Kernel User's Manual
Version 3.0
Appendix A: Ada Code

Judy Bamberger
Tim Coddington
Robert Firth
Daniel Klein
David Stinchcomb
Roger Van Scoy
December 1989
Kernel User's Manual
Version 3.0
Appendix A: Ada Code

Judy Bamberger
Tim Coddington
Robert Firth
Daniel Klein
David Stinchcomb
Roger Van Scoy

Distributed Ada Real-Time Kernel Project

Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
This report was prepared for the

SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Karl Shingler
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Copyright © 1989 Carnegie Mellon University

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.
# Table of Contents

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>package Hardware_Interface</td>
<td>6</td>
</tr>
<tr>
<td>generic package Generic_Kernel_Time</td>
<td>18</td>
</tr>
<tr>
<td>package Kernel_Time</td>
<td>45</td>
</tr>
<tr>
<td>generic package Generic_Time_Globals</td>
<td>49</td>
</tr>
<tr>
<td>package Time_Globals</td>
<td>83</td>
</tr>
<tr>
<td>generic package Generic_Schedule_Types</td>
<td>86</td>
</tr>
<tr>
<td>package Schedule_Types</td>
<td>91</td>
</tr>
<tr>
<td>generic package Generic_Network_Globals</td>
<td>94</td>
</tr>
<tr>
<td>package Network_Globals</td>
<td>99</td>
</tr>
<tr>
<td>generic package Generic_Network_Configuration</td>
<td>102</td>
</tr>
<tr>
<td>package Network_Configuration</td>
<td>110</td>
</tr>
<tr>
<td>generic package Generic_Processor_Management</td>
<td>118</td>
</tr>
<tr>
<td>package Ke</td>
<td>118</td>
</tr>
<tr>
<td>package Tg</td>
<td>118</td>
</tr>
<tr>
<td>package Processor_Management</td>
<td>128</td>
</tr>
<tr>
<td>generic package Generic_Processor_Managers_Globals</td>
<td>131</td>
</tr>
<tr>
<td>package Process_Managers_Globals</td>
<td>134</td>
</tr>
<tr>
<td>generic package Generic_Processor_Managers</td>
<td>140</td>
</tr>
<tr>
<td>package Process_Managers</td>
<td>152</td>
</tr>
<tr>
<td>generic package Generic_Communication_Globals</td>
<td>155</td>
</tr>
<tr>
<td>package Communication_Globals</td>
<td>157</td>
</tr>
<tr>
<td>generic package Generic_Communication_Management</td>
<td>162</td>
</tr>
<tr>
<td>package Ke</td>
<td>162</td>
</tr>
<tr>
<td>package Tg</td>
<td>162</td>
</tr>
<tr>
<td>package Communication_Management</td>
<td>192</td>
</tr>
<tr>
<td>generic package Generic_Process_Attribute_Modifiers</td>
<td>195</td>
</tr>
<tr>
<td>package Ke</td>
<td>195</td>
</tr>
<tr>
<td>package Ptb</td>
<td>195</td>
</tr>
<tr>
<td>package St</td>
<td>196</td>
</tr>
<tr>
<td>package Process_Attribute_Modifiers</td>
<td>206</td>
</tr>
<tr>
<td>generic package Generic_Process_Attribute_Readers</td>
<td>210</td>
</tr>
<tr>
<td>package Ptb</td>
<td>210</td>
</tr>
<tr>
<td>package Ke</td>
<td>210</td>
</tr>
<tr>
<td>package St</td>
<td>210</td>
</tr>
<tr>
<td>package Process_Attribute_Readers</td>
<td>217</td>
</tr>
<tr>
<td>generic package Generic_Interrupt_Globals</td>
<td>220</td>
</tr>
<tr>
<td>package Interrupt_Globals</td>
<td>227</td>
</tr>
<tr>
<td>generic package Generic_Interrupt_Management</td>
<td>230</td>
</tr>
<tr>
<td>package Ig</td>
<td>230</td>
</tr>
<tr>
<td>package Interrupt_Management</td>
<td>241</td>
</tr>
<tr>
<td>generic package Generic_Semaphore_Management</td>
<td>244</td>
</tr>
<tr>
<td>package Semaphore_Management</td>
<td>255</td>
</tr>
<tr>
<td>generic package Generic_Alarm_Management</td>
<td>258</td>
</tr>
<tr>
<td>package Ke</td>
<td>258</td>
</tr>
<tr>
<td>package St</td>
<td>258</td>
</tr>
<tr>
<td>package Tg</td>
<td>258</td>
</tr>
<tr>
<td>package Alarm_Management</td>
<td>266</td>
</tr>
<tr>
<td>generic package Generic_Time_Management</td>
<td>269</td>
</tr>
</tbody>
</table>
The following packages are specific to the
Motorola 68020 version of the Kernel.
pragma (page);

with System;
   -- to get address

with Unchecked_Conversion;

-- package short name: HI

package Hardware_Interface is

-- The following definitions might need modification after any change to
-- either compiler or hardware. The purpose of the definitions is to
-- provide a coordinated and clean view of the underlying basic machine
-- data types and addressing structure.

   -- define the bit interface to the underlying system
   ...

Bits_Per_Bit : constant := 8;

   -- define constants to be used as offsets in reg specs; these values
   -- indicate how many bytes in the named storage unit (i.e., byte, word,
   -- longword)
   ...
   -- a byte contains 1 byte
   -- a word contains 2 bytes
   -- a longword contains 4 bytes
   ...

Byte : constant := 1;
Word : constant := 2;
Longword : constant := 4;

   -- The type hw_duration must capture exactly the hardware
   -- representation of the current Ada type DURATION
   ...
   -- to be defined for hw_duration:
   -- range
   -- size
   -- 'small' (i.e., granularity of measure)
   ...
   -- In the event of any mismatch, the compiler should reject the
   -- representation clauses

type Hw_Duration is new Duration range -86.400.0 .. +86.400.0;
for Hw_Duration use 2.0 ** (14);
for Hw_Duration use 4 * Bits_Per_Bit;
pragma (page);
-- The types hw_byte and hw_bits8 are to be used to define essentially
-- untiyped data formats, such as those found in device interfaces.
--
-- hw_byte is intended to be an unsigned 8-bit byte
-- hw_bits8 is intended to be a byte considered as a sequence of bits
--
-- Note especially that the bit numbering in the type hw_bits8 should
-- be whatever is most convenient for subsequent use of the type. If
-- the bulk of the hardware and device documentation numbers the bits
-- a certain way, that way should be used in the type declaration.

```
type Hw_Byte is range 0..255;
```

-- NOTE: the Telesoft compiler rejects the following representation clause
-- it will allocate TWO bytes (16 bits) for simple hw_byte objects
-- However, it will correctly allocate ONE byte (8 bits) for record
-- components of type hw_byte, if asked to by a rep clause

-- for hw_byte size use 1 'bits_per_byte,'
-- the same, considered as a sequence of bits

-- NOTE: WE FOLLOW THE MC68020 CONVENTION OF NUMBERING THE BITS
-- FROM THE RIGHT (bit0 = least significant)
--
-- THE COMPILER USES THE OPPOSITE CONVENTION, SO THE REP SPEC
-- MUST REVERSE THE NUMBERING

```
type Hw_Bits8 is record
  Bit7 : Boolean;
  Bit6 : Boolean;
  Bit5 : Boolean;
  Bit4 : Boolean;
  Bit3 : Boolean;
  Bit2 : Boolean;
  Bit1 : Boolean;
  Bit0 : Boolean;
end record;
```

for Hw_Bits8 use record
```
Bit7 at 0 range 0..1;
Bit6 at 0 range 2..3;
Bit5 at 0 range 3..3;
Bit4 at 0 range 4..4;
Bit3 at 0 range 5..5;
Bit2 at 0 range 6..6;
Bit1 at 0 range 7..7;
end record;
```

-- SUBPROGRAM NAME:
-- to_hw_bits8
--
-- DESCRIPTION:
-- conversion routine from hw_byte to hw_bits8
--
-- REQUIREMENTS:
-- 8.1.4
--
-- PRIM: N/A
--
-- TIME: N/A
--
-- PARAMETERS:
-- Source
-- an 8 bit value of type hw_byte
-- (a small integer in the range 0..255)
-- default value:
-- none
-- function return value
-- the same bit pattern, considered as a record of eight
-- single-bit components
--
-- PRECONDITIONS:
-- N/A
--
-- ACTIONS:
-- return the same physical bit pattern as the parameter
-- return the same physical bit pattern as the parameter
--
-- POSTCONDITIONS:
-- N/A
--
-- ERROR PROCESSING:
-- N/A
--
-- EXAMPLES:
-- This conversion is to be used when it is necessary to inspect
-- an individual bit of a device register or other hardware value.
--
-- Consider for instance a device status byte of which Bit2 is the
-- 'ready' bit, asserted whenever the device is ready. The test
-- in Ada then looks like this:
--
-- if to_hw_bits8(status byte) bit2 then -- device is ready
--
-- function To_hw_bits8 new Unchecked_Conversion(Hw_Byte,Hw_Bits8);

pragma (page);
function To_Hw_Bit8_Ptr is new Unchecked_Conversion(System.Address, Hw_Bit8_Ptr);

function To_Hw_Bit8_Ptr is new Unchecked_Conversion(System.Address, Hw_Bit8_Ptr);

-- The following types and conversions are intended to allow the
-- code to set individual bits in device registers. This is
-- achieved by
-- (a) declaring the register to be a hw_byte
-- (b) computing its Address
-- (c) casting that address into an access value designating an
-- object of type hw_bit8
-- (d) accessing the individual bits of that object

-- these types are pointers to hw_byte or hw_bit8 objects
-- they are intended to allow unchecked conversion of values

type Hw_Byte_Ptr is access Hw_Byte;
type Hw_Bit8_Ptr is access Hw_Bit8;

-- SUBPROGRAM NAME:
-- to_hw_bit8_ptr
--
-- DESCRIPTION:
-- conversion routine from Ada address type to access type
--
-- --REQ: 5.1.4
-- --PRIM: N/A
-- --TIME: N/A
--
-- PARAMETERS:
-- Source
-- the address of a declared object
-- WHICH MUST BE A HW_BYTE OR A HW_BIT8
-- default value:
-- none
-- function return value
-- the same bit pattern, considered as an access value designating
-- a hw_bit8
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- return the same physical bit pattern as the parameter
--
-- POSTCONDITIONS:
-- NONE
--
-- ERROR PROCESSING:
-- NONE
--
-- EXAMPLES:
-- This conversion is to be used when it is necessary to change
-- an individual bit of a device register or other hardware object
-- Consider for instance a device status byte of which Bit4 is the
-- 'enable' bit, to be set to enable the device. The code
-- in Ada then looks like this:
--
-- function to enable the device, we must set Bit 4 of the status byte
--
-- to hw bits8_ptr(status byte address).bit4 := True;
--
-- function To_Hw_Bit8_Ptr is new Unchecked_Conversion(System.Address, Hw_Bit8_Ptr);
/* ************************************************************
 /* SUBPROGRAM NAME:  
 /* to_hw_bits8_pr  
 /* DESCRIPTION:  
 /* conversion routine from Ada address type to access type  
 /*  
 /* -REQ- 5.1.4  
 /* -PRIM- NA  
 /* -TIME- NA  
 /*  
 /* PARAMETERS:  
 /*  
 /* Source  
 /* the address of a declared object  
 /* WHICH MUST BE A HW_BYTE OR A HW_BITS8  
 /*  
 /* default value:  
 /* none  
 /*  
 /* function return value  
 /* the same bit pattern, considered as an access value designating  
 /* a hw_bits8  
 /*  
 /* PRECONDITIONS:  
 /* NONE  
 /*  
 /* ACTIONS:  
 /* return the same physical bit pattern as the parameter  
 /*  
 /* POSTCONDITIONS:  
 /* NONE  
 /*  
 /* ERROR PROCESSING:  
 /* NONE  
 /*  
 /* EXAMPLES:  
 /* see above  
 /*  
 /* ************************************************************

function To_Hw_BITS8_Pr is new Unchecked_Conversion(Hw_BYTE_Pr,Hw_BITS8_Pr);
pragma (page):

-- The integer types are intended to capture the natural efficient  
-- integer types of the target machine. Currently, these are  
-- 8-bit signed : hw_short_integer  
-- 16-bit signed : hw_integer  
-- 32-bit signed : hw_long_integer  
--  
-- These types are defined in terms of explicit ranges, and a size  
-- clause is provided as a check that the compiler has indeed mapped  
-- them onto the correct hardware type.  
--  
-- The Ada types defined in package STANDARD should not be used, since  
-- they are dependent on the compiler as well as the target  
--  
-- hw_integer is intended to be a 16-bit integer  
--  
-- hw_short_integer is intended to be an 8-bit integer  
--  
-- hw_long_integer is intended to be a 32-bit integer  
--  
-- type Hw_Integer is range -32768..32767;  
-- for Hw_Integer/Size use 2 * Bits_Per_Byte;

-- type Hw_Long_Integer is range -2**31..2**31-1;  
-- for Hw_Long_Integer/Size use 4 * Bits_Per_Byte;

-- type Hw_SHORT_INTEGER is range -128..127;  
-- for Hw_SHORT_INTEGER/Size use 1 * Bits_Per_Byte;

-- type Hw_LONG_INTEGER is range -2**31..2**31-1;  
-- for Hw_LONG_INTEGER/Size use 4 * Bits_Per_Byte;  

pragma (page);
I

\[ \text{For additional security, the appropriate subsets of the machine} \]
\[ \text{integer types are defined as full types, not as subtypes. The} \]
\[ \text{two subsets are} \]
\[ \text{natural: the non-negative half of the range (0, Last)} \]
\[ \text{positive: the strictly positive subset (1, Last)} \]
\[ \text{Again, a size clause is added as a check.} \]
\[ \text{hw_long.natural is intended to be a 32-bit integer with natural range} \]
\[ \text{to be defined for hw_long.natural:} \]
\[ \text{range} \]
\[ \text{size} \]
\[ \text{type Hw_Long Natural is range 0..2 147 483 647;} \]
\[ \text{for Hw_Long.Natural Size use 4 * Bits Per Byte;} \]
\[ \text{hw_long.positive is intended to be a 32-bit integer with positive range} \]
\[ \text{to be defined for hw_long.positive:} \]
\[ \text{range} \]
\[ \text{size} \]
\[ \text{type Hw_Long Positive is range 1..2 147 483 647;} \]
\[ \text{for Hw_Long_Positive Size use 4 * Bits Per Byte;} \]
\[ \text{the range of hw.natural is to be a subset of hw_integer} \]
\[ \text{(i.e., only 16-bits)} \]
\[ \text{to be defined for hw.natural:} \]
\[ \text{range} \]
\[ \text{size} \]
\[ \text{type Hw_Natural is range 0..32 767;} \]
\[ \text{for Hw_Natural Size use 2 * Bits Per Byte;} \]
\[ \text{the range of hw.positive is to be a subset of hw_integer} \]
\[ \text{(i.e., only 16-bits)} \]
\[ \text{to be defined for hw.positive:} \]
\[ \text{range} \]
\[ \text{size} \]
\[ \text{type Hw_Positive is range 1..32 767;} \]
\[ \text{for Hw_Positive Size use 2 * Bits Per Byte;} \]
\[ \text{pragma (page);} \]

---

\[ \text{hw_address is included just in the off chance that the compiler's} \]
\[ \text{implementation of system address is not acceptable to the Kernel and/or} \]
\[ \text{the application; this allows Kernel/application developers to provide} \]
\[ \text{their own version of address operators without needing to modify Kernel} \]
\[ \text{or application code} \]
\[ \text{T: we is also a function to convert an integer of the appropriate} \]
\[ \text{size to an address, and a null value that cannot be a legal address} \]
\[ \text{in the current implementation, it has been shown that the} \]
\[ \text{compiler-provided address type is satisfactory.} \]

\[ \text{type Hw_Address is new System Address;} \]
\[ \text{-- SUBPROGRAM NAME:} \]
\[ \text{-- io_hw_address} \]
\[ \text{-- DESCRIPTION:} \]
\[ \text{-- conversion routine from an integer type to type hw_address} \]
\[ \text{-- REQ: NONE} \]
\[ \text{-- PRM: NONE} \]
\[ \text{-- TIME: NONE} \]
\[ \text{-- PARAMETERS:} \]
\[ \text{-- Source} \]
\[ \text{-- an integer value OF THE SAME SIZE AS AN ADDRESS} \]
\[ \text{-- the value MUST be a legal address} \]
\[ \text{-- default value:} \]
\[ \text{-- none} \]
\[ \text{-- function return value} \]
\[ \text{-- the same bit pattern, considered as an address} \]
\[ \text{-- PRECONDITIONS:} \]
\[ \text{-- NONE} \]
\[ \text{-- ACTIONS:} \]
\[ \text{-- return the same physical bit pattern as the parameter} \]
\[ \text{-- POSTCONDITIONS:} \]
\[ \text{-- NONE} \]
\[ \text{-- ERROR PROCESSING:} \]
\[ \text{-- NONE} \]
\[ \text{-- EXAMPLES} \]
\[ \text{-- The most common use of this function is to construct a pointer} \]
\[ \text{-- to a specific part of the machine's address space, for instance} \]
function To_Hw_Address
  new Unchecked_Conversion(Hw_Long_Integer, Hw_Address);

Null_Hw_Address: constant Hw_Address
  := To_Hw_Address(Hw_Long_Integer(0));

-- hw_string is included just in the off chance that the compiler
-- implementation of string is not acceptable to the Kernel and/or the
-- application, this allows Kernel/application developers to provide their
-- own version of string operators without needing to modify Kernel or
-- application code

-- In this initial implementation, we assume that the compiler-provided
-- string type and operations are OK

type Hw_String is new String;

end Hardware_Interface;

pragma (page).

--- SUPPORTING DOCUMENTATION:
--- AUTHORS:
--- L. Beaulieu
--- D. M. Wasserstrom
--- J. P. Brown (page).
generic

... ...
- the number of TICKs in a second (this value is never used within the
- Kernel except for the initialization of the constant, nor should it be
- used without the Kernel)
...

Ticks_Per_Second_Value := Hw_Long_Integer;

/package short name: GKT
/package Generic_Kernel_Time

is ...
...
- the constant number of TICKs per second (thus providing a mechanism of
- converting from Kernel time to application time)
...

Ticks_Per_Second := Hw_Long_Integer =>Ticks_Per_Second_Value;
...
...
- type Kernel_time, the time on which elapsed time and epoch time
- abstractions are built
...
- this time representation allows applications beginning at time zero to
- execute for some 150,000 years, so no overflow is monitored
...
- ...
- the user should ensure that adjusting any times does not approach the
- limit of this time representation, proper Kernel functioning is NOT
- guaranteed if time overflows
...
...
- Kernel_Time is represented as a signed 64-bit binary integer,
- representing a count of microseconds, hence, a kernel time
- of 1,000,000 corresponds to one second.
...
- For the purposes of Ada definition, Kernel_time is a record of
- two components, being respectively the low order and high order
- 32 bits. The high order component can be correctly declared to
- be a signed 32-bit integer, of type hw long integer. The low order
- component, however, is properly an UNSIGNED 32 Bit INTEGER, which
- this Ada compiler will not accept. Accordingly, it must be declared
- to be SIGNED, which is wrong
...
- The consequences are these
...
- (a) if a kernel time value is printed in the "obvious" manner, ie by
- printing each component, the low order part may be printed as a
- negative number when in fact it is a large positive number

with Hardware Interface, use Hardware Interface;
... to get: hw duration
... hw long integer
... hw positive

pragma (page);
function Seconds1 (Ada_Duration : Hw_Duration)
   return Kernel_Time;
function Seconds (Ada_Duration : Hw_Duration)
   return Kernel_Time renames Seconds1;
pragma Interface(Assembly, Seconds1);
pragma Linkname(Seconds1, "GKT_seconds1");

-- SUBPROGRAM NAME:
   seconds

-- DESCRIPTION:
   // conversion routine from Ada duration type to Kernel_time

   // -REQ: 12.1.23, 12.1.24
   // -PRM: N/A
   // -TIME: N/A

   // PARAMETERS:
   // Ada duration
   // the time to convert in Ada duration format (note that this limits
   // legal values to 0 to 86400 seconds, 1 day)
   // default value:
   // none
   // function return value
   // the input duration in Kernel_time format (note that this contains
   // time up to 1 day)

   // PRECONDITIONS:
   // NONE

   // ACTIONS:
   // returns the Ada duration time in Kernel_time format

   // POSTCONDITIONS:
   // NONE
   // ERROR PROCESSING:
   // NONE

pragma page;
function Seconds2 (  
   An Integral Duration ; Integral Duration)  
   return Kernel.Time;

function Seconds (  
   An Integral Duration ; Integral Duration)  
   return Kernel.Time rename Seconds2;

pragma Interface(Assembly, Seconds2);
pragma Linkname(Seconds2, "GKT_seconds2");

--- SUBPROGRAM NAME:  
   seconds  
---

--- DESCRIPTION:  
   conversion routine from integral elapsed times to Kernel. time  
---
   - REQ. 12.1.23, 12.1.24  
   - PRIM. N/A  
   - TIME. N/A  

--- PARAMETERS:  
   - an integral duration  
   - the time to convert in integer format  
   - default value  
   - none  
   - function return value  
   - the input integral duration in Kernel. time format  

--- PRECONDITIONS:  
   - NONE  

--- ACTIONS:  
   - returns the integral duration time in Kernel. time format  

--- POSTCONDITIONS:  
   - NONE  

--- ERROR PROCESSING:  
   - NONE  

---

pragma (page);
function Microseconds (Integral_Microseconds, Integral_Duration)
    return Kernel_Time;
pragma Interface(Assembly, Microseconds);
pragma Linkname(Microseconds, "GKT_microseconds");

function Add (Left : Kernel_Time; Right : Kernel_Time)
    return Kernel_Time;
pragma Interface(Assembly, Add);
pragma Linkname(Add, "GKT_add");

-- SUBPROGRAM NAME: seconds
-- DESCRIPTION: conversion routine from integral elapsed time to Kernel_time
-- REQ: 12.1.23, 12.1.24
-- PRIM: N/A
-- TIME: N/A
-- PARAMETERS:
-- integral microseconds
-- the duration in microseconds to convert to Kernel_time
-- default value: none
-- function return value
-- the input integral duration in Kernel_time format
-- PRECONDITIONS:
-- NONE
-- ACTIONS:
-- returns the integral duration time in Kernel_time format
-- POSTCONDITIONS:
-- NONE
-- ERROR PROCESSING:
-- NONE

-- SUBPROGRAM NAME: +
-- DESCRIPTION: adds two Kernel_time values returning a Kernel_time value
-- REQ: 12.1.23, 12.1.24
-- PRIM: N/A
-- TIME: N/A
-- PARAMETERS:
-- left
-- left operand of the addition operator
-- default value: none
-- right
-- right operand of the addition operator
-- default value: none
-- function return value
-- result of the addition operator
-- PRECONDITIONS:
-- NONE
-- ACTIONS:
-- takes the mathematical sum of the two Kernel_time values, returning a properly formatted Kernel_time result
-- POSTCONDITIONS:
-- the sum of left and right is computed and returned
-- ERROR PROCESSING:
-- NONE

pragma (page);
function Subtract (  
  Left : Kernel Time;  
  Right : Kernel Time)  
return Kernel Time;

function " "(  
  Left : Kernel Time;  
  Right : Kernel Time)  
return Kernel Time renames Subtract;

pragma Interface(Assembly, Subtract);  
pragma Linkname(Subtract, "GKT_subtract");

="/---------------------------------------------------------------"/
"/ SUBPROGRAM NAME:
"/ " "
"/
"/ DESCRIPTION:
"/ subtracts the right Kernel time value from the left Kernel time
"/ value returning a Kernel time value
"/
"/ -REQ: 12.1.23, 12.1.24
"/
"/ -PRIM: N/A
"/
"/ -TIME: N/A
"/
"/ -PARAMETERS:
"/  left
"/    left operand of the subtraction operator (subtrahend)
"/    default value:
"/    none
"/  right
"/    right operand of the subtraction operator (minuend)
"/    default value:
"/    none
"/
"/ -function return value
"/ result of the subtraction operator
"/
"/ -PRECONDITIONS:
"/ N/A
"/
"/ -ACTIONS:
"/ takes the mathematical difference of the two Kernel time values by
"/ subtracting the right operand from the left operand, returning a
"/ properly formatted Kernel time result
"/
"/ -POSTCONDITIONS:
"/ the difference of left and right is computed and returned
"/
"/ -ERROR PROCESSING:
function Multiply1 ( 
  Left : Kernel_Time, 
  Right : Hw_Long_Integer) 
return Kernel_Time;

function ** ( 
  Left : Kernel_Time; 
  Right : Hw_Long_Integer) 
return Kernel_Time rename Multiply1;

pragma Interface(Assembly, Multiply1); 
pragma Linkname(Multiply1, "C01_multiply1");

/* SUBPROGRAM NAME: */
/* ** */

/* DESCRIPTION: */
/* multiplies a Kernel_time value by an Integer value returning a */
/* Kernel_time value */

/* REQ: 12.1.23, 12.1.24 */
/* PRIM: N/A */
/* TIME: N/A */

/* PARAMETERS: */
/* left */
/* left operand of the multiplication operator */
/* default value: */
/* none */
/* right */
/* right operand of the multiplication operator */
/* default value: */
/* none */

/* function return value */
/* result of the multiplication operator */

/* PRECONDITIONS: */
/* NONE */

/* ACTIONS: */
/* takes the mathematical product of the Kernel_time value and the */
/* Integer, returning a properly formatted Kernel_time result */

/* POSTCONDITIONS: */
/* the product of left and right is computed and returned */

/* ERROR PROCESSING: */
/* NONE */
function Multiply2 (Left : Hw Long Integer; Right : Kernel Time)
return Kernel Time;

pragma "" (Left : Hw Long Integer;
Right : Kernel Time)
return Kernel Time renames Multiply2;

pragma Interface(Assembly, Multiply2);
pragma Linkname(Multiply2, "GLK1_multiply2");

-- SUBPROGRAM NAME:
-- ""
-- 
-- DESCRIPTION:
-- multiplies an integer value by a Kernel time value returning a
-- Kernel time value
-- 
-- -REQ: 12.1.23, 12.1.24
-- 
-- -PRIM: N/A
-- -TIME: N/A
-- 
-- PARAMETERS:
-- left
-- left operand of the multiplication operator
-- 
-- default value:
-- none
-- 
-- right
-- right operand of the multiplication operator
-- 
-- default value:
-- none
-- 
-- function return value
-- result of the multiplication operator
-- 
-- PRECONDITIONS:
-- NONE
-- 
-- ACTIONS
-- takes the mathematical product of the integer and the Kernel time
-- value, returning a properly formatted Kernel time result
-- 
-- POSTCONDITIONS
-- the product of left and right is computed and returned
-- 
-- ERROR PROCESSING:
-- NONE
function Divide (
Left : Kernel_Time;
Right : Hw_Long_Integer)
return Kernel_Time;

function "/( 
Left : Kernel_Time;
Right : Hw_Long_Integer)
return Kernel_Time renames Divide;

pragma Interface(Assembly, Divide);
pragma Linkname(Divide, "GKT_divide");

--================================================================-----------
-- SUBPROGRAM NAME:
-- "/"
--
-- DESCRIPTION:
-- divides a Kernel time value by an integer value returning a
-- Kernel time value
--
-- - REQ: 12.1.23, 12.1.24
-- - PRIM: N/A
-- - TIME: N/A
--
-- PARAMETERS:
-- left
-- left operand of the division operator (dividend)
--
-- default value:
-- none
--
-- right
-- right operand of the division operator (divisor)
--
-- default value:
-- none
--
-- function return value
-- result of the division operator
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- takes the mathematical quotient of the Kernel time value and the
-- integer, truncating the result toward zero if necessary, returning
-- a properly formatted Kernel time result
--
-- POSTCONDITIONS:
-- the quotient of left divided by right is computed and returned
--
-- ERROR PROCESSING:
function Unary_Plus (  
    Left: Kernel_Time;  
return Kernel_Time;  
)

function "*" (  
    Left: Kernel_Time  
return Kernel_Time rename Unary_Plus;  
)

pragma Interface(Assembly, Unary_Plus);  
pragma Linkname(Unary_Plus, "KRT_unary_plus");  

-- SUBPROGRAM NAME:  
-- "*"  

-- DESCRIPTION:  
-- unary plus of a Kernel, time value returning a Kernel_time value  

-- --REQ: 12.1.23, 12.1.24  
-- --PRIM: N/A  
-- --TIME: N/A  

-- PARAMETERS:  
-- left  
-- left operand of the unary plus operator  
-- default value:  
-- none  
-- function return value  
-- result of the unary plus operator  

-- PRECONDITIONS:  
-- NONE  

-- ACTIONS:  
-- takes the mathematical unary plus of the Kernel, time value, returning  
-- a properly formatted Kernel_time result  

-- POSTCONDITIONS:  
-- the unary plus of left is computed and returned  

-- ERROR PROCESSING:  
-- NONE  

}
function Unary_Minus ( 
   Left : Kernel_Time) 
return Kernel_Time; 

function "-" ( 
   Left : Kernel_Time) 
return Kernel_Time rename Unary_Minus; 

pragma Interface(Assembly, Unary_Minus); 
pragma Linkname(Unary_Minus, "GKT_unary_minus"); 

-- SUBPROGRAM NAME: 
-- "-" 
-- 
-- DESCRIPTION: 
-- unary negation of a Kernel_time value returning a Kernel_time value
-- 
-- --REQ: 12.1.23, 12.1.24 
-- 
-- --PRIM: NA 
-- 
-- --TIME: NA 
-- 
-- PARAMETERS: 
-- left 
-- left operand of the unary negation operator 
-- 
-- default value: 
-- none 
-- 
-- function return value 
-- result of the negation operator 
-- 
-- PRECONDITIONS: 
-- NONE 
-- 
-- ACTIONS: 
-- takes the mathematical unary negation of the Kernel_time value, 
-- returning a properly formatted Kernel_time result
-- 
-- POSTCONDITIONS: 
-- the unary negation of left is computed and returned 
-- 
-- ERROR PROCESSING: 
-- NONE 
-- 
-- "-" 

pragma (page);
function Le (  
  Left : Kernel_Time;  
  Right : Kernel_Time)  
return Boolean;

function "<" (  
  Left : Kernel_Time;  
  Right : Kernel_Time)  
return Boolean renames Le;

pragma Interface(Assembly, Le);  
pragma Linkname(L, "GTK_le");

/** .......................................................... */  
/** $SUBPROGRAM NAME: */  
/** "<" */  
/** */  
/** DESCRIPTION: */  
/** compares two Kernel_time values returning a Boolean true if left is */  
/** less than or equal to right, false otherwise */  
/** */  
/** --REQ: 12.1.23, 12.1.24 */  
/** */  
/** --PRIM: N/A */  
/** */  
/** --TIME: N/A */  
/** */  
/** --PARAMETERS: */  
/** --left */  
/** --left operand of the less-than-or-equal-to operator */  
/** */  
/** --default value: */  
/** --none */  
/** */  
/** --right */  
/** --right operand of the less-than-or-equal-to operator */  
/** */  
/** --default value: */  
/** --none */  
/** */  
/** --function return value */  
/** --result of the less-than-or-equal-to operator */  
/** */  
/** --PRECONDITIONS: */  
/** --NONE */  
/** */  
/** --ACTIONS: */  
/** --compares the two Kernel_time values, returning Boolean true if the */  
/** --left operand is less than or equal to the right, returning false */  
/** --otherwise */  
/** */  
/** --POSTCONDITIONS: */  
/** --the values of left and right are compared */  
/** */  
/** --ERROR PROCESSING: */
```c
function Gr (left: Kernel.Time; right: Kernel.Time)
return Boolean;

function ">" (left: Kernel.Time; right: Kernel.Time)
return Boolean rename Gr;

pragma Interface(Assembly, Gr);
pragma Linkname(Gr, "GKT_gr");

-- SUBPROGRAM NAME:
-- ">
--
-- DESCRIPTION:
-- compares two Kernel.Time values returning a Boolean true if left is
-- greater than right, false otherwise
--
-- REQ: 12.1.23, 12.1.24
--
-- PRM: N/A
--
-- TIME: N/A
--
-- PARAMETERS:
-- left
-- left operand of the greater-than operator
-- default value:
-- none
--
-- right
-- right operand of the greater-than operator
-- default value:
-- none
--
-- function return value
-- result of the greater-than operator
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- compares the two Kernel.Time values, returning Boolean true if the
-- left operand is greater than the right; returning false otherwise
--
-- POSTCONDITIONS:
-- the values of left and right are compared
--
-- ERROR PROCESSING:
-- NONE
```
function Ge (  
  Left : Kernel.Time;  
  Right : Kernel.Time)  
return Boolean;

function ">=" (  
  Left : Kernel.Time;  
  Right : Kernel.Time)  
return Boolean renames Ge,  
pragma Interface(Assembly, Ge);  
pragma Linkname(Ge,"OKT_.ge");

/-*======================================================================*/
/-* SUBPROGRAM NAME: */
/-* ">=" */
/-* DESCRIPTION: */
/-* compares two Kernel time values returning a Boolean true if left is */
/-* greater than or equal to right, false otherwise */
/-* */
/-* -REQ: 12.1.23, 12.1.24 */
/-* */
/-* -PRIM: N/A */
/-* */
/-* -TIME: N/A */
/-* */
/-* PARAMETERS: */
/-* left */
/-* left operand of the greater-than-or-equal-to operator */
/-* */
/-* default value: */
/-* none */
/-* */
/-* right */
/-* right operand of the greater-than-or-equal-to operator */
/-* */
/-* default value: */
/-* none */
/-* */
/-* function return value */
/-* result of the greater-than-or-equal-to operator */
/-* */
/-* PRECONDITIONS: */
/-* NONE */
/-* */
/-* ACTIONS: */
/-* compares the two Kernel time values, returning Boolean true if the */
/-* left operand is greater than or equal to the right, returning */
/-* false otherwise */
/-* */
/-* POSTCONDITIONS: */
/-* the values of left and right are compared */
/-* */
/-* ERROR PROCESSING.*/

and Generic Kernel Time;

```
--- NONE
---
--- +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

--- /* MODULE NAME:
--- // Kernel time
--- */

--- /* MODULE TYPE:
--- // package specification
--- // package body
--- */

--- /* MODULE PURPOSE:
--- // an example of (and a template for) an instantiation of the generic
--- // package generic_Kernel_time
--- */

--- /* MODULE DESCRIPTION:
--- // package instantiating the Kernel_time abstract type and subroutines to
--- // manipulate it where:
--- // ticks_per_second_value is defined as 16,384
--- */

--- /* MODULE CONTENTS:
--- // generic Kernel time (generic package specification)
--- // Integral duration (type)
--- // Kernel time (private type)
--- // seconds (overloaded (2) function)
--- // ticks per second (constant)
--- // "*" (function)
--- // "*" (function)
--- // "**" (overloaded (2) function)
--- // "*" (function)
--- // generic Kernel time (generic package body)
--- */

--- /* NOTES:
--- // this package can be instantiated and used in place of the
--- // compiler-provided Package Calendar
--- */

--- /* MODIFICATION HISTORY:
--- // 18Jan89 bamberg created (compiles clean)
--- // 14Jan89 birch final doc. update for Beta Release (no change)
--- */

--- /* VERSION: 3.0
--- */

--- /* DISCLAIMER:
--- */

--- // The following copyright must be included in this software and
--- // all software utilizing this software.
--- */

--- /* DISTRIBUTION:
--- */

--- // Approved for public release, distribution is unlimited.
--- */

--- // Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
--- // PA. The Software Engineering Institute (SEI) is a federally
--- // funded research and development center established and operated
--- // by Carnegie Mellon University (CMU). Sponsored by the U.S
The following packages are specific to both the Motorola 68020 and VAX/VMS versions of the Kernel.

