```
──────────────────── 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 ≜ 3
Num_Interrupt_Priorities ≜ 2

Valid_Thread_Priority_Type ≜ 0 .. Num_Thread_Priorities − 1
Thread_Priority_Type ≜ Valid_Thread_Priority_Type ∪ {Invalid_Thread_Priority}

Threads ≜ {"Idle_Thread", "thread1", "thread2", "thread3"}
Mutexes ≜ {"mutex1"}
Condvars ≜ {"thread1_condvar", "thread2_condvar", "thread3_condvar", "condvar1"}
Timers ≜ {"thread1_timer", "thread2_timer", "thread3_timer"}
Interrupts ≜ {"Timer_Interrupt"}

Thread_Id_Type ≜ Threads ∪ {"Invalid_Thread_Id"}
Mutex_Id_Type ≜ Mutexes ∪ {"Invalid_Mutex_Id"}
Condvar_Id_Type ≜ Condvars ∪ {"Invalid_Condvar_Id"}
Timer_Id_Type ≜ Timers ∪ {"Invalid_Timer_Id"}
Interrupt_Id_Type  ≜ Interrupts ∪ {"Invalid_Interrupt_Id"}

Thread_State_Type ≜ {"Suspended", "Runnable", "Running",
                     "Blocked_On_Condvar", "Blocked_On_Mutex"}

Timer_State_Type ≜ {"Timer_Stopped", "Timer_Running"}

No_Duplicates_List_Type(S) ≜
    {x ∈ Seq(S)    : Len(x) ≠ 0 ⇒
                     ∀ i, j ∈ 1 .. Len(x) : i ≠ j ⇒ x[i] ≠ x[j]}

Range(f) ≜ {f[x] : x ∈ DOMAIN f}

Thread_Queue_Type ≜ No_Duplicates_List_Type(Thread_Id_Type)

Thread_Priority_Queue_Type ≜ [Valid_Thread_Priority_Type → Thread_Queue_Type]

HiRTOS_Type ≜ [
   Current_Thread_Id : Thread_Id_Type,
   Runnable_Threads_Queue : Thread_Priority_Queue_Type,
   Interrupts_Enabled : BOOLEAN
]
```

```
$HiRTOS\_Initializer \triangleq [$
    $Current\_Thread\_Id \mapsto$ "Invalid_Thread_Id",
    $Runnable\_Threads\_Queue \mapsto$
        $[p \in Valid\_Thread\_Priority\_Type \mapsto$
          $\text{CASE } p = 0 \rightarrow \langle$ "Idle_Thread" $\rangle$
          $\Box p = 1 \rightarrow \langle$ "thread1" $\rangle$
          $\Box p = 2 \rightarrow \langle$ "thread2", "thread3" $\rangle$
        $],$
    $Interrupts\_Enabled \mapsto \text{TRUE}$
$]$

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

$Thread\_Object\_Initializer(priority, timer\_id, condvar\_id) \triangleq [$
    $State \mapsto$ "Runnable",
    $Current\_Priority \mapsto priority,$
    $Base\_Priority \mapsto priority,$
    $Builtin\_Timer\_Id \mapsto timer\_id,$
    $Builtin\_Condvar\_Id \mapsto condvar\_id,$
    $Waiting\_On\_Condvar\_Id \mapsto$ "Invalid_Condvar_Id",
    $Waiting\_On\_Mutex\_Id \mapsto$ "Invalid_Mutex_Id",
    $Owned\_Mutexes \mapsto \langle\rangle,$
    $ghost\_Time\_Slice\_Consumed \mapsto \text{FALSE},$
    $ghost\_Condvar\_Wait\_Mutex\_Id \mapsto$ "Invalid_Mutex_Id"
$]$

$Mutex\_Object\_Type \triangleq [$
    $Owner\_Thread\_Id : Thread\_Id\_Type,$
    $Last\_Inherited\_Priority : Thread\_Priority\_Type,$
    $Waiting\_Threads\_Queue : Thread\_Priority\_Queue\_Type$
$]$

$Mutex\_Object\_Initializer \triangleq [$
    $Owner\_Thread\_Id \mapsto$ "Invalid_Thread_Id",
    $Last\_Inherited\_Priority \mapsto Invalid\_Thread\_Priority,$
    $Waiting\_Threads\_Queue \mapsto$
        $[p \in Valid\_Thread\_Priority\_Type \mapsto \langle\rangle]$
$]$

$Condvar\_Object\_Type \triangleq [$
    $Wakeup\_Mutex\_Id : Mutex\_Id\_Type,$
    $Waiting\_Threads\_Queue : Thread\_Priority\_Queue\_Type$
$]$

$Condvar\_Object\_Initializer \triangleq [$
    $Wakeup\_Mutex\_Id \mapsto \text{“Invalid\_Mutex\_Id”},$
    $Waiting\_Threads\_Queue \mapsto$
        $[p \in Valid\_Thread\_Priority\_Type \mapsto \langle\rangle]$
$]$

$Timer\_Object\_Type \triangleq [$
    $State : Timer\_State\_Type$
$]$

$Timer\_Object\_Initializer \triangleq [$
    $State \mapsto \text{“Timer\_Stopped”}$
$]$

$Is\_Thread\_Priority\_Queue\_Empty(prio\_queue) \triangleq$
   $\forall p \in Valid\_Thread\_Priority\_Type : prio\_queue[p] = \langle\rangle$

$Is\_Thread\_In\_Priority\_Queue(prio\_queue, thread\_id) \triangleq$
    $\exists p \in Valid\_Thread\_Priority\_Type : thread\_id \in Range(prio\_queue[p])$

$Is\_Thread\_In\_Priority\_Queue\_In\_Only\_One\_Queue(prio\_queue, thread\_id) \triangleq$
    $Is\_Thread\_In\_Priority\_Queue(prio\_queue, thread\_id) \Rightarrow$
    $Cardinality(\{p \in Valid\_Thread\_Priority\_Type : thread\_id \in Range(prio\_queue[p])\}) = 1$

$Get\_Highest\_Priority(prio\_queue) \triangleq$
    LET
        $Non\_Empty\_Queues \triangleq \{p \in Valid\_Thread\_Priority\_Type : prio\_queue[p] \neq \langle\rangle\}$
    IN
        CHOOSE $p1 \in Non\_Empty\_Queues :$
            $\forall p2 \in Non\_Empty\_Queues \setminus \{p1\} : p2 < p1$

$Get\_Highest\_Priority\_Queue(prio\_queue) \triangleq$
      $prio\_queue[Get\_Highest\_Priority(prio\_queue)]$

 ****************************************************************************
--algorithm $hirtos\_threads\_model$
   variables
      $HiRTOS = HiRTOS\_Initializer,$
      $Thread\_Objects = [$
    $Idle\_Thread \mapsto$
        $Thread\_Object\_Initializer(0,$ “Invalid_Timer_Id”, “Invalid_Condvar_Id”$),$
    $thread1 \mapsto$
        $Thread\_Object\_Initializer(1,$ “thread1_timer”, “thread1_condvar”$),$
    $thread2 \mapsto$
        $Thread\_Object\_Initializer(2,$ “thread2_timer”, “thread2_condvar”$),$
    $thread3 \mapsto$
        $Thread\_Object\_Initializer(2,$ “thread3_timer”, “thread3_condvar”$)$
  $],$
  $Mutex\_Objects = [m \in Mutexes \mapsto Mutex\_Object\_Initializer],$
  $Condvar\_Objects = [cv \in Condvars \mapsto Condvar\_Object\_Initializer],$
  $Timer\_Objects = [tm \in Timers \mapsto Timer\_Object\_Initializer],$
  $Global\_Resource\_Available = \text{FALSE}$ ;

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

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

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

  $Priority\_Queue\_Tail(priority\_queue) \triangleq [$
      $priority\_queue$ 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 $new\_prio \neq old\_prio$ ;
    Remove thread from its current queue:
  $priority\_queue[old\_prio] := SelectSeq(priority\_queue[old\_prio], \text{LAMBDA } x : x \neq thread\_id)$ ||
    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 $HiRTOS.Interrupts\_Enabled \land$
        $(context\_id \in Threads \Rightarrow$
            $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 $\neg HiRTOS.Interrupts\_Enabled$ ;
   if $HiRTOS.Current\_Thread\_Id \neq$ "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 $HiRTOS.Current\_Thread\_Id \neq$ "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 $\neg HiRTOS.Interrupts\_Enabled$ ;
  if $Mutex\_Objects[mutex\_id].Owner\_Thread\_Id$ = "Invalid_Thread_Id" then
    $acquire\_mutex\_acquire\_step$:
    $Mutex\_Objects[mutex\_id].Owner\_Thread\_Id := thread\_id$ ||
    $Thread\_Objects[thread\_id].Owned\_Mutexes :=$
        $\langle mutex\_id \rangle \circ Thread\_Objects[thread\_id].Owned\_Mutexes$ ;
    if $waking\_up\_thread\_after\_condvar\_wait$ then
      $acquire\_mutex\_make\_condvar\_wait\_awoken\_thread\_runnable\_step$:
      assert $thread\_id \neq HiRTOS.Current\_Thread\_Id$ ;
      assert $Thread\_Objects[thread\_id].State$ = "Blocked_On_Condvar" ;
      assert $thread\_id \notin$
          $Range(HiRTOS.Runnable\_Threads\_Queue[Thread\_Objects[thread\_id].Current\_Priority])$ ;
      $HiRTOS.Runnable\_Threads\_Queue :=$
        $Enqueue\_Thread(HiRTOS.Runnable\_Threads\_Queue,\ thread\_id)$ ||
      $Thread\_Objects[thread\_id].State$ := "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 $owner\_thread\_id \neq thread\_id$ ;
    $Mutex\_Objects[mutex\_id].Waiting\_Threads\_Queue :=$
      $Enqueue\_Thread(Mutex\_Objects[mutex\_id].Waiting\_Threads\_Queue,\ thread\_id)$ ;
    if $waking\_up\_thread\_after\_condvar\_wait$ then
      assert $thread\_id \neq HiRTOS.Current\_Thread\_Id$ ;
      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 $\neg waking\_up\_thread\_after\_condvar\_wait$ 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 $Do\_Acquire\_Mutex(thread\_id, mutex\_id, \text{FALSE})$ ;
    $exit\_critical\_section\_step$:
    $Exit\_Critical\_Section()$ ;
    $acquire\_mutex\_return\_step$:
    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 $\neg HiRTOS.Interrupts\_Enabled$ ;
    assert $Mutex\_Objects[mutex\_id].Owner\_Thread\_Id = thread\_id$ ;
    assert $Thread\_Objects[thread\_id].Owned\_Mutexes \neq \langle\rangle$ ;
    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 $Thread\_Objects[thread\_id].Owned\_Mutexes \neq \langle\rangle \land \neg doing\_condvar\_wait$ then
        with $prev\_mutex\_obj = Mutex\_Objects[Head(Thread\_Objects[thread\_id].Owned\_Mutexes)]$ do
            if $prev\_mutex\_obj.Last\_Inherited\_Priority \neq Invalid\_Thread\_Priority$ 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" ||
        $Thread\_Objects[awoken\_thread\_id].Owned\_Mutexes$ :=
            $\langle mutex\_id \rangle \circ Thread\_Objects[awoken\_thread\_id].Owned\_Mutexes$ ;

        if $\neg doing\_condvar\_wait$ 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 $Do\_Release\_Mutex(thread\_id, mutex\_id, \text{FALSE})$ ;
    $exit\_critical\_section\_step$:
    $Exit\_Critical\_Section()$ ;
    $release\_mutex\_return\_step$:
    return ;
  end procedure ;

  procedure $Do\_Wait\_On\_Condvar(thread\_id, condvar\_id, mutex\_id)$
  begin
    $wait\_on\_condvar\_wait\_step$:
    assert $\neg HiRTOS.Interrupts\_Enabled$ ;
    $Thread\_Objects[thread\_id].ghost\_Condvar\_Wait\_Mutex\_Id$ := $mutex\_id$ ||
    $Condvar\_Objects[condvar\_id].Waiting\_Threads\_Queue$ :=
        $Enqueue\_Thread(Condvar\_Objects[condvar\_id].Waiting\_Threads\_Queue, thread\_id)$ ||
    $Thread\_Objects[thread\_id].State$ := "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 $mutex\_id \neq$ "Invalid_Mutex_Id" then
        call $Do\_Release\_Mutex(thread\_id, mutex\_id, \text{TRUE})$ ;
    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** $\neg HiRTOS.Interrupts\_Enabled$ ;
  **if** $\neg Is\_Thread\_Priority\_Queue\_Empty(Condvar\_Objects[condvar\_id].Waiting\_Threads\_Queue)$
   **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** $awoken\_thread\_id \neq HiRTOS.Current\_Thread\_Id$ ;
    **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** $to\_reacquire\_mutex\_id \neq$ "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 $Do\_Signal\_Condvar(condvar\_id, \text{TRUE})$ ;
    $exit\_critical\_section\_step$:
    $Exit\_Critical\_Section()$ ;
    $condvar\_signaled\_step$:
    return ;
end procedure ;

procedure $Broadcast\_Condvar(context\_id, condvar\_id)$
    variable $thread\_was\_awaken = \text{FALSE}$ ;
begin
    $enter\_critical\_section\_step$:
    $Enter\_Critical\_Section(context\_id)$ ;

    $broadcast\_condvar\_step$:
    while $\neg Is\_Thread\_Priority\_Queue\_Empty(Condvar\_Objects[condvar\_id].Waiting\_Threads\_Queue)$
     do
        $broadcast\_condvar\_wakeup\_waiter\_step$:
        call $Do\_Signal\_Condvar(condvar\_id, \text{FALSE})$ ;
       $broadcast\_condvar\_after\_waking\_up\_one\_waiter\_step$:
       $thread\_was\_awaken := \text{TRUE}$ ;
    end while ;

    $broadcast\_condvar\_check\_if\_sync\_context\_switch\_needed\_step$:
    if $context\_id \in Threads \wedge thread\_was\_awaken$ 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 $Thread\_State\_Machine \in Threads \setminus \{$"Idle_Thread"$\}$
begin
    $thread\_state\_machine\_next\_state\_loop$:
    while TRUE do
        await $Thread\_Objects[self].State =$ "Running" $\land\ HiRTOS.Interrupts\_Enabled$ ;
        $context\_switch0$:
        either
            $acquire\_mutex\_step$:
            call $Acquire\_Mutex(self,$ "mutex1"$)$ ;
            $context\_switch1$:
            await $Thread\_Objects[self].State =$ "Running" $\land\ HiRTOS.Interrupts\_Enabled$ ;
            assert $(Mutex\_Objects[$"mutex1"$].Owner\_Thread\_Id = self)$ ;

            either
                $waiting\_for\_resource\_step$:
                while $\neg Global\_Resource\_Available$ do
                    call $Wait\_On\_Condvar(self,$ "condvar1", "mutex1"$)$ ;
                    $context\_switch2$:
                    await $Thread\_Objects[self].State =$ "Running" $\land\ HiRTOS.Interrupts\_Enabled$ ;
                    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 $\neq \{\}$ do
            with $t \in$ delayed_threads do
                delayed_threads := delayed_threads $\setminus \{t\}$ ;
                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*) = "*ec*6916*c*" $\land$ *chksum*(*tla*) = "5*dd*4*fa*09")

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

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

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

$$
\begin{array}{l}
\mathit{Priority\_Queue\_Head}(\mathit{priority\_queue}) \triangleq \\
\quad \mathit{Head}(\mathit{Get\_Highest\_Priority\_Queue}(\mathit{priority\_queue}))
\end{array}
$$

$$
\begin{array}{l}
\mathit{Priority\_Queue\_Tail}(\mathit{priority\_queue}) \triangleq [\\
\qquad \mathit{priority\_queue}\ \text{EXCEPT}\ ![\mathit{Get\_Highest\_Priority}(\mathit{priority\_queue})] = \mathit{Tail}(@) \\
]
\end{array}
$$

