["Timer_Interrupt"] = "timer_interupt_as
                                       ∧ UNCHANGED ⟨Timer_Objects, stack,
                                                    condvar_id_D,
                                                    do_context_switch,
                                                    awoken_thread_id,
                                                    to_reacquire_mutex_id,
```

$delayed\_threads\rangle$

$\land$ UNCHANGED $\langle HiRTOS,\ Thread\_Objects,$
$Mutex\_Objects,\ Condvar\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_,\ mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_,\ thread\_id\_A,$
$mutex\_id\_A,\ owner\_thread\_id,$
$thread\_id\_D,\ mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,\ thread\_id\_R,$
$mutex\_id\_R,\ thread\_id\_Do,$
$condvar\_id\_,\ mutex\_id\_Do,$
$thread\_id\_W,\ condvar\_id\_W,$
$mutex\_id,\ context\_id\_,$
$condvar\_id\_S,\ context\_id,$
$condvar\_id,\ thread\_was\_awaken,$
$thread\_id\rangle$

$timer\_interupt\_asynchronous\_context\_switch\_step \triangleq \land pc[\text{“Timer\_Interrupt”}] = \text{“timer\_interupt\_asynchron}$
$\land stack' = [stack \text{ EXCEPT } ![\text{“Timer\_Interrupt”}] = \langle[$
$\land pc' = [pc \text{ EXCEPT } ![\text{“Timer\_Interrupt”}] = \text{“check\_t}$
$\land$ UNCHANGED $\langle HiRTOS,$
$Thread\_Objects,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Available,$
$thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_$
$owner\_thread\_id\_,$
$thread\_id\_A,$
$mutex\_id\_A,$
$owner\_thread\_id,$
$thread\_id\_D,$
$mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_,$
$thread\_id\_R,$
$mutex\_id\_R,$
$thread\_id\_Do,$
$condvar\_id\_,$

$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$exit\_critical\_section\_step\_T \triangleq \land pc[\text{“Timer\_Interrupt”}] = \text{“exit\_critical\_section\_step\_T”}$
$\land HiRTOS' = [HiRTOS \text{ EXCEPT } !.Interrupts\_Enabled = \text{TRUE}]$
$\land pc' = [pc \text{ EXCEPT } ![\text{“Timer\_Interrupt”}] = \text{“timer\_interrupt\_next\_state\_loc}$
$\land \text{UNCHANGED } \langle Thread\_Objects, Mutex\_Objects,$
$Condvar\_Objects, Timer\_Objects,$
$Global\_Resource\_Available,$
$stack, thread\_id\_, mutex\_id\_,$
$waking\_up\_thread\_after\_condvar\_wait,$
$owner\_thread\_id\_, thread\_id\_A,$
$mutex\_id\_A, owner\_thread\_id,$
$thread\_id\_D, mutex\_id\_D,$
$doing\_condvar\_wait,$
$awoken\_thread\_id\_, thread\_id\_R,$
$mutex\_id\_R, thread\_id\_Do,$
$condvar\_id\_, mutex\_id\_Do,$
$thread\_id\_W, condvar\_id\_W,$
$mutex\_id, condvar\_id\_D,$
$do\_context\_switch,$
$awoken\_thread\_id,$
$to\_reacquire\_mutex\_id,$
$context\_id\_, condvar\_id\_S,$
$context\_id, condvar\_id,$
$thread\_was\_awaken, thread\_id,$
$delayed\_threads\rangle$

$Timer\_Interrupt \triangleq timer\_interrupt\_next\_state\_loop$
$\lor enter\_critical\_section\_step\_T \lor track\_time\_slice$
$\lor wakeup\_delay\_until\_waiters$