```
with Generic Kernel Time;
pragma Elaborate (Generic Kernel Time);
-- to get: see MODULE CONTENTS

-- package short name, KT
```

```
package Kernel_Time

is new Generic Kernel Time (Ticks Per Second_Value => 500_000);
pragma (page);
```
-- MODULE NAME:
-- generic_time_globals
--
-- MODULE TYPE:
-- generic package specification
--
-- MODULE PURPOSE:
-- exports the type "time" as an alternative to an application's having
-- to use package calendar and its type "time"
--
-- MODULE DESCRIPTION:
-- exports type time in two distinct flavors
-- epoch_time (conceptually related to Ada's type TIME); and
-- elapsed_time (conceptually related to Ada's type DURATION)
-- and the "expected" operations to preserve those two abstractions
--
-- MODULE CONTENTS:
-- generic_time_globals (package specification)
-- base_time (function)
-- create_epoch_time (function)
-- create_elapsed_time (function)
-- elapsed_time (function)
-- elapsed_time (private type)
-- epoch_time (function)
-- integral_duration (type)
-- milliseconds (function)
-- microseconds (function)
-- seconds (function)
-- ticks_per_second (constant)
-- to_Ad_time (function)
-- to_Ada_time (overloaded function)
-- to_Ada_time (overloaded (2) function)
-- to_epoch_time (function)
-- to_epoch_time (overloaded (2) function)
-- zero_elapsed_time (constant)
-- zero_epoch_time (constant)
-- "+" (overloaded (2) function)
-- "-" (overloaded (2) function)
-- "*" (overloaded (2) function)
-- "/" (overloaded (2) function)
-- "<" (overloaded (2) function)
-- "<" (overloaded (2) function)
-- ">" (overloaded (2) function)
-- ">" (overloaded (2) function)
-- REFERENCES
-- DESIGN DOCUMENTS
-- Kernel Facilities Definition Document
-- -- REG 12 1 1, 12 1 25
-- -- PRM N/A
-- -- TIME N/A
--
-- USER'S MANUAL
--
-- TESTING AND VALIDATION:
-- Tested via unit tests for generic_kernel_time
--
-- NOTES:
-- none
--
-- MODIFICATION HISTORY:
-- 18 Aug 88 bamberg created (compiles clean)
-- 4 Jan 89 first revised; added more functions following
-- model in Kernel time
-- 13 Jan 89 final check for beta release
-- 17 Jan 89 final minor format changes
--
-- VERSION: 3.0
--
-- DISCLAIMER:
--
-- The following copyright must be included in this software and
-- all software utilizing this software.
--
-- DISTRIBUTION:
--
-- Approved for public release; distribution is unlimited.
--
-- Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
-- PA. The Software Engineering Institute (SEI) is a federally
-- funded research and development center established and operated
-- by Carnegie Mellon University (CMU). Sponsored by the U.S.
-- Department of Defense under contract F16628-86-C-0003, the SEI
-- is supported by the services and defense agencies, with the U.S.
-- Air Force as the executive contracting agent.
--
-- Permission to use, copy, modify, or distribute this software and
-- its documentation for any purpose and without fee is hereby
-- granted, provided that the above copyright notice appear in all
-- copies and that both that copyright notice and this permission
-- notice appear in supporting documentation. Further, the names
-- Software Engineering Institute or Carnegie Mellon University may
-- not be used in advertising or publicity pertaining to
-- distribution of the software without specific, written prior
-- permission. CMU makes no claims or representations about the
-- suitability of this software for any purpose. This software is
-- provided "as is" and no warranty, express or implied, is made
-- by the SEI or CMU, as to the accuracy and functioning of the
-- program and related program material, nor the fact of
-- distribution constitute any such warranty. No responsibility is
-- assumed by the SEI or CMU in connection herewith.
--
---end---
with Hardware Interface, use Hardware_Interface:
  -- to get: hw duration
  -- hw integer
  -- hw long integer

with Kernel Time:
  -- to get: integral duration
  -- Kernel time
  -- zero Kernel time
  -- ticks per second

generic

/.*******************************************************************************/
// package short name: GTG
/.*******************************************************************************/

package Generic Time Globals
is

  -- elapsed time conceptually corresponds to the Ada type "duration"

  type Elapsed Time is private;

  -- there is a zero value for elapsed time

  Zero Elapsed Time: constant Elapsed Time;

  -- epoch time conceptually corresponds to the Ada type "time"

  type Epoch Time is private;

  -- there is a zero value for epoch time

  Zero Epoch Time: constant Epoch Time;

  -- an integral duration is an integer valued measure of duration in
  -- user specified units

  type Integral Duration is new Kernel Time Integral Duration;

  -- the value ticks per second is reexported for completeness' sake

Ticks Per Second: constant Hw Long Integer
function Create Elapsed Time (  
Day : Hw Long Integer;  
Second : Hw Duration)  
return Elapsed Time;  
  
/**  
  * create elapsed time  
  */  
  
/**DESCRIPTION:  
  * create an elapsed time value from its component parts  
  */  
  
/**  
  * - REQ: 12.1.9  
  * - PRIM: N/A  
  * - TIME: N/A  
  */  
  
/**PARAMETERS:  
  * - day  
  *   any integer number of days  
  * - default value: none  
  */  
  
/**second  
  * number of seconds within that day (note that this limits  
  * legal values to 0 to 86400 seconds, 1 day)  
  */  
  
/**function return value  
  * an elapsed time value created from its component parts  
  */  
  
/**PRECONDITIONS:  
  * - NONE  
  */  
  
/**ACTIONS:  
  * returns the specified time in elapsed_time format  
  */  
  
/**POSTCONDITIONS:  
  * - NONE  
  */  
  
/**ERROR PROCESSING:  
  * - NONE  
  */  
  
******************************************************************************
pragma (page);  

function Create Epoch Time (  
Day : Hw Long Integer;  
Second : Hw Duration)  
return Epoch.Time;  
  
/**  
  * create epoch, time  
  */  
  
/**DESCRIPTION:  
  * create an epoch time value from its component parts  
  */  
  
/**  
  * - REQ: 12.1.9  
  * - PRIM: N/A  
  * - TIME: N/A  
  */  
  
/**PARAMETERS:  
  * - day  
  *   any integer number of days  
  * - default value: none  
  */  
  
/**second  
  * number of seconds within that day (note that this limits  
  * legal values to 0 to 86400 seconds, 1 day)  
  */  
  
/**function return value  
  * an epoch time value created from its component parts  
  */  
  
/**PRECONDITIONS:  
  * - NONE  
  */  
  
/**ACTIONS:  
  * returns the specified time in epoch, time format  
  */  
  
/**POSTCONDITIONS:  
  * - NONE  
  */  
  
/**ERROR PROCESSING:  
  * - NONE  
  */  
  
******************************************************************************
pragma (page);
function Seconds (An Integral Duration: Integral Duration)
return Elapsed Time;
// DESCRIPTION:
// conversion routine from Integral elapsed time to elapsed time
// -REQ: 12.1.23, 12.1.24
// -PRIM: N/A
// -TIME: N/A

// PARAMETERS:
// An Integral duration
// the time to convert in Integral duration format (note that this limits
// legal values to 0 to 86400 seconds, 1 day)
// default value:
// none
// function return value
// the input duration in elapsed, time format (note that this contains
// time up to 1 day)
// PRECONDITIONS:
// NONE
// ACTIONS:
// returns the Integral duration time in elapsed, time format
// POSTCONDITIONS:
// NONE
// ERROR PROCESSING:
// NONE
// =---------------------------------------------------------------

premise (page):
function milliseconds ( Integral milliseconds : Integral Duration)
    return Elapsed time;

-- DESCRIPTION:
-- conversion routine from integral elapsed time to elapsed_time
--
-- --REQ. 12 1 23, 12 1 24
--
-- --PRIM: N/A
--
-- --TIME: N/A
--

-- PARAMETERS:
-- integral milliseconds
-- the duration in milliseconds to convert to elapsed time
-- default value
-- none

-- function return value
-- the input integral duration in elapsed_time format
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- returns the integral duration time in elapsed_time format
--
-- POSTCONDITIONS:
-- NONE
--
-- ERROR PROCESSING:
-- NONE
--

-- ====

pragma (page);

function microseconds ( Integral microseconds : Integral Duration)
    return Elapsed time;

-- DESCRIPTION:
-- conversion routine from integral elapsed time to elapsed_time
--
-- --REQ. 12 1 23, 12 1 24
--
-- --PRIM: N/A
--
-- --TIME: N/A
--

-- PARAMETERS:
-- integral microseconds
-- the duration in microseconds to convert to elapsed time
-- default value
-- none

-- function return value
-- the input integral duration in elapsed_time format
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- returns the integral duration time in elapsed_time format
--
-- POSTCONDITIONS:
-- NONE
--
-- ERROR PROCESSING:
-- NONE
--

-- ====

pragma (page);
function "*"
  (Left : Epoch Time;
   Right : Epoch Time)
return Epoch Time;
/------------------------------------------------------------------
/ "*
/  
/ DESCRIPTION
/  adds an epoch time value and an elapsed time value returning an
/  epoch time value
/  
/  --REQ 12 1 23, 12 1 24
/  
/  --PRIM N/A
/  
/  --TIME N/A
/  
/  --PARAMETERS
/  -- left
/  -- left operand of the addition operator
/  
/  -- default value
/  -- none
/  
/  -- right
/  -- right operand of the addition operator
/  
/  -- default value:
/  -- none
/  
/  -- function return value
/  -- result of the addition operator
/  
/  -- PRECONDITIONS
/  -- NONE
/  
/  -- ACTIONS:
/  -- takes the mathematical sum of the epoch time value and the
/  -- elapsed time value, returning a properly formatted epoch time
/  -- result
/  
/  -- POSTCONDITIONS
/  -- the sum of left and right is computed and returned
/  
/  -- ERROR PROCESSING
/  -- NONE
/  
/=================================================================
pragma (page).

function "-"
  (Left : Epoch Time;
   Right : Epoch Time)
return Epoch Time;
/------------------------------------------------------------------
/ "-
/  
/ DESCRIPTION
/  subtracts the right elapsed time value from the left epoch time value
/  returning an epoch time value
/  
/  --REQ 12 1 23, 12 1 24
/  
/  --PRIM N/A
/  
/  --TIME N/A
/  
/  --PARAMETERS:
/  -- left
/  -- left operand of the subtraction operator (subtrahend)
/  
/  -- default value:
/  -- none
/  
/  -- right
/  -- right operand of the subtraction operator (minuend)
/  
/  -- default value:
/  -- none
/  
/  -- function return value
/  -- result of the subtraction operator
/  
/  -- PRECONDITIONS:
/  -- NONE
/  
/  -- ACTIONS:
/  -- takes the mathematical difference of the two values by subtracting
/  -- the right operand from the left operand, returning a properly
/  -- formatted epoch time result
/  
/  -- POSTCONDITIONS:
/  -- the difference of left and right is computed and returned
/  
/  -- ERROR PROCESSING:
/  -- NONE
/  
/=================================================================
pragma (page).
function `**` ( 
  Left Epoch Time,
  Right Epoch Time)
return Elapsed Time;
```
// SUBPROGRAM NAME:
-- **
--
-- DESCRIPTION:
-- subtracts the right epoch time value from the left epoch time value
-- returning an elapsed time value
--
-- REQ: 12.1.23, 12.1.24
--
-- PRIM: N/A
--
-- TIME: N/A
--
-- PARAMETERS:
-- left
--  left operand of the subtraction operator (subtrahend)
--
-- default value
--  none
--
-- right
--  right operand of the subtraction operator (minuend)
--
-- default value
--  none
--
-- function return value
--  result of the subtraction operator
--
-- PRECONDITIONS:
--  NONE
--
-- ACTIONS:
--  takes the mathematical difference of the two values by subtracting
--   the right operand from the left operand, returning a properly
--   formatted elapsed time result
--
-- POSTCONDITIONS:
--  the difference of left and right is computed and returned
--
-- ERROR PROCESSING:
--  NONE
--
-- !********************************************************************************
```
pragma (page);
```
function "*" 
Left : Elapsed Time;
Right : Elapsed Time;
return Elapsed Time;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SUBPROGRAM NAME:
* 
DESCRIPTION:
adds two elapsed time values returning an elapsed time value
--REQ: 12.1.23, 12.1.24
PRIM: N/A
TIME: N/A

PARAMETERS:
left
left operand of the addition operator

right
right operand of the addition operator

default value:
none

function return value
result of the addition operator

PRECONDITIONS:
NONE

ACTIONS:
takes the mathematical sum of the two elapsed time values, returning
a properly formatted elapsed time result

POSTCONDITIONS:
The sum of left and right is computed and returned

ERROR PROCESSING:
NONE

pragma (page);

function "+" 
Left : Elapsed Time;
Right : Elapsed Time;
return Elapsed Time;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SUBPROGRAM NAME:
*
DESCRIPTION:
subtracts the right elapsed time value from the left elapsed time
value returning an elapsed time value
--REQ: 12.1.23, 12.1.24
PRIM: N/A
TIME: N/A

PARAMETERS:
left
left operand of the subtraction operator (subtrahend)

right
right operand of the subtraction operator (minuend)

default value:
none

function return value
result of the subtraction operator

PRECONDITIONS:
NONE

ACTIONS:
takes the mathematical difference of the two elapsed time values by
subtracting the right operand from the left operand, returning a
properly formatted elapsed time result

POSTCONDITIONS:
The difference of left and right is computed and returned

ERROR PROCESSING:
NONE

pragma (page);
function """" ( 
    Left : Elapsed_Time; 
    Right : HW,Integer)
return Elapsed_Time;
-- SUBPROGRAM NAME:
-- """"
--
-- DESCRIPTION:
-- multiplies an elapsed_time value by an integer value returning an
-- elapsed_time value
-- 
-- REQ : 12.1.23, 12.1.24
-- 
-- PRIM: N/A
-- 
-- TIME: N/A
--
-- PARAMETERS:
-- left
-- left_operand of the multiplication operator
--
-- default value:
-- none
--
-- right
-- right_operand of the multiplication operator
--
-- default value:
-- none
--
-- result of the multiplication operator
--
-- function return value
--
-- PRECONDITIONS:
-- none
--
-- ACTIONS:
-- takes the mathematical product of the elapsed_time value and the
-- integer, returning a properly formatted elapsed_time result
--
-- POSTCONDITIONS:
-- the product of left and right is computed and returned
--
-- ERROR PROCESSING:
-- none
--
-- ==---------------------------------------------------------------------
pragma (page);

function """" ( 
    Left : HW,Integer; 
    Right : Elapsed_Time)
return Elapsed_Time;
-- SUBPROGRAM NAME:
-- """"
--
-- DESCRIPTION:
-- multiplies an integer value by an elapsed_time value returning an
-- elapsed_time value
-- 
-- REQ : 12.1.23, 12.1.24
-- 
-- PRIM: N/A
-- 
-- TIME: N/A
--
-- PARAMETERS:
-- left
-- left_operand of the multiplication operator
--
-- default value:
-- none
--
-- right
-- right_operand of the multiplication operator
--
-- default value:
-- none
--
-- result of the multiplication operator
--
-- function return value
--
-- PRECONDITIONS:
-- none
--
-- ACTIONS:
-- takes the mathematical product of the integer and the elapsed_time
-- value, returning a properly formatted elapsed_time result
--
-- POSTCONDITIONS:
-- the product of left and right is computed and returned
--
-- ERROR PROCESSING:
-- none
--
-- ==---------------------------------------------------------------------
pragma (page);
function "i"(  
Left: Elapsed_TIME;  
Right: New_INTEGER) 
return Elapsed_TIME; 
/** * SUBPROGRAM NAME:  
"i" */ 
/** * DESCRIPTION:  
divides an elapsed_time value by an integer value returning an  
elapsed_time value */ 
/** * REQ:  12.1.23, 12.1.24 */ 
/** * PRIM: N/A */ 
/** * TIME: N/A */ 
/** * PARAMETERS:  
"i" left  
default_value: none  
right  
default_value: none */ 
/** * function return value  
result of the division operator */ 
/** * PRECONDITIONS:  
"i" NONE */ 
/** * ACTIONS:  
takes the mathematical quotient of the elapsed_time value and the  
integer, returning a properly formatted elapsed_time result */ 
/** * POSTCONDITIONS:  
the quotient of left divided by right is computed and returned */ 
/** * ERROR PROCESSING:  
"i" NONE */ 
/** */ 
pragma (page);;

function "<"(  
Left: Epoch_TIME;  
Right: Epoch_TIME) 
return BOOLEAN; 
/** * SUBPROGRAM NAME:  
"<" */ 
/** * DESCRIPTION:  
compares two epoch_time values returning a Boolean true if left is  
less than right, false otherwise */ 
/** * REQ:  12.1.23, 12.1.24 */ 
/** * PRIM: N/A */ 
/** * TIME: N/A */ 
/** * PARAMETERS:  
"<" left  
default_value: none  
right  
default_value: none */ 
/** * function return value  
result of the less-than operator */ 
/** * PRECONDITIONS:  
"<" NONE */ 
/** * ACTIONS:  
compares the two epoch_time values, returning Boolean true if the  
left operand is less than the right; returning false otherwise */ 
/** * POSTCONDITIONS:  
the values of left and right are compared */ 
/** * ERROR PROCESSING:  
"<" NONE */ 
/** */ 
pragma (page);;
function "<=" (Left : Epoch Time; Right : Epoch Time) return Boolean;

-- SUBPROGRAM NAME: "<="
--
-- DESCRIPTION:
-- compares two epoch time values returning a Boolean true if left is
-- less than or equal to right, false otherwise
--
-- --REQ: 12.1.23, 12.1.24
--
-- --PRIM: N/A
--
-- --TIME: N/A
--
-- --PARAMETERS:
-- left
-- left operand of the less-than-or-equal-to operator
--
-- default value:
-- none
--
-- right
-- right operand of the less-than-or-equal-to operator
--
-- default value:
-- none
--
-- function return value
-- result of the less-than-or-equal-to operator
--
-- --PRECONDITIONS:
-- NONE
--
-- --ACTIONS:
-- compares the two epoch, time values, returning Boolean true if the
-- left operand is less than or equal to the right, returning false
-- otherwise
--
-- --POSTCONDITIONS:
-- the values of left and right are compared
--
-- --ERROR PROCESSING:
-- NONE
--
-----------------------------------------------------------------------

pragma (page);
function ">=" (  
Left Epoch Time;  
Right Epoch Time)  
return Boolean;  
/ * SUBPROGRAM NAME.  
/ * ">="  
/ */  
/ * DESCRIPTION.  
/ * compares two epoch time values returning a boolean true if left is  
/ * greater than or equal to right, false otherwise  
/ */  
/ * -REQ: 12 1 23, 12 1 24  
/ */  
/ * -PRIM: N/A  
/ */  
/ * -TIME: N/A  
/ */  
/ * -PARAMETERS:  
/ * left  
/ * left operand of the greater-than-or-equal-to operator  
/ */  
/ * default value:  
/ * none  
/ */  
/ * right  
/ * right operand of the greater-than-or-equal-to operator  
/ */  
/ * default value:  
/ * none  
/ */  
/ * function return value  
/ * result of the greater-than-or-equal-to operator  
/ */  
/ * -PRECONDITIONS:  
/ * -NONE  
/ */  
/ * -ACTIONS:  
/ * -comparis the two epoch time values, returning boolean true if the  
/ * left operand is greater than or equal to the right, returning  
/ * false otherwise  
/ * -POSTCONDITIONS:  
/ * the values of left and right are compared  
/ */  
/ * -ERROR PROCESSING:  
/ * -NONE  
/ */  
/ */  
pragma (page);  
}
function "<" (  
Left : Elapsed Time;  
Right : Elapsed Time)  
return Boolean;  

/* SUBPROGRAM NAME:  
"<"  
/* DESCRIPTION:  
compares two elapsed time values returning a Boolean true if left is  
less than or equal to right, false otherwise  
/* --REQ: 12.1.23, 12.1.24  
/* --PRIM: N/A  
/* --TIME: N/A  
/* PARAMETERS:  
left  
left operand of the less-than-or-equal-to operator  
default value:  
none  
right  
right operand of the less-than-or-equal-to operator  
default value:  
none  
function return value  
result of the less-than-or-equal-to operator  
/* PRECONDITIONS:  
NONE  
/* ACTIONS:  
compares the two elapsed time values, returning Boolean true if the  
left operand is less than or equal to the right, returning false  
otherwise  
/* POSTCONDITIONS:  
the values of left and right are compared  
ERROR PROCESSING:  
NONE  
/*  
*/  
pragma (page).
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Parameters</th>
<th>Preconditions</th>
<th>Actions</th>
<th>Error Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&gt;=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is greater than or equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>==</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is not equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is less than or equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is less than the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is greater than the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is greater than or equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is not equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is less than or equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is less than the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is greater than the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is greater than or equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is not equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is less than or equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is less than the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is greater than the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is greater than or equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><code>=</code></td>
<td>Compares two elapsed time/TVS values, returning a <code>true</code> if the left operand is equal to the right operand</td>
<td>Elapsed Time, TVS Values</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
function To Ada_Duration (  
    An Elapsed_Time: Elapsed_Time)  
return the Duration;  
      
-- SUBPROGRAM NAME:  
-- to Ada_duration  
-- DESCRIPTION:  
-- converts values of Kernel type elapsed_time to Ada type duration  
--  
-- REQ: 12.1.23, 12.1.24  
-- BPRM: N/A  
-- TIME: N/A  
-- PARAMETERS:  
-- an elapsed_time  
-- a value of type elapsed_time  
-- default value:  
-- none  
-- function return value  
-- result of the conversion  
-- PRECONDITIONS:  
-- none  
-- ACTIONS:  
-- returns the value as an Ada duration type value  
-- POSTCONDITIONS:  
-- none  
-- ERROR PROCESSING:  
-- when the result of the conversion would cause overflow:  
-- the Ada predefined exception CONSTRAINT_ERROR is raised  
-- POSTCONDITIONS:  
-- the conversion is not performed  
pragma (page);  

function To Kernel_Time (  
    Elapsed: Elapsed_Time)  
return Kernel_Time;  
      
-- SUBPROGRAM NAME:  
-- to Kernel_time  
-- DESCRIPTION:  
-- "INTERNAL KERNEL USE ONLY"  
-- In keeping with maintenance of elapsed_time as an abstract,  
-- private type, this conversion operator is needed  
--  
-- REQ: 12.1.23, 12.1.24  
-- BPRM: N/A  
-- TIME: N/A  
-- PARAMETERS:  
-- elapsed  
-- the time as an elapsed_time value  
-- default value:  
-- none  
-- function return value  
-- the time as a Kernel_time value  
-- PRECONDITIONS:  
-- none  
-- ACTIONS:  
-- none; just re-typifies the elapsed_time value as a Kernel_time value  
-- POSTCONDITIONS:  
-- none  
-- ERROR PROCESSING:  
-- none  
pragma (page);
function To Kernel Time ( 
    Epoch : Epoch Time) 
return Kernel.Time Kernel.Time; 
--------------------------------------------------------------- 
/* SUBPROGRAM NAME: 
   */ to Kernel_time 
/* */ 
/* DESCRIPTION: 
   */ """ INTERNAL KERNEL USE ONLY """ 
   */ In keeping with maintenance of epoch, time as an abstract, 
   */ private type, this conversion operator is needed 
/* */ 
/* */ -REQ. 12.1.23, 12.1.24 /* */ 
/* */ -PRIM: N/A /* */ 
/* */ -TIME: N/A /* */ 
/* */ 
/* PARAMETERS: 
   */ - elapsed 
   */ the time as an epoch, time value 
   */ - default value: 
   */ none 
   */ function return value 
   */ the time as a Kernel time value 
/* */ 
/* PRECONDITIONS: 
   */ none 
/* */ 
/* ACTIONS: 
   */ none, just re-types the epoch, time value as a Kernel_time value 
/* */ 
/* POSTCONDITIONS: 
   */ none 
/* */ 
/* ERROR PROCESSING: 
   */ none 
/* */ 
------------------------------------------------------------------- 

function To Elapsed Time ( 
    KTime : Kernel.Time Kernel.Time) 
return Elapsed Time; 
--------------------------------------------------------------- 
/* SUBPROGRAM NAME: 
   */ to elapsed time 
/* */ 
/* DESCRIPTION: 
   */ """ INTERNAL KERNEL USE ONLY """ 
   */ In keeping with maintenance of elapsed time as an abstract, 
   */ private type, this conversion operator is needed 
/* */ 
/* */ -REQ. 12.1.23, 12.1.24 /* */ 
/* */ -PRIM: N/A /* */ 
/* */ -TIME: N/A /* */ 
/* */ 
/* PARAMETERS: 
   */ KTime 
   */ the time as an Internal Kernel time value 
   */ - default value: 
   */ none 
   */ function return value 
   */ the time as elapsed time value 
/* */ 
/* PRECONDITIONS: 
   */ none 
/* */ 
/* ACTIONS: 
   */ none, just re-types the Kernel_time value as an elapsed_time value 
/* */ 
/* POSTCONDITIONS: 
   */ none 
/* */ 
/* ERROR PROCESSING: 
   */ none 
/* */ 
------------------------------------------------------------------- 

program (page);
function To_Epoch_Time (Ktime: Kernel_Time; {Kernel_Time}return Epoch_Time, {Epoch_Time}
-- SUBPROGRAM NAME:
-- to epoch_time
--
-- DESCRIPTION:
-- *** INTERNAL KERNEL USE ONLY ***
-- in keeping with maintenance of epoch_time as an abstract,
-- private type, this conversion operator is needed
--
-- REQ: 12.1.23, 12.1.24
--
-- PRIM: N/A
--
-- TIME: N/A
--
-- PARAMETERS:
-- Ktime
-- the time as an internal Kernel_time value
--
-- default value:
-- none
--
-- function return value
-- the time as epoch_time value
--
-- PRECONDITIONS:
-- none
--
-- ACTIONS:
-- none, just re-types the Kernel_time value as an epoch_time value
--
-- POSTCONDITIONS:
-- none
--
-- ERROR PROCESSING:
-- none
--
-- /===============================================
pragma (page);

private

-- elapsed_time is based on Kernel_time
--

type Elapsed_Time is new Kernel_Time; {Kernel_Time}
--
-- the zero value is also based on Kernel_time
--

Zero Elapsed_Time : constant Elapsed_Time :=
Elapsed_Time (Kernel_Time.Zero_Kernel_Time);
--
-- epoch_time is based on Kernel_time
--

type Epoch_Time is new Kernel_Time; {Kernel_Time}
--
-- the zero value is also based on Kernel_time
--

Zero Epoch_Time : constant Epoch_Time :=
Epoch_Time (Kernel_Time.Zero_Kernel_Time);

end Generic_Time_Globals;
pragma (page);
module time_globals

package specification

package generic_time_globals

package instantiating the type time is two distinct flavors:
epoch time (conceptually related to Ada's type TIME), and
elapsed time (conceptually related to Ada's type DURATION)
and the "expected" operations to preserve those two abstractions

package contents:
- time_globals (package specification)
- base time (function)
- create epoch time (function)
- epoch time (private type)
- epoch time (private type)
- integral duration (type)
- milliseconds (function)
- microseconds (function)
- seconds (overloaded (2) function)
- ticks per second (constant)
- to Ada duration (function)
- to elapsed time (overloaded (2) function)
- to epoch time (function)
- to Kernel time (overloaded (2) function)
- zero elapsed time (constant)
- zero epoch time (constant)
- "" (overloaded (2) function)
- "" (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
- " " (overloaded (2) function)
is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic Time Globals,

is new Generic TimeGlobals,

is new Generic TimeGlobals,

is new Generic TimeGlobals,
Ic
UjU
IrL
Lu
a
~.
EE
±.
L
CL
z
414x504
.00
c
II
I
30
PA
The Software Engineering Institute (SEI) is a federally
funded research and development center established and operated
by Carnegie Mellon University (CMU). Sponsored by the U.S.
Department of Defense under contract F19628-85-C-0003, the SEI
is supported by the services and defense agencies, with the U.S.
Air Force as the executive contracting agent.

The following copyright must be included in this software and
all software utilizing this software.

------

Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
PA. The Software Engineering Institute (SEI) is a federally
funded research and development center established and operated
by Carnegie Mellon University (CMU). Sponsored by the U.S.
Department of Defense under contract F19628-85-C-0003, the SEI
is supported by the services and defense agencies, with the U.S.
Air Force as the executive contracting agent.

Permission to use, copy, modify, or distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation. Further, the names
of Software Engineering Institute or Carnegie Mellon University may
not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission. CMU makes no claims or representations about the
suitability of this software for any purpose. This software is
provided "as is" and no warranty, express or implied, is made
by the SEI or CMU, as to the accuracy and functioning of the
program and related program material, nor shall the fact of
distribution constitute any such warranty. No responsibility is
assumed by the SEI or CMU in connection herewith.

with Kernel Exceptions:

- to get: hw integer
- to set: hw positive

- REQ: 5.1.8

Generic:

- value indicating the minimum possible priority value (this value is
  never used within the Kernel except for the initialization of the
  exported constant and the lower bound on priority, nor should it be used
  without the Kernel)

- REQ: 5.1.6, 9.14, 9.1.5

Lowest Priority Value: In Hw Positive;

package Generic Schedule Types

is

Null Priority Range: exception
renames Kernel Exceptions Null Priority Range;

- KERNEL PRIORITY MODEL:
  - the highest priority that should be used by an application process is 1;
  - this is available to the application via the constant highest_priority
  - the lowest priority that should be used by an application process is
    specified by the tailoring parameter lowest_priority value
  - this is available to the application via the constant lowest_priority
  - the kernel reserves priority first (0) and
    priority last (lowest_priority value + 1) for its own use
  - thus, the application should ONLY use priorities in the range
    highest_priority...lowest_priority
  - PRIORITY_FIRST and PRIORITY_LAST are reserved for use by the Kernel
  - ANY USE OF THESE VALUES WILL GENERATE INCORRECT PRIORITY ASSIGNMENTS
    FROM THE APPLICATION VIEWPOINT

this abstraction can be built by the user on top of the Kernel
type: Priority is new HW Natural range
  0 : HW Natural [Lowest_Priority, Value + 1);

  -- the lowest priority possible
  -- REQ: 5.1.6, 9.1.3

Highest_Priority : constant Priority := Priority'First + 1;
Lowest_Priority := Priority (Lowest_Priority, Value);

-- when a default priority is given on Kernel primitives that could block,
-- this value is used to indicate that the process's current priority should
-- be assumed (the "edge")

Current_Process_Priority : constant Priority := 0;

-- the default priority is always the lowest possible value
-- REQ: 5.1.6, 9.1.2

Default_Priority : constant Priority := Lowest_Priority;

-- this flag indicates whether or not a process may be preempted by another
-- process at the same priority level
-- REQ: 9.1.10

type: Preemption is (Enabled, Disabled);

-- the default preemption is always that a process may be preempted
-- REQ: 9.1.2

Default_Preemption : constant Preemption := Enabled;

-- a process is always in one of 4 possible states
-- running: the process currently executing on the processor
-- suspended: eligible to be running but, due to its priority, not
currently running
MODELE NAME:
- schedule_types

MODULE TYPE:
- package specification

MODULE PURPOSE:
- an example of (and a template for) an instantiation of the generic
- package generic_schedule_types

MODULE DESCRIPTION:
- package instantiating the schedule abstract types where:
- error checking for a null priority range is enabled
- highest priority value is (arbitrarily) defined as 1
- lowest priority value is (arbitrarily) defined as 255

MODULE CONTENTS:
- schedule_types (package specification)
- current_process_priority (constant)
- default_preemption (constant)
- default_priority (constant)
- default_process_state (constant)
- highest_priority (constant)
- lowest_priority (constant)
- null_priority_range (exception)
- preemption (type)
- priority (type)
- process_state (type)
- schedule_types (package body)
- elaboration-time initialization code

REFERENCES
- DESIGN DOCUMENTS
  - Kernel Facilities Definition
- USER'S MANUAL

TESTING AND VALIDATION:
- none

NOTES:
- none

MODIFICATION HISTORY:
- 18aug88 bamberg created (compiles clean)
- 13dec88 dvk deleted maximum slices per quantum and
  ticks per slice. This functionality has been
  moved to generic timelace management
- 15feb89 dvk updated documentation to reflect prior changes,
  added highest priority to documentation - per
  bug report 0008 SEI

version 3.0

DISCLAIMER:
- The following copyright must be included in this software and
- all software utilizing this software

DISTRIBUTION:
- Approved for public release; distribution is unlimited
- Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
  PA. The Software Engineering Institute (SEI) is a federally
  funded research and development center established and operated
  by Carnegie Mellon University (CMU). Sponsored by the U.S.
  Department of Defense under contract F19628-85-C-0003, the SEI
  is supported by the services and defense agencies, with the U.S.
  Air Force as the executive contracting agent.
- Permission to use, copy, modify, or distribute this software and
  its documentation for any purpose and without fee is hereby
  granted, provided that the above copyright notice appear in all
  copies and that both that copyright notice and this permission
  notice appear in supporting documentation. Further, the names
  Software Engineering Institute or Carnegie Mellon University may
  not be used in advertising or publicity pertaining to
  distribution of the software without specific, written prior
  permission. CMU makes no claims or representations about the
  suitability of this software for any purpose. This software is
  provided "as is" and warranty, express or implied, is made
  by the SEI or CMU, as to the accuracy and functioning of the
  program and related program material, nor shall the fact of
  distribution constitute any such warranty. No responsibility is
  assumed by the SEI or CMU in connection herewith.

with Generic Schedule Types:
pragma Elaborate (Generic Schedule Types);
  - to get: see MODULE CONTENTS

package short name: ST

package Schedule Types
In new Generic Schedule Types( 
   Null Priority Range Enabled => True, 
   Lowest Priority Value => 255), 
pragma (page);
with Hardware Interface; use Hardware Interface;
    -- to get: hw, natural

with Kernel Exceptions;
    -- to get: bus_address_check

generic

    ...
    -- ERROR checking conditional compilation flags
    ...
    --REQ: 5.1.8

Bus_Address_Check Enabled: In Boolean := True;
...
    ...
    value indicating a null bus address (this value is never used within the ...
    Kernel except for the initialization of the exported constant, nor should ...
    it be used without the Kernel)
...