```
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 ≜ ⟨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⟩

ProcSet ≜ (Threads \ {"Idle_Thread"}) ∪ {"Idle_Thread"} ∪ {"Timer_Interrupt"} ∪ {"Other_Interrupt"}

Init ≜ Global 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
         Procedure Do_Acquire_Mutex
        ∧ thread_id_ = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ mutex_id_ = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ waking_up_thread_after_condvar_wait = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ owner_thread_id_ = [self ∈ ProcSet ↦ "Invalid_Thread_Id"]
         Procedure Acquire_Mutex
        ∧ thread_id_A = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ mutex_id_A = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ owner_thread_id = [self ∈ ProcSet ↦ "Invalid_Thread_Id"]
```

Procedure *Do_Release_Mutex*

$\land\ thread\_id\_D = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ mutex\_id\_D = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ doing\_condvar\_wait = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ awoken\_thread\_id\_ = [self \in ProcSet \mapsto \text{“Invalid\_Thread\_Id”}]$

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*

$\land\ condvar\_id\_D = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ do\_context\_switch = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ awoken\_thread\_id = [self \in ProcSet \mapsto \text{“Invalid\_Thread\_Id”}]$
$\land\ to\_reacquire\_mutex\_id = [self \in ProcSet \mapsto \text{“Invalid\_Mutex\_Id”}]$

Procedure *Signal_Condvar*

$\land\ context\_id\_ = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ condvar\_id\_S = [self \in ProcSet \mapsto defaultInitValue]$

Procedure *Broadcast_Condvar*

$\land\ context\_id \ = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ condvar\_id = [self \in ProcSet \mapsto defaultInitValue]$
$\land\ thread\_was\_awaken = [self \in ProcSet \mapsto \text{FALSE}]$

Procedure *Delay_Until*

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

Process *Timer_Interrupt*

$\land\ delayed\_threads = \{\}$
$\land\ stack = [self \in ProcSet \mapsto \langle\rangle]$
$\land\ pc = [self \in ProcSet \mapsto \text{CASE}\ self \in Threads \setminus \{\text{“Idle\_Thread”}\} \rightarrow \text{“thread\_state\_machine\_next\_st}$
$\qquad\qquad\qquad\qquad\qquad \Box\ self = \text{“Idle\_Thread”} \rightarrow \text{“idle\_thread\_next\_state\_loop”}$
$\qquad\qquad\qquad\qquad\qquad \Box\ self = \text{“Timer\_Interrupt”} \rightarrow \text{“timer\_interrupt\_next\_state\_loop”}$
$\qquad\qquad\qquad\qquad\qquad \Box\ self = \text{“Other\_Interrupt”} \rightarrow \text{“other\_interrupt\_next\_state\_loop”}]$

$check\_time\_slice\_step(self) \triangleq\ \land\ pc[self] = \text{“check\_time\_slice\_step”}$
$\qquad\qquad\qquad\qquad\qquad\quad \land\ Assert(\neg HiRTOS.Interrupts\_Enabled,$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{“Failure of assertion at line 218, column 7.”})$
$\qquad\qquad\qquad\qquad\qquad\quad \land\ \text{IF}\ HiRTOS.Current\_Thread\_Id \neq \text{“Invalid\_Thread\_Id”}$
$\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{THEN}\ \land\ Thread\_Objects' = [Thread\_Objects\ \text{EXCEPT}\ ![HiRTOS.Cu$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \land\ \text{IF}\ Thread\_Objects'[HiRTOS.Current\_Thread\_Id].ghost\_Tin$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{THEN}\ \land\ HiRTOS' = [HiRTOS\ \text{EXCEPT}\ !.Runnable\_Th$