$\lor timer\_interupt\_asynchronous\_context\_switch\_step$

$\lor$ *exit_critical_section_step_T*

other_interrupt_next_state_loop $\triangleq$ $\land$ *pc*["Other_Interrupt"] = "other_interrupt_next_state_loop"
$\land$ *pc*′ = [*pc* EXCEPT ![“Other_Interrupt”] = “enter_critical_section_step”
$\land$ UNCHANGED ⟨*HiRTOS*, *Thread_Objects*,
Mutex_Objects,
Condvar_Objects,
Timer_Objects,
Global_Resource_Available,
stack, *thread_id_*,
*mutex_id_*,
waking_up_thread_after_condvar_wait,
*owner_thread_id_*,
thread_id_A, *mutex_id_A*,
owner_thread_id,
thread_id_D, *mutex_id_D*,
doing_condvar_wait,
*awoken_thread_id_*,
thread_id_R, *mutex_id_R*,
thread_id_Do, *condvar_id_*,
mutex_id_Do, *thread_id_W*,
condvar_id_W, *mutex_id*,
condvar_id_D,
do_context_switch,
awoken_thread_id,
to_reacquire_mutex_id,
*context_id_*, *condvar_id_S*,
context_id, *condvar_id*,
thread_was_awaken,
thread_id, *delayed_threads*⟩

enter_critical_section_step $\triangleq$ $\land$ *pc*["Other_Interrupt"] = "enter_critical_section_step"
$\land$ *HiRTOS.Interrupts_Enabled* $\land$
(“Other_Interrupt” $\in$ *Threads* $\Rightarrow$
Thread_Objects[“Other_Interrupt”].*State* = “Running”)
$\land$ *HiRTOS*′ = [*HiRTOS* EXCEPT !.*Interrupts_Enabled* = FALSE]
$\land$ *pc*′ = [*pc* EXCEPT ![“Other_Interrupt”] = “other_interupt_asynchronous_co
$\land$ UNCHANGED ⟨*Thread_Objects*, *Mutex_Objects*,
Condvar_Objects, *Timer_Objects*,
Global_Resource_Available,
stack, *thread_id_*, *mutex_id_*,
waking_up_thread_after_condvar_wait,
*owner_thread_id_*, *thread_id_A*,
mutex_id_A, *owner_thread_id*,
thread_id_D, *mutex_id_D*,

$doing\_condvar\_wait$,
$awoken\_thread\_id\_$, $thread\_id\_R$,
$mutex\_id\_R$, $thread\_id\_Do$,
$condvar\_id\_$, $mutex\_id\_Do$,
$thread\_id\_W$, $condvar\_id\_W$,
$mutex\_id$, $condvar\_id\_D$,
$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$, $condvar\_id\_S$,
$context\_id$, $condvar\_id$,
$thread\_was\_awaken$, $thread\_id$,
$delayed\_threads\rangle$

$other\_interupt\_asynchronous\_context\_switch\_step \triangleq$ $\land\ pc[\text{“Other\_Interrupt”}] = \text{“other\_interupt\_asynchron}$
$\land\ stack' = [stack \text{ EXCEPT } ![\text{“Other\_Interrupt”}] = \langle [$
$\circ$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Other\_Interrupt”}] = \text{“check\_t}$
$\land$ UNCHANGED $\langle HiRTOS$,
$Thread\_Objects$,
$Mutex\_Objects$,
$Condvar\_Objects$,
$Timer\_Objects$,
$Global\_Resource\_Available$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_thread\_after\_condvar\_$
$owner\_thread\_id\_$,
$thread\_id\_A$,
$mutex\_id\_A$,
$owner\_thread\_id$,
$thread\_id\_D$,
$mutex\_id\_D$,
$doing\_condvar\_wait$,
$awoken\_thread\_id\_$,
$thread\_id\_R$,
$mutex\_id\_R$,
$thread\_id\_Do$,
$condvar\_id\_$,
$mutex\_id\_Do$,
$thread\_id\_W$,
$condvar\_id\_W$,
$mutex\_id$,
$condvar\_id\_D$,

$do\_context\_switch$,
$awoken\_thread\_id$,
$to\_reacquire\_mutex\_id$,
$context\_id\_$,
$condvar\_id\_S$,
$context\_id$,
$condvar\_id$,