Null Bus_Address_Value := In Hw Natural;

Pragmas(Page):
.. /********************************************************************************
.. / package short name: GNG
.. /********************************************************************************

package Generic_Network_Globals is
...
    ...
    bus_address_check raised by:
    ...
    Initialization code in corresponding package body
.......
    --REQ: 5.1.2, 5.1.4, 6.1.11

Bus_Address_Check: exception
renames Kernel_Exceptions.Bus_Address_Check;
.......
    *** begin pseudo generic formal parameters ***
...
    *** the following two values are hard wired because ada does not allow ...
    *** for non statically bounded types to be used within rep spec; the ...
    *** type bus_address is used within a rep spec in diagram globals, these ...
    *** should be treated as generic formal parameters (i.e., they should NOT ...
    *** be used outside this package)
...
    ...
    value indicating the lowest possible bus address
    ...

pragma (page):
First Bus Address Value := constant Hw_Natural := 0;

-- value indicating the highest possible bus address

Last Bus Address Value := constant Hw_Natural := 255;

... *** and pseudo generic formal parameters ***

... bus_address is a hw_integer with values within a user-specified range

... PRAM 15.1.4

type Bus Address is new Hw_Natural
range First_Bus_Address_Value .. Last_Bus_Address_Value;

... the following two declarations make available to the application the
... first and the last bus_address values

First Bus Address constant Bus Address :=
Bus_Address (First_Bus_Address_Value);

Last Bus Address constant Bus Address :=
Bus_Address (Last_Bus_Address_Value);

... this value is the null value for type bus_address

Null Address constant Bus Address :=
Bus_Address (Null_Bus_Address_Value);

pragma (page);
II

---

MODULE NAME:

network_globals

MODULE TYPE:

package specification

MODULE PURPOSE:

an example of (and a template for) an instantiation of the generic

package: generic_network_globals

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

MODULE DESCRIPTION

package instantiating the network global information where:

- checking for bus_address check is enabled
- the value of null_bus_address_value is initialized to 16#00

MODULE CONTENTS:

- network globals (package specification)
- bus address (type)
- bus address check (exception)
- first bus address (constant)
- last bus address (constant)
- null address (constant)
- process index type (type)
- processor identifier (type)

REFERENCES:

- DESIGN DOCUMENTS:

  - Kernel Facilities Definition
  - REQ.S.1.4, S.1.7, S.1.11
  - PRI: N/A
  - TIME: N/A

- USER'S MANUAL:


- TESTING AND VALIDATION:

  - Integration testing with generic processor management

- NOTES:

  - none

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

MODIFICATION HISTORY:

- Aug 89: bamberg: created (complies clean)
- Sep 89: bamberg: final doc. check for 01 Sep 89 freeze

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

VERSION: 3.0

DISCLAIMER:

- The following copyright must be included in this software and
  all software utilizing this software.
package Network_Globals

is new Generic_Network_Globals (Bus Address Check, Enabled => True,
Null Bus Address, Value => 16#000H),
pragma (page);

-- MODULE NAME:
-- generic_network_configuration
-- MODULE TYPE:
-- generic package specification
-- MODULE PURPOSE:
-- network configuration table (NCT) abstraction
-- MODULE DESCRIPTION:
-- provides the abstraction of the NCT and its component parts
-- MODULE CONTENTS:
-- generic_network_configuration (generic package specification)
-- configuration table (type)
-- get processor id (function)
-- maximum length of processor name (constant)
-- maximum length of processor_name_value (generic formal parameter)
-- NCT (data structure)
-- NCT entry (type)
-- number of nodes value (generic formal parameter)
-- number of nodes (constant)
-- REFERENCES:
-- DESIGN DOCUMENTS:
-- Kernel Facilities Definition
-- REQ: 5.1.5, 6.1.6, 6.1.10, 6.1.11;
-- REQ: 7.1.14, 7.1.16, 10.1.6, 10.1.15, 10.1.26
-- PRIM: 15.1.4
-- TIME: 6.2.1
-- USER'S MANUAL:
-- TESTING AND VALIDATION:
-- Integration testing with generic_processor_management
-- NOTES
-- none
-- MODIFICATION HISTORY:
-- 18aug88 bamberg created (complied clean)
-- 24aug88 bamberg change type of index of NCT to
-- network globals processor identifier (which it
-- should have been) and begin NCT origin at 1
-- 29aug88 bamberg move bus address stuff to NODE in break
-- circularity (DG, GP11B)
-- 01sep88 bamberg export maximum length of processor name
-- 13dec88 rvs provide complete default initialization for
-- NCT entries
-- 22mar99 rvs move get processor id from PIT to here
-- 30may99 rvs Bug fix 0063 set
-- name (i.e., upper bound on a string)

Maximum Length Of Processor_Name : constant Hw Natural :=
Maximum Length Of Processor_Name_Value;

-- each entry in the network control table (NCT) describes the physical
-- connectivity of one node in the network

-- the information needed about each node is:
-- logical name
-- the string-valued name given by the application engineer (this is
-- mapped to the device ID, which is a free index into the NCT
-- during initialization, once that is done, the Kernel refers to
-- the processor by device, ID)

-- default value:
-- none

-- this value should never change after initialization in Main Unit

-- physical address
-- the actual bus address at which the device is located; this value
-- is used in the datagram packet wrapper to identify the network
-- node that is to receive the packet containing the datagram

-- default value:
-- null address

-- this value should never change after initialization in Main Unit

-- Kernel device
-- indication of whether or not the device is also running the Kernel
-- (e.g., whether or not the device responds to the Kernel
-- communication protocols)

-- possible values:
-- true (the processor is running the Kernel)
-- false (the processor is not running the Kernel)

-- default value:
-- true (the processor is running the Kernel)

-- this value should never change after initialization in Main Unit

-- needed to run
-- indication of whether or not the device must successfully complete
-- the Kernel initialization protocol (i.e., always should be false
-- when Kernel device is false)

-- possible values:
-- true (the device does participate in the initialization
-- protocol)
-- false (the device does not participate in the initialization

-- protocol

-- default value:
-- true (the device does participate in the initialization
-- protocol)

-- this value should never change after initialization in Main Unit

-- allocated process ID
-- identifier of "surrogate process" allocated to receive messages
-- from the specified non-Kernel device.

-- default value:
-- null

-- this value may change via a call to the Kernel primitives:
-- allocate_device, receive
-- die
-- kill

-- Initialization order
-- order in which the processors identified in the NCT are to be
-- initialized

-- default value:
-- 0 (all NCT entries have the same initial value; initialization
-- proceeds following each entry in the NCT)

-- this value should never change after initialization in Main Unit

-- initialization complete
-- indication of whether or not the initialization protocol for this
-- processor has been completed

-- possible values:
-- true (Initialization has completed)
-- false (Initialization has not completed)

-- default value:
-- false (Initialization has not completed)

-- this value is set by the Kernel during initialization and should
-- never change after initialization is complete

type Nct Entry is record
Logical Name : Hw String;
(1..Positive (Maximum Length Of Processor Name Value));
< (0 others => ":"); Physical Address : Network Globals Bus Address :=
Network Globals Null Address;
Kernel Device : Boolean = true;
Needed To Run : Boolean = true;
Allocated Process Id : Process Table Process Identifier := null;
function Get_Processor_Id (Node_Address: In NetworkGlobals.Bus_Address) return NetworkGlobals.Processor_Identifier;
pragma Inline (Get_Processor_Id);

-- SUBPROGRAM NAME:
-- get_processo_id
--
-- DESCRIPTION:
-- Locate a node's entry in the NCT based on it's network address.
--
-- NOTES:
-- The need for this function arises because the entry for a node in
-- the NCT is not required to correspond to its network address.
-- Thus, a node with address 0 may be located in entry 0 of the NCT.
-- Also, care must be exercised when using this function. The NCT must
-- be initialized for this routine to return the correct result. Since
-- that can not be guaranteed until after a call to one of the
-- Initialize entries in processor_management, this function should
-- not be used during package elaboration.
--
-- REQ: 6.1.11, 10.1.6, 10.1.15
--
-- PRIM: N/A
--
-- TIME: 6.2.1
--
-- PARAMETERS:
-- node address
-- the network address of the node
-- default value:
-- none
-- return
-- the location of the node's entry in the NCT
-- default value:
-- none
--
-- PRECONDITIONS:
-- A valid node address is supplied
--
-- ACTIONS:
-- Search the NCT for the node address
-- Return the NCT location of the node
--
-- POSTCONDITIONS:
-- The returned location is a valid NCT entry
--
-- ERROR PROCESSING:
-- None
--
-- ============

Initialization_Order : Hw Natural := 0;
Initialization_Complete : Boolean := False;
end record;

...—template for the NCT: an array (bounds user-defined) of NCT_entry
—defined above
...

-- Configuration Table is array (''
NetworkGlobals.Processor_Identifier range <> of Nct_Entry;
...
...—this declares the actual NCT with the user-defined number of nodes
— in it, the user assigns values to each entry in the NCT at compile time
...
...—once this table is declared, the application should NOT make ANY changes
—to it. EVER, proper Kernel functioning is NOT guaranteed if any changes
—are made outside of calls to Kernel primitives
...
...
Nct : Configuration_Table (NetworkGlobals.Processor_Identifier
range 1..NetworkGlobals.Processor_Identifier (Number_Of_Nodes));

pragma (page);
and Generic Network Configuration;
pragma (page):

- -------------------------------
- MODULE NAME: network_configuration
- MODULE TYPE: package specification
- MODULE PURPOSE: network configuration table (NCT) abstraction
- MODULE DESCRIPTION: provides the abstraction of the NCT and its component parts
- MODULE CONTENTS: network configuration (package specification)
- configuration table (type)
- get processor id (function)
- maximum length of processor name (constant)
- maximum length of processor name_value
- (pseudo-generic formal parameter)
- NCT (data structure)
- NCT entry (type)
- number of nodes value (pseudo-generic formal parameter)
- number of nodes (constant)
- REFERENCES:
  - Kernel Facilities Definition
  - REQ. 5.1.6, 6.1.10, 6.1.11, 7.1.14, 7.1.16;
  - REQ. 10.1.6, 10.1.15, 10.1.26
  - PRIM 51.1.4
  - TIME 6.2.1
- USER'S MANUAL:
- TESTING AND VALIDATION:
  - Integration testing with generic processor management
- NOTES:
  - none
- MODIFICATION HISTORY:
  - 18aug88 bamberg created (compiles clean)
  - 24aug88 bamberg change type of index of NCT to
    network globals processor identifier (which it
    should have been) and begin NCT origin at 1
  - 29aug88 bamberg move bus address stuff to GNG to break
    circularity (OOG, GOP)
  - 01sep88 bamberg HAND INSTANTIATION due to telesoft bug!!!
  - export maximum length of processor name
  - 13dec88 rlv provide complete default initialization for
    NCT entries
  - 22mar89 rlv move get processor id from PIT to here
with Hardware Interface; use Hardware Interface;
-- to get: hw natural
  -- hw positive
  -- hw_string

with Network Global;
-- to get: bvs address
-- null address
-- processor identifier

with Process Table;
-- to get: null process
-- process identifier

package short name: NC
package Network_Configuration
package (page);

pragma (page);

108 December 1989

110 December 1989
Number Of Nodes: constant hw Positive = Number Of Nodes Value;

- the maximum length of the string used to represent a logical processor
- name (i.e., upper bound on a string)

Maximum Length Of Processor Name: constant hw Natural =

Maximum Length Of Processor Name Value;

- each entry in the network control table (NCT) describes the physical
- connectivity of one node in the network
- the information needed about each node is:
- logical name
- the string valued name given by the application engineer (this is
- mapped to the device ID which is just an index into the NCT
- during initialization, once that is done, the Kernel refers to
- the processor by device ID)
- default value
- none
- this value should never change after initialization in Main Unit
- physical address
- the actual bus address at which the device is located, this value
- is used in the datagram packet wrapper to identify the network
- node that is to receive the packet containing the datagram
- default value
- null address
- this value should never change after initialization in Main Unit
- Kernel device
- indication of whether or not the device is also running the Kernel
- (e.g., whether or not the device responds to the Kernel
- communication protocols)
- possible values:
- true (the processor is running the Kernel)
- false (the processor is not running the Kernel)
- default value
- true (the processor is running the Kernel)
- this value should never change after initialization in Main Unit
- needed to run
- indication of whether or not the device must successfully complete
- the Kernel initialization protocol (i.e., always should be false
- when Kernel device is false)

possible values:
- true (the device does participate in the initialization
- protocol)
- false (the device does not participate in the initialization
- protocol)
- default value:
- true (the device does participate in the initialization
- protocol)
- this value should never change after initialization in Main Unit
- allocated process ID
- identifier of "surrogate process" allocated to receive messages
- from the specified non-Kernel device
- default value:
- null
- this value may change via a call to the Kernel primitives:
- allocate device receiver
- die
- kill
- Initialization order
- order in which the processors identified in the NCT are to be
- initialized
- default value:
- 0 (all NCT entries have the same initial value; initialization
- proceeds following each entry in the NCT)
- this value should never change after initialization in Main Unit
- Initialization complete
- indication of whether or not the initialization protocol for this
- processor has been completed
- possible values:
- true (initialization has completed)
- false (initialization has not completed)
- default value:
- false (initialization has not completed)
- this value is set by the Kernel during initialization and should
- never change after initialization is complete

- REQ 6.1.10, 10.1.6, 10.1.26
- PRIM 15.1.4

type Nct Entry in record
Logical Name: hw String
(1: Positive (Maximum Length Of Processor Name Value))
function Get_Processor_Id (Node_Address: in Network_Globals.Bus_Address)
return Network_Globals.Processor_Identifier;
pragma Inline (Get_Processor_Id),
--============================================================================
-- SUBPROGRAM NAME:
-- get processor id
-- DESCRIPTION:
-- Locate a node's entry in the NCT based on it's network address.
--
-- NOTES:
-- Once the use of this function arises because the entry for a node in
-- the NCT is not required to correspond to its network address.
-- Thus, a node with address 0 may be located in entry 9 of the NCT.
-- Also, care must be exercised when using this function. The NCT must
-- be initialized for this function to return the correct result. Since
-- that can be guaranteed unless after a call to one of the
-- initialize entries in processor management, this function should
-- not be used during package elaboration.
--
-- --REQ 6.1.10, 10.1.6, 10.1.15
-- --PRIM: N/A
-- --TIME: 6.2.1
--
-- PARAMETERS:
-- node_address
-- the network address of the node
--
-- default value: none

Net_Configuration_Table (Network_Globals.Processor_Identifier
range 1..Network_Globals.Processor_Identifier (Number_Of_Nodes));
pragma (page);
and Network Configuration.

prolog (page):
with Kernel Exceptions;
-- to get: calling unit not Main Unit
-- configuration tables inconsistent
-- final sync Initialization_timeout expired
-- Master Initialization_timeout expired
-- network failure
-- process initialization failure
-- process maximum exceeded
-- processor failed to ACK go message
-- processor failed to transmit NCT
-- subordinate Initialization_timeout expired

with Hardware Interface; use Hardware Interface;
-- to get: hw duration
-- hw_positive

with Time Globals;

generic
--
-- ERROR checking conditional compilation flags
--
-- REQ: 5.1.8
--
-- Calling Unit Not Main Unit Enabled : In Boolean : = True;

Pragma(Page):
--...........................................................................
--package short name: GRM
--...........................................................................
package Generic Processor Management
in
package Ke
renames Kernel Exceptions;
package Tg
renames Time Globals;

--REQ: 6.1.8, 6.1.7, 6.1.8, 6.1.11
-- calling unit not Main Unit raised by:
-- initialize Master processor
-- initialize subordinate processor
-- initialization complete
-- PRIM 15.1.1, 15.1.2, 15.1.3

Calling Unit Not Main Unit : exception
renames Ka.Calling_Unit_Not_Main_Unit;

--
-- configuration_tables_inconsistent raised by:
-- initialize_Master_processor
-- PRIM: 15.1.1
Configuration Tables Inconsistent : exception
renames Ka.Configuration_Tables_Inconsistent;

-- final Sync_initialization_timeout_expired raised by:
-- initialization_complete
-- PRIM: 16.1.3
Final Sync_initialization_TIMEOUT_Expired : exception
renames Ka.Final_Sync_initialization_TIMEOUT_Expired;

-- Master_initialization_timeout_expired raised by:
-- initialize_Master_processor
-- PRIM: 15.1.1
Master Initialization_timeoutExpired: exception
renames Ka.Master_initialization_timeoutExpired;

-- Network_failure raised by:
-- initialize_Master_processor
-- initialize_subordinate_processor
-- initialization_complete
-- PRIM: 15.1.1
-- PRIM: 15.1.2
-- PRIM: 16.1.3
Network Failure : exception
renames Ka.Network.Failure;

-- process_initialization_failure raised by:
-- initialization_complete
-- PRIM: 16.1.3
Process Initialization Failure : exception
renames Ka.Process_initialization_failure;

-- process_maximum_exceeded raised by:
-- initialization_complete

-- PRIM: 16.1.3
Process Maximum_Expected : exception
renames Ka.Process.Maximum_Expected;

-- processor_failed_to_ACK_go_message raised by:
-- initialize_Master_processor
-- PRIM: 15.1.1
Processor Failed To Ack Go Message : exception
renames Ka.Processor_Failed_To_Ack_Go_Message;

-- processor_failed_to_transmit_NCT raised by:
-- initialize_Master_processor
-- PRIM: 15.1.1
Processor Failed To Transmit Nct : exception
renames Ka.Processor_Failed_To_Transmit_Nct;

-- subordinate_initialization_timeout_expired raised by:
-- initialize_subordinate_processor
-- PRIM: 15.1.2
Subordinate Initialization_TIMEOUT_Expired : exception
renames Ka.Subordinate_initialization_TIMEOUT_Expired;
procedure initialize_Master_Processor (Base Epoch, In Tgt Epoch, Time);
    SUBPROGRAM NAME: initialize_Master_processor
    DESCRIPTION:
    - begins Master Initialization sequence; identifies caller as Master
    - processor for duration of initialization only
    - REQ: 6.1.1, 6.1.3, 6.1.4, 6.1.5
    - PRIM 15.1.1
    - TIME 6.2.1, 6.2.2
    PARAMETERS:
    - base epoch
    - the base epoch time for the network
    - default value: none
    - timeout
    - duration to wait for a subordinate to respond to a message from the Master processor.
    - default value: none
    - NOTES:
    - Only those nodes that are Kernel devices and needed to run can result in a network failure.
    - PRECONDITIONS:
    - the network hardware is up and initialized
    - the processor intending to be Master for initialization purposes
    - may call this procedure
    - no other processor has yet called initialize_Master_processor
    - a previous call of initialize_Master_processor by this processor must not have been done
    - ACTIONS:
    - Initialize all internal Kernel data structures
    - Master identifies itself to all subordinate processors
    - Master waits for all subordinates to send back their NCTs
    - Master compares other NCTs with its own to ensure a network-wide consistent view of the physical network topology
    - Master tells all subordinate processors the current time of day
    - Master waits for all subordinates to acknowledge the go message
    - POSTCONDITIONS:
    - all NCTs are consistent (i.e., the physical topology is consistent across processors)
    - all the clocks are synchronized

    pragma (page);
procedure Initialize Subordinate Processor ; (Timeout After: In Tg Elapsed Time);

/*---------------------------------------------------------*/
/* SUBPROGRAM NAME:
/* Initialize subordinate processor
/*---------------------------------------------------------*/
/* DESCRIPTION:
/* begins subordinate initialization sequence; identifies caller as a
/* subordinate processor for duration of initialization only
/*---------------------------------------------------------*/
/* - REQ: 8.1.2, 8.1.3, 8.1.4
/* - PRIM: 15.1.2
/* - TIME: 8.2.1
/*---------------------------------------------------------*/

/* PARAMETERS:
/* timeout
/* duration to wait until subordinate asserts that network
/* initialization has failed
/* default value:
/* none
/*---------------------------------------------------------*/

/* PRECONDITIONS:
/* the network hardware is up and initialized
/* only the processors not intending to be Master for initialization
/* purposes may call this procedure
/* a previous call of Initialize Subordinate processor by this processor must not
/* have been done
/*---------------------------------------------------------*/

/* ACTIONS
/* Initialize all internal Kernel data structures
/* a subordinate waits for the Master to identify itself
/* a subordinate sends its NCT to the Master for consistency checking
/* a subordinate receives the epoch time from the Master and sets its
/* own local time
/*---------------------------------------------------------*/

/* POSTCONDITIONS:
/* all NCTs are consistent
/* The local clock is synchronized with the Master’s clock
/*---------------------------------------------------------*/

/* ERROR PROCESSING:
/* when the call is detected from a unit that is not the Main Unit:
/* the Kernel raises calling unit, not Main Unit exception
/* POSTCONDITIONS:
/* the requested action does not take place
/* when the subordinate’s initialization timeout has expired:
/* the subordinate processor informs all processors that there is a
/* network failure
/* the Kernel raises
/* subordinate initialization timeout expired exception and
/* control is returned to the Main Unit to inform the external
/* world that initialization has failed
/* POSTCONDITIONS:
/* the network is not functional
/*---------------------------------------------------------*/
procedure Initialization Complete (Timeout After: In Tg Elapsed Time);

/ / SUBPROGRAM NAME
/ / initialization_complete

/ / DESCRIPTION:
/ / asserts that the declaration and creation of all processes on this
/ / processor is complete

/ / -REQ: 7.1.15; 7.1.16; 7.1.17; 7.1.25; 7.1.26
/ /
/ / -PRIM: 16.1.3
/ /
/ / -TIME: 6.2.1
/ /

/ / PARAMETERS:
/ / timeout
/ / duration to wait until this processor asserts that its
/ / initialization has failed
/ / default value:
/ / none

/ / PRECONDITIONS:
/ / all declare_process and create_process calls have been completed for
/ / this processor

/ / ACTIONS:
/ / the Kernel informs all other Kernels that it has completed
/ / initialization
/ / wait for all other nodes to complete initialization (i.e.,
/ / invoke this primitive)
/ / the Kernel checks internal data structures for consistency and
/ / completeness
/ / the Kernel de-schedules the Main Unit
/ / the Kernel Scheduler is given control of the processor

/ / POSTCONDITIONS:
/ / processor-specific logical topology is consistent and complete
/ / Kernel data structures entries are consistent
/ / the Main Unit is no longer a schedulable entity

/ / ERROR PROCESSING:
/ / when the call is detected from a unit that is not the Main Unit:
/ / the Kernel raises calling unit not Main Unit exception

/ / POSTCONDITIONS:
/ / the requested action does not take place
/ / when the Kernel detects that the number of communication partners
/ / is specified.
/ / the Kernel informs all processors that there is a network failure
/ / the Kernel raises process_maximum_exceeded exception

/ / POSTCONDITIONS:
/ / the network is not functional
/ / the Kernel raises remote Kernel process undefined exception

end Generic Processor Management;
pragma (page),

December 1989
MODULE NAME:
generic process_managers_globals

MODULE TYPE:
generic package specification

MODULE PURPOSE:
collection of types used by process_manager subprograms

MODULE DESCRIPTION:
collection of types used by process_manager subprograms

MODULE CONTENTS:
generic process_managers_globals (generic package specification)
device_name_type (type)
how_to_handle_message_queue_overflow (type)
maximum_length_of_process_name (constant)
maximum_length_of_process_name_value (generic formal parameter)
process_name_type (type)

REFERENCES:
DESIGN DOCUMENTS:
Kernel Facilities Definition
-REQ: 10.1.22
-PIM: N/A
-TIME: N/A

USER'S MANUAL:
Kernel User's Manual

TESTING AND VALIDATION:
integration testing with generic_process_managers

NOTES:
one

MODIFICATION HISTORY:
15aug89 banberg created (compiles clean)
23may89 lac Bug 0048 SEI add overwrite oldest message
 enumeration literal to 'overflow type'
27aug89 bmacberg; final doc. check for 01sep89 freeze

VERSION: 3.0

DISCLAIMER:
The following copyright must be included in this software and
all software utilizing this software:

DISTRIBUTION:
Approved for public release; distribution is unlimited.

Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh.

PA. The Software Engineering Institute (SEI) is a federally
funded research and development center established and operated
by Carnegie Mellon University (CMU). Sponsored by the U.S.
Department of Defense under contract F18625-85-C-0003, the SEI
is supported by the services and defense agencies, with the U.S.
Air Force as the executive contracting agent.

Permission to use, copy, modify, or distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation. Further, the names
Software Engineering Institute or Carnegie Mellon University may
not be used in advertising or publicity pertaining to
distribution of the software without specific written prior
permission. CMU makes no claims or representations about the
suitability of this software for any purpose. This software is
provided "as is" and without warranty, express or implied, is made
by the SEI or CMU, as to the accuracy and functioning of the
program and related program material, nor shall the fact of
distribution constitute any such warranty. No responsibility is
assumed by the SEI or CMU in connection herewith.

#pragma (page);
with Hardware_Interface; use Hardware_Interface;
  get: hw natural
  hw_string

generic
  process table (i.e., upper bound on a string)
  note that null strings are allowed, so the application need not carry
  extra string space around with it

Maximum Length Of Process_Name_Value: in Hw_Natural;

Pragma(Page);

-- package short name: GPMG

package Generic_Process_Managers_Globals

is

-- indication of how to handle message queue overflow (should it occur)
-- choices are
-- drop newest message
-- if the incoming message is larger than the available space in the
-- message queue, the incoming message is "dropped," and all required
-- Kernel to Kernel negative acknowledgements are automatically made
-- overwrite oldest message
-- if the message queue is full when a message is received, the oldest
-- message is to be removed from the message queue, and all required
-- Kernel to Kernel negative acknowledgements are automatically made
-- for the removed message.
-- user-selectable option on a per-process basis at process creation time

-- REQ: 10.1.22

type How To Handle Message Queue Overflow is
  Drop Newest Message,
  Overwrite Oldest Message

Maximum Length Of Process_Name: constant Hw Natural :=
  Maximum Length Of Process_Name_Value,
MODULE NAME:

process.managersGlobals

MODULE TYPE:

package specification

MODULE PURPOSE:

an example of (and a template for) an instantiation of the generic

package generic_process_managersGlobals

MODULE DESCRIPTION:

package instantiating the process managers globals information where:

maximum_length_of_process_name_value is defined as 32 characters

MODULE CONTENTS:

process_managersGlobals (package specification)

device_name_type (type)

how_to_handle_message_queue_overflow (type)

maximum_length_of_process_name (constant)

process_name_type (type)

REFERENCES:

DESIGN DOCUMENTS:

Kernel Facilities Definition

USER'S MANUAL:

Kernel User's Manual

TESTING AND VALIDATION:

none

NOTES:

none

MODIFICATION HISTORY:

18aug89 bamberg created (complies clean)

25aug89 bamberg final doc. check for 01sep89 freeze

VERSION, 3.0

DISCLAIMER:

The following copyright must be included in this software and
all software utilizing this software.

DISTRIBUTION:

Approved for public release; distribution is unlimited

Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
PA. The Software Engineering Institute (SEI) is a federally
funded research and development center established and operated
by Carnegie Mellon University (CMU). Sponsored by the U.S.
Department of Defense under contract F19628-85-C-0003, the SEI
is supported by the services and defense agencies, with the U.S.
Air Force as the executive contracting agent.

Permission to use, copy, modify, or distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation. Further, the names
Software Engineering Institute or Carnegie Mellon University may
not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission. CMU makes no claims or representations about the
suitability of this software for any purpose. This software is
provided "as is" and no warranty, express or implied, is made
by the SEI or CMU, as to the accuracy and functioning of the
program and related program material, nor shall the fact of
distribution constitute any such warranty. No responsibility is
assumed by the SEI or CMU in connection herewith.

with Generic Process Managers Globals:
prgama Elaborate (Generic Process ManagersGlobals);

to get: see MODULE CONTENTS

package short name: PMG

package Process ManagersGlobals

is new Generic Process Managers Globals (Maximum Length Of Process Name Value <= 32);

prgama (page):
In declare process:

... if checking for calling unit is not Main Unit is disabled, the
application is asserting that it guarantees the calling unit is the
Main Unit (except for Child of DARK, when this restriction may be
lifted); if the calling unit is NOT the Main Unit and checking is
disabled, currently, a null process ID is returned (this is
required to maintain the integrity of the Kernel)

... in create process:

... if checking for calling unit is not Main Unit is disabled, the
application is asserting that it guarantees the calling unit is the
Main Unit (except for Child of DARK, when this restriction may be
lifted); if the calling unit is NOT the Main Unit and checking is
disabled, currently, this procedure simply returns (this is
required to maintain the integrity of the Kernel)

... if checking for illegal process address is disabled, the application is
asserting that it guarantees the input process address is good;
however, since the Kernel cannot sufficiently protect itself if this
address is really a bad address, it must do the check in all cases;
... if the address is truly a bad address, the Kernel assigns the address
of a null procedure to be process address; the process will then do
nothing, but do without affecting the integrity of the Kernel

... if checking for illegal process Identifier is disabled, the application
is asserting that the input process ID was, in fact, created via a
prior call to declare process; if that is not the case, the input
value is some null value (null or null_process), and the call simply
returns, or it is

... assumed that somewhere within the rest of create process, something
is bound to cause an exceptional event (thus maintaining the integrity
of the Kernel); this latter case is deemed to be "OK" because the
only way this
could happen is if the application used unchecked programming to
call a non-process identifier value into a process ID

... in declare process:

... if checking for insufficient space is disabled, the application is
asserting that there is sufficient storage to allocate the data
structures for the process table entry; if this is not the case and
checking is disabled, a null value is returned

... in create process:

... if checking for insufficient space is disabled, the application is
asserting that there is sufficient space for:

... the process stack; if this is not the case and checking is
disabled, the Kernel assigns the address of a null procedure to
be the process address; the process will then do nothing, but
... do so without affecting the integrity of the Kernel

... the message queue, if the Kernel is not the case and checking is
... disabled, the Kernel assigns the address of a null procedure to
... be the process address; the process will then do nothing, but
... do so without affecting the integrity of the Kernel

... and that there is no overflow (cycling around) of process numbers; if
... overflow occurs and checking is disabled, this procedure simply
... returns (this is required to maintain the integrity of the Kernel)

... if checking for no Kernel process on non KERNEL device is disabled, the
application is asserting that the process to be created is a Kernel
process on a Kernel processor; if this is not the case and checking is
disabled, this procedure simply returns (this is required to maintain
It fat, aU ES 4 ~ c 0 c L
Iu -M c q "

MODIFICATION HISTORY:
18aug88 bambarg created (compiles clean)
27aug88 bambarg final doc. check for 01sep88 freeze

VERSION: 3.0

DISCLAIMER:
The following copyright must be included in this software and all software utilizing this software.

Approved for public release; distribution is unlimited.

Copyright (c) 1988 by Carnegie Mellon University, Pittsburgh, PA. The Software Engineering Institute (SEI) is a federally funded research and development center established and operated by Carnegie Mellon University (CMU). Sponsored by the U.S. Department of Defense under contract F19628-86-C-0003, the SEI is supported by the services and defense agencies, with the U.S. Air Force as the executive contracting agent.