```
                                                          !.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_Th
                                                                           ![HiRTOS'.Current_Th
                                ∧ 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,
```

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

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

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

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

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

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

$acquire\_mutex\_make\_condvar\_wait\_awoken\_thread\_runnable\_step(self) \triangleq$ $\land\ pc[self] =$ "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
$\land\ Thread\_Objects' = [Threa$
$\land\ pc' = [pc$ EXCEPT $![self] =$
$\land$ UNCHANGED $\langle Mutex\_Objec$
$Condvar\_Obj$
$Timer\_Objec$
$Global\_Resou$
$stack$,
$thread\_id\_$,
$mutex\_id\_$,
$waking\_up\_th$
$owner\_thread$
$thread\_id\_A$,
$mutex\_id\_A$,

```
                                                                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_L
                                                                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_wait_on_mutex_step(self) ≜ ∧ pc[self] = "acquire_mutex_wait_on_mutex_step"
                                         ∧ owner_thread_id_′ = [owner_thread_id_ EXCEPT ![self] =
                                         ∧ Assert(owner_thread_id_′[self] ≠ thread_id_[self],
                                                  "Failure of assertion at line 270, column 10.")
                                         ∧ Mutex_Objects′ = [Mutex_Objects EXCEPT ![mutex_id_[se
                                         ∧ IF waking_up_thread_after_condvar_wait[self]
                                              THEN ∧ Assert(thread_id_[self] ≠ HiRTOS.Current_T
                                                            "Failure of assertion at line 274, column
                                                   ∧ Assert(Thread_Objects[thread_id_[self]].State
                                                            "Failure of assertion at line 275, column
                                                   ∧ UNCHANGED HiRTOS
                                              ELSE ∧ Assert(thread_id_[self] = HiRTOS.Current_T
                                                            "Failure of assertion at line 277, column
                                                   ∧ Assert(Thread_Objects[thread_id_[self]].State
                                                            "Failure of assertion at line 278, column
                                                   ∧ HiRTOS′ = [HiRTOS EXCEPT !.Current_Th
                                         ∧ Thread_Objects′ = [Thread_Objects EXCEPT ![thread_id_[
                                                                                    ![thread_id_[
                                         ∧ pc′ = [pc EXCEPT ![self] = "acquire_mutex_check_if_priorit
                                         ∧ UNCHANGED ⟨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⟩

acquire_mutex_priority_inheritance_step(self) ≜ ∧ pc[self] = "acquire_mutex_priority_inheritance_step"
                                               ∧ Mutex_Objects' = [Mutex_Objects EXCEPT ![mutex_
                                               ∧ IF Thread_Objects[owner_thread_id_[self]].State = "
                                                    THEN ∧ pc' = [pc EXCEPT ![self] = "acquire_mut
                                                    ELSE ∧ IF Thread_Objects[owner_thread_id_[sel
                                                              THEN ∧ pc' = [pc EXCEPT ![self] = 
                                                              ELSE ∧ Assert(Thread_Objects[own
                                                                              "Failure of assertion a
                                                                   ∧ pc' = [pc EXCEPT ![self] = 
                                               ∧ UNCHANGED ⟨HiRTOS,
                                                            Thread_Objects,
                                                            Condvar_Objects,
                                                            Timer_Objects,
                                                            Global_Resource_Available,
                                                            stack,
```

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

$$
acquire\_mutex\_priority\_inheritance\_if\_mutex\_owner\_runnable\_step(self) \triangleq \begin{array}{l}
\land\ pc[self] = \text{“acquire\_mutex\_} \\
\land\ Assert((Thread\_Objects[t \\
\qquad \text{“Failure of assertio} \\
\land\ \land\ HiRTOS' = [HiRTOS \\
\\
\quad \land\ Thread\_Objects' = [Th \\
\land\ pc' = [pc \text{ EXCEPT } ![self] = \\
\land\ \text{UNCHANGED } \langle Mutex\_Obj \\
\qquad\qquad Condvar\_O \\
\qquad\qquad Timer\_Obj \\
\qquad\qquad Global\_Res \\
\qquad\qquad stack, \\
\qquad\qquad thread\_id\_, \\
\qquad\qquad mutex\_id\_,
\end{array}
$$

```
                                                                                  waking_up_
                                                                                  owner_thre
                                                                                  thread_id_A
                                                                                  mutex_id_A
                                                                                  owner_thre
                                                                                  thread_id_I
                                                                                  mutex_id_I
                                                                                  doing_cond
                                                                                  awoken_thr
                                                                                  thread_id_F
                                                                                  mutex_id_F
                                                                                  thread_id_I
                                                                                  condvar_id_
                                                                                  mutex_id_I
                                                                                  thread_id_V
                                                                                  condvar_id_
                                                                                  mutex_id,
                                                                                  condvar_id_
                                                                                  do_context_
                                                                                  awoken_thr
                                                                                  to_reacquir
                                                                                  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) ≜  ∧ pc[self] = “acqui
                                                                             ∧ Assert((Thread_
                                                                                        “Failure c
                                                                             ∧  ∧ Mutex_Object