$thread\_was\_awaken$,
$thread\_id$,
$delayed\_threads\rangle$

$$
\begin{aligned}
exit\_critical\_section\_step \triangleq\ & \land pc[\text{“Other\_Interrupt”}] = \text{“exit\_critical\_section\_step”} \\
& \land HiRTOS' = [HiRTOS \text{ EXCEPT } !.Interrupts\_Enabled = \text{TRUE}] \\
& \land pc' = [pc \text{ EXCEPT } ![\text{“Other\_Interrupt”}] = \text{“other\_interrupt\_next\_state\_loop”}] \\
& \land \text{UNCHANGED } \langle Thread\_Objects,\ Mutex\_Objects, \\
& \qquad Condvar\_Objects,\ Timer\_Objects, \\
& \qquad Global\_Resource\_Available,\ stack, \\
& \qquad thread\_id\_,\ mutex\_id\_, \\
& \qquad waking\_up\_thread\_after\_condvar\_wait, \\
& \qquad owner\_thread\_id\_,\ thread\_id\_A, \\
& \qquad mutex\_id\_A,\ owner\_thread\_id, \\
& \qquad thread\_id\_D,\ mutex\_id\_D, \\
& \qquad doing\_condvar\_wait, \\
& \qquad awoken\_thread\_id\_,\ thread\_id\_R, \\
& \qquad mutex\_id\_R,\ thread\_id\_Do, \\
& \qquad condvar\_id\_,\ mutex\_id\_Do, \\
& \qquad thread\_id\_W,\ condvar\_id\_W, \\
& \qquad mutex\_id,\ condvar\_id\_D, \\
& \qquad do\_context\_switch, \\
& \qquad awoken\_thread\_id, \\
& \qquad to\_reacquire\_mutex\_id, \\
& \qquad context\_id\_,\ condvar\_id\_S, \\
& \qquad context\_id,\ condvar\_id, \\
& \qquad thread\_was\_awaken,\ thread\_id, \\
& \qquad delayed\_threads\rangle
\end{aligned}
$$

$$
\begin{aligned}
Other\_Interrupt \triangleq\ & other\_interrupt\_next\_state\_loop \\
& \lor enter\_critical\_section\_step \\
& \lor other\_interupt\_asynchronous\_context\_switch\_step \\
& \lor exit\_critical\_section\_step
\end{aligned}
$$

$$
\begin{aligned}
Next \triangleq\ & Idle\_Thread \lor Timer\_Interrupt \lor Other\_Interrupt \\
& \lor (\exists\, self \in ProcSet : \ \lor Run\_Thread\_Scheduler(self) \\
& \qquad\qquad\qquad\qquad \lor Do\_Acquire\_Mutex(self) \\
& \qquad\qquad\qquad\qquad \lor Acquire\_Mutex(self) \\
& \qquad\qquad\qquad\qquad \lor Do\_Release\_Mutex(self)
\end{aligned}
$$

$$
\begin{array}{l}
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\ Release\_Mutex(self) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\ Do\_Wait\_On\_Condvar(self) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\ Wait\_On\_Condvar(self) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\ Do\_Signal\_Condvar(self) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\ Signal\_Condvar(self) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\ Broadcast\_Condvar(self) \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \lor\ Delay\_Until(self)) \\
\quad\quad\quad\quad\quad\quad\quad \lor\ (\exists\ self\ \in\ Threads \setminus \{\text{“Idle\_Thread”}\} : Thread\_State\_Machine(self))
\end{array}
$$

$$
\begin{array}{lll}
Spec\ \triangleq & \land\ Init \land \Box[Next]_{vars} & \\
& \land\ \forall\ self\ \in\ Threads \setminus \{\text{“Idle\_Thread”}\} : & \land\ \mathrm{WF}_{vars}(Thread\_State\_Machine(self)) \\
& & \land\ \mathrm{WF}_{vars}(Acquire\_Mutex(self)) \\