Permission to use, copy, modify, or distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Further, the names of Software Engineering Institute or Carnegie Mellon University may not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. CMU makes no claims or representations about the suitability of this software for any purpose. This software is provided "as is" and no warranty, express or implied, is made by the SEI or CMU, as to the accuracy and functioning of the program and related program material, nor shall the fact of distribution constitute any such warranty. No responsibility is assumed by the SEI or CMU in connection herewith...
with Kernel Exceptions:
  -- to get calling unit not Main Unit
  -- Illegal process address
  -- Illegal process identifier
  -- insufficient space
  -- no Kernel process on non_Kernel_device
  -- process already created
  -- process already exists
  -- unknown_non_Kernel_device

with Hardware Interface; use Hardware.Interface;
  -- hw_long positive
  -- hw_long natural

with Process Managers Globals;
  -- to get device_name type
  -- how to handle message queue overflow
  -- process name type

with Process Table;
  -- to get process identifier

with Schedule Types;
  -- to get default preemption
  -- default priority
  -- preemption
  -- priority

pragma (page);

generic:

Caling Unit Not Main Unit Enabled: In Boolean := True;
Illegal Process Address Enabled: In Boolean := True;
Illegal Process Identifier Enabled: In Boolean := True;
Insufficient Space Enabled: In Boolean := True;
No Kernel Process On Non Kernel Device Enabled: In Boolean := True;
Process Already Created Enabled: In Boolean := True;
Process Already Exists Enabled: In Boolean := True;
Unknown Non Kernel Device Enabled: In Boolean := True;

-- value indicating the maximum size of the incoming message queue in number
-- of bytes (this value is never used within the Kernel except for the
-- initialization of the size of the message queue, nor should it be used
-- without the Kernel)

-- REQ: 5.1.6, (7.1.11)

Maximum Message Queue Size Value: In Hw_Long Natural;

-- value indicating the maximum size of the process stack in number of
-- bytes (this value is never used within the Kernel except for the
-- initialization of the size of the process stack, nor should it be used
-- without the Kernel)

-- REQ: 5.1.6, 7.1.8

Maximum Process Stack Size Value: In Hw_Long Positive;
Pragma(Page);

package Generic Process Managers

is

-- REQ: 8.1.9, 7.1.1; 7.1.2; 7.1.13

-- calling unit not main Unit raised by:
-- declare process
-- create process
-- -PRIM 16.1.1; 16.1.2

Calling Unit Not Main Unit: exception
renames Kernel Exceptions.Calling Unit Not Main Unit;
Illegal process address raised by:
  - create process

--PRIM: 16.1.2

Illegal Process Address exception
  renames Kernel.Exceptions.Illegal_Process_Address;

  ...
  -- Illegal process identifier raised by:
  - create_process

--PRIM: 16.1.2

Illegal Process Identifier exception
  renames Kernel.Exceptions.Illegal_Process_Identifier;

  ...
  -- insufficient space raised by:
  - declare_process
  - create_process

--PRIM: 16.1.1, 16.1.2

Insufficient Space exception
  renames Kernel.Exceptions.Insufficient_Space;

  ...
  -- no Kernel process on non Kernel device raised by
  - create_process

--PRIM: 16.1.2

No Kernel Process On Non Kernel Device exception

  ...
  -- process already created raised by:
  - create_process

--PRIM: 16.1.2

Process Already Created exception
  renames Kernel.Exceptions.Process_Already_Created;

  ...
  -- process already exists raised by:
  - declare_process

--PRIM: 16.1.1

Process Already Exists exception
  renames Kernel.Exceptions.Process_Already_Exists,
function Declare Process (Name, Process Manager, Global Process_Name, Type)
return Process Table_Process_Identifer;
-- ugl-----------------------------
-- SUBPROGRAM NAME:
-- declare_process
--
-- DESCRIPTION:
-- declares all remote Kernel processes with which communication may
-- occur, and any local processes to be created on this processor;
-- associates a logical, user-defined "name" (string valued) with a
-- process ID (internally generated handle) and initialized Kernel data
-- structures
--
-- REQ: 7.1.3, 10.1.1, 10.1.8, 10.1.15
--
-- PRIM: 18.1.1
--
-- TIME: 6.2.1
--
-- PARAMETERS:
-- name
-- user-defined string value by which the process is referred in the
-- user's problem domain space
-- default value: none
--
-- function return value
-- the process ID; the ID (handle) by which this process is
-- referenced in all other Kernel calls
-- default value: none
--
-- ASSUMPTIONS:
-- prior network initialization sequence completed successfully (i.e.,
-- calls to Initialize_"subprocessor and start_subordinates as
-- appropriate)
--
-- PRECONDITIONS:
-- the calling unit is the Ada Main Unit
-- the user specified name has not been declared by a prior
-- invocation of declare_process
--
-- ACTIONS:
-- the Kernel initializes the process Information for the process
-- the Kernel generates a processor unique process ID - the handle by
-- which this process is referenced for all ensuing activities - and
-- returns that handle to the user
--
-- POSTCONDITIONS:
-- updated Kernel data structures
-- processor unique process ID returned to the user

-- ERROR PROCESSING:
-- when the call is detected from a unit that is not the Main Unit:
-- the Kernel raises calling_unit_not_Main_Unit exception
-- POSTCONDITIONS:
-- the process is not declared
-- when there is insufficient memory in which to construct internal
-- Kernel data structures:
-- the Kernel raises Insufficient_space exception
-- POSTCONDITIONS:
-- the process is not declared
-- when the value for the name parameter has appeared in a prior
-- declare process call:
-- the Kernel raises process_already_exists exception
-- POSTCONDITIONS:
-- the process is not declared

pragma (page);
function Declare Process ( 
Name : Process Managers Globals Device Name Type) 
return Process Table Process Identifier; 
// SUBPROGRAM NAME: 
// declare process 
// DESCRIPTION: 
// declares any remote, non-Kernel device with which communication may 
// occur, associates a logical, user-defined "name" (string-valued) 
// with a process ID (internally generated handle) and initialized 
// Kernel data structures 
// -REQ: 7.1.1; 10.1.1; 10.1.6; 10.1.15, 10.1.26 
// -PRIM: 10.1.1 
// -TIME: 6.2.1 
// PARAMETERS: 
// name 
// user-defined string value by which the non-Kernel device is 
// referred in the user's problem domain space 
// default value: 
// none 
// function return value 
// the process ID, the ID (handle) by which this process is 
// referenced in all other Kernel calls 
// default value: 
// none 
// ASSUMPTIONS 
// prior network initialization sequence completed successfully (i.e., 
// calls to initialize processor and start subordinates as 
// appropriate) 
// PRECONDITIONS: 
// the calling unit is the Ada Main Unit 
// the indicated non-Kernel device is a valid non-Kernel device (i.e., 
// has been identified in the NCT) 
// the user-specified name has not been declared by a prior 
// invocation of declare process 
// ACTIONS: 
// the Kernel initializes the process information for the process 
// the Kernel generates a processor unique process ID - the handle by 
// which this process is referenced for all ensuing activities - and 
// returns that handle to the user 
// POSTCONDITIONS: 
// updated Kernel data structures 
// processor-unique process ID returned to the user 

--/ ERROR PROCESSING: 
--/ when the call is detected from a unit that is not the Main Unit: 
--/ the Kernel raises calling_unit_not_Main_Unit exception 
--/ POSTCONDITIONS: 
--/ the process is not declared 
--/ when there is insufficient memory in which to construct internal 
--/ Kernel data structures: 
--/ the Kernel raises insufficient_space exception 
--/ POSTCONDITIONS: 
--/ the process is not declared 
--/ when the value for the name parameter has appeared in a prior 
--/ declare process call: 
--/ the Kernel raises process_already_exists exception 
--/ POSTCONDITIONS: 
--/ the process is not declared 
--/ when the communication partner being declared is a non-Kernel device 
--/ and that non-Kernel device is not known as such (i.e., there is no 
--/ entry for it in the NCT): 
--/ the Kernel raises unknown_non_Kernel_device 
--/ POSTCONDITIONS: 
--/ the communication partner is not declared 

---
procedure Create Process (;
    ProcessId: Process Table.Process Identifier;
    Address: In Hw Address;
    Stack Size: In Hw Long, Positive = Maximum Process Stack Size Value;
    Message Queue Size: In Hw Long, Natural = Maximum Message Queue Size Value;
    Message Queue Overflow Handling: In
    Process Managers Globals Drop Newest Message,
    Initial Priority: In Schedule Types.Priority =
    Schedule Types Default Priority,
    Preemptable: In Schedule Types.Preemption =
    Schedule Types Default Preemption);
    /-****************************************************************************-
    / SUBPROGRAM NAME:
    /- create process
    /- DESCRIPTION:
    /- creates an independent thread of control, creates the execution
    /- environment in which the thread is to execute, this thread is called
    /- a process
    /- REQ. 7.14; 7.15; 7.16; 7.17; 7.18; 7.19; 7.1.10
    /- REQ. 7.11; 7.12; 7.25; 7.26; 7.27
    /- REQ. 9.1.1; 9.1.2; 9.1.7; 9.1.11; 10.1.22; 10.1.23
    /- PRINT: 16.1.2
    /- TIME: 6.2.1; 7.2.1; 7.2.5; 7.2.6
    /- PARAMETERS:
    /- process ID
    /- the handle by which this process is referenced; generated by a
    /- previous call to declare process
    /- default value:
    /- none
    /- address
    /- a value of hw address that corresponds to the 'address attribute
    /- of the procedure being identified as a Kernel process
    /- default value:
    /- none
    /- stack size
    /- the maximum number of bytes reserved for process local storage
    /- (i.e., storage for the non-Kernel portion of the runtime stack
    /- for the process - as generated by the Ada compiler)
    /- default value:
    /- maximum process stack size value, a user-specified maximum
    /- message queue size
    /- the maximum number of bytes reserved for incoming messages to this
    /- process from all other processes
    /- default value:
    /- maximum process queue size value, a user-specified maximum
    /- message queue overflow handling
    /- an indication of how overflow of the incoming message queue is to
    /- be handled for this process
    /- values include:
    /- drop newest_message (no change to the contents of the message
    /- queue; messages existing in message queue are preserved;
    /- the newest message is simply dropped)
    /- others TBD (thus, the only option right now is to drop the
    /- newest message)
    /- default value:
    /- drop newest_message (messages existing in message queue are
    /- preserved)
    /- Initial priority
    /- the priority at which this process is initially registered with
    /- the Scheduler
    /- default value:
    /- the Kernel-defined default priority (the lowest priority
    /- available)
    /- preemptable
    /- indication of whether or not a process may be preempted by
    /- another process at the same priority
    /- values include:
    /- enabled (the process is preemptable)
    /- disabled (the process is not preemptable)
    /- default value:
    /- the Kernel-defined default preemption (enabled, preemptable)
    /- ASSUMPTIONS:
    /- all parameters have legal values
    /- PRECONDITIONS:
    /- declare process was previously called to create a legal process, ID
    /- calling unit is the Ada Main Unit
    /- this Kernel process is being created on a Kernel processor
    /- this process is not already created
    /- the address provided is a legal address
    /- there is sufficient space to create this process and its execution
    /- environment
    /- ACTIONS:
    /- updates Kernel data structures, indicating the creation of this
    /- process
    /- constructs the execution environment for this process, including:
the local process stack (stack plug, dummy call frame, space or local process variables), and the process control block (message queue, schedule attributes, process code, and context save area pointers)

the local Kernel informs all other Kernels about the creation of this process

POSTCONDITIONS:
- Kernel data structures updated to reflect the creation of this process
- Initial process characteristics recorded in appropriate internal data structures
- Execution environment created for this process
- This process is registered as “suspended” with the Kernel Scheduler and is queued with all other processes of the same priority level
- To begin execution at the discretion of the Scheduler
- All other Kernels informed about the existence of this process

ERROR PROCESSING:
- When the call is detected from a unit that is not the Main Unit:
  - The Kernel raises calling_unit_not_Main_Unit exception

POSTCONDITIONS:
- The process is not created
- When the process ID cannot be found within Kernel data structures or is a null value:
  - The Kernel raises illegal_process_ID exception

POSTCONDITIONS:
- The process is not created
- When there is insufficient memory in which to construct the execution environment for the process being created (process number wrap around, not enough room for the process stack or the incoming message queue):
  - The Kernel raises insufficient_space exception

POSTCONDITIONS:
- The process is not created
- When the creation of a Kernel process on a non-Kernel device is detected:
  - The Kernel raises no_Kernel_process_on_non_Kernel_device exception

POSTCONDITIONS:
- The process is not created
- When a process has previously been created for this process_ID value:
  - The Kernel raises process_already_created exception

POSTCONDITIONS:
- The process is not created

and Generic Process Managers;

---

MODULE NAME:
- process_managers

MODULE TYPE:
- package specification

MODULE PURPOSE:
- an example of (and a template for) an instantiation of the generic package generic_process_managers

---

MODULE DESCRIPTION:
- package instantiating the declaration and creation of the logical processor configuration capabilities where:
  - error checking for calling_unit_not_Main_Unit is enabled
  - error checking for illegal_process_ID is enabled
  - error checking for insufficient_space is enabled
  - error checking for no_Kernel_process_on_non_Kernel_device is enabled
  - error checking for process_already_created is enabled
  - error checking for process_alreadyExists is enabled
  - error checking for unknown_non_Kernel_device is enabled
  - maximum_message_queue_size_value is defined as 1024
  - maximum_process_stack_size_value is defined as 4096

---

MODULE CONTENTS:
- process_managers(package specification)
  - calling_unit_not_Main_Unit(exception)
  - create_process(procedure)
  - declare_process overloaded(2) function
  - illegal_process_ID(exception)
  - illegal_process_ID(exception)
  - insufficient_space(exception)
  - no_Kernel_process_on_non_Kernel_device(exception)
  - process_already_created(exception)
  - process_alreadyExists(exception)
  - unknown_non_Kernel_device(exception)

---

REFERENCES:
- DESIGN DOCUMENTS:
  - none

- USER'S MANUAL:

- TESTING AND VALIDATION:
  - none

- NOTES:
  - none

---

MODIFICATION HISTORY:
- 18aug88 bamberg created (compiles clean)

---

...
I

/* 27aug89 bmbwv  final doc. check for 01sep89 freeze */

/* VERSION: 3.0 */

/* DISCLAIMER: */

/* The following copyright must be included in this software and */
/* all software utilizing this software */

/* DISTRIBUTION: */

/* Approved for public release; distribution is unlimited */

/* Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh, */
/* PA. The Software Engineering Institute (SEI) is a federally */
/* funded research and development center established and operated */
/* by Carnegie Mellon University (CMU). Sponsored by the U.S. */
/* Department of Defense under contract F19628-85-C-0003, the SEI */
/* is supported by the services and defense agencies, with the U.S. */
/* Air Force as the executive contracting agent. */

/* Permission to use, copy, modify, or distribute this software and */
/* its documentation for any purpose and without fee is hereby */
/* granted, provided that the above copyright notice appear in all */
/* copies and that both that copyright notice and this permission */
/* notice appear in supporting documentation. Further, the names */
/* Software Engineering Institute or Carnegie Mellon University may */
/* not be used in advertising or publicity pertaining to */
/* distribution of the software without specific, written prior */
/* permission. CMU makes no claims or representations about the */
/* suitability of this software for any purpose. This software is */
/* provided "as is" and no warranty, express or implied, is made */
/* by the SEI or CMU, as to the accuracy and functioning of the */
/* program and related program material. Nor shall the fact of */
/* distribution constitute any such warranty. No responsibility is */
/* assumed by the SEI or CMU in connection herewith. */

/* with Generic Process Managers; */
/* pragma Elaborate (Generic Process Managers); */
/* to get: see MODULE CONTENTS */

/* package short name: PM */

/* package Process Managers */
PA. The Software Engineering Institute (SEI) is a federally
funded research and development center established and operated
by Carnegie Mellon University (CMU). Sponsored by the U.S.
Department of Defense under contract F19628-85-C-0003, the SEI
is supported by the services and defense agencies, with the U.S.
Air Force as the executive contracting agent.

Permission to use, copy, modify, or distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation. Further, the names
Software Engineering Institute or Carnegie Mellon University may
not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission. CMU makes no claims or representations about the
suitability of this software for any purpose. This software is
provided "as is" and no warranty, express or implied, is made
by the SEI or CMU, as to the accuracy and functioning of the
program and related program material, nor shall the fact of
distribution constitute any such warranty. No responsibility is
assumed by the SEI or CMU in connection herewith.

/* module name: generic communicationglobals */
/* module type: generic package specification */
/* module purpose: exports communication global information */
/* module description: collection of globals related to communication management but which can be used generally by anyone */
/* module contents: generic communicationglobals (generic package specification)
   maximum message length (constant)
   maximum message length value (generic formal parameter)
   message length type (type)
   message tag type (type) */
/* references: */
/* design documents: */
/* kernel facilities definition */
/* primary 5.1.4, 5.1.7 */
/* primary N/A */
/* primary N/A */
/* user's manual: */
/* kernel user's manual */
/* testing and validation: */
/* integration testing with generic communication management */
/* notes: */
/* none */
/* modification history: */
/* 8/8/88 barger created (complies clean) */
/* version: 30 */
/* disclaimer: */
/* the following copyright must be included in this software and all software utilizing this software */
/* distribution */
/* approved for public release, distribution is unlimited */
/* copyright (c) 1989 by Carnegie Mellon University, Pittsburgh, PA. */
with Hardware Interface; see Hardware_Interface;
  -- to get: hw_integer
  -- hw_natural

generic
  -- the maximum length of a message that can be sent across the network

Maximum_Message_Length_Value := \text{In Hw Natural};

-- package short name: GCG
-- package specification

package Generic_Communication_Globals
is
  -- the maximum length of a message that can be sent across the network

Maximum_Message_Length := constant \text{Hw_Natural} :=
  Maximum_Message_Length_Value;
  -- this type defines the limits of message length

type Message_Length_Type is new Hw_Natural;
  -- this type defines the limits of message tags

type Message_Tag_Type is new Hw_Integer;

end Generic_Communication_Globals;

pragma (page):
/* MODULE NAME: */
generic_communication_management

/* MODULE TYPE: */
generic_package_specification

/* MODULE PURPOSE: */
support communication among Kernel processes

-----------------------------------------------
/* MODULE DESCRIPTION: */
collection of subprograms to send and receive messages among processes

/* MODULE CONTENTS: */
generic_communication_management (generic package specification)
allocate_device_receiver (procedure)
buffer too small for message (exception)
buffer too small for message enabled (generic formal parameter)
illegal context for call (exception)
illegal context for call enabled (generic formal parameter)
illegal process identifier (exception)
illegal process identifier enabled (generic formal parameter)
message not received (exception)
message not received enabled (generic formal parameter)
message timed out (exception)
message timed out enabled (generic formal parameter)
network failure (exception)
network failure enabled (generic formal parameter)
no message available (exception)
o message available enabled (generic formal parameter)
o such device exists (exception)
o such device exists enabled (generic formal parameter)
receive_message (overloaded (3) procedure)
receiver dead (exception)
receiver dead enabled (generic formal parameter)
receiver is sender (exception)
receiver is sender enabled (generic formal parameter)
receiver never existed (exception)
receiver never existed enabled (generic formal parameter)
replacing previous allocation (exception)
replacing previous allocation enabled (generic formal parameter)
send_message (procedure)
send message and wait (overloaded (3) procedure)

/***** REFERENCES *****/
/ DESIGN DOCUMENTS /
/ Kernel Facilities Definition /
/ - REQ: 6.1.9, 6.1.11, 10.1.19, 10.1.22, 10.1.23, 10.1.28, 10.1.29, 10.1.30, 10.1.31, 10.1.33, 10.1.34 /
/ - PRIM: N/A /
/ - TIME: N/A /
/ USER'S MANUAL: 
/ Kernel User's Manual
with Kernel Exceptions;
    \begin{verbatim}
    : to get buffer too small for message
    \end{verbatim}
    \begin{verbatim}
    illegal context for call
    illegal process identifier
    message not received
    message timed out
    network failure
    no message available
    no such device exists
    receiver dead
    receiver is sender
    receiver never existed
    replacing previous allocation
    \end{verbatim}

with Hardware Interface; use Hardware Interface;
    \begin{verbatim}
    : to get hw address
    null_hw_address
    \end{verbatim}

with Communication globals;
    \begin{verbatim}
    : to get message length type
    message_tag_type
    \end{verbatim}

with Network Globals;
    \begin{verbatim}
    : to get processor identifier
    \end{verbatim}

with Process Table;
    \begin{verbatim}
    : to get process identifier
    \end{verbatim}

with Schedule Types;
    \begin{verbatim}
    : to get current process priority
    priority
    \end{verbatim}

with Time Globals;
    \begin{verbatim}
    : to get elapsed_time
    epoch_time
    to_kernel_time
    \end{verbatim}

\begin{verbatim}
generic

    : ERROR checking conditional compilation flags

    \end{verbatim}

\begin{verbatim}
REQ: 5.1.8

Buffer Too Small For Message: exception
Receiver Never Existed Enabled: In Boolean := True;
Replacing_Previous_Allocation Enabled: In Boolean := True;
\end{verbatim}

\begin{verbatim}
Receiver Never Existed Enabled: In Boolean := True;
Replacing_Previous_Allocation Enabled: In Boolean := True;
\end{verbatim}

\begin{verbatim}
REMARK: 6.1.9, 6.1.11;
REMARK: 10.1.19, 10.1.22, 10.1.23, 10.1.28, 10.1.29, 10.1.30, 10.1.31
\end{verbatim}

\begin{verbatim}
buffer too small for message re-sized by;
receive message
receive message (waiting for)
receive message (waiting until)
\end{verbatim}

\begin{verbatim}
PRIM: 19.1.3
\end{verbatim}

\begin{verbatim}
Buffer Too Small For Message: exception
renames Ke.Buffer Too Small For Message;
\end{verbatim}

\begin{verbatim}
\begin{verbatim}
\end{verbatim}

\begin{verbatim}
illegal context for call raised by;
send message and wait
send message and wait (for)
send message and wait (until)
receive message
receive message (waiting for)
receive message (waiting until)
\end{verbatim}

\begin{verbatim}
PRIM: 19.1.2, 19.1.3
\end{verbatim}

\begin{verbatim}
\end{verbatim}
Illegal Context For Call: exception
  rename to Illegal_Context_For_Call;

  illegal process identifier raised by:
    send message
    send message and wait
    send message and wait (for)
    send message and wait (until)
    allocate device receiver

   -PRIM: 10.1.31

Illegal Process Identifier: exception
  rename to Illegal_Process_Identifier;

  message not received raised by:
    send message and wait
    send message and wait (for)
    send message and wait (until)

   -PRIM: 19.1.2

Message Not Received: exception
  rename to Message_Not_Received;

  message timed out raised by:
    send message and wait (for)
    send message and wait (until)
    receive_message (waiting for)
    receive_message (waiting until)

   -PRIM: 19.1.2; 19.1.3

Message Timed Out: exception
  rename to Message_Timed_Out;

    network failure raised by:
      send message and wait
      send message and wait (for)
      send message and wait (until)

   -PRIM: 19.1.2

Network Failure: exception
  rename to Network_Failure;

  no message available raised by:
    receive message (waiting for)
    receive message (waiting until)

   -PRIM: 19.1.3

No Message Available: exception
  rename to No_Message_Available;

  no such device exists raised by:
    allocate_device receiver

   -PRIM: 16.1.4

No Such Device Exists: exception
  rename to No_Such_Device_Exists;

  receiver dead raised by:
    send message
    send message and wait
    send message and wait (for)
    send message and wait (until)

   -PRIM: 19.1.1; 19.1.2

Receiver Dead: exception
  rename to Receiver_Dead;

  receiver is sender raised by:
    send message
    send message and wait
    send message and wait (for)
    send message and wait (until)

   -PRIM: 19.1.2

Receiver Is Sender: exception
  rename to Receiver_Is_Sender;

  receiver never existed raised by:
    send message
    send message and wait
    send message and wait (for)
    send message and wait (until)

   -PRIM: 19.1.1; 19.1.2
procedure Send_Message ;

  Receiver : in Pr.Sender ;
  Message : in Pkg.Message ;

  if Receiver = NULL then
    -- return
  else
    
    // Subprogram Name: Send_Message
    // Description:
    // - this subprogram sends a message to a specified receiver
    // - there is no waiting for any kind of acknowledgement by the receiver
    // - REQ: 13.1.2, 13.2.2; 10.1.2, 10.1.5, 10.1.6, 10.1.25; 10.1.28
    // - TIME: 14.2.3, 14.2.5, 14.2.6
    // - PARAMETERS:
    // - receiver
    // - the local process ID of the intended receiver of this message
    // - default value: none
    // - message tag
    // - an application-defined tag that allows the receiver to interpret
    // - the message in the format the sender intended
    // - default value: none
    // - message length
    // - the number of bytes of text that should be interpreted as
    // - containing message-specific data
    // - default value: 0
    // - message text
    // - the address at which the message-specific data begin
    // - specified by message_text
    // - default value: NULL
    // - PRECONDITIONS:
    // - NONE

  end Send_Message ;
procedure Send Message And Wait {
    Receiver: In Pid Process Identifier;
    Message Tag: In Cg Message Tag Type;
    Message Length: In Cg Message Length Type := 0;
    Message Text: In Hw Address := Null Hw Address;
    Resumption Priority: In St Priority := St Current Process Priority;
}

#pragma (page);
when the local Kernel detects that the specified receiver is dead:
  
  the Kernel raises receiver_dead exception
  
  POSTCONDITIONS:
  
  the send does not occur
  
  the priority of this process is its own priority
  
  the process is blocked and is either suspended or running
  
  depending on the discretion of the Kernel Scheduler
  
  when a process specifies itself as the intended receiver:
  
  the Kernel raises receiver_is_sender exception
  
  POSTCONDITIONS:
  
  the send does not occur
  
  the priority of this process is its own priority
  
  the process is blocked and is either suspended or running
  
  depending on the discretion of the Kernel Scheduler
  
  when the local Kernel has no knowledge of the specified receiver:
  
  the Kernel raises receiver_never_existed exception
  
  POSTCONDITIONS:
  
  the send does not occur
  
  the priority of this process is its own priority
  
  the process is blocked and is either suspended or running
  
  depending on the discretion of the Kernel Scheduler

postm (page);
procedure Send Message And Wait ( 
  Receiver: In PtB Process Identifier; 
  Message Tag: In Cg MESSAGE_TAG_TYPE; 
  Message Length: In Cg MESSAGE_LENGTH_TYPE = 0; 
  Message Text: In Hw Address = null Hw Address; 
  Timeout After: In Tg Elapsed Time; 
  Resumption Priority: In St.Priority = St.Current Process Priority); 

  /* SUBPROGRAM NAME: */ 
  /* send_message_and_wait */ 
  /* DESCRIPTION: */ 
  /* does a synchronous send of a message to a specified receiver; the */ 
  /* calling process blocks until a Kernel-to-Kernel acknowledgement of */ 
  /* message receipt (or lack thereof) by the receiving process is */ 
  /* returned or until the specified elapsed time has elapsed */ 
  /* */ 
  /* - REQ: 7.1.21, 7.1.22, 7.1.25, 7.1.26 */ 
  /* - REQ: 9.1.8, 9.1.8, 9.1.22, 10.1.2 */ 
  /* - REQ: 10.1.3, 10.1.4, 10.1.5, 10.1.6, 10.1.7, 10.1.9; */ 
  /* - REQ: 10.1.10, 10.1.11, 10.1.25, 10.1.27, 10.1.32 */ 
  /* - REQ: 11.1.12, 14.1.5 */ 
  /* */ 
  /* - PRM: 10.1.31, 18.1.2 */ 
  /* - */ 
  /* - TIME: 10.2.2, 10.2.4, 10.2.7, 10.2.8, 10.2.11 */ 
  /* - */ 
  /* - TIME: 14.2.3, 14.2.5, 14.2.6 */ 
  /* */ 
  /* PARAMETERS: */ 
  /* receiver */ 
  /* the process ID of the intended receiver of this message */ 
  /* */ 
  /* - default value: */ 
  /* none */ 
  /* */ 
  /* message tag */ 
  /* an application-defined tag that allows the receiver to interpret */ 
  /* the message in the format the sender intended */ 
  /* */ 
  /* - default value: */ 
  /* none */ 
  /* */ 
  /* message length */ 
  /* the number of bytes of text that should be interpreted as */ 
  /* containing message-specific data */ 
  /* */ 
  /* - default value: */ 
  /* 0 */ 
  /* */ 
  /* message text */ 
  /* the address at which the message-specific data begins; the Kernel */ 
  /* collects message length bytes of data beginning at the address */ 
  /* specified by message_text */ 
  /* */ 
  /* - default value: */ 
  /* null_hw_address */ 
  /* */ 
  /* timeout after */ 
  /* duration after which the attempt to send is abandoned and the */ 
  /* process becomes unblocked */ 
  /* */ 
  /* - default value: */ 
  /* none */ 
  /* */ 
  /* resumption priority */ 
  /* the priority to be assigned to this process when the */ 
  /* acknowledgement is received (i.e., the process becomes unblocked) */ 
  /* */ 
  /* - default value: */ 
  /* the calling process's current priority */ 
  /* */ 
  /* PRECONDITIONS: */ 
  /* invoked from a non-blocking context */ 
  /* the receiver is not the sender */ 
  /* */ 
  /* ACTIONS: */ 
  /* the Kernel validates the receiver field */ 
  /* the Kernel indicates that a Kernel-to-Kernel acknowledgement is */ 
  /* required when the receiver receives the message */ 
  /* the Kernel bundles the elapsed time value with the message (i.e., the */ 
  /* timeout is a timeout at the receiver's site and measured with */ 
  /* respect to the remote processor's clock) */ 
  /* the Kernel sends the message from the caller to the specified */ 
  /* receiver using local optimization if possible */ 
  /* the calling process blocks until a Kernel-to-Kernel positive */ 
  /* acknowledgement is returned, indicating that the message was */ 
  /* received by the receiver */ 
  /* the Kernel possibly changes the process priority */ 
  /* */ 
  /* POSTCONDITIONS: */ 
  /* the message is sent to the intended receiver */ 
  /* the priority of this process is its resumption priority */ 
  /* the process is unblocked and is either suspended or running depending */ 
  /* on the discretion of the Kernel Scheduler */ 
  /* */ 
  /* ERROR PROCESSING: */ 
  /* when send_message_and_wait is called within an interrupt handler: */ 
  /* the Kernel raises illegal_context_for_call */ 
  /* POSTCONDITIONS: */ 
  /* the call is rejected, no message is sent; no change in priority */ 
  /* is registered */ 
  /* when this primitive is called with an undeclared process ID */ 
  /* the Kernel raises illegal_process_identifier exception */ 
  /* POSTCONDITIONS: */ 
  /* the call is rejected, no message is sent; no change in priority */ 
  /* is registered */ 
  /* when the specified receiver of the message does not actually receive */ 
  /* the message (e.g., the message is dropped or the process dies); */ 
  /* the Kernel raises message_not_received exception */ 
  /* POSTCONDITIONS: */
procedure Send Message And Wait (  
Receiver : In Pib.Process-Identifier;  
Message Tag : In Cg.Message.Tag.Type;  
Message Length : In Cg.Message.Length.Type := 0;  
Message Text : In Hw.Address := Null.Hw.Address ;  
Timeout_At : In Tg.Epoch.Time;  
Resumption_Priority : In St.Priority := St.Current.Process.Priority);  
--------------------------------------------------------------------------  
SUBPROGRAM_NAME:  
/ send, message, and wait  
/--------------------------------------------------------------------------  
/ DESCRIPTION:  
/ does a synchronous send of a message to a specified receiver; the  
/ calling process blocks until a Kernel-to-Kernel acknowledgement of  
/ message receipt (or lack thereof) by the receiving process is  
/ returned or until the specified `epoch (absolute) time has arrived  
/  
/ / -REQ: 7.1.21; 7.1.22; 7.1.25; 7.1.26  
/ / -REQ: 9.1.6; 9.1.8; 9.1.22; 10.1.2  
/ / -REQ: 10.1.5; 10.1.6; 10.1.10; 10.1.11  
/ / -REQ: 10.1.25, 10.1.27, 10.1.32  
/ / -REQ: 11.1.12; 14.1.5  
/ / -PRIM: 10.1.31; 10.1.2  
/ / -TIME: 10.2.3; 10.2.4; 10.2.7; 10.2.8; 10.2.11  
/ / -TIME: 14.2.3, 14.2.5, 14.2.6  
/--------------------------------------------------------------------------  
/ PARAMETERS:  
/ / receiver  
/ / the process ID of the intended receiver of this message  
/ / default value:  
/ / none  
/ / message tag  
/ / an application-defined tag that allows the receiver to interpret  
/ / the message in the format the sender intended  
/ / default value:  
/ / none  
/ / message length  
/ / the number of bytes of text that should be interpreted as  
/ / containing message-specific data  
/ / default value:  
/ / 0  
/ / message text  
/ / the address at which the message-specific data begin, the Kernel  
/ / collects message length bytes of data beginning at the address  
/ / specified by message text  
/ / default value:  

pragma (page).
null_hv_address

- timeout at
- absolute time at which the attempt to send is abandoned and the
- process becomes unblocked
- default value: none
- resumption priority
- the priority to be assigned to this process when the
- acknowledgement is received (i.e., the process becomes unblocked)
- default value: the calling process's current priority

-- PRECONDITIONS:
- invoked from a non-blocking context
- the receiver is not the sender

-- ACTIONS:
- the Kernel validates the receiver field
- the Kernel indicates that a Kernel-to-Kernel acknowledgement is
  required when the receiver receives the message
- the Kernel bundles the elapsed time until the specified epoch, time
  value (i.e., the local Kernel converts "until epoch time" to the
  appropriate elapsed time duration) with the message (i.e., the
  timeout is a timeout at the receiver's site and measured with
  respect to the remote processor's clock)
- the Kernel sends the message from the caller to the specified
  receiver using local optimization if possible
- the calling process blocks until a Kernel-to-Kernel positive
- acknowledgement is returned, indicating that the message was
  received by the receiver
- the Kernel possibly changes the process priority

-- POSTCONDITIONS:
- the message is sent to the intended receiver
- the priority of this process is its resumption priority
- the process is unblocked and is either suspended or running depending
  on the discretion of the Kernel Scheduler

-- ERROR PROCESSING:
- when send message and wait is called within an interrupt handler:
- the Kernel raises illegal_context_for_call

-- POSTCONDITIONS:
- the call is rejected; no message is sent; no change in priority
- is registered
- when this primitive is called with an undeclared process ID
- the Kernel raises illegal_process_identifier_exception

-- POSTCONDITIONS:
- the call is rejected; no message is sent; no change in priority
- is registered
- when the specified receiver of the message does not actually receive
- the message (e.g., the message is dropped or the process dies):

- the Kernel raises message_not_received_exception
- POSTCONDITIONS:
- the send message is never received by the receiver
- the priority of this process is its resumption priority
- the process is unblocked and is either suspended or running
- depending on the discretion of the Kernel Scheduler
- when the remote timeout expires
- the Kernel raises message_timed_out_exception

-- POSTCONDITIONS:
- the send message is never received by the receiver
- the priority of this process is its resumption priority
- the process is unblocked and is either suspended or running
- depending on the discretion of the Kernel Scheduler
- when a network problem is detected:
- the Kernel raises network_failure_exception

-- POSTCONDITIONS:
- the send does not occur
- the priority of this process is its resumption priority
- the process is unblocked and is either suspended or running
- depending on the discretion of the Kernel Scheduler
- when the local Kernel detects that the specified receiver is dead:
- the Kernel raises receiver_dead_exception

-- POSTCONDITIONS:
- the send does not occur
- the priority of this process is its resumption priority
- the process is unblocked and is either suspended or running
- depending on the discretion of the Kernel Scheduler
- when a process specifies itself as the intended receiver:
- the Kernel raises receiver_is_sender_exception

-- POSTCONDITIONS:
- the send does not occur
- the priority of this process is its resumption priority
- the process is unblocked and is either suspended or running
- depending on the discretion of the Kernel Scheduler
- when the local Kernel has no knowledge of the specified receiver:
- the Kernel raises receiver_never_existed_exception

-- POSTCONDITIONS:
- the send does not occur
- the priority of this process is its resumption priority
- the process is unblocked and is either suspended or running
- depending on the discretion of the Kernel Scheduler

--======================================================================

grams (page);
procedureReceiveMessage(
  Sender: out Pcb, Process Identifier;
  Message Tag: out CgMessageTagType;
  Message Length: out CgMessageLengthType;
  Message Buffer: in Hw.Address;
  Buffer Size: in CgMessageLengthType;
  Messages Lost: out Boolean);

buffer size
the number of bytes of text into which message-specific data are placed
default value:
none
resumption priority
the priority to be assigned to this process when it becomes unblocked
default value:
the calling process's current priority
messages lost
an indication of whether or not the incoming message queue is sufficiently full that newly arriving messages were unable to fit into it in a non-destructive manner before the current receive...values include:
true (messages have been lost)
false (messages have not been lost)

PRECONDITIONS:
NONE

POSTCONDITIONS:
the calling process has the message text and message attributes
the calling process has an indication of whether a previous message was lost owing to the incoming message queue becoming full
the message is removed from the receiver's incoming message queue
the Kernel to *Kernel acknowledgement has been sent to the message sender if required
the priority of this process is its resumption priority
if a message was available, the process is either suspended or running depending on the discretion of the Kernel Scheduler
otherwise (if the proc has been blocked waiting for a message)
the process becomes unblocked and is either suspended or running depending on the discretion of the Kernel Scheduler

ERROR PROCESSING:
procedure Receive Message (  
  Sender : out PtBProcessIdentifier;  
  Message Tag : out Cg.Message.Tag Type;  
  Message Length : out Cg.Message.Length Type;  
  Message Buffer : in Hw Address;  
  Buffer Size: in Cg.Message.Length Type;  
  Timeout Alter : in TgL.Elapsed.Time;  
  Messages Lost : out Boolean);  
-------------------------------------------------------------------------------------  
/\ SUBPROGRAM NAME:  
/\ receive_message  
/\  
/\ DESCRIPTION:  
/\ a receive of a message from its message queue; the calling  
/\ process blocks until a message is available for receipt or until the  
/\ specified elapsed time has elapsed  
/\  
/\ - REQ: 9.1.6; 9.1.8; 9.1.22  
/\ - REQ: 10.1.12; 10.1.13; 10.1.14; 10.1.15; 10.1.16; 10.1.18  
/\ - REQ: 10.1.20; 10.1.21; 10.1.24; 10.1.26; 10.1.35; 10.1.36  
/\ - REQ: 11.1.12; 14.1.9  
/\  
/\ - PRM: 19.1.3  
/\  
/\ - TIME: 10.2.9; 10.2.10  
/\ - TIME: 14.2.3, 14.2.5, 14.2.6  
/\  
/\ - PARAMETERS:  
/\  
/\ - sender  
/\  
/\ - the process ID of the sender of this message  
/\  
/\ - default value:  
/\  
/\ - none  
/\  
/\ - message_tag  
/\  
/\ - an application-defined tag that allows the receiver to interpret  
/\ the message in the format the sender intended  
/\  
/\ - default value:  
/\  
/\ - none  
/\  
/\ - message_length  
/\  
/\ - the number of bytes of text that should be interpreted as  
/\ containing message-specific data  
/\  
/\ - default value:  
/\  
/\ - none  
/\  
/\ - message_buffer  
/\  
/\ - the address at which the message-specific data are to be put,  
/\ the Kernel places no more than buffer-size bytes of data beginning  
/\ at the address specified by message_buffer  
/\  
/\ - default value:
The calling process has an indication of whether a previous message was lost owing to the incoming message queue becoming full.