                                                                                ∧ Thread_Objec
                                                                             ∧ pc′ = [pc EXCEPT
                                                                             ∧ UNCHANGED ⟨Ha
                                                                                           Co
                                                                                           Ti
                                                                                           Ga
                                                                                           st
```

th
m
wo
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

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

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

$acquire\_mutex\_priority\_inheritance\_update\_prio\_step(self) \triangleq$ $\land\ pc[self] =$ "acquire_mutex_priority_inherit
$\land\ Thread\_Objects' = [Thread\_Objects$ EXC
$\land\ pc' = [pc$ EXCEPT $![self] =$ "acquire_mut
$\land$ UNCHANGED $\langle HiRTOS,$
$Mutex\_Objects,$
$Condvar\_Objects,$
$Timer\_Objects,$
$Global\_Resource\_Availabl$
$stack,$
$thread\_id\_,$
$mutex\_id\_,$
$waking\_up\_thread\_after\_c$

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

$acquire\_mutex\_check\_if\_synchronous\_context\_switch\_needed\_step(self) \triangleq$ $\land\ pc[self] =$ “acquire_mutex_ch
                                                                 $\land$ IF $\neg waking\_up\_thread\_after$
                                                                        THEN $\land\ pc' = [pc$ EXCE
                                                                        ELSE $\land\ pc' = [pc$ EXCE
                                                                 $\land$ 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_s
                                                                                        awoken_threa
                                                                                        to_reacquire_r
                                                                                        context_id_,
                                                                                        condvar_id_S
                                                                                        context_id,
                                                                                        condvar_id,
                                                                                        thread_was_a
                                                                                        thread_id,
                                                                                        delayed_threa

acquire_mutex_synchronous_context_switch_step(self) ≜ ∧ pc[self] = "acquire_mutex_synchronous_contex
                                                      ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure
                                                                                           pc
                                                                                         ○ stack[sel
                                                      ∧ pc′ = [pc EXCEPT ![self] = "check_time_slice_
                                                      ∧ UNCHANGED ⟨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⟩

do_acquire_mutex_return_step(self) ≜ ∧ pc[self] = "do_acquire_mutex_return_step"
                                      ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                                      ∧ owner_thread_id_′ = [owner_thread_id_ EXCEPT ![self] = Hea
                                      ∧ thread_id_′ = [thread_id_ EXCEPT ![self] = Head(stack[self]).t
                                      ∧ mutex_id_′ = [mutex_id_ EXCEPT ![self] = Head(stack[self]).m
                                      ∧ waking_up_thread_after_condvar_wait′ = [waking_up_thread_a
                                      ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
                                      ∧ UNCHANGED ⟨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⟩

Do_Acquire_Mutex(self) ≜ acquire_mutex_step_(self)
                           ∨ acquire_mutex_acquire_step(self)
                           ∨ acquire_mutex_make_condvar_wait_awoken_thread_runnable_step(self)
                           ∨ acquire_mutex_wait_on_mutex_step(self)
                           ∨ acquire_mutex_check_if_priority_inheritance_needed_step(self)
                           ∨ acquire_mutex_priority_inheritance_step(self)
                           ∨ acquire_mutex_priority_inheritance_if_mutex_owner_runnable_step(sel
                           ∨ acquire_mutex_priority_inheritance_if_mutex_owner_blocked_on_mutex
                           ∨ acquire_mutex_priority_inheritance_if_mutex_owner_blocked_on_condv
                           ∨ acquire_mutex_priority_inheritance_update_prio_step(self)
                           ∨ acquire_mutex_check_if_synchronous_context_switch_needed_step(self)
                           ∨ acquire_mutex_synchronous_context_switch_step(self)
                           ∨ do_acquire_mutex_return_step(self)

enter_critical_section_step_(self) ≜ ∧ pc[self] = "enter_critical_section_step_"
                                     ∧ HiRTOS.Interrupts_Enabled ∧
                                       (thread_id_A[self] ∈ Threads ⇒
                                          Thread_Objects[thread_id_A[self]].State = "Running")
                                     ∧ HiRTOS' = [HiRTOS EXCEPT !.Interrupts_Enabled = FALSE]
                                     ∧ Assert(HiRTOS'.Current_Thread_Id = thread_id_A[self],
                                                   "Failure of assertion at line 338, column 7.")
                                     ∧ ∧ mutex_id_' = [mutex_id_ EXCEPT ![self] = mutex_id_A[self]]
                                      ∧ stack' = [stack EXCEPT ![self] = ⟨[procedure ↦  "Do_Acquire_M
                                                                          pc          ↦  "exit_critical_se
                                                                          owner_thread_id_ ↦  owner
                                                                          thread_id_ ↦  thread_id_[se
                                                                          mutex_id_ ↦  mutex_id_[se
                                                                          waking_up_thread_after_con
                                                                          ∘ stack[self]]
                                      ∧ thread_id_' = [thread_id_ EXCEPT ![self] = thread_id_A[self]]
                                      ∧ waking_up_thread_after_condvar_wait' = [waking_up_thread_af
                                     ∧ owner_thread_id_' = [owner_thread_id_ EXCEPT ![self] = "Invalid
                                     ∧ pc' = [pc EXCEPT ![self] = "acquire_mutex_step_"]
                                     ∧ UNCHANGED ⟨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⟩

exit_critical_section_step_(self) ≜ ∧ pc[self] = "exit_critical_section_step_"
                                    ∧ HiRTOS' = [HiRTOS EXCEPT !.Interrupts_Enabled = TRUE]
                                    ∧ pc' = [pc EXCEPT ![self] = "acquire_mutex_return_step"]
                                    ∧ 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⟩

acquire_mutex_return_step(self) ≜ ∧ pc[self] = "acquire_mutex_return_step"
                                  ∧ pc' = [pc EXCEPT ![self] = Head(stack[self]).pc]
                                  ∧ owner_thread_id' = [owner_thread_id EXCEPT ![self] = Head(stac
                                  ∧ thread_id_A' = [thread_id_A EXCEPT ![self] = Head(stack[self]).th
                                  ∧ mutex_id_A' = [mutex_id_A EXCEPT ![self] = Head(stack[self]).m
                                  ∧ stack' = [stack EXCEPT ![self] = Tail(stack[self])]
                                  ∧ 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⟩

Acquire_Mutex(self) ≜ enter_critical_section_step_(self)
                        ∨ exit_critical_section_step_(self)
                        ∨ acquire_mutex_return_step(self)

release_mutex_step_(self) ≜ ∧ pc[self] = "release_mutex_step_"
                            ∧ Assert(¬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⟩

release_mutex_check_if_mutex_waiters_step(self) ≜ ∧ pc[self] = "release_mutex_check_if_mutex_waiters_s
                                                   ∧ IF Is_Thread_Priority_Queue_Empty(Mutex_Obj
                                                        THEN ∧ Mutex_Objects' = [Mutex_Objects EX
                                                             ∧ pc' = [pc EXCEPT ![self] = "do_releas
                                                        ELSE ∧ pc' = [pc EXCEPT ![self] = "release_m
                                                             ∧ UNCHANGED Mutex_Objects
                                                   ∧ UNCHANGED ⟨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⟩

release_mutex_wakeup_mutex_waiter_step(self) ≜ ∧ pc[self] = "release_mutex_wakeup_mutex_waiter_step
                                              ∧ awoken_thread_id_' = [awoken_thread_id_ EXCEPT
                                              ∧ Assert(Thread_Objects[awoken_thread_id_'[self]].W
                                                        "Failure of assertion at line 374, column 10.")
                                              ∧ ∧ HiRTOS' = [HiRTOS EXCEPT !.Runnable_Thre
                                                ∧ Mutex_Objects' = [Mutex_Objects EXCEPT ![mu
                                                                                          ![mu
                                                ∧ Thread_Objects' = [Thread_Objects EXCEPT ![a
                                                                                            ![a
                                                                                            ![a
                                              ∧ IF ¬doing_condvar_wait[self]
                                                    THEN ∧ pc' = [pc EXCEPT ![self] = "release_mu
                                                    ELSE ∧ pc' = [pc EXCEPT ![self] = "do_release_
                                              ∧ UNCHANGED ⟨Condvar_Objects,
                                                           Timer_Objects,
                                                           Global_Resource_Available,
                                                           stack,
                                                           thread_id_,
                                                           mutex_id_,
                                                           waking_up_thread_after_condvar_wa
                                                           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⟩

release_mutex_synchronous_context_switch_step(self) ≜ ∧ pc[self] = "release_mutex_synchronous_context
                                                           ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure
                                                                                                pc
                                                                                                ∘ stack[sel
                                                           ∧ pc′ = [pc EXCEPT ![self] = "check_time_slice_
                                                           ∧ UNCHANGED ⟨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"
        $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = Head(stack[self]).pc]$
        $\land\ awoken\_thread\_id\_' = [awoken\_thread\_id\_\ \text{EXCEPT}\ ![self] = He$
        $\land\ thread\_id\_D' = [thread\_id\_D\ \text{EXCEPT}\ ![self] = Head(stack[self$
        $\land\ mutex\_id\_D' = [mutex\_id\_D\ \text{EXCEPT}\ ![self] = Head(stack[self$
        $\land\ doing\_condvar\_wait' = [doing\_condvar\_wait\ \text{EXCEPT}\ ![self] =$
        $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = Tail(stack[self])]$
        $\land\ \text{UNCHANGED}\ \langle HiRTOS,\ Thread\_Objects,$
                    $Mutex\_Objects,$
                    $Condvar\_Objects,$
                    $Timer\_Objects,$
                    $Global\_Resource\_Available,$
                    $thread\_id\_,\ mutex\_id\_,$
                    $waking\_up\_thread\_after\_condvar\_wait,$
                    $owner\_thread\_id\_,$
                    $thread\_id\_A,\ mutex\_id\_A,$
                    $owner\_thread\_id,$
```

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

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

$enter\_critical\_section\_step\_R(self) \triangleq \land pc[self] = \text{“enter\_critical\_section\_step\_R”}$
$\quad \land HiRTOS.Interrupts\_Enabled \land$
$\qquad (thread\_id\_R[self] \in Threads \Rightarrow$
$\qquad\quad Thread\_Objects[thread\_id\_R[self]].State = \text{“Running”})$
$\quad \land HiRTOS' = [HiRTOS \text{ EXCEPT } !.Interrupts\_Enabled = \text{FALSE}]$
$\quad \land Assert(HiRTOS'.Current\_Thread\_Id = thread\_id\_R[self],$
$\qquad\qquad \text{“Failure of assertion at line 399, column 7.”})$
$\quad \land \ \land doing\_condvar\_wait' = [doing\_condvar\_wait \text{ EXCEPT } ![self] =$
$\qquad \land mutex\_id\_D' = [mutex\_id\_D \text{ EXCEPT } ![self] = mutex\_id\_R[s$
$\qquad \land stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \mapsto \ \text{“Do\_Release\_}$
$\qquad\qquad pc \mapsto \ \text{“exit\_critical\_}$
$\qquad\qquad awoken\_thread\_id\_ \mapsto \ aw$
$\qquad\qquad thread\_id\_D \mapsto \ thread\_id\_$
$\qquad\qquad mutex\_id\_D \mapsto \ mutex\_id\_$
$\qquad\qquad doing\_condvar\_wait \mapsto \ do$
$\qquad\qquad \circ\ stack[self]]$
$\qquad \land thread\_id\_D' = [thread\_id\_D \text{ EXCEPT } ![self] = thread\_id\_R[s$
$\quad \land awoken\_thread\_id\_' = [awoken\_thread\_id\_ \text{ EXCEPT } ![self] = \text{“In}$
$\quad \land pc' = [pc \text{ EXCEPT } ![self] = \text{“release\_mutex\_step\_”}]$
$\quad \land \text{UNCHANGED } \langle Thread\_Objects,$
$\qquad Mutex\_Objects,$
$\qquad Condvar\_Objects,$

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

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

$$\begin{array}{l}
\quad condvar\_id\_W,\ mutex\_id, \\
\quad condvar\_id\_D, \\
\quad do\_context\_switch, \\
\quad awoken\_thread\_id, \\
\quad to\_reacquire\_mutex\_id, \\
\quad context\_id\_, \\
\quad condvar\_id\_S,\ context\_id, \\
\quad condvar\_id, \\
\quad thread\_was\_awaken, \\
\quad thread\_id, \\
\quad delayed\_threads\rangle
\end{array}$$

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

$$\begin{array}{lll}
Release\_Mutex(self) & \triangleq & enter\_critical\_section\_step\_R(self) \\
 & & \lor\ exit\_critical\_section\_step\_R(self) \\
 & & \lor\ release\_mutex\_return\_step(self)
\end{array}$$

```
wait_on_condvar_wait_step(self) ≜ ∧ pc[self] = "wait_on_condvar_wait_step"
                                  ∧ Assert(¬HiRTOS.Interrupts_Enabled,
                                           "Failure of assertion at line 410, column 7.")
                                  ∧ ∧ Condvar_Objects' = [Condvar_Objects EXCEPT ![condvar_id_[s
                                    ∧ Thread_Objects' = [Thread_Objects EXCEPT ![thread_id_Do[sel
                                                                               ![thread_id_Do[sel
                                                                               ![thread_id_Do[sel
                                  ∧ HiRTOS' = [HiRTOS EXCEPT !.Current_Thread_Id = "Invalid_T
                                  ∧ pc' = [pc EXCEPT ![self] = "wait_on_condvar_release_mutex_step"]
                                  ∧ UNCHANGED ⟨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⟩

wait_on_condvar_release_mutex_step(self) ≜ ∧ pc[self] = "wait_on_condvar_release_mutex_step"
                                           ∧ IF mutex_id_Do[self] ≠ "Invalid_Mutex_Id"
                                                THEN ∧ ∧ doing_condvar_wait' = [doing_condvar_u
                                                       ∧ mutex_id_D' = [mutex_id_D EXCEPT ![s
                                                       ∧ stack' = [stack EXCEPT ![self] = ⟨[proced
                                                                                            pc
                                                                                            awoke
                                                                                            thread
                                                                                            mutex
                                                                                            doing_
                                                                                          ∘ stack
```

```
                                              ∧ thread_id_D′ = [thread_id_D EXCEPT ![s
                                            ∧ awoken_thread_id_′ = [awoken_thread_id_ E
                                            ∧ pc′ = [pc EXCEPT ![self] = "release_mutex_s
                                    ELSE    ∧ pc′ = [pc EXCEPT ![self] = "wait_on_condva
                                            ∧ UNCHANGED ⟨stack,
                                                         thread_id_D,
                                                         mutex_id_D,
                                                         doing_condvar_wait,
                                                         awoken_thread_id_⟩
                               ∧ 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_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⟩

wait_on_condvar_synchronous_context_switch_step(self) ≜  ∧ pc[self] = "wait_on_condvar_synchronous_c
                                                         ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedu
                                                                                              pc
```