& \land\ \mathrm{WF}_{vars}(Idle\_Thread) & \\
& \land\ \land\ \mathrm{WF}_{vars}(Timer\_Interrupt) & \\
& \quad \land\ \mathrm{WF}_{vars}(Do\_Signal\_Condvar(\text{“Timer\_Interrupt”})) & \\
& \quad \land\ \mathrm{WF}_{vars}(Run\_Thread\_Scheduler(\text{“Timer\_Interrupt”})) & \\
& \quad \land\ \mathrm{WF}_{vars}(Do\_Acquire\_Mutex(\text{“Timer\_Interrupt”})) & \\
& \land\ \land\ \mathrm{WF}_{vars}(Other\_Interrupt) & \\
& \quad \land\ \mathrm{WF}_{vars}(Run\_Thread\_Scheduler(\text{“Other\_Interrupt”})) &
\end{array}
$$

END TRANSLATION

Correctness Properties

$$
\begin{array}{l}
TypeInvariant\ \triangleq \\
\quad \land\ HiRTOS \in HiRTOS\_Type \\
\quad \land\ Thread\_Objects \in [Threads \to Thread\_Object\_Type] \\
\quad \land\ Mutex\_Objects \in [Mutexes \to Mutex\_Object\_Type] \\
\quad \land\ Condvar\_Objects \in [Condvars \to Condvar\_Object\_Type] \\
\quad \land\ Timer\_Objects \in [Timers \to Timer\_Object\_Type]
\end{array}
$$

There can be at most only one “running” thread

$$
\begin{array}{l}
SafetyInvariant1\ \triangleq \\
\quad HiRTOS.Interrupts\_Enabled \Rightarrow \\
\qquad \text{IF}\ HiRTOS.Current\_Thread\_Id \neq \text{“Invalid\_Thread\_Id”}\ \text{THEN} \\
\qquad\quad \land\ Cardinality(\{t \in Threads : Thread\_Objects[t].State = \text{“Running”}\}) = 1 \\
\qquad\quad \land\ HiRTOS.Current\_Thread\_Id = \\
\qquad\qquad \text{CHOOSE}\ t \in Threads : Thread\_Objects[t].State = \text{“Running”} \\
\qquad \text{ELSE} \\
\qquad\quad \{t \in Threads : Thread\_Objects[t].State = \text{“Running”}\} = \{\}
\end{array}
$$

Running thread not in any queue

$$
\begin{array}{l}
SafetyInvariant2\ \triangleq \\
\quad (HiRTOS.Interrupts\_Enabled \land
\end{array}
$$

```
    HiRTOS.Current_Thread_Id ≠ "Invalid_Thread_Id") ⇒
      ( ∧ Thread_Objects[HiRTOS.Current_Thread_Id].State = "Running"
        ∧ Thread_Objects[HiRTOS.Current_Thread_Id].Waiting_On_Condvar_Id = "Invalid_Condvar_Id"
        ∧ Thread_Objects[HiRTOS.Current_Thread_Id].Waiting_On_Mutex_Id = "Invalid_Mutex_Id"
        ∧ ¬Is_Thread_In_Priority_Queue(HiRTOS.Runnable_Threads_Queue, HiRTOS.Current_Thread_
        ∧ ∀ m ∈ Mutexes :
            ¬Is_Thread_In_Priority_Queue(Mutex_Objects[m].Waiting_Threads_Queue, HiRTOS.Curren
        ∧ ∀ cv ∈ Condvars :
            ¬Is_Thread_In_Priority_Queue(Condvar_Objects[cv].Waiting_Threads_Queue, HiRTOS.Curr
      )
```

All *Runnable* threads are in the *Runnable* threads queue and no other queue

```
SafetyInvariant3 ≜
    HiRTOS.Interrupts_Enabled ⇒
    ∀ t ∈ Threads : Thread_Objects[t].State = "Runnable" ⇒
        ∧ Thread_Objects[t].Waiting_On_Condvar_Id = "Invalid_Condvar_Id"
        ∧ Thread_Objects[t].Waiting_On_Mutex_Id = "Invalid_Mutex_Id"
        ∧ Is_Thread_In_Priority_Queue(HiRTOS.Runnable_Threads_Queue, t)
        ∧ Is_Thread_In_Priority_Queue_In_Only_One_Queue(HiRTOS.Runnable_Threads_Queue, t)
        ∧ ∀ m ∈ Mutexes :