The message is removed from the receiver's incoming message queue.

The Kernel-to-Kernel acknowledgement has been sent to the message sender.

The priority of this process is its resumption priority.

If a message was available, the process is either suspended or running depending on the discretion of the Kernel Scheduler.

Otherwise, if the process had been blocked waiting for a message, the process becomes unblocked and is either suspended or running depending on the discretion of the Kernel Scheduler.

If the buffer size parameter is too small (i.e., it is smaller than the message length), the Kernel raises buffer too_small_for_message exception.

POSTCONDITIONS:

- The message "envelope" (i.e., its attributes: sender, message tag, message length) is returned, no message text is returned.
- The calling process has an indication of whether a previous message was lost owing to the incoming message queue becoming full.
- The message is removed from the receiver's incoming message queue.
- The Kernel-to-Kernel negative acknowledgement has been sent to the message sender if required.
- The priority of this process is its resumption priority.
- If a message was available, the process is either suspended or running depending on the discretion of the Kernel Scheduler; otherwise, if the process had been blocked waiting for a message, the process becomes unblocked and is either suspended or running depending on the discretion of the Kernel Scheduler.

When receive message is called within an interrupt handler:

- The Kernel raises illegal_context_for_call exception.

POSTCONDITIONS:

- The call is rejected; no message is received; no change in priority is registered.
- When the receive timeout expires:
  - The Kernel raises message_timed_out exception.
  - No receive is done (there is nothing to receive within the specified elapsed time).
- The priority of this process is its resumption priority.
- When the timeout value is less than or equal to zero and there is no message in the receiver's incoming message queue:
  - The Kernel raises no_message_available exception.
  - No receive is done (there is nothing to receive within the specified elapsed time).
- The priority of this process is its resumption priority.

NOTES:

When a receive fails for any reason (expiry of time out, etc.)
I

```
procedure Receive Message (
  Sender : out PtB.Process_ID;
  Message_Tag : out Cg.Message_Tag_Type;
  Message_Length : out Cg.Message_Length_Type;
  Message_Buffer : in Hw.Address;
  Buffer_Size : in Cg.Message_Length_Type;
  Timeout_Ax : in Tg.Epoch.Time;
  Messages_Lost : out Boolean);
```

```
pragma (page);
```
./buffer_size
./the number of bytes of text into which message-specific data are placed
./default value: none
./timeout at
./absolute time at which the process resinds its attempt to receive a message and becomes unblocked
./default value: none
./resumption_priority
./the priority to be assigned to this process when it becomes unblocked
./default value: the calling process's current priority
./messages_lost
./an indication of whether or not the incoming message queue is sufficiently full that newly arriving messages were unable to fit into it in a non-destructive manner before the current receive
./values include:
./true (messages have been lost)
./false (messages have not been lost)

/PRECONDITIONS:
./NONE

/DESCRIPTION:
./the calling process has an indication of whether a previous message was lost owing to the incoming message queue becoming full
./the message is removed from the receiver's incoming message queue
./the Kernel-to-Kernel acknowledgement has been sent to the message sender
./the priority of this process is its resumption priority
./If a message was available, the process is either suspended or running depending on the discretion of the Kernel Scheduler; otherwise (if the process had been blocked waiting for a message) the process becomes unblocked and is either suspended or running depending on the discretion of the Kernel Scheduler

/ERROR_PROCESSING:
./when the buffer_size parameter is too small (i.e., it is smaller than the message length):
./the Kernel raises buffer too_small_for_message exception
./POSTCONDITIONS:
./the message "envelope" (i.e., its attributes: sender, message tag, message length) is returned, no message text is returned
./the calling process has an indication of whether a previous message was lost owing to the incoming message queue becoming full
./the message is removed from the receiver's incoming message queue
./the Kernel-to-Kernel acknowledgement has been sent to the message sender if required
./the priority of this process is its resumption priority
./If a message was available, the process is either suspended or running depending on the discretion of the Kernel Scheduler; otherwise (if the process had been blocked waiting for a message) the process becomes unblocked and is either suspended or running depending on the discretion of the Kernel Scheduler

/WHEN receive_message_is_called_within_anInterrupt_handler:
./the Kernel raises illegal_context_for_call
./POSTCONDITIONS:
./the call is permitted, no message is received; no change in priority is registered
./when the receive_timeout expires:
./the Kernel raises message_timed_out exception
./POSTCONDITIONS:
./no receive is done (there is nothing to receive within the specified epoch time)
./the priority of this process is its resumption priority
./when the timeout value is for a non-future epoch time and there is no message in the receiver's incoming message queue:
./the Kernel raises no_message_available exception
./POSTCONDITIONS:
./no receive is done (there is nothing to receive before the specified epoch time)
./the priority of this process is its resumption priority

/NOTES:
./When a receive fails for any reason (expiry of time out,
procedure Allocate Device Receiver (  
Receiver Process ID, In Pkt Process Identifier;  
Device ID, In Ng Processor Identifier;  
/*==========================================*/  
/* SUBPROGRAM NAME:  
allocate_device_receiver */  
/* DESCRIPTION:  
assigns a specified process to be the receiver of all messages from  
a specified non-Kernel device */  
/* REQ: 7.1.14 */  
/* PRIM: 10.1.31; 16.1.4 */  
/* TIME: 7.2.4 */  
/* PARAMETERS:  
receiver process ID  
the process ID of the process to receive messages from the  
non-Kernel device  
default value:  
one  
device ID  
the processor ID of the device from which messages are sent  
default value:  
one  
/* PRECONDITIONS:  
receiver_process_ID was generated by a previous call to  
declare_process  
the device ID corresponds to an existing entry in the NCT for that  
device (again, the 1-to-1 mapping of non-Kernel devices to  
non-Kernel processors) */  
/* ACTIONS:  
the Kernel registers the specified receiver_process_ID as the sole  
receiver of messages from the specified device */  
/* POSTCONDITIONS:  
update to the Kernel's routing tables for messages from non-Kernel  
deVICES */  
/* ERROR PROCESSING:  
when the Kernel does not recognize the device ID as a non-Kernel  
device.  
the Kernel raises no such_device exists exception  
/* POSTCONDITIONS:  
the pretend does not occur  
when this primitive is called with an undeclared process ID  
the Kernel raises illegal_process_identifier exception  
/* POSTCONDITIONS: */  
/* message too big, illegal context for call, etc.) the  
following values will be set in the output parameters:  
/* sender = PTB_null_process  
message_length = CG.message_length.type(0)  
message_tag = CG.message_tag.type(0)  
messages_lost = false  
Since the various exceptions can be disabled, this is  
another way of testing for a failed receive.  
--------------------------------------------------------
pragma (page);
..: the (re)binding does not occur
..: when any call other than the first call is made:
..: the Kernel raises replacing_previous_allocation exception
..: POSTCONDITIONS:
..: the rebinding occurs
..: the Kernel routing tables are updated accordingly
..: 
..: =================================================

eend Generic_Communication_Management;
begin (page);

..: ============================================================================
..: MODULE NAME:
..: communication_management
..: 
..: MODULE TYPE:
..: package specification
..: 
..: MODULE PURPOSE:
..: an example of (and a template for) an instantiation of the generic
..: package generic_communication_management
..: 
..: ============================================================================
..: MODULE DESCRIPTION:
..: package instantiating the capabilities to send and receive messages
..: among processes where:
..: - error checking for buffer too small for message is enabled
..: - error checking for illegal context for call is enabled
..: - error checking for illegal process identifier is enabled
..: - error checking for message not received is enabled
..: - error checking for message timed out is enabled
..: - error checking for network failure is enabled
..: - error checking for no message available is enabled
..: - error checking for no such device exists is enabled
..: - error checking for receiver dead is enabled
..: - error checking for receiver is sender is enabled
..: - error checking for receiver never existed is enabled
..: - error checking for replacing_previous_allocation is enabled
..: 
..: MODULE CONTENTS:
..: communication_management (package specification)
..: allocate_device (procedure)
..: allocate_device_receiver (procedure)
..: buffer too small for message (exception)
..: illegal_context_for_call (exception)
..: illegal_process_identifier (exception)
..: message_not_received (exception)
..: message_timed_out (exception)
..: network_failure (exception)
..: no_message_available (exception)
..: no_such_device_exists (exception)
..: receive_message (procedure)
..: receiver_dead (exception)
..: receiver_is_sender (exception)
..: receiver_never_existed (exception)
..: replacing_previous_allocation (exception)
..: send_message (procedure)
..: send_message_and_wait (procedure)

..: REFERENCES:
..: DESIGN DOCUMENTS:
..: none
..: 
..: USER'S MANUAL:
..: Kernel User's Manual
..: 
..: TESTING AND VALIDATION:
package Communication Management

Le new Generic Communication Management ()
Illegal Call Enabled => True,
Illegal Interface Enabled => True,
Message Not Received Enabled => True,
Message Timed Out Enabled => True,
Network Failure Enabled => True,
No Message Available Enabled => True,
No Such Device Exists Enabled => True,
Receiver Dead Enabled => True,
Receiver Is Sender Enabled => True,
Receiver Never Existed Enabled => True,
Replacing Previous Allocation Enabled => True);
pragma (page);
/* MODULE NAME: generic_process_attribute_modifiers 
/* generic package specification 
/* 
/* MODULE PURPOSE: support the modification of attributes of already existing processes 
/* 
/* MODULE DESCRIPTION: 
/* collection of subprograms to modify the attributes of already existing processes 
/* 
/* MODULE CONTENTS: 
/* generic_process_attribute_modifiers (generic package specification) 
/* die (procedure) 
/* illegal context for call (exception) 
/* illegal context for call enabled (generic formal parameter) 
/* illegal_process_identifier (exception) 
/* illegal_process_identifier_enabled (generic formal parameter) 
/* kill (procedure) 
/* set_process_preemption (procedure) 
/* set_process_priority (procedure) 
/* wait (overloaded) (procedure) 
/* 
/* REFERENCES: DESIGN DOCUMENTS: 
/* Kernel Facilities Definition 
/* -REQ: 5.1.5, 6.1.9, 7.1.1, 7.1.2, 7.1.13 
/* -PRIM: 10.1.3, 10.1.5, 16.1.1, 18.1.1, 18.1.3, 18.1.5 
/* -TIME: N/A 
/* USER'S MANUAL: 
/* TESTING AND VALIDATION: 
/* Unit testing 
/* NOTES: 
/* in die, set_process_preemption, set_process_priority, and wait, 
/* if checking for illegal context for call is disabled, the application 
/* is asserting that it guarantees that the call is NOT from within an 
/* interrupt handler. If the call IS from within an interrupt handler 
/* and checking is disabled, this procedure simply returns (the 
/* interrupt handler does NOT die). 
/* in kill 
/* if checking for illegal process identifier is disabled, the application 
/* will not be notified if an attempt is made to kill a non-existent 
/* process and that an attempt to do so will be a null action). 
/* 
/* MODIFICATION HISTORY: 
/* 18aug88 bamberg created (compiles clean) 
/* */ 

/* 24sep88 dvk changed to use package short names (and conform 
/* with package body ) 
/* 15feb99 bamberg update documentation per bug 0012 SEI 
/* 29jun99 dvk Bug 0040 SEI: add checks for illegal PIDs 
/* 27aug99 bamberg final doc. check for 01sep99 freeze 
/* */ 

/* VERSION: 3.0 
/* 
/* DISCLAIMER: 
/* The following copyright must be included in this software and 
/* all software utilizing this software. 
/* 
/* DISTRIBUTION: 
/* Approved for public release; distribution is unlimited. 
/* 
/* Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh, 
/* PA. The Software Engineering Institute (SEI) is a federally 
/* funded research and development center established and operated 
/* by Carnegie Mellon University (CMU). Sponsored by the U.S. 
/* Department of Defense under contract F19628-85-C-0003, the SEI 
/* is supported by the services and defense agencies, with the U.S. 
/* Air Force as the executive contracting agent. 
/* 
/* Permission to use, copy, modify, or distribute this software and 
/* its documentation for any purpose and without fee is hereby 
/* granted, provided that the above copyright notice appear in all 
/* copies and that both that copyright notice and this permission 
/* notice appear in supporting documentation. Further, the names 
/* Software Engineering Institute or Carnegie Mellon University may 
/* not be used in advertising or publicity pertaining to 
/* distribution of the software without specific written prior 
/* permission. CMU makes no warranties, express or implied, to the 
/* suitability of the software for any purpose. This software is 
/* provided "as is" and no warranty, express or implied, is made 
/* by the SEI or CMU, as to the accuracy and functioning of the 
/* program and related program material, nor shall the fact of 
/* distribution constitute any such warranty. No responsibility is 
/* assumed by the SEI or CMU in connection herewith. 
/* */ 

pragma (pas);
with Kernel Exceptions;
  -- lo get: illegal_context_for_call
  --     illegal_process_identifier

with Process_Table;
  -- lo get: process_identifier
  --     pending_activity_type

with Schedule_Types;
  -- lo get: current_process_priority
  --     preemption
  --     priority

with TimeGlobals;
  -- lo get: elapsed_time
  --     epoch_time
  --     to_kernel_time

generic

  -- <ERROR checking conditional compilation flags

  --

  -- REQ 5.1.8

Illegal_Context_For_Call: Enabled: In Boolean := True;
Illegal_Process_Identifier: Enabled: In Boolean := True;

declare
package short name GPAM

package Generic_Process_Attribute_Modifiers
is
  package Ke
renames Kernel Exceptions;
  package Pib
renames Process Table;
  package Si
renames Schedule Types;
  package Tg
renames TimeGlobals;

  -- REQ: 6.1.9, 7.1.1, 7.1.2, 7.1.13
  ...
  -- illegal_context_for_call raised by:
  --     die
  --     set process_preemption
  --     set process_priority
  --     wait
  --
  -- PRIM: 10.1.31
Illegal_Context_For_Call: exception
renames Ke.illegal_Context_For_Call;
renames Si.illegal_Process_Identifier:

pragma (page):
procedure Die;
    /* =================================================================== */
    /* SUBPROGRAM NAME:      */
    /* die */
    /* */
    /* DESCRIPTION:          */
    /* terminate the calling process */
    /* */
    /* -REQ: 7.1.18, 7.1.21, 7.1.25, 7.1.26, 8.1.12 */
    /* */
    /* -PRIM: 16.1.5 */
    /* */
    /* -TIME: 7.2.2 */
    /* */
    /* PARAMETERS:           */
    /* none */
    /* */
    /* PRECONDITIONS: */
    /* none (other than the calling process must not already be dead, but */
    /* if it were dead, it couldn't make this call) */
    /* */
    /* ACTIONS: */
    /* the Kernel updates this process's state to dead (i.e., it may never */
    /* be rescheduled) */
    /* no additional messages are accepted for the dying process */
    /* all pending messages for this process are discarded */
    /* all pending messages for this process that require a 1 acknowledgement */
    /* are negatively acknowledged */
    /* any binding of this process to any non-Kernel device is unbound */
    /* */
    /* POSTCONDITIONS: */
    /* this process's state is dead */
    /* */
    /* ERROR PROCESSING: */
    /* when die is called within an interrupt handler: */
    /* the Kernel raises illegal_context_for_call */
    /* POSTCONDITIONS: */
    /* the call is rejected; nothing dies */
    /* */
    /* EXAMPLES: */
    /* example calls and explanations */
    /* */
    /* =================================================================== */

#pragma (page);

procedure Kill (  
    Process_kid: Pp.Process_Identifier);  
    /* =================================================================== */  
    /* SUBPROGRAM NAME:      */  
    /* kill */  
    /* */  
    /* DESCRIPTION: */  
    /* asynchronously aborts the specified Kernel process */  
    /* */  
    /* -REQ: 7.1.18, 7.1.19, 7.1.20, 7.1.22, 7.1.26, 7.1.20, 8.1.12 */  
    /* -REQ: 14.1.5 */  
    /* */  
    /* -PRIM: 10.1.31; 16.1.6 */  
    /* */  
    /* -TIME: 7.2.2; (7.2.3) */  
    /* -TIME: 14.2.3, 14.2.5, 14.2.6 */  
    /* */  
    /* PARAMETERS: */  
    /* - process ID */  
    /* the ID of the process to be killed (may be local or remote) */  
    /* */  
    /* default value: */  
    /* none */  
    /* */  
    /* PRECONDITIONS: */  
    /* NONE */  
    /* */  
    /* ACTIONS: */  
    /* the Kernel updates the specified process's state to dead */  
    /* (i.e., it may never be rescheduled) (this is done asynchronously */  
    /* if the process to be killed is not a local process) */  
    /* the appropriate Kernel data structures are updated to note that the */  
    /* specified process is dead IF AND ONLY IF the process is local */  
    /* no additional messages are accepted for the process being killed */  
    /* all pending messages for the specified process are discarded */  
    /* all pending messages for the specified process that require an */  
    /* acknowledgement are negatively acknowledged */  
    /* any binding of the specified process to any non-Kernel device is */  
    /* unbound */  
    /* if the process is already dead, nothing happens */  
    /* */  
    /* POSTCONDITIONS: */  
    /* the specified process's state is dead */  
    /* */  
    /* ERROR PROCESSING: */  
    /* if an illegal process identifier is passed as a parameter */  
    /* the exception illegal_process_identifier is raised (if enabled) */  
    /* no action is performed (if not) */  
    /* */  
    /* =================================================================== */

#pragma (page);
procedure Set Process Preemption (Preemptable : in St.Preemption := St.Enabled);

-- DESCRIPTION:
-- changes the preemption status of the calling process
--
-- REQ: 9.1.22
--
-- PRIM: 18.1.1
--
-- TIME: 9.2.2
--
-- PARAMETERS:
-- preemptable
-- indication of whether or not a process may be preempted by
-- another process of the same priority
--
-- values include
-- enabled (the process is preemptable)
-- disabled (the process is not preemptable)
--
-- default value
-- enabled (the process is preemptable)
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- if the preemption is not changed, nothing happens
-- if preemption is being disabled, time slicing is disregarded for
-- the process, and this process continues execution
-- if preemption is being enabled, the Kernel changes the process state
-- to suspended, forcing a reschedule point
-- the Kernel records the specified value for this process's preemption
-- status
--
-- POSTCONDITIONS:
-- the process's preemption status is as specified in the call
-- the process state is either suspended or running, depending on the
-- discretion of the Kernel Scheduler
--
-- ERROR PROCESSING:
-- when this primitive is called in an interrupt handler:
-- the Kernel raises illegal context for call exception
--
-- PRECONDITIONS:
-- the call is rejected, the set does not occur

procedure Set Process Priority (NewPriority : in St.Priority);

-- DESCRIPTION:
-- changes the priority of the calling process
--
-- REQ: 9.1.6; 9.1.8; 9.1.22
--
-- PRIM: 18.1.3
--
-- TIME: 9.2.1
--
-- PARAMETERS:
-- new priority
-- priority at which the process is now to run
--
-- default value:
-- none
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- if there is no change in priority, this simply returns
-- if the priority is raised,
-- the Kernel just records the specified value
-- if the process priority is lowered, the Kernel changes the process
-- state to suspended and records the specified value for the
-- process's priority
--
-- POSTCONDITIONS:
-- the process's priority is as specified in the call
-- the process state is either suspended or running depending on the
-- discretion of the Kernel Scheduler
--
-- ERROR PROCESSING:
-- when this primitive is called in an interrupt handler:
-- the Kernel raises illegal context for call exception
--
-- POSTCONDITIONS:
-- the call is rejected, the set does not occur

package (page):
procedure Wait (  
  until Epoch Time : In TgEpoch Time;  
  Resumption Priority : In St.Priority := St.Current Process Priority);  
  -----------------------------------------------------------------------------  
  /SUBPROGRAM NAME:  
  / wait  
  /  
  / DESCRIPTION  
  / allows the caller to suspend its own execution  
  /  
  / -REQ: 9.1.8, 9.1.6, 9.1.14, 9.1.16, 9.1.17, 9.1.22, 9.1.29; 11.1.12  
  /  
  / -PRM: 18.1.5  
  /  
  / -TIME: 9.2.3  
  /  
  / PARAMETERS  
  / until epoch time  
  / absolute time at which the process becomes unblocked  
  / default value:  
  / none  
  /  
  / resumption_priority  
  / the priority to be assigned to this process when it becomes  
  / unblocked at until_epoch_time  
  / default value:  
  / the calling process's current priority  
  /  
  / PRECONDITIONS:  
  / NONE  
  /  
  / ACTIONS:  
  / if the epoch time until which to wait is now or in the past, the Kernel  
  / changes the process state of the caller to suspended, forcing a  
  / reschedule  
  / if the epoch time until which to wait is in the future,  
  / the Kernel changes the process state of the caller to blocked  
  / the Kernel notes that the process is to be unblocked at the specified  
  / epoch time  
  / the Kernel possibly changes the process priority  
  /  
  / POSTCONDITIONS:  
  / the calling process is suspended or blocked, depending on the value of  
  / the epoch time until which to wait  
  / the epoch time at which the process is to be unblocked is noted by  
  / the Kernel  
  / the resumption priority at which the process is to be unblocked is  
  / noted by the Kernel  
  /  
  / ERROR PROCESSING:  
  / when this primitive is called in a context that prohibits blocking  
  / (e.g., within an interrupt handler) and this call would block  
  / (i.e., the epoch time is for some future time).  
  
pragma (page);
procedure Wait (  
    For Elapsed Time : In Tg.Elapsed Time;  
    Resumption Priority : In ST.Priority = ST.Current Process Priority);  
===============================================================================  
// SUBPROGRAM NAME:  
//    wait  
// DESCRIPTION  
//    allows the caller to suspend its own execution  
//    - REO: 9.1.6, 9.1.8, 9.1.14, 9.1.15, 9.1.17, 9.1.22, 9.1.29, 11.1.12  
//    - PRIM: 18.1.5  
//    - TIME: 9.2.3  
// PARAMETERS:  
//    for elapsed time  
//       duration after which the process becomes unblocked  
//       default value:  
//       none  
//       resumption_priority  
//       the priority to be assigned to this process when it becomes  
//       unblocked after elapsed_time  
//       default value:  
//       the calling process's current priority  
// PRECONDITIONS:  
//    NONE  
// ACTIONS:  
//    if the elapsed time for which to wait is now or in the past, the Kernel  
//    changes the process state of the caller to suspended, forcing a  
//    reschedule  
//    if the elapsed time for which to wait is in the future:  
//    the Kernel changes the process state of the caller to blocked  
//    the Kernel notes that the process is to be unblocked at the specified  
//    elapsed time  
//    the Kernel possibly changes the process priority  
// POSTCONDITIONS:  
//    the calling process is suspended or blocked, depending on the value of  
//    the elapsed time until which to wait  
//    the elapsed time at which the process is to be unblocked is noted by  
//    the Kernel  
//    the resumption priority at which the process is to be unblocked is  
//    noted by the Kernel  
// POSTCONDITIONS:  
//    the calling process is blocked  
//    the elapsed time after which the process is to be unblocked is noted  
//    by the Kernel  
//    the resumption priority at which the process is to be unblocked is  
//    noted by the Kernel  
//    the Kernel raises Illegal_context_for_call exception  
//    POSTCONDITIONS:  
//    the wait does not occur  
//    no change in priority is registered  
===============================================================================  
and Generic Process_AttributeModifiers;  
pragma (page);
null
pragma (page);

with Kernel Exceptions;
   ... to get: illegal context for call
   ... illegal process identifier

with Hardware Interface; use Hardware Interface;
   ... to get: hw_string

with Process Table;
   ... to get: process_identifier

with Schedule Types;
   ... to get: preemption
   ... priority

generic

   ... ERROR checking conditional compilation flags
   ...
   --REQ: 5.1.8

Illegal Context For Call Enabled : in Boolean := True;

Illegal Process Identifier Enabled : in Boolean := True;

*/                      *package short name: GPAR
/                      *

package Generic; Process_Attribute, Readers
is

package Pth

renames Process Table;
package Ke

renames Kernel Exceptions;
package Si

renames Schedule Types;

   ...
   ... illegal context for call raised by:
   ... who am i
   ... get process preemption
   ... get process priority
   ...

--PRI: 16.1.7; 16.1.2; 16.1.4

Illegal Context For Call : exception
renames KeIllegal Context For Call;
function Name_Of (
    Process_Id : Pll.Process_Identifier;
  return The_String;
  {//-=================================================================
  // SUBPROGRAM NAME:
  // name_of
  //-DESCRIPTION:
  // allows a process to obtain the logical (string) name of a process
  //-REQ: 7.1.24, 7.1.25, 7.1.26, 10.1.1
  //-PRIM: 16.1.8
  //--TIME: N/A
  //--PARAMETERS:
  // process ID
  // the ID of the process for which the logical name is to be obtained
  // default value:
  // none
  // function return value
  // the logical (string) name of the process that corresponds to the
  // process ID
  //--PreCONDITIONS:
  // the process ID must be a legal value
  //--ACTIONS:
  // return the logical name of the specified process
  //--POSTCONDITIONS:
  // no change in Kernel data structures
  //--ERROR PROCESSING:
  // when this primitive is called with an undeclared process ID:
  // the Kernel raises illegal_process_identifier exception
  // POSTCONDITIONS:
  // the call is rejected; no meaningful value is returned
  //}---=================================================================

#pragma (page):
function Who Am I
  return PtB.Process Identifier:
  
  /* SUBPROGRAM NAME: */
  /* who_am_I */
  
  /* DESCRIPTION: */
  /* it allows a process to obtain its own process ID */
  
  /* --REQ: 7.1.23 */
  /* */
  /* --PRM: 16.1.7 */
  /* */
  /* --TIME: N/A */
  /* */
  /* --PARAMETERS: */
  /* -- function return value */
  /* -- the Kernel assigned process ID for this process */
  /* */
  /* --PRECONDITIONS: */
  /* -- none (other than the calling process must be alive, but if */
  /* -- weren't, it couldn't make this call) */
  /* */
  /* --ACTIONS: */
  /* -- return the process ID of the calling process */
  /* */
  /* --POSTCONDITIONS: */
  /* -- no change in Kernel data structures */
  /* */
  /* --ERROR PROCESSING: */
  /* -- when this primitive is called in an interrupt handler: */
  /* -- the Kernel raises illegal_context_for_call exception */
  /* */
  /* --POSTCONDITIONS: */
  /* -- the call is rejected, no meaningful value is returned */
  /* */
  /* */

#pragma (page);

function Get Process Preemption
  return St.Preemption:
  
  /* SUBPROGRAM NAME: */
  /* get_process_preemption */
  /* */
  /* DESCRIPTION: */
  /* it queries the current value of the preemption status of the calling */
  /* process */
  /* */
  /* --REQ: 9.1.13 */
  /* */
  /* --PRM: 18.1.2 */
  /* */
  /* --TIME: N/A */
  /* */
  /* --PARAMETERS: */
  /* -- function return value */
  /* -- the preemption status of the calling process */
  /* */
  /* --PRECONDITIONS: */
  /* -- none */
  /* */
  /* --ACTIONS: */
  /* -- the Kernel returns the value of the preemption status */
  /* */
  /* --POSTCONDITIONS: */
  /* -- no change in Kernel data structures */
  /* */
  /* --ERROR PROCESSING: */
  /* -- when this primitive is called in an interrupt handler: */
  /* -- the Kernel raises illegal_context_for_call exception */
  /* */
  /* --POSTCONDITIONS: */
  /* -- the call is rejected, no meaningful value is returned */
  /* */
  /* */

#pragma (page);
function Get_Priority return St.Priority;

-- SUBPROGRAM NAME:
-- get_process_priority
--
-- DESCRIPTION:
-- queries the current value of the priority of the calling process
--
-- REQ: 9.1.9
--
-- PRIM: 18.1.4
--
-- TIME: N/A
--
-- PARAMETERS:
-- function return value
-- the current priority of the calling process
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- the Kernel returns the value of the priority level
--
-- POSTCONDITIONS:
-- no change in Kernel data structures
--
-- ERROR PROCESSING:
-- the Kernel raises illegal_context_for_call exception
-- the call is rejected; no meaningful value is returned

-- --------

end Get_Priority;

pragma (page);

-- --------
/* DISTRIBUTION: */
/* */
/* Approved for public release; distribution is unlimited. */
/* */
/* Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh, */
/* PA. The Software Engineering Institute (SEI) is a federally */
/* funded research and development center established and operated */
/* by Carnegie Mellon University (CMU). Sponsored by the U.S */
/* Department of Defense under contract F19628-85-C-0003, the SEI */
/* is supported by the services and defense agencies, with the U.S. */
/* Air Force as the executive contracting agent. */
/* */
/* Permission to use, copy, modify, or distribute this software and */
/* its documentation for any purpose and without fee is hereby */
/* granted, provided that the above copyright notice appear in all */
/* copies and that both that copyright notice and this permission */
/* notice appear in supporting documentation. Further, the names */
/* Software Engineering Institute or Carnegie Mellon University may */
/* not be used in advertising or publicity pertaining to */
/* distribution of the software without specific, written prior */
/* permission. CMU makes no claims or representations about the */
/* suitability of this software for any purpose. This software is */
/* provided "as is" and no warranty, express or implied, is made */
/* by the SEI or CMU, as to the accuracy and functioning of the */
/* program and related program material, nor shall the fact of */
/* distribution constitute any such warranty. No responsibility is */
/* assumed by the SEI or CMU in connection herewith. */
/* */

with Generic Process Attribute Readers;
pragma Elaborate (Generic Process Attribute Readers);
  to get: see MODULE CONTENTS

package short name: PAR

package Process Attribute Readers

is new Generic Process Attribute Readers (Illegal Context For Call Enabled => True,
Illegal Process Identifier Enabled => True);
pragma (page)

/***************************************************************************/

/* MODULE NAME: */
/* */
/* generic Interrupt, globals */
/* */
/* MODULE TYPE: */
/* */
/* generic package specification */
/* */
/* MODULE PURPOSE: */
/* */
/* exports interrupt global information */
/* */

/***************************************************************************/

/* MODULE DESCRIPTION: */
/* */
/* collection of globals related to interrupt management but which can be */
/* used generally by anyone */
/* */
/* MODULE CONTENTS: */
/* */
/* generic interrupt globals (generic package specification) */
/* */
/* Interrupt condition (type) */
/* */
/* Interrupt name (type) */
/* */
/* Interrupt owner (type) */
/* */
/* Interrupt source (type) */
/* */
/* Interrupt state (type) */
/* */
/* Interrupt table (data structure) */
/* */
/* Interrupt table entry (type) */
/* */
/* Interrupt table type (type) */
/* */
/* Interrupt vector (data structure) */
/* */
/* null handler (constant) */
/* */
/* null handler body (procedure) */
/* */

REFERENCES:
/* */
/* DESIGN DOCUMENTS: */
/* */
/* Kernel Facilities Definition */
/* */
/* -REQ: 11.1.1, 11.1.4, 11.1.5, 11.1.6, 11.1.14, 11.1.17 */
/* */
/* -REQ: 11.1.19, 11.1.21 */
/* */
/* -PRIM: 20.1.1, 20.1.2, 20.1.3 */
/* */
/* -TIME: N/A */
/* */
/* USER'S MANUAL: */
/* */
/* Kernel User's Manual */
/* */
/* TESTING AND VALIDATION: */
/* */
/* Integration testing with Kproc and Time_Keep */
/* */
/* NOTES: */
/* */
/* none */
/* */

/* MODIFICATION HISTORY */
/* */
/* 1/4/88 bamberg created (compiles clean) */
/* */
/* 1/1988 firth revised bind interface */
/* */
/* 2/28/88 firth revised interrupt table */
/* */
/* 3/8/88 firth final cleanup */
/* */
/* 3/8/88 firth final doc. check for beta release (no change) */
/* */
/* 3/8/88 firth made null handler a true handler with no body */
/* */
/* 3/8/88 firth removed obsolete generic format params */
with Hardware Interface; use Hardware_Interface;
  -- to get: hw address
  .. hw integer
  .. hw positive
  .. hw byte
  .. null hw address

with Process Table;
  -- to get: Process_identifier
  -- to get: null process

generic
  ...
  ... package short name. GIG
  ...

package Generic Interrupt_Globals

is
  ...
  .. Interrupt_name is an integer value in the range 0 .. 255
  ... these values correspond to the hardware vector assignments
  ... of the target, and are generally sparse. Each Interrupt
  ... name corresponds to one of four kinds of Interrupt, as indicated
  ... by the type Interrupt_owner

  type Interrupt_Name is new Hw_BYTE;
  ... an interrupt vector can be one of four kinds
  ... absent (not assigned to any interrupting device)
  ... reserved (reserved to Ada runtime or other non-Kernel system code
  ... Kernel (owned by Kernel)
  ... application (owned by application)
  ...
  ... Note that the owner must still bind a handler and enable the Interrupt

  type Interrupt_Owner is (Absent, Reserved, Kernel, Application);
  ...
  ... any interrupt may be in one of the following conditions:
  ... bound (a handler has been associated with the named interrupt via
  ... a Kernel call)
  ... unbound (a handler has not been associated with the named interrupt
  ... via a Kernel call)

  type Interrupt Condition is (Bound, Unbound);
  ...
  ... this type indicates the state of an interrupt
  ...
  ... values include:
  ... enabled (the Interrupt is enabled)
  ... disabled (the Interrupt is disabled)
...
type Interrupt_State is (Enabled, Disabled);
  -- this type indicates the source of an interrupt
  -- values include
  -- Internal - interrupt was generated by a simulate Interrupt call
  -- External - interrupt was generated by hardware the the hardware
  -- Interrupt handling mechanism

  type Interrupt_Source is (Internal, External);
  -- the null interrupt handler, ie a procedure with null body
  procedure Null_Handler_Body;
  -- this constant designates the null interrupt handler
  Null_Handler : constant Hw_Address := Hw_Address(Null_Handler_Body/Address);

pragma (page);

-- the information maintained about Interrupts includes:
  -- owner
  -- the owner of the interrupt
  -- condition
  -- the condition of the interrupt
  -- Values include:
  -- Bound (a handler has been associated with the named interrupt
  -- via a Kernel call)
  -- Unbound (a handler has not been associated with the named
  -- interrupt via a Kernel call)
  -- state
  -- The state of the interrupt
  -- Values include:
  -- Enabled (the interrupt is enabled)
  -- Disabled (the interrupt is disabled)
  -- Interrupt source
  -- Indication of where the interrupt sequence originated from
  -- Values include:
  -- Internal (the interrupt was generated by a simulate Interrupt
  -- call)
  -- External (the interrupt was generated external the processor
  -- as some hardware Interrupt)
  -- can preempt
  -- after executing the handler, an Interrupt can do one of
  -- two things
  -- a: return control to the interrupted process
  -- b: return control to the DARK scheduler, which of course
  -- may elect not to resume the interrupted process
  -- a value False for the attribute 'can preempt' means that
  -- the interrupt will always do (a); a value True that it will
  -- always do (b).
  -- Interrupt handler
  -- Interrupt service routine for the Interrupt
  -- tool interface
  -- Indication of whether a tool interface has been enabled
  -- Values include:
  -- True (tool Interface established)
  -- False (tool Interface not established)
  -- process ID
  -- the monitoring process's identifier

type Interrupt Table Entry is record
Owner : Interrupt.Owner = Application;
Condition : Interrupt.Condition = Unbound;
State : Interrupt.State = Disabled;
Source : Interrupt.Source = External;
Can Preempt : Boolean = True;
Handler : Hw.Address = Null Handler;
Tool Interface Enabled : Boolean = False;
Monitoring Process Id : Process Table Process Identifier :=
Process Table Null Process;
end record;
pragma (page);

-- this is the template for declaring an Interrupt table. It is indexed
-- by the range of interrupt names
-- each interrupt entry contains all of the information on an interrupt;

type Interrupt Table Type is array (Interrupt_Name) of Interrupt Table Entry;

Interrupt Table : Interrupt Table Type;

-- this is the transfer vector actually used by the interrupt
-- encapsulation routines. Whenever a handler is enabled,
-- its code address is set in this vector. Whenever a handler
-- is disabled (including initially), the vector designates
-- a null handler, i.e. a procedure with null body

Interrupt Vector : array (Interrupt_Name) of Hw.Address
:= (others => Null_Handler);

end Generic Interrupt Globals;
pragma (page);
/* MODULE NAME */
interrupt_globals

/* MODULE TYPE */
package specification

/* MODULE PURPOSE */
an example of (and a template for) an instantiation of the generic
package generic_interrupt_globals

/* MODULE DESCRIPTION */
package instantiating the interrupt globals (notably the
legal interrupt table)

/* MODULE CONTENTS */
interrupt globals (package specification)
interrupt condition (type)
interrupt name (type)
interrupt source (type)
interrupt state (type)
interrupt table (data structure)
interrupt table entry (type)
interrupt table type (type)

/* NOTES */
none

/* MODIFICATION HISTORY */
18aug88 bamberg created (completes clean)
11jan89 bth revised doc. for beta release
19apr89 bth removed obsolete generic actual params
14may89 bth removed old documentation per bug 0071, SEI

/* VERSION */ 3.0

/* DISCLAIMER */

The following copyright must be included in this software and
all software utilizing this software.