$\circ\ stack[$

$\land\ pc' = [pc \text{ EXCEPT } ![self] = $ "check_time_sli

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

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

```
                              ∧ 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_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⟩

Do_Wait_On_Condvar(self) ≜ wait_on_condvar_wait_step(self)
                             ∨ wait_on_condvar_release_mutex_step(self)
                             ∨ wait_on_condvar_synchronous_context_switch_step(self)
                             ∨ do_wait_on_condvar_return_step(self)

enter_critical_section_step_W(self) ≜ ∧ pc[self] = "enter_critical_section_step_W"
                                      ∧ HiRTOS.Interrupts_Enabled ∧
                                        (thread_id_W[self] ∈ Threads ⇒
                                             Thread_Objects[thread_id_W[self]].State = "Running")
                                      ∧ HiRTOS' = [HiRTOS EXCEPT !.Interrupts_Enabled = FALSE]
                                      ∧ ∧ condvar_id_'  = [condvar_id_ EXCEPT ![self]       = condvar
                                        ∧ mutex_id_Do' = [mutex_id_Do EXCEPT ![self]      = mutex_
                                        ∧ stack' = [stack EXCEPT ![self] = ⟨[procedure ↦  "Do_Wait_C
                                                                             pc          ↦  "exit_critical
                                                                             thread_id_Do ↦  thread_
```

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

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

```
                                                  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⟩

wait_on_condvar_return_step(self) ≜ ∧ pc[self] = "wait_on_condvar_return_step"
                                    ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                                    ∧ thread_id_W′ = [thread_id_W EXCEPT ![self] = Head(stack[sel
                                    ∧ condvar_id_W′ = [condvar_id_W EXCEPT ![self] = Head(stack
                                    ∧ mutex_id′ = [mutex_id EXCEPT ![self] = Head(stack[self]).mut
                                    ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
                                    ∧ 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_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 $![condvar$
                                         $\land Assert(awoken\_thread\_id'[self] \neq HiRTOS.Current\_Threa$
                                                 "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.")
```

```
                                          ∧ Assert(Thread_Objects[awoken_thread_id'[self]].Waiting_
                                                     "Failure of assertion at line 456, column 10.")
                                          ∧ to_reacquire_mutex_id' = [to_reacquire_mutex_id EXCEPT
                                          ∧ Thread_Objects' = [Thread_Objects EXCEPT ![awoken_thr
                                                                                       ![awoken_thr
                                          ∧ pc' = [pc EXCEPT ![self] = "signal_condvar_check_if_mutex
                                          ∧ UNCHANGED ⟨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⟩

signal_condvar_check_if_mutex_reacquire_needed_step(self) ≜ ∧ pc[self] = "signal_condvar_check_if_mut
                                                              ∧ IF to_reacquire_mutex_id[self] ≠ "Inval
                                                                    THEN ∧ pc' = [pc EXCEPT ![self] =
                                                                    ELSE ∧ pc' = [pc EXCEPT ![self] =
                                                              ∧ UNCHANGED ⟨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⟩

signal_condvar_reacquire_mutex_step(self) ≜ ∧ pc[self] = "signal_condvar_reacquire_mutex_step"
                                            ∧ ∧ mutex_id_′ = [mutex_id_ EXCEPT ![self] = to_reacqu
                                              ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure ↦  "Do_
                                                                                    pc          ↦  "sign
                                                                                    owner_thread_id_ 
                                                                                    thread_id_ ↦  thre
                                                                                    mutex_id_ ↦  mut
                                                                                    waking_up_thread_
                                                                                  ∘ stack[self]]
```

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

$signal\_condvar\_awoken\_thread\_runnable\_step(self) \triangleq$ $\land\ pc[self] = \text{“signal\_condvar\_awoken\_thread\_runnab}$
                                                     $\land\ \land\ HiRTOS' = [HiRTOS\ \textsc{except}\ !.Runnable\_T$
                                                        $\land\ Thread\_Objects' = [Thread\_Objects\ \textsc{except}$
                                                     $\land\ pc' = [pc\ \textsc{except}\ ![self] = \text{“signal\_condvar\_chec}$
                                                     $\land\ \textsc{unchanged}\ \langle Mutex\_Objects,$
                                                                           $Condvar\_Objects,$
                                                                           $Timer\_Objects,$
                                                                           $Global\_Resource\_Available,$
```

```
                                        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⟩
```

$signal\_condvar\_check\_if\_sync\_context\_switch\_needed\_step(self) \triangleq$

```
∧ pc[self] = "signal_condvar_check_if_s
∧ IF do_context_switch[self]
      THEN ∧ pc' = [pc EXCEPT ![sel
      ELSE ∧ pc' = [pc EXCEPT ![sel
∧ UNCHANGED ⟨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⟩