            ¬Is_Thread_In_Priority_Queue(Mutex_Objects[m].Waiting_Threads_Queue, t)
        ∧ ∀ cv ∈ Condvars :
            ¬Is_Thread_In_Priority_Queue(Condvar_Objects[cv].Waiting_Threads_Queue, t)
```

Each thread blocked on a mutex in only one mutex's wait queue and no other queue

```
SafetyInvariant4 ≜
    HiRTOS.Interrupts_Enabled ⇒
    ∀ t ∈ Threads :
      LET
        thread_obj ≜ Thread_Objects[t]
      IN
        thread_obj.State = "Blocked_On_Mutex" ⇒
            ∧ thread_obj.Waiting_On_Mutex_Id ≠ "Invalid_Mutex_Id"
            ∧ thread_obj.Waiting_On_Condvar_Id = "Invalid_Condvar_Id"
            ∧ (LET
                 mutex_obj ≜ Mutex_Objects[thread_obj.Waiting_On_Mutex_Id]
               IN
                 ∧ Is_Thread_In_Priority_Queue(mutex_obj.Waiting_Threads_Queue, t)
                 ∧ Is_Thread_In_Priority_Queue_In_Only_One_Queue(mutex_obj.Waiting_Threads_Qu
            ∧ ¬Is_Thread_In_Priority_Queue(HiRTOS.Runnable_Threads_Queue, t)
            ∧ ∀ m ∈ Mutexes \ {thread_obj.Waiting_On_Mutex_Id} :
                ¬Is_Thread_In_Priority_Queue(Mutex_Objects[m].Waiting_Threads_Queue, t)
            ∧ ∀ cv ∈ Condvars :
                ¬Is_Thread_In_Priority_Queue(Condvar_Objects[cv].Waiting_Threads_Queue, t)
```

Each thread blocked on a condvar is in only one condvar's wait queue and no other queue

$SafetyInvariant5 \triangleq$
$HiRTOS.Interrupts\_Enabled \Rightarrow$
$\forall\, t \in Threads :$
LET
$thread\_obj \triangleq Thread\_Objects[t]$
IN
$thread\_obj.State = \text{“Blocked\_On\_Condvar”} \Rightarrow$
$\land\ thread\_obj.Waiting\_On\_Condvar\_Id \neq \text{“Invalid\_Condvar\_Id”}$
$\land\ thread\_obj.Waiting\_On\_Mutex\_Id = \text{“Invalid\_Mutex\_Id”}$
$\land$ (LET
$condvar\_obj \triangleq Condvar\_Objects[thread\_obj.Waiting\_On\_Condvar\_Id]$
IN
$\land\ Is\_Thread\_In\_Priority\_Queue(condvar\_obj.Waiting\_Threads\_Queue,\ t)$
$\land\ Is\_Thread\_In\_Priority\_Queue\_In\_Only\_One\_Queue(condvar\_obj.Waiting\_Threads\_Q$
$\land \neg Is\_Thread\_In\_Priority\_Queue(HiRTOS.Runnable\_Threads\_Queue,\ t)$
$\land \forall\, cv \in Condvars \setminus \{thread\_obj.Waiting\_On\_Condvar\_Id\} :$
$\neg Is\_Thread\_In\_Priority\_Queue(Condvar\_Objects[cv].Waiting\_Threads\_Queue,\ t)$
$\land \forall\, m \in Mutexes :$
$\neg Is\_Thread\_In\_Priority\_Queue(Mutex\_Objects[m].Waiting\_Threads\_Queue,\ t)$

Each mutex that is currently owned by a thread must be in the list of mutexes owned by that thread
$SafetyInvariant6 \triangleq$
$HiRTOS.Interrupts\_Enabled \Rightarrow$
$\forall\, m \in Mutexes :$
LET
$t \triangleq Mutex\_Objects[m].Owner\_Thread\_Id$
IN
$t \neq \text{“Invalid\_Thread\_Id”} \Rightarrow$
$m \in Range(Thread\_Objects[t].Owned\_Mutexes)$

If a mutex is not owned by a thread, its wait queue should be empty
$SafetyInvariant7 \triangleq$
$HiRTOS.Interrupts\_Enabled \Rightarrow$
$\forall\, m \in Mutexes :$
$Mutex\_Objects[m].Owner\_Thread\_Id = \text{“Invalid\_Thread\_Id”} \Rightarrow$