/* DISTRIBUTION */

Approved for public release, distribution is unlimited.

/* Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
PA. The Software Engineering Institute (SEI) is a federally
funded research and development center established and operated
by Carnegie Mellon University (CMU). Sponsored by the U.S.
Department of Defense under contract F 19628 85-C-0003, the SEI
is supported by the services and defense agencies, with the U.S.
Air Force as the executive contracting agent.

225 December 1989

226 December 1989
| MODULE NAME:   |enic Interrupt, Global
|---------------|-------------------
| MODULE TYPE:  |Generic Interrupt, Global
| MODULE PURPOSE: | To block, enable, and simulate interrupts
| MODULE DESCRIPTION: | Estimate error checking appropriately by generic parameters

<table>
<thead>
<tr>
<th>MODULE PARAMETERS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERIC PARAMETERS:</td>
</tr>
<tr>
<td>- Booleans flag</td>
</tr>
</tbody>
</table>
| - Illegal interrupt handler bound enabled (Boolean flag)
| - Reserved interrupt enabled (Boolean flag)

<table>
<thead>
<tr>
<th>MODULE CONTENTS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Enable procedure</td>
</tr>
<tr>
<td>- Disable procedure</td>
</tr>
<tr>
<td>- Bind interrupt handler procedure</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REFERENCES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESIGN DOCUMENTS:</td>
</tr>
</tbody>
</table>
| - NMI 1.2.12.14.1.5

<table>
<thead>
<tr>
<th>TESTING AND VALIDATION:</th>
</tr>
</thead>
</table>
| - Kernal User Manual

<table>
<thead>
<tr>
<th>MODIFICATION HISTORY:</th>
</tr>
</thead>
</table>
| - 22 Sep 9/9 revised Interface
| - 20 Oct 89 revised Interface
with Hardware Interface; use Hardware Interface; -
with Interrupt Global;
with Interrupt_Name;
with Kernel Exceptions;
- to get: Illegal Interrupt
- to get: illegal interrupt handler address
- to get: no Interrupt_handler bound
- to get: replacing previous interrupt_handler
- to get: reserved_interrupt
generic
- error checking conditional compilation flags
Illegal Interrupt Enabled : In Boolean := True;
Illegal Interrupt Handler Address Enabled : In Boolean := True;
No Interrupt Handler Bound Enabled : In Boolean := True;
Replcacing Previous Interrupt Handler Enabled : In Boolean := True;
Reserved Interrupt Enabled : In Boolean := True;

package short name: GIM

package Generic Interrupt Management
is
package ig
renames Interrupt Global;
- Error Status Codes 20.3
- illegal interrupt is intended to be raised by any of
  - enable
  - disable
  - enabled
  - simulate Interrupt
  - bind Interrupt handler
- if the interrupt name parsed as a parameter does not designate a
  - legal interrupt.
- - In the current implementation, all values of the subtype interrupt
    - are legal, so this exception cannot be raised.
Illegal Interrupt exception
renames Kernel Exceptions Illegal Interrupt;
- illegal interrupt handler address is raised by
  - bind interrupt handler
  - if the subprogram address passed as the handler code parameter
procedure Enable (Interrupt : in Interrupt_Name);

-- SUBPROGRAM NAME:

-- enable

-- DESCRIPTION:

-- allows the specified interrupt to occur

-- REQUESTED TIME: N/A

-- PARAMETERs:

-- Interrupt

-- the name of the interrupt to be enabled

-- default value:

-- none

-- PRECONDITIONs:

-- The interrupt is legal

-- The interrupt is not reserved

-- A handler has already been bound to the interrupt

-- ACTIONS:

-- Indicates that the interrupt is allowed to occur

-- POSTCONDITIONs:

-- the Interrupt may now occur

-- the Kernel notes that the interrupt state is enabled

-- ERRORS:

-- Legality of the Interrupt is checked by the Ada language,

-- since the type Interrupt global Interrupt_Name embraces

-- all legal interrupts. The exception Constraint_Error is

-- raised for an illegal Interrupt.

-- The subprogram checks that the interrupt is free to the

-- application, and raises Reserved Interrupt if it is not.

-- The subprogram checks that a handler has been bound, and

-- raises No Interrupt Handler Bound if one has not

-- If an error occurs but its reporting is disabled, the subprogram

-- silently returns without changing the state of the interrupt.
procedure Disable (Interrupt in Ig Interrupt Name);
/* -------------------------------------------------------------
/* SUBPROGRAM NAME:
/* disable
/* DESCRIPTION:
/* ignores the specified interrupt whenever it occurs
/* -REQ. 11.1.5
/* -PRIM. 20.1.2
/* -TIME: N/A
/* PARAMETERS:
/* interrupt
/* the name of the interrupt to be disabled
/* default value:
/* none
/* PRECONDITIONS:
/* The interrupt is legal
/* The interrupt is not reserved
/* NOTE that a handler need not have been bound
/* ACTIONS:
/* indicates that the Interrupt is ignored whenever it occurs
/* POSTCONDITIONS:
/* the interrupt is disabled
/* the Kernel notes that the interrupt state is disabled
/* ERROR PROCESSING:
/* Legality of the interrupt is checked by the Ada language,
/* since the type interrupt globals interrupt name embraces
/* all legal interrupts. The exception Constraint Error is
/* raised for an Illegal interrupt.
/* If an error occurs but its reporting is disabled, the subprogram
/* silently returns without changing the state of the interrupt.
/* */
 pragma page.]

function Enabled (Interrupt in Ig Interrupt Name)
return Boolean;
/* -------------------------------------------------------------
/* SUBPROGRAM NAME:
/* enabled
/* DESCRIPTION:
/* queries
/* -REQ. 11.1.7
/* -PRIM. 20.1.3
/* -TIME: N/A
/* PARAMETERS:
/* interrupt
/* the name of the Interrupt being queried
/* default value:
/* none
/* function return value
/* indication of whether or not the specified interrupt is enabled
/* values include:
/* true (the specified Interrupt is enabled)
/* false (the specified interrupt is disabled)
/* default value:
/* none
/* PRECONDITIONS:
/* The interrupt is legal
/* The interrupt is not reserved
/* ACTIONS:
/* the Kernel returns an indication of the interrupt state
/* POSTCONDITIONS:
/* no change in Kernel data structures
/* ERROR PROCESSING:
/* Legality of the interrupt is checked by the Ada language,
/* since the type interrupt globals interrupt name embraces
/* all legal interrupts. The exception Constraint Error is
/* raised for an Illegal interrupt.
/* If an error occurs but its reporting is disabled, the subprogram
/* returns false, regardless of the actual state of the interrupt.
/* */
/* */

pragma (page);

procedure Simulate Interrupt (%
  Interrupt: in g Interrupt Name;%
  --- SUBPROGRAM NAME:%
  // simulate interrupt%
  ---
  // DESCRIPTION:%
  // simulates the occurrence of a specified interrupt in software%
  ---
  // --REQ: 11.1.2, 11.1.3, 11.1.8%
  ---
  // --PRIM: 20.1.4%
  ---
  // --TIME: N/A%
  ---
  // PARAMETERS:%
  // Interrupt%
  // the name of the interrupt being simulated%
  // default value:%
  // none%
  ---
  // PRECONDITIONS:%
  // The interrupt is legal%
  // The interrupt is not reserved%
  // A handler has already been bound to the interrupt%
  ---
  // ACTIONS:%
  // the interrupt handler associated with the specified interrupt name%
  // is executed as though the actual hardware interrupt had occurred%
  ---
  // POSTCONDITIONS:%
  // upon completion of the interrupt handler execution,%
  // if can_preempt is False%
  // the interrupted code is resumed%
  // if can_preempt is True%
  // the Scheduler is invoked and selects the process to be resumed%
  ---
  // ERROR PROCESSING:%
  // Legality of the interrupt is checked by the Ada language,%
  // since the type Interrupt globals Interrupt name embraces%
  // all legal interrupts. The exception Constraint_Error is%
  // raised for an illegal interrupt.%
  // The subprogram checks that the interrupt is free to the%
  // application, and raises Reserved Interrupt if it is not.%
  // The subprogram checks that a handler has been bound, and%
  // raises No Interrupt Handler Bound if one has not%
  // If the interrupt handler tries to propagate an exception the%
  // subprogram simulate interrupt immediately considers the handler to%
  // have completed, and continues with the normal completion action.%
  // (This is the same as the action taken by a real interrupt)%;
  Interrupt)
  return;
end Simulate Interrupt;
procedure Bind Interrupt Handler (
  Interrupt : In Interrupt Name;
  Handler Code : In Hw Address;
  Can Preempt : In Boolean := True
);

pragma (page);
--/ constructs the execution environment for this interrupt handler,
--/ including the local interrupt stack and interrupt entry
--/ and exit code. Note that no dummy call frame or stack plug
--/ is needed, since the handlers will be themselves Kernel code
--/
--/ POSTCONDITIONS:
--/ execution environment created for this Interrupt handler
--/ note that this postcondition is established even if the
--/ prior state of the Interrupt was Bound, is any previous
--/ handler will be replaced.
--/
--/ ERROR PROCESSING:
--/ Legality of the interrupt is checked by the Ada language,
--/ since the type Interrupt_globals.Interrupt_name embraces
--/ all legal interrupts. The exception Constraint_Error is
--/ raised for an illegal interrupt.
--/
--/ The subprogram checks that the interrupt is free to the
--/ application, and raises Reserved_Interrupt if it is not.
--/
--/ The subprogram checks that a handler has not already been bound.
--/ If one has, it is REPLACED by the new handler, after which
--/ Replacing, Previous Interrupt_Handler is raised. The handler
--/ is replaced even if the error report is disabled.
--/
--/ If other error reporting is disabled, the subprogram silently
--/ returns without changing the state of the interrupt.
--/
--/ NOTES:
--/ The exception Insufficient_Space cannot be raised by this
--/ subprogram. All data structures are static, and all real
--/ Interrupts execute on a single stack that is allocated at
--/ program load time. Binding a handler requires no further
--/ allocation of space.
--/
--/ ***********************************************************************
--/ and Generic_Interrupt_Management;
--/ pragma (page).
--/ ***********************************************************************
--/ "********************************************************************************
--/ MODULE NAME:
--/ "********************************************************************************
--/ Interrupt_management
--/ "********************************************************************************
--/ MODULE TYPE:
--/ "********************************************************************************
--/ package specification
--/ "********************************************************************************
--/ MODULE PURPOSE:
--/ "********************************************************************************
--/ an example of (and a template for) an instantiation of the generic
--/ package generic_Interrupt_management
--/ "********************************************************************************
--/ "********************************************************************************
--/ MODULE DESCRIPTION:
--/ "********************************************************************************
--/ package instantiating the capabilities to create the interrupt
--/ execution environment and to manipulate interrupts where:
--/ error checking for illegal interrupt is enabled
--/ error checking for illegal interrupt handler address is enabled
--/ error checking for Insufficient_space is enabled
--/ error checking for no Interrupt_handler bound is enabled
--/ error checking for replacing previous interrupt_handler is enabled
--/ error checking for reserved Interrupt is enabled
--/ "********************************************************************************
--/ MODULE CONTENTS:
--/ "********************************************************************************
--/ interrupt management (generic package specification)
--/ bind interrupt handler (procedure)
--/ disable (procedure)
--/ enable (procedure)
--/ enabled (function)
--/ illegal interrupt (exception)
--/ illegal interrupt handler address (exception)
--/ insufficient space (exception)
--/ no interrupt handler bound (exception)
--/ replacing previous interrupt handler (exception)
--/ reserved interrupt (exception)
--/ simulate interrupt (procedure)
--/ "********************************************************************************
--/ REFERENCES:
--/ "********************************************************************************
--/ DESIGN DOCUMENTS:
--/ Kernel Facilities Definition
--/ USER'S MANUAL:
--/ Kernel User's Manual
--/ "********************************************************************************
--/ TESTING AND VALIDATION:
--/ none
--/ "********************************************************************************
--/ NOTES:
--/ none
--/ "********************************************************************************
--/ MODIFICATION HISTORY:
--/ 18aug93 bamberg created (compiles clean)
--/ 12jan93 frth final revision for beta release
--/ "********************************************************************************
/* VERSION 3.0 */
/* DISCLAIMER: */
/* The following copyright must be included in this software and */
/* all software utilizing this software. */
/* */
/* DISTRIBUTION */
/* */
/* Approved for public release; distribution is unlimited. */
/* */
/* Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh, */
/* PA. The Software Engineering Institute (SEI) is a federally */
/* funded research and development center established and operated */
/* by Carnegie Mellon University (CMU). Sponsored by the U.S */
/* Department of Defense under contract F18652-85-C-0003, the SEI */
/* is supported by the services and defense agencies, with the U.S. */
/* Air Force as the executive contracting agent. */
/* */
/* Permission to use, copy, modify, or distribute this software and */
/* its documentation for any purpose and without fee is hereby */
/* granted, provided that the above copyright notice appear in all */
/* copies and that both that copyright notice and this permission */
/* notice appear in supporting documentation. Further, the names */
/* Software Engineering Institute or Carnegie Mellon University may */
/* not be used in advertising or publicity pertaining to */
/* distribution of the software without specific, written prior */
/* permission. CMU makes no claims or representations about the */
/* suitability of this software for any purpose. This software is */
/* provided "as is" and no warranty, express or implied, is made */
/* by the SEI or CMU, as to the accuracy and functioning of the */
/* program and related program material, nor shall the fact of */
/* distribution constitute any such warranty. No responsibility is */
/* assumed by the SEI or CMU in connection herewith. */
/* */
with Generic Interrupt Management;
   -- to get: see MODULE CONTENTS
/* */
/* package short name: IM */
/* */
package Interrupt_Management

In new Generic Interrupt Management
   Illegal Interrupt Enabled => True,
   Illegal Interrupt Handler Address Enabled => True,
   No Interrupt Handler Bound Enabled => True,
   Replacing Previous Interrupt Handler Enabled => True,
   Reserved Interrupt Enabled => True);
with Kernel Exceptions;
  ... to get: claim timed out
  ... illegal context for call
  ... not_my_semaphore

with Process Table;
  ... to get: semaphore

with Schedule Types;
  ... to get: current_process_priority
  ... priority

with Time Globals;
  ... to get: elapsed_time
  ... epoch_time

generic

  ... ERROR checking conditional compilation flags
  ...
  ... -REQ: 5.1.8

Claim Timed Out Enabled: In Boolean = True;
Illegal Context For Call Enabled: In Boolean = True;
Not My Semaphore Enabled: In Boolean = True;
Pragma(Page);
...
/package short name. GMM
...

package GenericSemaphoreManagement

is
...
  ... claim_timed_out raised by:
  ... claim
  ... -PRIM: 17.1.2

Claim Timed Out: exception
renames Kernel Exceptions.Claim_Timed_Out;
...
  illegal context for call raised by:
  claim
  release
  ...
  -PRIM: 17.1.2, 17.1.3

Illegal Context For Call: exception
renames Kernel_Exceptions.Legal_Context_For_Call;

-- not my semaphore raised by:
-- release
--PRIM: 17.1.3

Not My Semaphore : exception
renames Kernel_Exceptions.Not_My_Semaphore;

procedure Claim (
Semaphore Name : In Process Table.Semaphore;
Resumption Priority : In Schedule_Type.Priority :=
Schedule_Type.Current_Process.Priority);
//----------------------------------------------------------------------
//-- SUBPROGRAM NAME:
//-- claim
//-- // DESCRIPTION:
//-- allows a process to try to claim a semaphore, blocking until the
//-- semaphore is claimed
//-- ...
//-- // REQ: 8.1.3, 8.1.4; 8.1.7, 8.1.8; 8.1.9, 8.1.16;
//-- // REQ: 9.1.9; 9.1.8;
//-- ...
//-- //PRIM: 17.1.2
//-- ...
//-- //TIME: 8.2.2
//-- ...
//-- //PARAMETERS:
//-- // semaphore name
//-- // a semaphore object created by a declaration using type
//-- // semaphore
//-- ...
//-- // default value:
//-- // none
//-- ...
//-- // resumption priority
//-- // the priority to be assigned to this process when it becomes
//-- // unblocked
//-- ...
//-- // default value:
//-- // the calling process's current priority
//-- ...
//-- // PRECONDITIONS:
//-- // the semaphore has been declared
//-- ...
//-- // ACTIONS:
//-- // the process registers a request to claim a semaphore
//-- // if the semaphore is free, the process claims the semaphore;
//-- // otherwise, the calling process blocks (if permitted, see ERROR
//-- // PROCESSING below) until the semaphore becomes free
//-- // the Kernel possibly changes the process priority
//-- ...
//-- // POSTCONDITIONS:
//-- // the semaphore is not free
//-- // the priority of this process is its resumption priority
//-- // if the semaphore was free when the process called claim, then the
//-- // process has control of the semaphore, the process is either
//-- // suspended or running depending on the discretion of the Kernel
//-- // Scheduler, otherwise (if the process was blocked waiting for the
//-- // semaphore to become free) the process becomes unblocked and is
//-- // either suspended or running depending on the discretion of the
//-- // Kernel Scheduler
//-- //
procedure Claim {
    Semaphore Name : In Process Table.Semaphore;
    Within Elapsed Time : In Time Globals.Elapsed Time;
    Resumption Priority : In Schedule Types.Priority :=
} ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
/. SUBPROGRAM NAME:
/. claim
/. /
/. DESCRIPTION:
/. allows a process to try to claim a semaphore within a specified
/. elapsed time
/. /
/. /-REQ: 8.1.3, 8.1.4, 8.1.5, 8.1.7, 8.1.8, 8.1.9, 8.1.16
/. /-REQ: 9.1.5, 9.1.8
/. /-PRIM: 17.12
/. /-TIME: 8.2.2
/. /
/. PARAMETERS:
/. semaphore name
/. a semaphore object created by a declaration using type
/. semaphore
/. default value:
/. none
/. /
/. within elapsed time
/. duration within which the request to claim a semaphore must be
/. honored or the request is withdrawn
/. /
/. default value:
/. none
/. /
/. resumption priority
/. the priority to be assigned to this process when it becomes
/. unblocked
/. /
/. default value:
/. the calling process's current priority
/. /
/. PRECONDITIONS:
/. the semaphore has been declared
/. /
/. ACTIONS:
/. the process registers a request to claim a semaphore by a specified
/. elapsed time
/. if the semaphore is free, the process claims the semaphore.
/. otherwise, the calling process blocks (if permitted, see ERROR
/. PROCESSING below) until the semaphore becomes free or until the
/. specified timeout expires (for the latter case, see ERROR
/. PROCESSING below)
/. if the semaphore is not free and the timeout is for zero or less
/. elapsed time, then the process does not wait for the semaphore to
procedure Claim {
    Semaphore Name : In Process Table Semaphore;
    By Epoch Time : In Time Globals Epoch Time;
    Resumption Priority : In Schedule Type Priority :=
    Schedule Types.Current Process Priority);
/>.-----------------------------------------------
/>. SUBPROGRAM NAME:
/>. claim
/>.-----------------------------------------------
/>. DESCRIPTION:
/>. allows a process to try to claim a semaphore by a specified epoch
/>. time
/>.-----------------------------------------------
/>. -REQ: 8.1.3, 8.1.4, 8.1.6, 8.1.7, 8.1.8, 8.1.9, 8.1.16;
/>. -REQ: 9.1.6, 9.1.8
/>. -PRIM: 17.1.2
/>. -TIME: 8.2.2
/>.-----------------------------------------------
/>. PARAMETERS:
/>. semaphore name
/>. a semaphore object created by a declaration using type
/>. semaphore
/>. default value:
/>. none
/>.-----------------------------------------------
/>. by epoch, time
/>. absolute time by which the request to claim a semaphore must be
/>. honored or the request is withdrawn
/>. default value:
/>. none
/>.-----------------------------------------------
/>. resumption priority
/>. the priority to be assigned to this process when it becomes
/>. unblocked
/>. default value:
/>. the calling process's current priority
/>.-----------------------------------------------
/>. PRECONDITIONS
/>. the semaphore has been declared
/>.-----------------------------------------------
/>. ACTIONS
/>. the process registers a request to claim a semaphore by a specified
/>. epoch (absolute) time
/>. if the semaphore is free, the process claims the semaphore;
/>. otherwise, the calling process blocks (if permitted, see ERROR
/>. PROCESSING below) until the semaphore becomes free or until the
/>. specified timeout expires (for the latter case, see ERROR
/>. PROCESSING below)
/>. if the semaphore is not free and the timeout is for a non future
/>. epoch time, then the process does not wait for the semaphore to
procedure Release (Semaphore Name: In Process Table Semaphore);

SUBPROGRAM NAME:

DESCRIPTION:

allows a process to release a semaphore it currently controls

REQ: 8.1.10, 8.1.11, 8.1.13, 8.1.15

PRM: 17.1.3

TIME: 8.2.3

PARAMETERS:

semaphore name

a semaphore object created by a declaration using type semaphore
default value: none

PRECONDITIONS:

the semaphore has been declared

the semaphore has been claimed by the calling process

ACTIONS:

the Kernel releases the semaphore (i.e., makes it available to the
next process in the waiting queue)

POSTCONDITIONS:

the semaphore is no longer held by the calling process

If there are no other processes in the waiting queue for this
semaphore, the semaphore becomes free

If there are processes in the waiting queue, then the process at the
head of the queue is given the semaphore

ERROR PROCESSING:

When release is called within an interrupt handler:

the Kernel raises Illegal_context_for_call

POSTCONDITIONS:

the call is rejected, no attempt to release a semaphore is noted;

no change in priority is registered

when a request to release a semaphore is made by a process that does
not own the semaphore:

the Kernel raises not_my_semaphore_exception

POSTCONDITIONS:

the requested action does not take place

end

Generic Semaphore Management;

pragma (page);
--- Approved for public release; distribution is unlimited. ---
--- Copyright (c) 1999 by Carnegie Mellon University, Pittsburgh, ---
--- PA. The Software Engineering Institute (SEI) is a federally ---
--- funded research and development center established and operated ---
--- by Carnegie Mellon University (CMU). Sponsored by the U.S. ---
--- Department of Defense under contract F19628-95-C-0003, the SEI ---
--- is supported by the services and defense agencies, with the U.S. ---
--- Air Force as the executive contracting agent. ---
--- Permission to use, copy, modify, or distribute this software and ---
--- its documentation for any purpose and without fee is hereby ---
--- granted, provided that the above copyright notice appear in all ---
--- copies and that both that copyright notice and this permission ---
--- notice appear in supporting documentation. Further, the names ---
--- Software Engineering Institute or Carnegie Mellon University may ---
--- not be used in advertising or publicity pertaining to ---
--- distribution of the software without specific, written prior ---
--- permission. CMU makes no claims or representations about the ---
--- suitability of this software for any purpose. This software is ---
--- provided "as is" and no warranty, express or implied, is made ---
--- by the SEI or CMU, as to the accuracy and functioning of the ---
--- program and related program material, nor shall the fact of ---
--- distribution constitute any such warranty. No responsibility is ---
--- assumed by the SEI or CMU in connection herewith. ---
--- 
--- pragma (page);
with Generic Semaphore Management;
pragma Elaborate (Generic Semaphore Management);
-- to get: see MODULE CONTENTS
--
-- package short name: MM
--

package Semaphore.Management

is new Generic Semaphore Management (Claim Timed Out Enabled => True, Illegal Context For Call Enabled => True, Not My Semaphore Enabled => True);
pragma (page);

-- ************************************************************
-- MODULE NAME:
-- generic_alarm_management
--
-- MODULE TYPE:
-- generic package specification
--
-- MODULE PURPOSE:
-- provide the capability to set and cancel alarms (time-triggered events)
-- and to detect the expiration of an alarm
--
-- MODULE DESCRIPTION:
-- collection of subprograms to set and cancel alarm and an exception to
-- report the expiration of an alarm
--
-- MODULE CONTENTS:
-- generic alarm management (generic package specification)
-- alarm expired (exception)
-- cancel alarm (procedure)
-- illegal context for call (exception)
-- illegal context for call enabled (generic formal parameter)
-- no alarm set (exception)
-- no alarm set enabled (generic formal parameter)
-- resetting existing alarm (exception)
-- resetting existing alarm enabled (generic formal parameter)
-- set alarm (overloaded (2) procedure)
--
-- REFERENCES:
-- DESIGN DOCUMENTS:
-- Kernel Facilities Definition
--
-- USER’S MANUAL:
-- Kernel User’s Manual
--
-- TESTING AND VALIDATION:
-- Unit testing
--
-- NOTES:
-- the exception: alarm expired is raised when an alarm expires, when
-- using alarms, the user should take care that the alarm, expired is not
-- masked (or mishandled) by a "when others" clause in an exception
-- handler
--
-- MODIFICATION HISTORY:
-- 18aug88bamberg created (compiles clean)
-- 24dec88dvk changed to use short names
--
-- VERSION: 3.0
--
-- DISCLAIMER:
--
-- The following copyright must be included in this software and
package Generic Alarm Management

is

package Ke

renames Kernel Exceptions;

package Si

renames Schedule Types;

package Tg

renames Time Globals;

renames Ke.Alarm Expired;

package Seq

renames Ke.Alarm Set;

with Kernel Exceptions;

.. to get: alarm expired
... illegal context for call
... no alarm set
... resetting existing alarm

with Schedule Types;

.. to get: current process priority
... priority

with Time Globals;

.. to get: elapsed time
... epoch time

generic

.. ERROR checking conditional compilation flags

.. Req: 5.1.8

Illegal Context For Call Enabled: In Boolean := True;
No Alarm Set Enabled: In Boolean := True;
Resetting Existing Alarm Enabled: In Boolean := True;

.. package short name: CAM

.. package short name: CAM

package Alarm Management

is

package Ke

renames Kernel Exceptions;

package Si

renames Schedule Types;

package Tg

renames Time Globals;

.. Req: 13.1.4; 13.1.5; 13.2.3

.. Req: 13.1.1; 13.1.4; 13.1.5

.. Prim: 22; 22.1.1

Alarm Expired: exception
renames Ke.Alarm Expired;

..
procedure Set Alarm
   After Elapsed Time; In Tg, Elapsed Time;
   Expiration Priority; In St Priority; St Current_Process Priority;
SUBPROGRAM NAME:
   set, alarm
   /
   / DESCRIPTION:
   / defines an alarm that interrupts the calling process if it expires
   /
   / - REQ: 9.1.6, 9.1.8, 9.1.22
   /
   
   /
   / PARAMETERS:
   / - after elapsed time
   / - the elapsed time immediately after which the alarm expired
   / - exception is raised
   /
   /
   / default value:
   / none
   /
   /
   / PRECONDITIONS:
   / none
   /
   /
   / ACTIONS:
   / an alarm event is registered with the Kernel
   / the Kernel sets its internal timer to note when the specified
   / elapsed time expires
   / if the alarm expires, the Kernel raises alarm expired exception in
   / the calling process, thereby causing an enforced transfer of
   / control to a user provided exception handler and a possible
   / change in process priority
   / if the alarm becomes cancelled, there is no transfer or change of
   / priority
   /
   /
   / POSTCONDITIONS:
   / the alarm event is registered with the Kernel
   / if the alarm expires:
   / the priority of this process is its expiration priority
   / control is transferred to a user provided handler, and the process
   / state is either suspended or running depending on the
   / discretion of the Kernel Scheduler
   / if the alarm becomes cancelled, the internal timer is de-activated
   /
   /
   / ERROR PROCESSING:
   / when set alarm is called within an interrupt handler:
   / the Kernel raises illegal context, for call
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   /
   

procedure Set Alarm {
  For Epoch Time : In Tg Epoch Time;
  ________________________________
  SUBPROGRAM NAME:
  - set alarm
  __________
  DESCRIPTION:
  - defines an alarm that interrupts the calling process if it expires
  - - REQ: 9.1.6; 9.1.8; 9.1.22
  - - REQ: 13.1.1; 13.1.3; 13.1.5; 13.1.6; 13.1.7; 13.1.9
  - - PRM: 22.1.1
  __________
  - TIME: 13.2.1
  __________
  PARAMETERS:
  - for epoch time
  - the epoch time immediately at which the alarm expires
  - exception is raised
  - default value:
  - none
  __________
  PRECONDITIONS:
  - NONE
  __________
  ACTIONS:
  - an alarm event is registered with the Kernel
  - the Kernel sets its internal timer to note when the specified
    epoch time arrives
  - if the alarm expires, the Kernel raises alarm expired exception in
    the calling process, thereby causing an enforced transfer of
    control to a user provided exception handler and a possible
    change in process priority
  - if the alarm becomes cancelled, there is no transfer or change of
    priority
  __________
  POSTCONDITIONS:
  - the alarm event is registered with the Kernel
  - if the alarm expires:
    - the priority of this process is its expiration priority
    - control is transferred to a user provided handler, and the process
      state is either suspended or running depending on the
      discretion of the Kernel Scheduler
    - if the alarm becomes cancelled, the internal timer is deactivated
  __________
  ERROR PROCESSING:
  - when set alarm is called within an interrupt handler,
    the Kernel raises illegal context for call
  __________
  POSTCONDITIONS:
  - the call is rejected, no alarm is set
  - when an active alarm is reset
    the Kernel raises resetting existing alarm exception
procedure Cancel_Alarm;