do_condvar_signal_return_step(self) ≜ ∧ pc[self] = "do_condvar_signal_return_step"
                                      ∧ pc′ = [pc EXCEPT ![self] = Head(stack[self]).pc]
                                      ∧ awoken_thread_id′ = [awoken_thread_id EXCEPT ![self] = Hea
                                      ∧ to_reacquire_mutex_id′ = [to_reacquire_mutex_id EXCEPT ![se
                                      ∧ condvar_id_D′ = [condvar_id_D EXCEPT ![self] = Head(stack
                                      ∧ do_context_switch′ = [do_context_switch EXCEPT ![self] = Hea
                                      ∧ stack′ = [stack EXCEPT ![self] = Tail(stack[self])]
                                      ∧ 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_A, mutex_id_A,
                                                   owner_thread_id,
                                                   thread_id_D, mutex_id_D,
                                                   doing_condvar_wait,
```

$$\begin{array}{l}
awoken\_thread\_id\_,\\
thread\_id\_R,\ mutex\_id\_R,\\
thread\_id\_Do,\\
condvar\_id\_,\\
mutex\_id\_Do,\\
thread\_id\_W,\\
condvar\_id\_W,\ mutex\_id,\\
context\_id\_,\\
condvar\_id\_S,\\
context\_id,\ condvar\_id,\\
thread\_was\_awaken,\\
thread\_id,\\
delayed\_threads\rangle
\end{array}$$

$$\begin{array}{lll}
Do\_Signal\_Condvar(self) & \triangleq & signal\_condvar\_step(self)\\
 & & \quad \lor\ signal\_condvar\_wakeup\_waiter\_step(self)\\
 & & \quad \lor\ signal\_condvar\_check\_if\_mutex\_reacquire\_needed\_step(self)\\
 & & \quad \lor\ signal\_condvar\_reacquire\_mutex\_step(self)\\
 & & \quad \lor\ signal\_condvar\_awoken\_thread\_runnable\_step(self)\\
 & & \quad \lor\ signal\_condvar\_check\_if\_sync\_context\_switch\_needed\_step(self)\\
 & & \quad \lor\ signal\_condvar\_synchronous\_context\_switch\_step(self)\\
 & & \quad \lor\ do\_condvar\_signal\_return\_step(self)
\end{array}$$

$$\begin{array}{lll}
enter\_critical\_section\_step\_S(self) & \triangleq & \land\ pc[self] = \text{“enter\_critical\_section\_step\_S”}\\
 & & \land\ HiRTOS.Interrupts\_Enabled\ \land\\
 & & \quad (context\_id\_[self] \in Threads \Rightarrow\\
 & & \qquad Thread\_Objects[context\_id\_[self]].State = \text{“Running”})\\
 & & \land\ HiRTOS' = [HiRTOS\ \text{EXCEPT}\ !.Interrupts\_Enabled = \text{FALSE}]\\
 & & \land\ \land\ condvar\_id\_D' = [condvar\_id\_D\ \text{EXCEPT}\ ![self] = condvar\_ic\\
 & & \quad \land\ do\_context\_switch' = [do\_context\_switch\ \text{EXCEPT}\ ![self] = \text{TR}\\
 & & \quad \land\ stack' = [stack\ \text{EXCEPT}\ ![self] = \langle[procedure \mapsto\ \text{“Do\_Signal\_C}\\
 & & \qquad\qquad pc \mapsto\ \text{“exit\_critical\_}\\
 & & \qquad\qquad awoken\_thread\_id \mapsto\ awol\\
 & & \qquad\qquad to\_reacquire\_mutex\_id \mapsto\\
 & & \qquad\qquad condvar\_id\_D \mapsto\ condvar\\
 & & \qquad\qquad do\_context\_switch \mapsto\ do\_c\\
 & & \qquad\qquad \circ\ stack[self]]\\
 & & \land\ awoken\_thread\_id' = [awoken\_thread\_id\ \text{EXCEPT}\ ![self] = \text{“Inval}\\
 & & \land\ to\_reacquire\_mutex\_id' = [to\_reacquire\_mutex\_id\ \text{EXCEPT}\ ![self\\
 & & \land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“signal\_condvar\_step”}]\\
 & & \land\ \text{UNCHANGED}\ \langle Thread\_Objects,\\
 & & \qquad Mutex\_Objects,\\
 & & \qquad Condvar\_Objects,\\
 & & \qquad Timer\_Objects,\\
 & & \qquad Global\_Resource\_Available,
\end{array}$$

$$\begin{array}{l}
\quad \mathit{thread\_id\_}, \mathit{mutex\_id\_}, \\
\quad \mathit{waking\_up\_thread\_after\_condvar\_wait}, \\
\quad \mathit{owner\_thread\_id\_}, \\
\quad \mathit{thread\_id\_A}, \mathit{mutex\_id\_A}, \\
\quad \mathit{owner\_thread\_id}, \\
\quad \mathit{thread\_id\_D}, \mathit{mutex\_id\_D}, \\
\quad \mathit{doing\_condvar\_wait}, \\
\quad \mathit{awoken\_thread\_id\_}, \\
\quad \mathit{thread\_id\_R}, \mathit{mutex\_id\_R}, \\
\quad \mathit{thread\_id\_Do}, \\
\quad \mathit{condvar\_id\_}, \\
\quad \mathit{mutex\_id\_Do}, \\
\quad \mathit{thread\_id\_W}, \\
\quad \mathit{condvar\_id\_W}, \mathit{mutex\_id}, \\
\quad \mathit{context\_id\_}, \\
\quad \mathit{condvar\_id\_S}, \\
\quad \mathit{context\_id}, \mathit{condvar\_id}, \\
\quad \mathit{thread\_was\_awaken}, \\
\quad \mathit{thread\_id}, \\
\quad \mathit{delayed\_threads}\rangle
\end{array}$$

$$\begin{array}{lcl}
\mathit{exit\_critical\_section\_step\_S}(\mathit{self}) & \triangleq & \land\ \mathit{pc}[\mathit{self}] = \text{“exit\_critical\_section\_step\_S”} \\
& & \land\ \mathit{HiRTOS}' = [\mathit{HiRTOS}\ \text{EXCEPT}\ !.\mathit{Interrupts\_Enabled} = \text{TRUE}] \\
& & \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \text{“condvar\_signaled\_step”}] \\
& & \land\ \text{UNCHANGED}\ \langle \mathit{Thread\_Objects}, \\
& & \qquad \mathit{Mutex\_Objects}, \\
& & \qquad \mathit{Condvar\_Objects}, \\
& & \qquad \mathit{Timer\_Objects}, \\
& & \qquad \mathit{Global\_Resource\_Available}, \\
& & \qquad \mathit{stack}, \mathit{thread\_id\_}, \\
& & \qquad \mathit{mutex\_id\_}, \\
& & \qquad \mathit{waking\_up\_thread\_after\_condvar\_wait}, \\
& & \qquad \mathit{owner\_thread\_id\_}, \\
& & \qquad \mathit{thread\_id\_A}, \mathit{mutex\_id\_A}, \\
& & \qquad \mathit{owner\_thread\_id}, \\
& & \qquad \mathit{thread\_id\_D}, \mathit{mutex\_id\_D}, \\
& & \qquad \mathit{doing\_condvar\_wait}, \\
& & \qquad \mathit{awoken\_thread\_id\_}, \\
& & \qquad \mathit{thread\_id\_R}, \mathit{mutex\_id\_R}, \\
& & \qquad \mathit{thread\_id\_Do}, \\
& & \qquad \mathit{condvar\_id\_}, \mathit{mutex\_id\_Do}, \\
& & \qquad \mathit{thread\_id\_W}, \\
& & \qquad \mathit{condvar\_id\_W}, \mathit{mutex\_id}, \\
& & \qquad \mathit{condvar\_id\_D}, \\
& & \qquad \mathit{do\_context\_switch},
\end{array}$$

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

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

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

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

```
                                  $\land$ 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⟩

broadcast_condvar_wakeup_waiter_step(self) ≜  ∧ pc[self] = "broadcast_condvar_wakeup_waiter_step"
                                              ∧ ∧ condvar_id_D' = [condvar_id_D EXCEPT ![self] = 
                                                ∧ do_context_switch' = [do_context_switch EXCEPT ![
                                                ∧ stack' = [stack EXCEPT ![self] = ⟨[procedure ↦  "D
                                                                                      pc          ↦  "br
                                                                                      awoken_thread_i
                                                                                      to_reacquire_mut
                                                                                      condvar_id_D ↦
                                                                                      do_context_switc
                                                                                      ∘ stack[self]]
                                              ∧ awoken_thread_id' = [awoken_thread_id EXCEPT ![sel
                                              ∧ to_reacquire_mutex_id' = [to_reacquire_mutex_id EXC
                                              ∧ pc' = [pc EXCEPT ![self] = "signal_condvar_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_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⟩

broadcast_condvar_after_waking_up_one_waiter_step(self) ≜ ∧ pc[self] = "broadcast_condvar_after_wakin
                                                          ∧ thread_was_awaken′ = [thread_was_awak
                                                          ∧ pc′ = [pc EXCEPT ![self] = "broadcast_co
                                                          ∧ UNCHANGED ⟨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⟩

broadcast_condvar_check_if_sync_context_switch_needed_step(self) ≜ ∧ pc[self] = "broadcast_condvar_ch
                                                                  ∧ IF context_id[self] ∈ Threads ∧
                                                                        THEN ∧ pc′ = [pc EXCEPT !
                                                                        ELSE  ∧ pc′ = [pc EXCEPT !
                                                                  ∧ UNCHANGED ⟨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_awak
                                                                                thread_id,
                                                                                delayed_threads⟩

broadcast_condvar_synchronous_context_switch_step(self) ≜ ∧ pc[self] = "broadcast_condvar_synchronous
                                                          ∧ stack' = [stack EXCEPT ![self] = ⟨[proced
                                                                                               pc
                                                                                             ∘ stack
                                                          ∧ pc' = [pc EXCEPT ![self] = "check_time_s
                                                          ∧ UNCHANGED ⟨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,
```

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

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

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

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

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

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

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

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

$$\begin{array}{l}
\mathit{condvar\_id\_W},\ \mathit{mutex\_id},\ \mathit{condvar\_id\_D},\\
\mathit{do\_context\_switch},\ \mathit{awoken\_thread\_id},\\
\mathit{to\_reacquire\_mutex\_id},\ \mathit{context\_id\_},\\
\mathit{condvar\_id\_S},\ \mathit{context\_id},\ \mathit{condvar\_id},\\
\mathit{thread\_was\_awaken},\ \mathit{thread\_id},\\
\mathit{delayed\_threads}\rangle
\end{array}$$

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

$$\begin{array}{lll}
\mathit{after\_delay\_until\_step}(\mathit{self}) & \triangleq & \land\ \mathit{pc}[\mathit{self}] = \text{“after\_delay\_until\_step”}\\
& & \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{pc}]\\
& & \land\ \mathit{thread\_id}' = [\mathit{thread\_id}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{thread\_id}]\\
& & \land\ \mathit{stack}' = [\mathit{stack}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Tail}(\mathit{stack}[\mathit{self}])]\\
& & \land\ \text{UNCHANGED}\ \langle \mathit{HiRTOS},\ \mathit{Thread\_Objects},
\end{array}$$

```
                                                  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⟩

Delay_Until(self) ≜ enter_critical_section_step_D(self)
                      ∨ delay_until_step(self)
                      ∨ exit_critical_section_step_D(self)
                      ∨ after_delay_until_step(self)

thread_state_machine_next_state_loop(self) ≜ ∧ pc[self] = "thread_state_machine_next_state_loop"
                                              ∧ Thread_Objects[self].State = "Running" ∧ HiRTOS.Inte
                                              ∧ pc′ = [pc EXCEPT ![self] = "context_switch0"]
                                              ∧ 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⟩

context_switch0(self) ≜ ∧ pc[self] = "context_switch0"
                        ∧ ∨ ∧ pc′ = [pc EXCEPT ![self] = "acquire_mutex_step"]
                              ∧ UNCHANGED ⟨Global_Resource_Available, stack, context_id_, condvar_id
                          ∨ ∧ Global_Resource_Available′ = TRUE
                              ∧ ∨ ∧ ∧ condvar_id_S′ = [condvar_id_S EXCEPT ![self] = "condvar1"]
                                      ∧ context_id_′ = [context_id_ EXCEPT ![self] = self]
                                      ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure ↦  "Signal_Condvar'
                                                                          pc          ↦  "thread_iteration
                                                                          context_id_ ↦  context_id_[se
                                                                          condvar_id_S ↦  condvar_id_
                                                                          ∘ stack[self]]
                                    ∧ pc′ = [pc EXCEPT ![self] = "enter_critical_section_step_S"]
                                    ∧ UNCHANGED ⟨context_id, condvar_id, thread_was_awaken⟩
                                ∨ ∧ ∧ condvar_id′ = [condvar_id EXCEPT ![self] = "condvar1"]
                                      ∧ context_id′  = [context_id EXCEPT ![self]  = self]
                                      ∧ stack′ = [stack EXCEPT ![self] = ⟨[procedure ↦  "Broadcast_Conc
                                                                          pc          ↦  "thread_iteration
                                                                          thread_was_awaken ↦  thread
                                                                          context_id  ↦  context_id[self
                                                                          condvar_id ↦  condvar_id[sel
                                                                          ∘ stack[self]]
                                    ∧ thread_was_awaken′ = [thread_was_awaken EXCEPT ![self] = 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_Io
                           ∧ 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
                                                                                          $thread\_id\_W \mapsto$
                                                                                          $condvar\_id\_W \mapsto$
                                                                                          $mutex\_id$     $\mapsto$ $mu$
```