$Is\_Thread\_Priority\_Queue\_Empty(Mutex\_Objects[m].Waiting\_Threads\_Queue)$

The thread owning a mutex can never have lower priority than any thread waiting for the mutex
$SafetyInvariant8 \triangleq$
$(HiRTOS.Interrupts\_Enabled \land$
$HiRTOS.Current\_Thread\_Id \neq \text{“Invalid\_Thread\_Id”}) \Rightarrow$
$\forall\, m \in Mutexes :$
LET
$t \triangleq Mutex\_Objects[m].Owner\_Thread\_Id$
$prio\_queue \triangleq Mutex\_Objects[m].Waiting\_Threads\_Queue$

IN

$(t \neq \text{“Invalid\_Thread\_Id”} \land \neg Is\_Thread\_Priority\_Queue\_Empty(prio\_queue)) \Rightarrow$
$\forall\, wt \in \text{UNION}\ \{Range(q) : q \in Range(prio\_queue)\} :$
$Thread\_Objects[wt].State = \text{“Blocked\_On\_Mutex”} \land$
$Thread\_Objects[t].Current\_Priority \geq Thread\_Objects[wt].Current\_Priority$

A thread not owning any mutex and not waiting on a condvar always has its current priority set to its base priority

$SafetyInvariant9 \triangleq$
$HiRTOS.Interrupts\_Enabled \Rightarrow$
$\forall\, t \in Threads :$
$(Thread\_Objects[t].Owned\_Mutexes = \langle\rangle \land Thread\_Objects[t].State \neq \text{“Blocked\_On\_Condvar”}) \Rightarrow$
$Thread\_Objects[t].Current\_Priority = Thread\_Objects[t].Base\_Priority$

$LivenessProperty1 \triangleq$
$\neg HiRTOS.Interrupts\_Enabled \Rightarrow \Diamond HiRTOS.Interrupts\_Enabled$

$LivenessProperty2 \triangleq$
$\forall\, t \in Threads \setminus \{\text{“Idle\_Thread”}\},\ p \in Valid\_Thread\_Priority\_Type \setminus \{0\} :$
$(Thread\_Objects[t].Current\_Priority = p \land Thread\_Objects[t].State = \text{“Runnable”}) \Rightarrow$
$\Diamond(Thread\_Objects[t].State = \text{“Running”})$

$LivenessProperty3 \triangleq$
$\forall\, t \in Threads \setminus \{\text{“Idle\_Thread”}\} :$
$\Diamond(pc[t] = \text{“thread\_iteration\_completed\_step”})$

$LivenessProperty4 \triangleq$
$\forall\, t \in Threads :$
$Thread\_Objects[t].State = \text{“Waiting\_On\_Mutex”} \Rightarrow \Diamond(Thread\_Objects[t].State = \text{“Runnable”})$

$LivenessProperty5 \triangleq$
$\forall\, t \in Threads :$
$Thread\_Objects[t].State = \text{“Waiting\_On\_Condvar”} \Rightarrow \Diamond(Thread\_Objects[t].State = \text{“Runnable”})$

THEOREM $Spec \Rightarrow \Box TypeInvariant$
THEOREM $Spec \Rightarrow \Box SafetyInvariant1$
THEOREM $Spec \Rightarrow \Box SafetyInvariant2$
THEOREM $Spec \Rightarrow \Box SafetyInvariant3$
THEOREM $Spec \Rightarrow \Box SafetyInvariant4$
THEOREM $Spec \Rightarrow \Box SafetyInvariant5$
THEOREM $Spec \Rightarrow \Box SafetyInvariant6$
THEOREM $Spec \Rightarrow \Box SafetyInvariant7$
THEOREM $Spec \Rightarrow \Box SafetyInvariant8$
THEOREM $Spec \Rightarrow \Box SafetyInvariant9$
THEOREM $Spec \Rightarrow \Box LivenessProperty1$
THEOREM $Spec \Rightarrow \Box LivenessProperty2$
THEOREM $Spec \Rightarrow \Box LivenessProperty3$

THEOREM $Spec \Rightarrow \Box LivenessProperty4$
THEOREM $Spec \Rightarrow \Box LivenessProperty5$