/* SUBPROGRAM NAME: */
/* cancel_alarm */
/* */
/* DESCRIPTION: */
/* turns off an active alarm (i.e., one that was set but has not yet */
/* expired) */
/* */
/* --REQ: 9.1.22, 13.1.10 */
/* */
/* --PRIM: 22.1.2 */
/* */
/* --TIME: 13.2.2 */
/* */
/* PARAMETERS: */
/* none */
/* */
/* POSTCONDITIONS: */
/* NONE */
/* */
/* ACTIONS: */
/* the Kernel deactivates the timer */
/* */
/* POSTCONDITIONS: */
/* no alarm event exists (any longer) for the calling process */
/* */
/* ERROR PROCESSING: */
/* when cancel_alarm is called within an interrupt handler: */
/* the Kernel raises illegal context for call */
/* POSTCONDITIONS: */
/* the call is rejected, no alarm is canceled */
/* when there is no alarm set for the calling process: */
/* the Kernel raises no alarm, set exception */
/* POSTCONDITIONS: */
/* the requested action does not take place */
/* */
/* */
end Generic Alarm Management;

pragma (page);
/ * _________________________________ 
/ * MODULE NAME: 
/ * alarm_management 
/ * / 
/ * MODULE TYPE: 
/ * package specification 
/ * / 
/ * MODULE PURPOSE: 
/ * an example of (and a template for) an instantiation of the generic 
/ * package generic_alarm_management 
/ * / 
/ * _________________________________ 
/ * MODULE DESCRIPTION: 
/ * package instantiating the capabilities to set and cancel an alarm and to 
/ * report the expiration of an alarm where 
/ * error checking for no alarm set is enabled 
/ * error checking for resetting existing alarm is enabled 
/ * / 
/ * MODULE CONTENTS: 
/ * alarm_management (package specification) 
/ * alarm_expired (exception) 
/ * cancel_alarm (procedure) 
/ * illegal_context_for_call (exception) 
/ * no_alarm_set (exception) 
/ * resetting_existing_alarm (exception) 
/ * set_alarm (overloaded (2) procedure) 
/ * / 
/ * REFERENCES: 
/ * DESIGN DOCUMENTS: 
/ * none 
/ * / 
/ * USER'S MANUAL: 
/ * / 
/ * TESTING AND VALIDATION: 
/ * none 
/ * / 
/ * NOTES: 
/ * the exception: alarm_expired is raised when an alarm expires; when 
/ * using alarms, the user should take care that the alarm is not 
/ * masked (or mishandled) by a "when others" clause in an exception 
/ * handler. 
/ * / 
/ * _________________________________ 
/ * MODIFICATION HISTORY: 
/ * 18aug88 bamberg created (completed clean) 
/ * 02aug89 tkw added pragma to make VMS happy 
/ * / 
/ * _________________________________ 
/ * VERSION: 3.0 
/ * / 
/ * DISCLAIMER: 
/ * The following copyright must be included in this software and 
/ * all software utilizing this software 
/ * / 
/ * DISTRIBUTION: 
/ * / 
/ * Approved for public release; distribution is unlimited. 
/ * / 
/ * Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh, 
/ * PA. The Software Engineering Institute (SEI) is a federally 
/ * funded research and development center established and operated 
/ * by Carnegie Mellon University (CMU). Sponsored by the U.S. 
/ * Department of Defense under contract F19628-88-C-0003, the SEI 
/ * is supported by the services and defense agencies, with the U.S. 
/ * Air Force as the executive contracting agent. 
/ * / 
/ * Permission to use, copy, modify, or distribute this software and 
/ * its documentation for any purpose and without fee is hereby 
/ * granted, provided that the above copyright notice appear in all 
/ * copies and that both that copyright notice and this permission 
/ * notice appear in supporting documentation. Further, the names 
/ * Software Engineering Institute or Carnegie Mellon University may 
/ * not be used in advertising or publicity pertaining to 
/ * distribution of the software without specific, written prior 
/ * permission. CMU makes no claims or representations about the 
/ * suitability of this software for any purpose. This software is 
/ * provided "as is" and no warranty, express or implied, is made 
/ * by the SEI or CMU, as to the accuracy and functioning of the 
/ * program and related program material, nor shall the fact of 
/ * distribution constitute any such warranty. No responsibility is 
/ * assumed by the SEI or CMU in connection herewith. 
/ * / 
/ * with Generic Alarm Management; 
/ * to get: see MODULE CONTENTS 
/ * 
/ * package Elaborate (Generic Alarm Management); 
/ * / 
/ * package Alarm_Management 
/ * is new Generic Alarm Management ( 
/ * Illegal Context For Call Enabled => True, 
/ * No Alarm Set Enabled => True, 
/ * Resetting Existing Alarm Enabled => True); 
/ * pragma (page);
-/- MODULE NAME:
-/- generic time management
-/-
-/- MODULE TYPE:
-/- generic package specification
-/-
-/- MODULE PURPOSE:
-/- provide the abstraction of Kernel time (i.e., SLICE) and the capability
-/- of setting/resetting epoch and elapsed time
-/-
-/-
-/- MODULE DESCRIPTION:
-/- collection of constants and subprograms to define, modify, and read
-/- Kernel time
-/-
-/- MODULE CONTENTS:
-/- generic time management (generic package specification)
-/- adjust elapsed time (procedure)
-/- adjust epoch time (procedure)
-/- change results in negative elapsed time (exception)
-/- generic formal parameter
-/- change results in negative epoch time (exception)
-/- change results in negative epoch time enabled
-/- generic formal parameter
-/- network failure (exception)
-/- network failure enabled (generic formal parameter)
-/- OK but time already passed (exception)
-/- OK but time already passed enabled (generic formal parameter)
-/- read clock (function)
-/- synchronize (overloaded) (procedure)
-/- synchronization in progress (exception)
-/- synchronization in progress enabled (generic formal parameter)
-/- synchronize timeout (exception)
-/- synchronize timeout enabled (generic formal parameter)
-/-
-/- REFERENCES:
-/- DESIGN DOCUMENTS:
-/- Kernel Facilities Definition
-/- - REQ 5 1.8 6.1.8
-/- - REQ 12.1.2 12.1.3 12.1.7 12.1.8 12.1.9 12.1.11
-/- - REQ 12.1.12 12.1.14
-/- - PRM 21 1.3 21.1.4 21.1.6
-/- - ATME: N/A
-/-
-/- USER'S MANUAL:
-/- Kernel User's Manual
-/-
-/- TESTING AND VALIDATION:
-/- Unit testing
-/-
-/- NOTES
-/- Network failure will never be raised in the current implementation.
-/- It is left in the code as place holder for future expansion.
-/-
with Kernel Exceptions;
   -- to get: change results in negative elapsed time
   -- change results in negative_epoch_time
   -- network failure
   -- OK but time already passed
   -- synchronization_in_progress
   -- synchronization_timeout_expired

with Schedule Types;
   -- to get: current_process_priority
       -- priority

with Time Global;
   -- to get: elapsed_time
       -- epoch_time

generic
   
   -- error checking conditional compilation flags
   
   -- REQ: 5.1.8
   
   Change Results In Negative Elapsed Time Enabled : In Boolean := True;
   Change Results In Negative_Epoch Time Enabled : In Boolean := True;
   Network Failure Enabled : In Boolean := True;
   OK But Time Already Passed Enabled : In Boolean := True;
   Synchronization In Progress Enabled : In Boolean := True;
   Synchronization Timeout Enabled : In Boolean := True;

   Pragma(Page);
   ...
   "package short name: GTM"
   ...

package Generic_Time_Management
is
   
   -- REQ: 6.1.6, 12.1.2, 12.1.3, 12.1.7, 12.1.9, 12.1.11
   -- REQ: 12.1.13, 12.1.14
   
   -- change results in negative_elapsed_time raised by:
      -- adjust_elapsed_time
      -- PRIM: 21.1.3
   
   Change Results In Negative Elapsed Time : exception
      renames Kernel_Exceptions.Change_Results_In_Negative_Elapsed_Time;
   
      -- change results in negative_epoch_time raised by:
      -- adjust_epoch_time
      -- PRIM: 21.1.4

   Change Results In Negative Epoch Time : exception
      renames Kernel_Exceptions.Change_Results_In_Negative_Epoch_Time;
   
      -- network failure raised by:
      -- synchronize
      -- PRIM: 21.1.6

   Network Failure : exception
      renames Kernel_Exceptions.Network_Failure;
   
      -- OK but time already passed raised by:
      -- adjust_epoch_time
      -- PRIM: 21.1.4

   Ok But Time Already Passed : exception
      renames Kernel_Exceptions.Ok_But_Time_Already_Passed;
   
      -- synchronization_in_progress raised by:
      -- synchronize
      -- PRIM: 21.1.6

   Synchronization In Progress : exception
      renames Kernel_Exceptions.Synchronization_In_Progress;
   
      -- synchronization_timeout expired raised by:
      -- synchronize
      -- PRIM: 21.1.6

   Synchronization Timeout : exception
      renames Kernel_Exceptions.Synchronization_Timeout_Expired;

   pragma (page);
procedure Adjust Elapsed Time (Adjustment: In TimeGlobals.Elapsed_Time);
    
    /* SUBPROGRAM NAME:
    */
    
    /* DESCRIPTION:
    */
    
    /* allows the application to increment or decrement the current local
    */
    
    /* elapsed time by a specified amount of elapsed time
    */
    
    /*
    */
    
    /*-REQ: 12.1.10
    */
    
    /*-PRIM: 21.1.3
    */
    
    /*-TIME: 12.2.1
    */
    
    /* PARAMETERS:
    */
    
    /* adjustment
    */
    
    /* the amount of elapsed time by which the elapsed time should be
    */
    
    /* advanced (if adjustment is greater than or equal to zero) or
    */
    
    /* backed up (if adjustment is less than zero)
    */
    
    /* default value:
    */
    
    /* none
    */
    
    /* PRECONDITIONS:
    */
    
    /* NONE
    */
    
    /* ACTIONS:
    */
    
    /* the elapsed time counter is adjusted (advanced or backed up) by the
    */
    
    /* specified amount of elapsed time
    */
    
    /* POSTCONDITIONS:
    */
    
    /* the elapsed time counter is adjusted
    */
    
    /* this affects ALL events pending a time-triggered action - both
    */
    
    /* elapsed time and epoch time
    */
    
    /* ERROR PROCESSING:
    */
    
    /* when the requested change results in negative elapsed time since
    */
    
    /* start up:
    */
    
    /* the Kernel raises change_results_in_negative_elapsed_time
    */
    
    /* exception
    */
    
    /* POSTCONDITIONS:
    */
    
    /* the requested change does not occur
    */
    
    /*
    */
    
    pragma (page);

procedure Adjust Epoch Time (New Epoch Time: In TimeGlobals.Epoch_Time);
    
    /* SUBPROGRAM NAME:
    */
    
    /* DESCRIPTION:
    */
    
    /* allows the application to set the current local epoch time to a
    */
    
    /* specified new epoch time
    */
    
    /*-REQ: 12.1.12
    */
    
    /*-PRIM: 21.1.4
    */
    
    /*-TIME: 12.2.2
    */
    
    /* PARAMETERS:
    */
    
    /* new_epoch_time
    */
    
    /* the new value for the local epoch time
    */
    
    /* default value:
    */
    
    /* none
    */
    
    /* PRECONDITIONS:
    */
    
    /* NONE
    */
    
    /* ACTIONS:
    */
    
    /* the epoch time counter is set to the specified new_epoch_time
    */
    
    /* POSTCONDITIONS:
    */
    
    /* the epoch time counter is adjusted
    */
    
    /* this affects ONLY those events pending an epoch-time-triggered action
    */
    
    /* ERROR PROCESSING:
    */
    
    /* when the requested change results in a negative epoch time:
    */
    
    /* the Kernel raises change_results_in_negative_epoch_time
    */
    
    /* POSTCONDITIONS:
    */
    
    /* the change does not occur
    */
    
    /* when the requested change results in a backward flow of time:
    */
    
    /* the Kernel raises OK but_time_already_passed exception
    */
    
    /* POSTCONDITIONS:
    */
    
    /* the change occurs as requested
    */
    
    /*
    */
    
    pragma (page);
function Read_Clock
  return Time_Globals.Epoch_Time;

procedure Synchronize (Resumption_Priority; In Schedule Types.Priority :=
  Schedule_Type.Current; Process_Priority);

pragma (page):
procedure Synchronize (Timeout After : In Time, Globals Elapsed Time;
Resumption Priority : In Schedule Types.Priority :=
Schedule Types.Current Process.Priority);
-------------------------------------------------------------------------------
// SUBPROGRAM NAME:
// synchronize

// DESCRIPTION:
// forces all local processor clocks on Kernel devices to synchronize
// with the local clock on the invoking processor
//
// --REQ: 12.1.16, 12.1.17, 12.1.18, 12.1.20, 12.1.21, 12.1.22, 12.1.27
// ...
// --PRIM: 21.1.6
// ...
// --TIME: 12.2.4, 12.2.5

// PARAMETERS:
// timeout after
// duration after which the process rescinds its attempt to
// synchronize processor clocks and becomes unblocked
//
// default value:
// none
//
// resumption priority
// the priority to be assigned to this process when it becomes
// unblocked
//
// default value:
// the calling process's current priority

// PRECONDITIONS:
// there is no call to synchronize currently executing
//
// ACTIONS:
// the calling processor indicates that all network traffic is to
// cease
// the calling processor sends its elapsed time and epoch time to each
// processor on the network
// the calling processor waits until the full two-phase commit protocol
// (ready and received/implemented change) has been performed by all
// processors on the network, then unblocks network traffic

// POSTCONDITIONS:
// all clocks are synchronized

// ERROR PROCESSING:
// when this primitive is called in a context that prohibits blocking
// (e.g., within an interrupt handler) and this call would block
// (i.e., all cases)
// the Kernel raises blocking prohibited exception
// POSTCONDITIONS
// the synchronize does not occur
procedure Synchronize (
  Timeout At: In Time Global's Epoch Time,
  Resumption Priority: In Schedule Types Priority —
  Schedule Types Current Process Priority);
/---------------------------------------------------------------------------
/ SYNOPSIS NAME:
/ synchronize
/ DESCRIPTION
/ forces all local processor clocks on Kernel devices to synchronize
/ time with the local clock on the invoking processor
/ REQUIREMENTS:
/ - REQ: 12.1.16, 12.1.17, 12.1.18, 12.1.20, 12.1.21, 12.1.22, 12.1.27
/ - PRI: 21.1.6
/ - TIME: 12.2.4, 12.2.5
/ PARAMETERS:
/ timeout at
/ absolute time at which the process resinds its attempt to
/ synchronize processor clocks and becomes unblocked
/ default value:
/ none
/ resumption priority
/ the priority to be assigned to this process when it becomes
/ unblocked
/ default value:
/ the calling process's current priority
/ PRECONDITIONS:
/ there is no call to synchronize currently executing
/ ACTIONS
/ the calling processor indicates that all network traffic is to
/ cease
/ the calling processor sends its elapsed time and epoch time to each
/ processor on the network
/ the calling processor waits until the full two-phase commit protocol
/ (ready and received/implemented change) has been performed by all
/ processors on the network, then unblocks network traffic
/ POSTCONDITIONS
/ all clocks are synchronized
/ ERROR PROCESSING
/ when this primitive is called in a context that prohibits blocking
/ (e.g. within an interrupt handler) and this call would block
/ (i.e. all cases)
/ the Kernel raises blocking prohibited exception
/ POSTCONDITIONS
/ the synchronize does not occur
the exact state of network time is not known

the Kernel raises network failure exception

POSTCONDITIONS

the synchronization does not occur

when more than one call to synchronize occurs

when the synchronization is abandoned

when the synchronization timed out exception:

the exact state of network time is not known

the Kernel raises synchronization in progress exception

POSTCONDITIONS

the synchronization is abandoned

when the synchronization timed out exception

when the synchronization is abandoned

the exact state of network time is not known

end Generic Time Management;

pragma (page);
with Generic Time Management;
pragma Elaborate (Generic Time Management);
...to get see MODULE CONTENTS

package Time Management

in new Generic Time Management |
Change Results In Negative Elapsed Time Enabled => True,
Change Results In Negative Epoch Time Enabled => True,
Ok But Time Already Passed Enabled => True;
pragma (page);

MODULE NAME:
generic_timeslice_management

MODULE TYPE:
generic package specification

MODULE PURPOSE:
support timeslice scheduling of processes

MODULE DESCRIPTION:
collection of subprograms to define the timeslice quantum and to
enable and disable timeslicing

MODULE CONTENTS:
generic timeslice management (generic package specification)
disable time slicing (procedure)
enable time slicing (procedure)
illegal quantum (exception)
illegal quantum enabled (generic formal parameter)
minimum slice time (constant)
minimum slice time value (generic formal parameter)
set timeslice (procedure)
Default timeslice value (generic formal parameter)

REFERENCES:
DESIGN DOCUMENTS:
Kernel Facilities Definition
REQ 5.1.8
REQ 9.1.1, 9.1.4, 9.1.5, 9.1.10, 9.1.11, 9.1.24, 9.1.26
PRIM 18.1.6
TIME 9.2.4, 9.2.6, 9.2.7

USER'S MANUAL:
Kernel User's Manual

TESTING AND VALIDATION:
Unit testing

NOTES:
none

MODIFICATION HISTORY:
18aug88 bamberg : created (completes clean)
28sep88 dna Changed to use package short names (to
conform to package body)
13dec88 dna Added generic parameter minimum_slice_time
24mar88 dna Bug report 0003 set
Added ability to specify the initial default
timeslice.

VERSION 3.0
Disclaimer:
The following copyright must be included in this software and all software utilizing this software.

DISTRIBUTION:
Approved for public release, distribution is unlimited.

Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh, PA. The Software Engineering Institute (SEI) is a federally funded research and development center established and operated by Carnegie Mellon University (CMU). Sponsored by the U.S. Department of Defense under contract F19628-85-C-0003, the SEI is supported by the services and defense agencies, with the U.S. Air Force as the executive contracting agent.

Permission to use, copy, modify, or distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission appear in supporting documentation. Further, the names of Software Engineering Institute or Carnegie Mellon University may not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. CMU makes no claims or representations about the suitability of this software for any purpose. This software is provided "as is" and without warranty, express or implied, it is made by the SEI or CMU, as to the accuracy and functioning of the program and related program material, nor shall the fact of distribution constitute any such warranty. No responsibility is assumed by the SEI or CMU in connection herewith.

pragma (page);
with Kernel Exceptions;
-- to get: illegal quantum

with Time Globals;
-- to get: elapsed_time, to_elapsed_time

with Kernel Time;
-- to get: kernel_time

Generic
--
-- ERROR checking conditional compilation flags
--
--REQ: 5.1.8

Illegal Quantum Enabled: In Boolean := True;

-- minimum amount of time that can be specified as a timeslice interval

Minimum Slice Time Value: In Time,Globals,Elapsed Time;

-- the default amount of time allocated to a process before it
-- may be preempted

-- default value:
-- 1 second

Default Timeslice Value: Time,Globals,Elapsed Time :=
Time(Globals,To,Elapsed Time)
Kernel Time,Kernel Time,
(High => 0, Low => 1 000 000));

package short name: GTSM
package Tg
renames Time,Globals;
package Ke
renames Kernel Exceptions:

- TIME: 9.2.4, 9.2.6, 9.2.7

- Illegal quantum raised by:
  - set_timeslice
  - PRRM: 19.1.6

Illegal Quantum: exception renames KeIllegal Quantum;

- The minimum amount of time that can be specified as a timeslice interval

Minimum Slice Time: constant TgElapsed_Time = Minimum_Slice_Time_Value;

procedure set Timeslice (Quantum in TgElapsed Time);

- SUBPROC: PARAM NAME:
- set_timeslice

- DESCRIPTION:
- set the timeslice quantum in elapsed time units
- REQ: 9.1.18
- PRRM: 19.1.7
- TIME: N/A
- PARAMETERS:
- quantum
- the quantum of time allocated to a process for execution
- default value:
- none
- PRECONDITIONS:
- NONE
- ACTIONS:
- the Kernel changes the length of the timeslice immediately and does not cause a reschedule point
- POSTCONDITIONS:
- the length of the timeslice is adjusted as specified (which does not affect a previously pending timeslice for the currently running process)
- ERROR PROCESSING:
- when a quantum is specified that is less than the value of minimum slice time
- the Kernel raises illegal quantum exception
- POSTCONDITIONS:
- the requested change does not take place
- the length of the timeslice is set to the minimum slice time value

premata (page):
procedure Enable Time Slicing;

/\ SUBPROGRAM NAME:
  /\ enable_time_slicing
  /\ DESCRIPTION:
  /\ enables round-robin, timeslice scheduling among processes of equal
  /\ priority
  /\ -REQ: 9.1.19
  /\ -PRIM: 18.1.8
  /\ -TIME: 9.2.5
  /\ PARAMETERS:
  /\ none
  /\ PRECONDITIONS:
  /\ NONE
  /\ ACTIONS:
  /\ the Kernel enables time slicing immediately, which may later affect
  /\ the currently running process, even though this is not a reschedule
  /\ point
  /\ POSTCONDITIONS:
  /\ time slicing is enabled for the processor
  /\ CAVEATS:
  /\ it is possible to call enable_time_slicing without first calling
  /\ set_timeslice. In this case, the default value specified for
  /\ TSP timeslice duration is used as the timeslice amount. Note
  /\ that this default value is not user configurable (due to a
  /\ design error, and not due to any implementation restrictions).
  /\ it is recommended that the first use of enable_time_slicing always
  /\ be preceded by a call to set_timeslice.
  /\ ERROR PROCESSING:
  /\ none
  /\ ________________________________

pragma (page);

procedure Disable Time Slicing;

/\ SUBPROGRAM NAME:
  /\ disable_time_slicing
  /\ DESCRIPTION:
  /\ disables round-robin, timeslice scheduling among processes of equal
  /\ priority
  /\ -REQ: 9.1.20
  /\ -PRIM: 18.1.9
  /\ -TIME: 9.2.5
  /\ PARAMETERS:
  /\ none
  /\ PRECONDITIONS:
  /\ NONE
  /\ ACTIONS:
  /\ the Kernel disables time slicing immediately, which may affect the
  /\ currently running process, even though this is not a reschedule
  /\ point
  /\ POSTCONDITIONS:
  /\ time slicing is disabled for the processor
  /\ ERROR PROCESSING:
  /\ none
  /\ ________________________________

end Generic Timeslice Management;
pragma (page);
package Timeslice Management

is new Generic Timeslice Management {
  Illegal Quantum Enabled -> True,
  This value is set to 77 microseconds. Note that this is less than the minimum context switch time (KFD : 2.6).
}

Minimum Slice Time Value -> Time Globals.T1 Elapsed Time;

Kernel Time Kernel Time((4,12)+0,1,low>77); pragma (page);

package Timeslice Management
**MODULE NAME:**
- context save area

**MODULE TYPE:**
- package specification

**MODULE PURPOSE:**
- to encapsulate the hardware-specific information for saving the
  context of a Kernel process

**MODULE DESCRIPTION:**
- types and template for the hardware-specific information for saving the
  context of a Kernel process

**MODULE CONTENTS:**
- context save area (package specification)
  - address registers area (type)
  - context save area contents (type)
  - data registers area (type)
  - floating point coprocessor area (type)

**REFERENCES:**
- DESIGN DOCUMENTS:
  - Kernel Facilities Definition
  - -REQ 5.14
  - -TIME N/A

**USER'S MANUAL:**

**TESTING AND VALIDATION:**
- Integration testing with Scheduler

**NOTES:**
- *** THIS IS HARDWARE SPECIFIC ***

- *** NOTE THAT THE REP SPECS FOR ALL RECORDS ARE AT MOD 2 - WHICH LEAVES
  *** STUFF AT 16 BIT BOUNDARIES ... THESE REALLY SHOULD BE A MOD 4 FOR
  *** THE HARDWARE, BUT TELESOFT COMPILED DOESN'T SUPPORT THAT!!!

**MODIFICATION HISTORY:**
- 18aug88 tammberg created (compiles clean)
- 02sep88 tammberg added reg spec required by hardware instructions
- 11jan89 8th final doc update for beta release (no change)

**VERSION 3.0**

**DISCLAIMER:**

The following copyright must be included in this software and

---

all software utilizing this software.

---

**DISTRIBUTION:**

---

Approved for public release; distribution is unlimited

---

Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
-PA. The Software Engineering Institute (SEI) is a federally
-funded research and development center established and operated
-by Carnegie Mellon University (CMU). Sponsored by the U.S.
-Department of Defense under contract F19628-85-C-0003, the SEI
-is supported by the services and defense agencies, with the U.S.
-Air Force as the executive contracting agent.

---

Permission to use, copy, modify, or distribute this software and
-its documentation for any purpose and without fee is hereby
-granted, provided that the above copyright notice appear in all
-copies and that both this copyright notice and this permission
-notice appear in supporting documentation. Further, the names
-Software Engineering Institute or Carnegie Mellon University may
-not be used in advertising or publicity pertaining to
-distribution of the software without specific written prior
-permission. CMU makes no claims or representations about the
-suitability of this software for any purpose. This software is
-provided "as is" and no warranty, express or implied, is made
-by the SEI or CMU, as to the accuracy and functioning of the
-program and related program material, nor shall the fact of
-distribution constitute any such warranty. No responsibility is
-assumed by the SEI or CMU in connection herewith.

---

pragma (page);
pragma (page);

with Hardware Interface; use Hardware Interface;
... to get: bits per byte
... hw_long Integer
... longword

package Context_Save_Area

is

... "THIS IS HARDWARE SPECIFIC"

... on the 68020, there are eight address registers, each 32 bits long

... 

type Address Registers_Area is record
A0 : Hw Long Integer;
A1 : Hw Long Integer;
A2 : Hw Long Integer;
A3 : Hw Long Integer;
A4 : Hw Long Integer;
A5 : Hw Long Integer;
A6 : Hw Long Integer;
A7 : Hw Long Integer;
end record;

for Address Registers_Area
... TS use record at mod Hardware Interface longword;
... TS this is to get around a telesoft limitation: ideally, this should be
... TS allocated on a longword boundary, but telesoft does not support that
... TS use reps
... use record at mod Hardware Interface Word;
A0 at 0 * Hardware Interface Longword range 0 .. 31;
A1 at 1 * Hardware Interface Longword range 0 .. 31;
A2 at 2 * Hardware Interface Longword range 0 .. 31;
A3 at 3 * Hardware Interface Longword range 0 .. 31;
A4 at 4 * Hardware Interface Longword range 0 .. 31;
A5 at 5 * Hardware Interface Longword range 0 .. 31;
A6 at 6 * Hardware Interface Longword range 0 .. 31;
A7 at 7 * Hardware Interface Longword range 0 .. 31;
end record;
... this assumes a byte-addressable machine

Number Address Registers : constant := 8;

Address Registers Area Size : constant :=
Number Address Registers * Hardware Interface Bits Per Byte *
Hardware Interface Longword;

for Address Registers Area Size use Address Registers Area Size,
```
-- """" THIS IS HARDWARE SPECIFIC """

-- on the 68020, there are eight data registers, each 32 bits long

type Data Registers Area is record
  D0 : Hw Long Integer;
  D1 : Hw Long Integer;
  D2 : Hw Long Integer;
  D3 : Hw Long Integer;
  D4 : Hw Long Integer;
  D5 : Hw Long Integer;
  D6 : Hw Long Integer;
  D7 : Hw Long Integer;
end record;

for Data Registers Area
  TS use record at mod hardware Interface.Longword;
  TS this is to get around a telesoft limitation: ideally, this should be
  TS allocated on a longword boundary, but telesoft does not support that
  TS via rep specs
    use record at mod Hardware Interface Word;
    D0 at 0 * Hardware Interface.Longword range 0..31;
    D1 at 1 * Hardware Interface.Longword range 0..31;
    D2 at 2 * Hardware Interface.Longword range 0..31;
    D3 at 3 * Hardware Interface.Longword range 0..31;
    D4 at 4 * Hardware Interface.Longword range 0..31;
    D5 at 5 * Hardware Interface.Longword range 0..31;
    D6 at 6 * Hardware Interface.Longword range 0..31;
    D7 at 7 * Hardware Interface.Longword range 0..31;
end record;

Number Data Registers : constant := 8;

Data Registers Area Size : constant :=
  Number Data Registers * Hardware Interface.Bits_Per_Byte * Hardware Interface.Longword;

for Data Registers Area Size use Data Registers Area Size;
```

```
-- """" THIS IS HARDWARE SPECIFIC """

-- on the 68020, up to 280 bytes are used to save the state of the
-- M68881 floating point coprocessor, when context is saved or restored,
-- the address of this save area is passed to a hardware instruction that
-- performs all the necessary manipulation for saving and restoring the
-- M68881 floating point coprocessor

Number Bytes Floating Point Coprocessor Save Area : constant := 280;

type Floating Point Coprocessor_Type is array (1..Hw Long Integer(1)) of Hw Long Integer;
  -- this assumes 4 bytes per word (hw long integer)
  -- this assumes that ONLY 280 bytes are allocated for this type;
  -- if this proves to be false, it must be changed to a record type and
  -- have rep specs written for it (as ada does not provide any rep specs
  -- to assert sizes of array types)

Floating Point Coprocessor Size : constant :=
  Number Bytes Floating Point Coprocessor Save Area *
  Hardware Interface.Bits_Per_Byte;

for Floating Point Coprocessor_Type Size use Floating Point Coprocessor_Size;

pragma (page);
```
*** THIS IS HARDWARE SPECIFIC ***
* the information required to be saved for a process context switch
* includes:
  * data registers
    * the contents of the 68020's eight data registers, this is used by
      * the Kernel's context switcher
    * default value:
      * none
  * address registers
    * the contents of the 68020's eight address registers, this is used
      * by the Kernel's context switcher
    * default value:
      * none
  * status register
    * the contents of the 68020's status register; this is used by the
      * Kernel's context switcher
    * default value:
      * none
  * program counter
    * the contents of the 68020's program counter; this is used by the
      * Kernel's context switcher
    * default value:
      * none
  * floating point coprocessor
    * the contents of the 68020's floating point coprocessor
    * default value:
      * none

---

Address Registers Offset : constant := 0;
Status Register Offset : constant := 14;
Program Counter Offset : constant := 17;
Floating Point Coprocessor Offset : constant := 18;

Status Register Size : constant :=
  Hardware_Interface.Longword * Hardware_Interface.Bits_Per_Byte;

Program Counter Size : constant :=
  Hardware_Interface.Longword * Hardware_Interface.Bits_Per_Byte;

for Context Save Area Contents
  - TS use record at mod hardware Interface.Longword;
  - TS this is to get around a telosoft limitation: ideally, this should be
  - TS allocated on a longword boundary, but telosoft does not support that
  - TS via rep spec

use record at mod Hardware_Interface.Word;
Data Registers at Data Registers_Offset *
  Hardware_Interface.Longword
range 0 .. Data Registers.Area.Size - 1;
Address Registers at Address Registers_Offset *
  Hardware_Interface.Longword
range 0 .. Address Registers.Area.Size - 1;
Status Register at Status Register_Offset *
  Hardware_Interface.Longword
range 0 .. Status Register.Size - 1;
Program Counter at Program Counter_Offset *
  Hardware_Interface.Longword
range 0 .. Program Counter.Size - 1;
Floating Point Coprocessor at Floating Point Coprocessor_Offset *
  Hardware_Interface.Longword
range 0 .. Floating Point Coprocessor.Size - 1;

end record;

for Context Save Area Contents .size use
Data Registers .Size *
Address Registers .Size *
Status Register .Size *
Program Counter .Size *
Floating Point Coprocessor .Size *

end Context Save Area;

287 288
December 1989  December 1989
// single enumerated type, added acknowledged
// message information record
// 14sep88 dhv moved message identifier to datagram globals
// 14sep88 rvs change type of maximum message queue size to
// tw long, natural (to allow zero queue length)
// 14sep88 rvs changed null_process to a real null process,
// and not just "null"
// 17nov88 rvs added locate_process function, used to locate
// a process by a brute force search of the process
// table for the logical name
// 12jan89 brth modified definition of semaphore and added
// ancillary types semaphore head (ptr)
// 15feb89 bamberg updated documentation on stack high address
// to have example consistent with longword
// alignment
// 03apr89 rvs added next field to scheduling attributes
// 03apr89 rvs added current_receive_buffer_address and
// current_receive_buffer_size
// 12may89 rvs added real task interface structure
// 16may89 rvs added size of process table entry
// 27aug89 bamberg final doc. check for 01sep89 freeze

// VERSION 3.0

// DISCLAIMER:

// The following copyright must be included in this software and
// all software utilizing this software.

// DISTRIBUTION

// Approved for public release, distribution is unlimited.

// Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh.
// PA. The Software Engineering Institute (SEI) is a federally
// funded research and development center established and operated
// by Carnegie Mellon University (CMU). Sponsored by the U.S.
// Department of Defense under contract F19628-85-C-0003, the SEI
// is supported by the services and defense agencies, with the U.S.
// Air Force as the executive contracting agent.

// Permission to use, copy, modify, or distribute this software and
// its documentation for any purpose and without fee is hereby
// granted, provided that the above copyright notice appear in all
// copies and that both that copyright notice and this permission
// notice appear in supporting documentation. Further, the names
// Software Engineering Institute or Carnegie Mellon University may
// not be used in advertising or publicity pertaining to
// distribution of the software without specific written prior
// permission. CMU makes no claims or representations about the
// suitability of this software for any purpose. This software is
// provided "as is" and no warranty, express or implied, is made
// by the SEI or CMU, as to the accuracy and functioning of the
with Communication Globals;
  ... to get: message, length, type

with Context Save Area;
  ... to get: context, save_area, contents

with Context Switcher Globals;
  ... to get: context, saved_type

with Datagram Globals;
  ... to get: datagram pointer
          ... message, identifier

with Generic Queue Manager;
  ... to get: all utilities

with Hardware Interface; use Hardware Interface;
  ... to get: hw address
          ... hw integer
          ... hw long natural
          ... hw natural
          ... hw positive
          ... hw string

with Kernel Time;
  ... to get: Kernel time
          ... zero, Kernel time

with Kernel Exceptions;
  ... to get: Kernel exceptions

with Network Globals;
  ... to get: process, index, type

with Process Managers Globals;
  ... to get: how to handle message, queue overflow
          ... maximum length, of process name

with Schedule Types;
  ... to get: preemption
          ... priority
          ... process state

with Time Keeper Globals;
  ... to get: event type
          ... time class type

with Tool Interface Globals;
  ... to get: collection type,

pragma (page);

generic

  ... value indicating the maximum number of processes in the Process Table
  ... (this value is never used within the Kernel except for the initialization
  ... of the size of the Process Table, nor should it be used without the
  ... Kernel)

  ... this value indicates the maximum size of the Process Table as determined
  ... when initialization is complete, not during the initialization process

  ... REQ: 5.1.8

Maximum Number_Of_Processes_Value : In Hw_Positive;

package Generic_Process_Table

  is
    ... short name: GPTB

    ... value indicating the maximum number of processes in the Process Table
    ... this value indicates the maximum size of the Process Table as determined
    ... when initialization is complete, not during the initialization process

    ... REQ: 5.1.8

Maximum Number_Of_Processes : constant Hw_Positive :=
    Maximum Number_Of_Processes_Value;

/***************************************************************************************

    ... a process identifier is used to access process table entries efficiently

    ... type Process Information Record;

    ... type Process Identifier is access Process Information Record;

    ... a null process identifier is just a pointer to an empty process (this
    ... is set in initialize_process_table)

    Null Process : Process Identifier;

pragma (page):
An event identifier is used by the Kernel's time keeper to access time event queue entries efficiently.

Event information is the template used to represent each of the sequence of events that are understood by the Kernel to happen some time in the future (i.e., the template for each entry in the time event queue).