```
                                                                  ∘ stack[self]]
                                        ∧ thread_id_W' = [thread_id_W EXCEPT ![self] = self
                                      ∧ pc' = [pc EXCEPT ![self] = "enter_critical_section_step_
                                      ∧ UNCHANGED Global_Resource_Available
                                 ELSE ∧ Global_Resource_Available' = FALSE
                                      ∧ pc' = [pc EXCEPT ![self] = "release_mutex_step"]
                                      ∧ UNCHANGED ⟨stack,
                                                   thread_id_W,
                                                   condvar_id_W,
                                                   mutex_id⟩
                          ∧ 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, 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⟩

context_switch2(self) ≜ ∧ pc[self] = "context_switch2"
                        ∧ Thread_Objects[self].State = "Running" ∧ HiRTOS.Interrupts_Enabled
                        ∧ Assert(Mutex_Objects["mutex1"].Owner_Thread_Id = self,
                                 "Failure of assertion at line 562, column 21.")
                        ∧ pc' = [pc EXCEPT ![self] = "waiting_for_resource_step"]
                        ∧ 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,
                                     $thread\_id\_D$, $mutex\_id\_D$,
                                     $doing\_condvar\_wait$, $awoken\_thread\_id\_$,
                                     $thread\_id\_R$, $mutex\_id\_R$, $thread\_id\_Do$,
                                     $condvar\_id\_$, $mutex\_id\_Do$, $thread\_id\_W$,
                                     $condvar\_id\_W$, $mutex\_id$, $condvar\_id\_D$,
                                     $do\_context\_switch$, $awoken\_thread\_id$,
                                     $to\_reacquire\_mutex\_id$, $context\_id\_$,
                                     $condvar\_id\_S$, $context\_id$, $condvar\_id$,
                                     $thread\_was\_awaken$, $thread\_id$,
                                     $delayed\_threads\rangle$

$release\_mutex\_step(self) \triangleq$ $\land\ pc[self] =$ "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 $\langle HiRTOS$,
```

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

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

```
idle_thread_next_state_loop ≜ ∧ pc["Idle_Thread"] = "idle_thread_next_state_loop"
                              ∧ Thread_Objects["Idle_Thread"].State = "Running" ∧ HiRTOS.Interrupts
                              ∧ pc′ = [pc EXCEPT ![“Idle_Thread”] = “idle_thread_next_state_loop”]
                              ∧ 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⟩

Idle_Thread ≜ idle_thread_next_state_loop

timer_interrupt_next_state_loop ≜ ∧ pc["Timer_Interrupt"] = "timer_interrupt_next_state_loop"
                                  ∧ pc′ = [pc EXCEPT !["Timer_Interrupt"] = "enter_critical_section_step
                                  ∧ 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_T ≜ ∧ pc["Timer_Interrupt"] = "enter_critical_section_step_T"
                                ∧ HiRTOS.Interrupts_Enabled ∧
                                  ("Timer_Interrupt" ∈ Threads ⇒
                                      Thread_Objects["Timer_Interrupt"].State = "Running")
                                ∧ HiRTOS' = [HiRTOS EXCEPT !.Interrupts_Enabled = FALSE]
                                ∧ pc' = [pc EXCEPT !["Timer_Interrupt"] = "track_time_slice"]
                                ∧ 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⟩

track_time_slice ≜ ∧ pc["Timer_Interrupt"] = "track_time_slice"
                   ∧ IF HiRTOS.Current_Thread_Id ≠ "Invalid_Thread_Id"
                         THEN ∧ Assert(¬Thread_Objects[HiRTOS.Current_Thread_Id].ghost_Time_Sl
                                "Failure of assertion at line 606, column 13.")
                             ∧ Thread_Objects′ = [Thread_Objects EXCEPT ![HiRTOS.Current_Threa
                        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_
                                            ∧ ∧ condvar_id_D′ = [condvar_id_D EXCEPT !["Timer_
                                              ∧ do_context_switch′ = [do_context_switch EXCEPT ![
                                              ∧ stack′ = [stack EXCEPT !["Timer_Interrupt"] = ⟨[pro
                                                                                               pc
                                                                                               aw
                                                                                               to_
                                                                                               con
                                                                                               do_
                                                                                               ∘ st
                                            ∧ awoken_thread_id′ = [awoken_thread_id EXCEPT !["Ti
                                            ∧ to_reacquire_mutex_id′ = [to_reacquire_mutex_id EXC
                                            ∧ 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⟩
                                ∧ UNCHANGED ⟨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⟩

timer_interupt_asynchronous_context_switch_step ≜ ∧ pc[“Timer_Interrupt”] = “timer_interupt_asynchron
                                                  ∧ stack′ = [stack EXCEPT ![“Timer_Interrupt”] = ⟨[

                                                  ∧ pc′ = [pc EXCEPT ![“Timer_Interrupt”] = “check_
                                                  ∧ UNCHANGED ⟨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⟩

exit_critical_section_step_T ≜ ∧ pc["Timer_Interrupt"] = "exit_critical_section_step_T"
                                ∧ HiRTOS′ = [HiRTOS EXCEPT !.Interrupts_Enabled = TRUE]
                                ∧ pc′ = [pc EXCEPT !["Timer_Interrupt"] = "timer_interrupt_next_state_loo
                                ∧ 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⟩

Timer_Interrupt ≜ timer_interrupt_next_state_loop
                    ∨ enter_critical_section_step_T ∨ track_time_slice
                    ∨ wakeup_delay_until_waiters
                    ∨ timer_interupt_asynchronous_context_switch_step
```

$$\lor\ \mathit{exit\_critical\_section\_step\_T}$$

$$
\begin{array}{lll}
\mathit{other\_interrupt\_next\_state\_loop} & \triangleq & \land\ \mathit{pc}[\text{"Other\_Interrupt"}] = \text{"other\_interrupt\_next\_state\_loop"} \\
& & \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{"Other\_Interrupt"}] = \text{"enter\_critical\_section\_step} \\
& & \land\ \textsc{unchanged}\ \langle \mathit{HiRTOS},\ \mathit{Thread\_Objects}, \\
& & \qquad \mathit{Mutex\_Objects}, \\
& & \qquad \mathit{Condvar\_Objects}, \\
& & \qquad \mathit{Timer\_Objects}, \\
& & \qquad \mathit{Global\_Resource\_Available}, \\
& & \qquad \mathit{stack},\ \mathit{thread\_id\_}, \\
& & \qquad \mathit{mutex\_id\_}, \\
& & \qquad \mathit{waking\_up\_thread\_after\_condvar\_wait}, \\
& & \qquad \mathit{owner\_thread\_id\_}, \\
& & \qquad \mathit{thread\_id\_A},\ \mathit{mutex\_id\_A}, \\
& & \qquad \mathit{owner\_thread\_id}, \\
& & \qquad \mathit{thread\_id\_D},\ \mathit{mutex\_id\_D}, \\
& & \qquad \mathit{doing\_condvar\_wait}, \\
& & \qquad \mathit{awoken\_thread\_id\_}, \\
& & \qquad \mathit{thread\_id\_R},\ \mathit{mutex\_id\_R}, \\
& & \qquad \mathit{thread\_id\_Do},\ \mathit{condvar\_id\_}, \\
& & \qquad \mathit{mutex\_id\_Do},\ \mathit{thread\_id\_W}, \\
& & \qquad \mathit{condvar\_id\_W},\ \mathit{mutex\_id}, \\
& & \qquad \mathit{condvar\_id\_D}, \\
& & \qquad \mathit{do\_context\_switch}, \\
& & \qquad \mathit{awoken\_thread\_id}, \\
& & \qquad \mathit{to\_reacquire\_mutex\_id}, \\
& & \qquad \mathit{context\_id\_},\ \mathit{condvar\_id\_S}, \\
& & \qquad \mathit{context\_id},\ \mathit{condvar\_id}, \\
& & \qquad \mathit{thread\_was\_awaken}, \\
& & \qquad \mathit{thread\_id},\ \mathit{delayed\_threads}\rangle
\end{array}
$$

$$
\begin{array}{lll}
\mathit{enter\_critical\_section\_step} & \triangleq & \land\ \mathit{pc}[\text{"Other\_Interrupt"}] = \text{"enter\_critical\_section\_step"} \\
& & \land\ \mathit{HiRTOS}.\mathit{Interrupts\_Enabled} \land \\
& & \quad (\text{"Other\_Interrupt"} \in \mathit{Threads} \Rightarrow \\
& & \qquad \mathit{Thread\_Objects}[\text{"Other\_Interrupt"}].\mathit{State} = \text{"Running"}) \\
& & \land\ \mathit{HiRTOS}' = [\mathit{HiRTOS}\ \textsc{except}\ !.\mathit{Interrupts\_Enabled} = \textsc{false}] \\
& & \land\ \mathit{pc}' = [\mathit{pc}\ \textsc{except}\ ![\text{"Other\_Interrupt"}] = \text{"other\_interupt\_asynchronous\_co} \\
& & \land\ \textsc{unchanged}\ \langle \mathit{Thread\_Objects},\ \mathit{Mutex\_Objects}, \\
& & \qquad \mathit{Condvar\_Objects},\ \mathit{Timer\_Objects}, \\
& & \qquad \mathit{Global\_Resource\_Available}, \\
& & \qquad \mathit{stack},\ \mathit{thread\_id\_},\ \mathit{mutex\_id\_}, \\
& & \qquad \mathit{waking\_up\_thread\_after\_condvar\_wait}, \\
& & \qquad \mathit{owner\_thread\_id\_},\ \mathit{thread\_id\_A}, \\
& & \qquad \mathit{mutex\_id\_A},\ \mathit{owner\_thread\_id}, \\
& & \qquad \mathit{thread\_id\_D},\ \mathit{mutex\_id\_D},
\end{array}
$$

```
                                                    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⟩

other_interupt_asynchronous_context_switch_step ≜ ∧ pc["Other_Interrupt"] = "other_interupt_asynchrono
                                                     ∧ stack′ = [stack EXCEPT ![“Other_Interrupt”] = ⟨[p
                                                                                                        p
                                                                                                        ∘
                                                     ∧ pc′ = [pc EXCEPT ![“Other_Interrupt”] = “check_t
                                                     ∧ UNCHANGED ⟨HiRTOS,
                                                                  Thread_Objects,
                                                                  Mutex_Objects,
                                                                  Condvar_Objects,
                                                                  Timer_Objects,
                                                                  Global_Resource_Available,
                                                                  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,
```

$$\begin{array}{l}
\mathit{do\_context\_switch}, \\
\mathit{awoken\_thread\_id}, \\
\mathit{to\_reacquire\_mutex\_id}, \\
\mathit{context\_id\_}, \\
\mathit{condvar\_id\_S}, \\
\mathit{context\_id}, \\
\mathit{condvar\_id}, \\
\mathit{thread\_was\_awaken}, \\
\mathit{thread\_id}, \\
\mathit{delayed\_threads}\rangle
\end{array}$$

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

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

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

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

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

END TRANSLATION

Correctness Properties

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

There can be at most only one “running” thread

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

The running thread is not in any queue

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

```
    $HiRTOS.Current\_Thread\_Id \neq \text{“Invalid\_Thread\_Id”}) \Rightarrow$
      $(\land\ Thread\_Objects[HiRTOS.Current\_Thread\_Id].State = \text{“Running”}$
       $\land\ Thread\_Objects[HiRTOS.Current\_Thread\_Id].Waiting\_On\_Condvar\_Id = \text{“Invalid\_Condvar\_Id”}$
       $\land\ Thread\_Objects[HiRTOS.Current\_Thread\_Id].Waiting\_On\_Mutex\_Id = \text{“Invalid\_Mutex\_Id”}$
       $\land \lnot Is\_Thread\_In\_Priority\_Queue(HiRTOS.Runnable\_Threads\_Queue,\ HiRTOS.Current\_Thread\_$
       $\land \forall\, m \in Mutexes :$
           $\lnot Is\_Thread\_In\_Priority\_Queue(Mutex\_Objects[m].Waiting\_Threads\_Queue,\ HiRTOS.Curren$
       $\land \forall\, cv \in Condvars :$
           $\lnot Is\_Thread\_In\_Priority\_Queue(Condvar\_Objects[cv].Waiting\_Threads\_Queue,\ HiRTOS.Curr$
      $)$
```

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

```
$SafetyInvariant3 \triangleq$
    $HiRTOS.Interrupts\_Enabled \Rightarrow$
   $\forall\, t \in Threads : Thread\_Objects[t].State = \text{“Runnable”} \Rightarrow$
       $\land\ Thread\_Objects[t].Waiting\_On\_Condvar\_Id = \text{“Invalid\_Condvar\_Id”}$
       $\land\ Thread\_Objects[t].Waiting\_On\_Mutex\_Id = \text{“Invalid\_Mutex\_Id”}$
       $\land\ Is\_Thread\_In\_Priority\_Queue(HiRTOS.Runnable\_Threads\_Queue,\ t)$
       $\land\ Is\_Thread\_In\_Priority\_Queue\_In\_Only\_One\_Queue(HiRTOS.Runnable\_Threads\_Queue,\ t)$
       $\land \forall\, m \in Mutexes :$
           $\lnot Is\_Thread\_In\_Priority\_Queue(Mutex\_Objects[m].Waiting\_Threads\_Queue,\ t)$
       $\land \forall\, cv \in Condvars :$
           $\lnot Is\_Thread\_In\_Priority\_Queue(Condvar\_Objects[cv].Waiting\_Threads\_Queue,\ t)$
```

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

```
$SafetyInvariant4 \triangleq$
    $HiRTOS.Interrupts\_Enabled \Rightarrow$
   $\forall\, t \in Threads :$
      LET
         $thread\_obj \triangleq Thread\_Objects[t]$
      IN
         $thread\_obj.State = \text{“Blocked\_On\_Mutex”} \Rightarrow$
             $\land\ thread\_obj.Waiting\_On\_Mutex\_Id \neq \text{“Invalid\_Mutex\_Id”}$
             $\land\ thread\_obj.Waiting\_On\_Condvar\_Id = \text{“Invalid\_Condvar\_Id”}$
             $\land$ (LET
                     $mutex\_obj \triangleq Mutex\_Objects[thread\_obj.Waiting\_On\_Mutex\_Id]$
                 IN
                     $\land\ Is\_Thread\_In\_Priority\_Queue(mutex\_obj.Waiting\_Threads\_Queue,\ t)$
                     $\land\ Is\_Thread\_In\_Priority\_Queue\_In\_Only\_One\_Queue(mutex\_obj.Waiting\_Threads\_Qu$
             $\land \lnot Is\_Thread\_In\_Priority\_Queue(HiRTOS.Runnable\_Threads\_Queue,\ t)$
             $\land \forall\, m \in Mutexes \setminus \{thread\_obj.Waiting\_On\_Mutex\_Id\} :$
                 $\lnot Is\_Thread\_In\_Priority\_Queue(Mutex\_Objects[m].Waiting\_Threads\_Queue,\ t)$
             $\land \forall\, cv \in Condvars :$
                 $\lnot Is\_Thread\_In\_Priority\_Queue(Condvar\_Objects[cv].Waiting\_Threads\_Queue,\ t)$
```

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

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

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

$$\begin{array}{l}
\mathit{SafetyInvariant}9 \triangleq \\
\quad \mathit{HiRTOS}.\mathit{Interrupts\_Enabled} \Rightarrow \\
\quad \forall\, t \in \mathit{Threads} : \\
\quad\ (\mathit{Thread\_Objects}[t].\mathit{Owned\_Mutexes} = \langle\rangle \land \mathit{Thread\_Objects}[t].\mathit{State} \neq \text{“Blocked\_On\_Condvar”}) \Rightarrow \\
\quad\ \mathit{Thread\_Objects}[t].\mathit{Current\_Priority} = \mathit{Thread\_Objects}[t].\mathit{Base\_Priority}
\end{array}$$

Interrupts are not disabled indefinitely:

$$\begin{array}{l}
\mathit{LivenessProperty}1 \triangleq \\
\quad \neg \mathit{HiRTOS}.\mathit{Interrupts\_Enabled} \Rightarrow \Diamond \mathit{HiRTOS}.\mathit{Interrupts\_Enabled}
\end{array}$$

All runnable app threads of the same proirity eventually get the CPU:

$$\begin{array}{l}
\mathit{LivenessProperty}2 \triangleq \\
\quad \forall\, t \in \mathit{Threads} \setminus \{\text{“Idle\_Thread”}\},\ p \in \mathit{Valid\_Thread\_Priority\_Type} \setminus \{0\} : \\
\quad\ (\mathit{Thread\_Objects}[t].\mathit{Current\_Priority} = p \land \mathit{Thread\_Objects}[t].\mathit{State} = \text{“Runnable”}) \Rightarrow \\
\quad\quad \Diamond(\mathit{Thread\_Objects}[t].\mathit{State} = \text{“Running”})
\end{array}$$

All app threads complete at least one iteration

$$\begin{array}{l}
\mathit{LivenessProperty}3 \triangleq \\
\quad \forall\, t \in \mathit{Threads} \setminus \{\text{“Idle\_Thread”}\} : \\
\quad\quad \Diamond(\mathit{pc}[t] = \text{“thread\_iteration\_completed\_step”})
\end{array}$$

Every thread waiting to acquire a mutex, eventually gets it:

$$\begin{array}{l}
\mathit{LivenessProperty}4 \triangleq \\
\quad \forall\, t \in \mathit{Threads} : \\
\quad\quad \mathit{Thread\_Objects}[t].\mathit{State} = \text{“Waiting\_On\_Mutex”} \Rightarrow \Diamond(\mathit{Thread\_Objects}[t].\mathit{State} = \text{“Runnable”})
\end{array}$$

Every thread waiting on a condvar is eventually awaken:

$$\begin{array}{l}
\mathit{LivenessProperty}5 \triangleq \\
\quad \forall\, t \in \mathit{Threads} : \\
\quad\quad \mathit{Thread\_Objects}[t].\mathit{State} = \text{“Waiting\_On\_Condvar”} \Rightarrow \Diamond(\mathit{Thread\_Objects}[t].\mathit{State} = \text{“Runnable”})
\end{array}$$

$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{TypeInvariant}$
$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{SafetyInvariant}1$
$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{SafetyInvariant}2$
$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{SafetyInvariant}3$
$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{SafetyInvariant}4$
$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{SafetyInvariant}5$
$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{SafetyInvariant}6$
$\text{THEOREM}\ \mathit{Spec} \Rightarrow \Box \mathit{SafetyInvariant}7$

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 $Spec \Rightarrow \Box LivenessProperty5$