The information maintained for each time event is:

- Kind of event
- The event type causing the creation of this time event
- Possible values:
  - Alarm (via a call to set alarm)
  - Receive timeout (via a call to receive message)
  - Semaphore timeout (via a call to claim)
  - Wait timeout (via a call to wait)
  - Send with ACK timeout (via receiving a message indicating a remote timeout is required)
  - Active expiration (time slicing was previously enabled)
- Default value:
  - None

This value is assigned when an event information record is added to the time event queue and should never change as long as it is still in the time event queue.

Time class:
The type of time specified by the application (via a call to any Kernel primitive that provides a time parameter).

Possible values:
- Elapsed (the application specified an elapsed time)
- Epoch (the application specified an epoch time)

Default value:
- None

This value is assigned when an event information record is added to the time event queue and should never change as long as it is still in the time event queue.

Expiry:
The Kernel time at which the time event expires; the time in each entry is that which was specified via the Kernel primitive called.

Default value:
- None

This value is assigned when an event information record is added to the time event queue; it may change only via a call to one of the Kernel primitives: adjust elapsed time (for all entries in the time event queue) or reset epoch time (only for those time event queue entries with time class = epoch).

Process ID:
The concrete process identifier for which the time event is maintained (i.e., a pointer into the process table for the process "owning" this event).

Default value:
- None

This value is assigned when an event information record is added to the time event queue and should never change as long as it is still in the time event queue.

Type Event Information is record
Kind Of Event: Time Keeper Globals Event Type;
Time Class: Time Keeper Globals Time Class Type;
Expiry: Kernel Time Kernel Time;
Process Id: Process Identifier;
End Record;
End Record;
```plaintext
function "<=" (left: Event Information; right: Event Information) return Boolean,

"=": ---------------------------------------------------------------

SUBPROGRAM NAME:
"<="

DESCRIPTION:
defines the "less than or equal to" operation on two
event information records in terms of comparing their expiry
components.
- REQ: 5.14
- PRIM: NA
- TIME: NA

PARAMETERS:
- left
  the left operand of the comparison between two event information
  records
  default value:
  none
- right
  the right operand of the comparison between two event information
  records
  default value:
  none
- function return value
  result of the comparison
  values include:
  true (the left operand orders "less than or equal to" the right)
  false (the left operand does not order "less than or equal to"
  the right)

PRECONDITIONS:
- none

ACTIONS:
return true if left expiry <= right expiry

POSTCONDITIONS:
- none

ERROR PROCESSING:
- none

--------------------------------------------------------------------------------
prefix (page);
```

package Time_Event_Queue

in new Generic Queue Manager {
  Element Type => Event Information,
  "<=" => "<=";
}

subtype Event Identifier is Time_Event_Queue.Queue_Pointer;

Null Event : Event Identifier => null;
prefix (page);
To allow a semaphore to function reliably, two things
must be guaranteed

- (a) operations upon semaphores must use sharing semantics, ie
to operations applied concurrently to the same semaphore
must access the exact same data structure
- (b) all semaphores must be initialized upon declaration

To ensure (a), the implementation allocates all semaphore objects
and references them by access values

To ensure (b), all types associated with semaphores are record types
with default initialization of components

The three types that are declared below are

- Semaphore
  - the type of a semaphore object
- Semaphore Head
  - an access type designating a semaphore head
- Semaphore
  - a record type whose only component is a semaphore_head, pointer
    with appropriate initialization
    - The last type - Semaphore - is the only one that the primitives
      operate on, and the only one of interest to the application
    - the information maintained each semaphore head is:
      number of waiting processes
      the number of processes in the waiting queue for the semaphore
      default value:
      - 0 (there are no processes waiting and the semaphore is free);
      - the value 0 indicates that the semaphore is claimed and
        there are no processes waiting for it, any positive value
        indicates that the semaphore is currently claimed and there
        are positive value number of processes waiting for the
        semaphore
      - this value is incremented via a call to the Kernel primitive:
        claim and decremented via a call to the Kernel primitive: release,
      - otherwise, this value should never change
- queue: head
  - the first process waiting in the queue for this semaphore
  - default value:
    - null process (there are no processes waiting in the queue for
      this semaphore)
  - this value may be set via a call to the Kernel primitive: claim
    (if the call is the first process waiting for the semaphore)
    and reset via a call to the Kernel primitive: release (if the call
    is for the last process waiting for the semaphore); otherwise,
    this value should never change
  - sema previously claimed
    - the last semaphore claimed by the process that owns this semaphore
      (ie, if process P claims semas S1 and S2 in that order, then this
      component of S2 head will designate S1)
  - default value: null semaphore
  - The purpose of this component is to chain together in LIFO order
    all semas currently owned by a process.

- REQ: 8.1.1, 8.1.2
  - PRIM: 17.1.1
  - TIME: 8.2.1

- preliminary incomplete type declaration

  type Semaphore Head;
  - the access type, to allow sharing semantics on semaphore objects

  type Semaphore Head_Ptr is access Semaphore Head;
  - the type of a semaphore object proper, with correct initialization

  type Semaphore Head is record
  Number Of Waiting Processes : HW Integer := -1;
  Queue Head : Process Identifier := Null Process;
  Sema Previously Claimed : Semaphore Head_Ptr := null;
  end record;
  - the type visible to the application, with correct initialization
    - each declaration thus creates a new semaphore object, which will
      be automatically initialized

  type Semaphore is record
  Head : Semaphore Head_Ptr := new Semaphore Head;
  end record;
  - the null semaphore value
-- this must be the initial value of all semaphore variables
-- that are "working" variables, i.e. that are not associated with
-- unique semaphore objects but rather that are assigned semaphore
-- values dynamically

Null Semaphore : constant Semaphore := (Head => null);
pragma (page);

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

-- Indication of current status of process initialization protocol
-- (i.e., process declaration and creation)
-- components include
-- declared
-- indication of whether or not declare_process successfully completed
--
-- values include:
-- true (successful completion of declare_process)
-- false (unsuccessful completion of declare_process)
--
-- default value:
-- false (unsuccessful completion of declare_process)
--
-- this value is set via a call to the Kernel primitive:
-- declare_process; it should never change after this call
--
-- created
-- indication of whether or not create_process successfully completed
--
-- values include:
-- true (successful completion of create_process)
-- false (unsuccessful completion of create_process)
--
-- default value:
-- false (unsuccessful completion of create_process)
--
-- this value is set via a call to the Kernel primitive:
-- create_process; it should never change after this call
--
-- remotely_created
-- indication of whether or not the process was created on another
-- node
--
-- values include:
-- true (process was created on a remote node)
-- false (process was not created remotely)
--
-- default value:
-- false (no knowledge yet about where the process was created)
--
-- This value is set by the receive_datagram_interrupt_handler when
-- over a process_create message arrives.

-- REQ: 7.1.4
-- PRIM: 16.1.1, 16.1.2

type Process Initialization Status Type is record
   Declared : Boolean := False;
   Created  : Boolean := False;
   Remotely Created : Boolean := False;
end record;
... indication of whether or not a process is running on a Kernel processor
... (and thus follows all Kernel protocols) or a process is really just a
... non-Kernel device (and thus follows none of the Kernel protocols)
... values include:
... Kernel_process (follows Kernel protocols)
... Non_Kernel_device (does not follow Kernel protocols)
... PRM: 16.1.1
...

define Process_Type is
    Kernel_Process, Non_Kernel_Device;
pragma (page);

... process attributes include:
... logical_name
... the string valued name provided by the application for a process;
... the length of the string is limited by the user-provided value
... for the maximum length of a process name
... default_value:
... none
... this value is set via a call to the Kernel primitive:
... declare_process; it should never change after that call
... kind_of_process
... indication of whether a process is running on a Kernel processor
... (and thus follows all Kernel protocols) or a process is really
... just a non-Kernel device (and thus follows none of the Kernel
... protocols)
... values include:
... Kernel_process (follows Kernel protocols)
... Non_Kernel_device (does not follow Kernel protocols)
... default_value:
... Kernel_process
... this value is set via a call to the Kernel primitive:
... declare_process; it should never change after that call
... process_initialization_status
... indication of current status of process initialization protocol
... (i.e., process declaration and creation)
... default_value:
... declared => false (set by declare_process)
... created => false (set by create_process)
... this value is set via a call to the Kernel primitives:
... declare_process (the declared component) and create_process (the
... created component); it should never change after the call
... process_index
... another way of referencing a process (in addition to a process
... identifier) via its owning processor and an identifier unique with
... respect to that processor
... default_value:
... none
... for a non-Kernel device, this value is set via a call to the Kernel
... primitive: declare_process; it should never change after that call;
... for a Kernel device, this value is set via a call to the Kernel
... primitive: create_process; it should never change after the call
... code_address
the address of the code that comprises this process

default value:
none

this value is set via a parameter to the Kernel primitive:
create_process, it should never change after that call

stack low address
the system low address (e.g., $160000$) of the Kernel-created
process stack; this is the FIRST longword address (i.e., aligned
on a 32-bit boundary) at which the Kernel may safely store a
longword (i.e., 32 bits) of data in the Kernel-maintained
process stack

*** the Kernel-maintained process stack is always longword-aligned

default value:
none

this value is set via a call to the Kernel primitive:
create_process, it should never change after that call

stack high address
the system high address (e.g., $16FFFF$) of the Kernel-created
process stack; this is the LAST longword address (i.e., aligned
on a 32-bit boundary) at which the Kernel may safely store a
longword (i.e., 32 bits) of data in the Kernel-maintained
process stack

*** the Kernel-maintained process stack is always longword-aligned

default value:
none

this value is set via a call to the Kernel primitive:
create_process, it should never change after that call

context, saved
indication of whether or not the current context of this process
may be assumed to be saved (i.e., that the context_save_area has
contents that are currently valid)

values include:
via call (context is saved and was saved via the
procedure/function calling protocol)
via interrupt (context is saved as was saved via the
interrupt handling protocol)
not saved (context must not be assumed to be saved)

default value:
via call

this value is set initially via a call to the Kernel primitive:
create_process, it is modified by the Kernel as process context is

saved and restored (when a process context switch occurs as
directed by the Scheduler or when an interrupt occurs)

process_context_save_area
place where the context of a process is saved (e.g., registers,
program counter)

default value:
none

this value is modified by the Kernel as process context is
saved and restored (when a process context switch occurs as
directed by the Scheduler or when an interrupt occurs)

-REQ: 7.1.4, 7.1.6, 7.1.7

type Process_Attributes Information 1e record
Logical Name: Hw String (1..0
-0sive (Process_Managers_Globals.Maximun_Lenelve:Of_Process_Name)) :=
others = "";
Kind Of Process := Process_Type := Kernel Process;
Process Initialization Status := Process Initialization_Status_Type;
Process Index := Network_Globals.Process_Index_Type;
Code_Address := Hw_Address;
Stack Low Address := Hw_Address;
Stack High Address := Hw_Address;
Context Switcher_Globals.Context_Saved_Type :=
Context Switcher_Globals_Via_Call;
Process_Context Save Area := Context_Save_Area_Context_Save_Area_Contents;
and record ;
pragma (page);
... schedule attributes include:
  state
    the current state of this process, this is used by the Kernel's
    Scheduler
    values include:
    running (this process controls the processor and is the
      currently running process)
    suspended (this process is able to run but another process is
      currently running)
    blocked (this process is unable to run)
    dead (this process is no longer able to run)
    default value
    blocked (this process is unable to run, since it has not yet
    been inserted into the scheduler)
    this value is set by the Scheduler as the process state changes
    (due to a call to a Kernel primitive, the passage of time, the
    occurrence of an event)
    priority
    the current priority of this process, this is used by the Kernel's
    Scheduler, and the primitives
    process attribute readers get process priority, and all primitives
    take a (resumption) priority as a parameter
    default value
    none
    this value is set initially via a parameter to the Kernel primitive:
    create process, it may be modified by the Scheduler as the process
    priority changes
    (due to a call to a Kernel primitive, the passage of time, the
    occurrence of an event)
    preemptable
    an indication of whether or not this process may be preempted; this
    is used by the Kernel's Scheduler (for time slicing), and the
    primitives
    process attribute readers get process preemptable
    process attribute modifiers set process preemptable
    values include
    true (this process may be preempted)
    false (this process may not be preempted)
    default value
    none, provided by initial call to create process
    this value is set initially via a parameter to the Kernel primitive:
    create process, it may be modified via a call to the Kernel
    primitive set process preemptable

... next
   a pointer to the next process in the scheduler's run queue
   values include
   any legal process identifier for a Kernel process
   default value:
   null
   -- REQ: 9.1.7; 9.1.8; 9.1.9; 9.1.11; 9.1.12; 9.1.13; 9.1.23
...

... type Schedule Attributes Information is record
  State ; Schedule Types Process State = Schedule Types Blocked;
  Priority ; Schedule Types Priority;
  Preemptable ; Schedule Types Preemptable;
  Next Process_Identifier = null;
end record;
pragma (page).
communication attributes include:
next available message ID
the message ID that may be used for the next message sent by this
process via send, message, and, wait; this value is constantly
increasing
default value
  first (lowest) message identifier available
  this value is modified only via a call to the Kernel primitive:
  send, message, and, wait
  maximum message queue size
  the maximum number of messages that may be queued awaiting receipt
  for this process
  default value
  none
  this value is set by a parameter to the Kernel primitive
  create, process; it should never be modified after that call
message queue
pointer to the first message in the message queue for this process;
this is used by the Kernel primitives to send and receive messages
default value:
  null
  this value is set via a call to the Kernel primitive
  create, process; it should never be modified after that call
current send buffer
  pointer to the current buffer being used to send a message, via
  an application call to send, message* 
  default value:
  null
  this value is set via a call to the Kernel primitives:
  send, message and send, message and, wait; it is also used by
  GPAM purge, message, queue to clean out the message of a
  recipient of a "kill" message
queue overwrite rule
  indication of how this process is to handle incoming message queue
  overflow
values include
  drop newest message (the most recently received message is lost)
  default value
  none

  this value is set by a parameter to the Kernel primitive:
  create, process, it should never change after that call
  message queue, overflow
  indication of whether or not the incoming message queue for this
  process is currently full and messages are being lost or in danger
  of being lost; this is used by the Kernel primitive:
  receive, message
values include
  true (at least one message has been lost already)
  false (no messages have been lost since last call to
  receive, message)
  default value:
  false (no messages have been lost since last call to
  receive, message)
  this value may be set by the Kernel as messages are received, its
  value may be reset via a call to the Kernel primitive:
  receive, message
  REQ: 7.1.6; 7.1.21; (7.1.22) 7.1.27
  REQ: 10.1.2; 10.1.3; 10.1.12; 10.1.22

  type CommunicationAttributes Information record
  Next Available Message Id : Datagram Globals.Message Identifier := 0;
  Maximum Message Queue Size : Hw.Long; Natural;
  Message Queue : Datagram Globals.Datagram Pointer := null;
  Current Send Buffer : Datagram Globals.Datagram.Pointer := null;
  Queue Overflow Rule :
  Process Managers Globals How To Handle Message Queue Overflow;
  Message Queue Overflow : Boolean := False;
  end record ;
  pragma (page);
... the kinds of mutually exclusive activities that can be pending for a
... single process are
... receive pending (the application called the Kernel primitive:
... receive message and is blocked until a message is received or until
... the timeout expires)
... semaphore pending (the application called the Kernel primitive:
... claim and is blocked until the requested semaphore is free or until
... the timeout expires)
... send with ACK pending (the application called the Kernel primitive:
... send message and wait and is blocked until an ACK or a NAK is
... returned or until the timeout expires)
... wait pending (the application called the Kernel primitive: wait and is
... blocked until the timeout expires)
... nothing pending (there is no activity on which this process is
... currently pending)
...


type Pending Activity Type is {
  Receive Pending,
  Semaphore Pending,
  Send With Ack Pending,
  Wait Pending,
  Nothing Pending};

pragma (page);

... pending activity attributes includes:
... pending activity
... indication of what kind of event has caused the process to block
... values include:
... see pending activity type just above
... default value:
... nothing pending
... this value is set via a call to any blocking Kernel primitive:
... receive message, claim, send message and, wait, wait; if it is
... reset by the Kernel upon expiry of the timeout, occurrence of
... the event awaited (e.g., receipt of message or ACK/NAK, availability
... of the semaphore)
... pending event ID
... an index into the time keeper's time event queue indicating the
... event entry corresponding to the value of pending activity for this
... process;
... used by Kernel internals as a link into the Kernel's time keeper
... default value:
... nil event
... this value is set via a call to any blocking Kernel primitive (as
... enumerated above); it is reset by the Kernel as described above
... current pending message
... if pending activity indicates send with ACK pending, this is the
... message identifier for which an ACK or a NAK is expected
... default value:
... none
... this value is set via a call to the Kernel primitive:
... send message, and, wait, it is reset by the Kernel upon receipt of
... the ACK/NAK for the identified message. It is valid if and only
... if pending activity indicates send with ACK pending
... current receive buffer address
... address of the receiver's message buffer. Used to perform local
... optimizations
... default value:
... none
... current receive buffer size
... size of the data space accessed via current receive buffer address.
... Used to perform local optimizations
... default value:
... none
alarm event ID
- an index into the time keeper's time event queue indicating the
  alarm expiration event for this process
- default value:
  null event
- this value is set via a call to the Kernel primitive: set_alarm;
  it may be reset either via a call to the Kernel primitive:
  cancel_alarm or by the Kernel upon the expiry of the alarm
- alarm resumption priority
  - if alarm event ID is not the null event, the priority at which
    this process is to be resumed upon the expiration of the alarm
  - default value:
    none
- this value is set via a call to the Kernel primitive: set_alarm;
  it should never change otherwise, it is valid if and only if
  alarm event ID is not the null event
- exception name
  - indication of whether or not the Kernel is raising an exception
    for this process, if not no exception, then also an indication of
    which exception is to be raised
  - values include:
    - there is an enumeration literal corresponding to each exception
      the Kernel may raise, see package Kernel.exceptions
  - default value:
    no_exception
- this value is set whenever the Kernel Internals detect a Kernel
  exception that is to be raised and reset to no_exception upon
  completion of internal exception processing
- REQ: 8.1.5, 8.1.6, 8.1.10, 9.1.15, 9.1.16, 9.1.19, 9.1.20, 9.1.29
- REQ: 10.1.7, 10.1.8, 10.1.10, 10.1.17, 10.1.18, 10.1.37, 10.1.38, 10.1.39
- REQ: 12.1.1, 12.1.2, 12.1.3, 12.1.6, 12.1.9, 12.1.27

type Pending Activity Attributes Information is record
  Pending Activity : Pending Activity Type (Nothing Pending,
  Pending Event Id : Event Identifier => Null Event;
  Current Pending Message : Datagram Globals.Message Identifier;
  Current Receive Buffer Size : Communication Globals.Message Length Type;
  Current Receive Buffer Address : Hw Address;
  Alarm Event Id : Event Identifier => Null Event;
  Alarm Resumption Priority : Schedule Types.Priority;
  Exception Name : Kernel Exceptions.Kernel Exceptions =>
  Kernel Exceptions.No Exception;
end record;
Acknowledged message information is contained in the process table entry corresponding to the SENDING process, this information refers to data about the RECEIVING process's incoming message queue, this is done to facilitate ready access to message queue information to process timeout expiration efficiently.

Acknowledged message information includes:

- Event ID
- Indication that this process is the SENDING process and sent a message via the Kernel primitive send message and wait, thus an ACK is required to be returned to THIS process upon receipt of the corresponding message

- Default value:
  - null event
  - this value is set by the Kernel when it receives a message that was sent via a call to the Kernel primitive send message and wait;
  - it is reset via a call to the Kernel primitive receive message or by the Kernel when the corresponding timeout expires

- Message
  - a list into the RECEIVING process's incoming message queue
  - indicating the message that this process sent via the Kernel primitive send message and wait

- Default value:
  - null

- This value is set by the Kernel when it receives a message that was sent via a call to the Kernel primitive send message and wait;
  - it is reset via a call to the Kernel primitive receive message or by the Kernel when the corresponding timeout expires

- Queue
  - a pointer to the head of the RECEIVING process's message queue - i.e., the message queue that contains the message field just above

- Default value:
  - null

- This value is set by the Kernel when it receives a message that was sent via a call to the Kernel primitive send message and wait;
  - it is reset via a call to the Kernel primitive receive message or by the Kernel when the corresponding timeout expires

REQ: 10.1.3

Type Acknowledged Message Information is record
Event Id: Event Identifier - null Event
Message: Datagram Globals Datagram Pointer = null
Queue: Datagram Globals Datagram Pointer = null
end record;
semaphore attributes include:

- semaphore last claimed
  - the identity of the semaphore most recently claimed by
    - the process, and still owned by it.
- default value
  - null semaphore
- semaphore name
  - the identity of the semaphore on which this process is currently
    - waiting
- default value
  - null semaphore
- this value is set via a call to the Kernel primitive: claim;
  - it is reset by release.
- next process pending claim
  - the process identifier for the process that called the Kernel
    - primitive claim after this process did
- default value
  - null process ID
- this value is set by the next call of the Kernel primitive: claim;
  - it may be reset by the Kernel if the timeout of that claim expires.
- previous process pending claim
  - the process identifier for the process that called the Kernel
    - primitive claim before this process did
- default value
  - null process ID
- this value is set via the current call to the Kernel primitive:
  - claim, it is reset by the Kernel if the timeout of the claim
    - expires or once the previous process releases the semaphore
- REQ: 8.1.3, 8.1.4, 8.1.9, 8.1.10, 8.1.11

These structures support the collection of data related to process
activities, i.e., they support the tool interface into the Kernel.

- monitoring process, type
  - This array holds process identifiers. One process identifier
    - for each possible attribute that can be monitored, this allows
      - different attributes to be sent to different tool processes.

- default value:
  - none

- monitoring message tag, type
  - this array holds the message tags. One message tag for each
    - possible attribute that can be monitored (this is tag that the
      - tool process will receive when a message is sent about the
        - attribute). This allows each tool interface message to be
          - tagged in a manner defined by the application.

- default value:
  - none

- tool interface attributes information
  - this record contains all the information the tool process
    - needs to collect data on other processes in the system.

- enabled
  - this variable indicates whether or not some tool process
    - is monitoring the activity of this particular process.

- default value:
  - false

- this value is set to true when begin collection is called
  - with this process's pid and set to false when
    - cease collection is for this process and ALL its
      - collection attributes are disabled.

- monitoring process
  - this array holds the pid of the tool process that will
    - receive the index attribute message.

- default value:
  - null

- this value is set by a call to begin collection and
  - reset to null by a call to cease collection

- message tag
  - this array holds the tags for each tool interface message
    - generated for the index attribute for this process.

- default value:
  - null.
each process information record comprises the following:
    process attributes
        the collection of process attribute information defined above
    schedule attributes
        the collection of schedule attribute information defined above
    communication attributes
        the collection of communication attribute information defined above
    pending activity attributes
        the collection of pending activity attribute information defined above
    send w. ACK attributes
        the collection of attributes relating to the sending process of
        a message sent via the Kernel primitive: send_message and wait
    semaphore attributes
        the collection of semaphore attribute information defined above
    tool interface attributes
        the collection of tool interface attribute information defined above

\[
\text{type Process Information Record is record}
\]
\[
\begin{align*}
\text{Process Attributes} & : \text{Process Attributes Information}; \\
\text{Schedule Attributes} & : \text{Schedule Attributes Information}; \\
\text{Communication Attributes} & : \text{Communication Attributes Information}; \\
\text{Pending Activity Attributes} & : \text{Pending Activity Attributes Information}; \\
\text{Send WAck Attributes} & : \text{Acknowledged Message Information}; \\
\text{Semaphore Attributes} & : \text{Semaphore Attributes Information}; \\
\text{Tool Interface Attributes} & : \text{Tool Interface Attributes Information}; \\
\end{align*}
\]
\end{align*}

the process table is a collection of process information records, built

dynamically: the process identifier points to a

process information record, which was allocated and initialized during

process initialization time (e.g., via \texttt{declare}, \texttt{process} and \texttt{create}, \texttt{process}

calls)

the list of process table entries contains a processor identifier and

is maintained by an instantiation of the generic \texttt{queue} manager

entries in the process table comprise:

    process ID
    this is the real reference to the information specific to this
    process; this is the value, cast as a
    process types process identifier, that the application uses
    when referencing a process anywhere in the application program
function "<" (Left, Right: Process Table Entry)
return Boolean;

FUNCTION "<"

DESCRIPTION:
- defines a less-than-or-equal-to ordering for process table entries,
- which always returns false (i.e., new entries will always be inserted
  at the head of the list)

PARAMETERS:
- left
  - left operand of the comparison between process table entries
  - default value: none

- right
  - right operand of the comparison between process table entries
  - default value: none

function return value
  - result of the comparison; always false

PRECONDITIONS:
- none

ACTIONS:
- return true

POSTCONDITIONS:
- none

ERROR PROCESSING:
- none
I4
A
I
ct
O
U
f
N
b
A
Cc
a
U

function Create Process Information Record return Process Identifier;

SUBPROGRAM NAME:
create_process_information_record

DESCRIPTION:
creates a new process information record, returns a pointer to it
(this may be a direct call on the Ada allocator or may shield a call
to a Kernel provided storage manager)

--REQ: 5.1.4
--PRIM: N/A
--TIME: N/A

PARAMETERS:
function return value
pointer to a new process information record

PRECONDITIONS:
none

ACTIONS:
create a new process information record and return a pointer to it

POSTCONDITIONS:
the storage is allocated and returned to the caller

ERROR PROCESSING:
when the allocator fails due to there not being sufficient storage
available
the exception Insufficient_space is redefined to be raised in the
calling process
POSTCONDITIONS:
the requested storage must be assumed to have not been allocated

pragma (page);
procedure Destroy Process Information Record (  
  Process id: In out Process Identifier);  
/ subdivision date: 1986  
/ SUBPROC NAME  
/ destroy process, information record  
/ DESCRIPTION  
/ destroys a specified process, information record, sets the pointer to  
/ it to null process  
/ - REQ 5.14  
/ - PRIM: N/A  
/ - TIME: N/A  
/ PARAMETERS:  
/ process ID  
/ pointer to process, information record to destroy  
/ default value:  
/ none  
/ PRECONDITIONS:  
/ none  
/ ACTIONS:  
/ destroy a process, information record and set the pointer to it to  
/ null process  
/ POSTCONDITIONS:  
/ the storage is returned to the storage manager  
/ ERROR PROCESSING:  
/ none  
/ -------------------------------  
pragma (page);  

procedure Initialize Process Table (  
Initial Allocation: In the Natural);  
/ subdivision date: 1986  
/ SUBPROC NAME:  
/ initialize process table  
/ DESCRIPTION:  
/ sets the initial number of entries in the process table to the  
/ specified value  
/ - REQ 5.14  
/ - PRIM: N/A  
/ - TIME: N/A  
/ PARAMETERS:  
/ initial allocation  
/ number of entries to initially allocate  
/ default value:  
/ none  
/ PRECONDITIONS:  
/ none  
/ ACTIONS:  
/ allocates the specified number of process, table, entry records  
/ creates a process table entry for the null, process  
/ POSTCONDITIONS:  
/ the storage is allocated  
/ the null, process is initialized and entered into the process table  
/ ERROR PROCESSING:  
/ when the null process process table entry cannot be created:  
/ the Kernel raise insufficient, space  
/ POSTCONDITIONS:  
/ the storage is not allocated, process table initialization  
/ fails (this should never happen)  
/ -------------------------------  
pragma (page);  

335  December 1986  
336  December 1986  


function Locate Process (Name: In Hw_String) return Process Identifier;
    /* SUBPROGRAM NAME:
    locate process
    */
    /* DESCRIPTION:
    Find the process table entry of a process by searching for its
    logical name in the process table.
    */
    /* REQ: 5.1.4
    */
    /* PRIM: N/A
    */
    /* TIME: N/A
    */
    /* PARAMETERS:
    name
    logical name of the process to find
    */
    /* default value:
    none
    */
    /* return
    process table index of the process
    */
    /* values
    if found: a legal process index
    if not found: null
    */
    /* default value:
    none
    */
    /* Preconditions:
    none
    */
    /* Actions:
    Locate the process in the process table
    */
    /* PostConditions:
    None
    */
    /* Error Processing:
    None
    */
    /* -------------------------------
    */

pragma (page);

function Size Of Process Table return Hw_Natural;
pragma inline (Size_Of_Process_Table);
    /* SUBPROGRAM NAME:
    size of process table
    */
    /* DESCRIPTION:
    Returns the number of process entries in the process table
    (at the time of invocation)
    */
    /* REQ: 5.1.4, 14.1.15
    */
    /* PRIM: N/A
    */
    /* TIME: N/A
    */
    /* PARAMETERS:
    */
    /* return
    the size of the process table
    */
    /* default value:
    none
    */
    /* Preconditions:
    none
    */
    /* Actions:
    return the size of the process table
    */
    /* PostConditions:
    None
    */
    /* Error Processing:
    None
    */
    /* -------------------------------
    */

end Generic Process Table;
pragma (page);
-- MODULE NAME
-- process table
--
-- MODULE TYPE
-- package specification
--
-- MODULE PURPOSE
-- an example of (and a template for) an instantiation of the generic
-- package generic_process_table
--
-- MODULE DESCRIPTION:
-- package instantiating the definition of the process table data structure
--
-- where
-- maximum_number_of_processes_value is defined as 100
--
-- MODULE CONTENTS:
-- process table (package specification)
-- acknowledged_message_information (type)
-- adjacent_process_table_entry (type)
-- communication_attributes_information (type)
-- create_process_information_record (function)
-- destroy_process_information_record (procedue)
-- event_identifier (subtype)
-- event_information (type)
-- initialize_process_table (procedure)
-- locate_process (function)
-- maximum_number_of_processes (constant)
-- null_event (constant)
-- null_semaphore (constant)
-- pending_activity_attributes_information (type)
-- process_attributes_information (type)
-- process_information_record (type)
-- process_initialization_status (type)
-- process_table (generic_process_table)
-- process_table_entry (type)
-- process_type (type)
-- schedule_attributes_information (type)
-- semaphore (type)
-- semaphore_head (type)
-- semaphore_attributes_information (type)
-- time_event_queue (generic_process_table)
-- tool_interface_attributes_information (type)
-- "x" (overloaded (2) function)
--
-- REFERENCES
-- DESIGN DOCUMENTS:
-- none
--
-- USER'S MANUAL
--
--
-- TESTING AND VALIDATION:
-- none
--
-- NOTES:
-- none
--
-- MODIFICATION HISTORY:
-- 18aug88 bamberg created (compiles clean)
-- 26aug88 bamberg final doc. check for 01sep89 freeze
--
-- VERSION: 3.0
--
-- DISCLAIMER:
--
-- The following copyright must be included in this software and
-- all software utilizing this software.
--
-- DISTRIBUTION:
--
-- Approved for public release; distribution is unlimited.
--
-- Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
-- PA. The Software Engineering Institute (SEI) is a federally
-- funded research and development center established and operated
-- by Carnegie Mellon University (CMU). Sponsored by the U.S.
-- Department of Defense under contract F19628-86-C-0003, the SEI
-- is supported by the services and defense agencies, with the U.S.
-- Air Force as the executive contracting agent.
--
-- Permission to use, copy, modify, or distribute this software and
-- its documentation for any purpose and without fee is hereby
-- granted, provided that the above copyright notice appear in all
-- copies and that both that copyright notice and this permission
-- notice appear in supporting documentation. Further, the names
-- Software Engineering Institute or Carnegie Mellon University may
-- not be used in advertising or publicity pertaining to
-- distribution of the software without specific, written prior
-- permission. CMU makes no claims or representations about the
-- suitability of this software for any purpose. This software is
-- provided "as is" and no warranty, express or implied, is made
-- by the SEI or CMU, as to the accuracy and functioning of the
-- program and related program material, nor shall the fact of
-- distribution constitute any such warranty. No responsibility is
-- assumed by the SEI or CMU in connection herewith.
--
-- with Generic Process Table;
-- pragma Elaborate (Generic Process Table);
-- to get: see MODULE CONTENTS
--
--****************************************************
package Process_Table

is new Generic Process Table (Maximum Number Of Processes Value => 25);
pragma (page);
// MODIFICATION HISTORY
// 18apr88 bamberg created (compiles clean)
// 12may89 rvs implement the tool interface

// VERSION: 3.0

// DISCLAIMER:

// The following copyright must be included in this software and
// all software utilizing this software.

// DISTRIBUTION:

// Approved for public release; distribution is unlimited.

// Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
// PA. The Software Engineering Institute (SEI) is a federally
// funded research and development center established and operated
// by Carnegie Mellon University (CMU). Sponsored by the U.S.
// Department of Defense under contract F19628-85-C-0003, the SEI
// is supported by the services and defense agencies, with the U.S.
// Air Force as the executive contracting agent.

// Permission to use, copy, modify, or distribute this software and
// its documentation for any purpose and without fee is hereby
// granted, provided that the above copyright notice appear in all
// copies and that both that copyright notice and this permission
// notice appear in supporting documentation. Further, the names
// Software Engineering Institute or Carnegie Mellon University may
// not be used in advertising or publicity pertaining to
// distribution of the software without specific, written prior
// permission. CMU makes no claims or representations about the
// suitability of this software for any purpose. This software is
// provided "as is" and no warranty, express or implied, is made
// by the SEI or CMU, as to the accuracy and functioning of the
// program and related program material, nor shall the fact of
// distribution constitute any such warranty. No responsibility is
// assumed by the SEI or CMU in connection herewith.

// *********************************************************************************

// pragma (page);

/* with Communication Globals;
   -- to get: message, tag, type;

with Hardware Interface; use Hardware Interface;
   -- to get: hw address;
   -- hw integer
   -- hw positive
   -- hw string

with Interrupt Globals;
   -- to get: interrupt condition
   -- interrupt state

with Kernel Exceptions;
   -- to get: kernel exceptions

with Network Globals;
   -- to get: bus, address

with Process Table;
   -- to get: process identifier

with Schedule Types;
   -- to get: preemption
   -- priority
   -- process state

with Time Globals;
   -- to get: elapsed_time

with Tool Interface Globals;
   -- to get: collection type

with Unchecked_Conversion;

generic

   -- the number of characters that are saved in the tables for the Kernel
   -- primitive name

   First_N_Primitive_Name_Chas = In_Hw_Positive = 32;

Pragma(Page);
*/

package Generic_Tool_Interface
package Cg
renames Communication_Globals;
package Ig
renames Interrupt_Globals;
package Ke
renames Kernel_Exceptions;
package Ng
renames Network_Globals;
package Pb
renames Process_Table;
package Si
renames Schedule_Types;
package Tg
renames Time_Globals;
package Tg
renames Tool_Interface_Globals;

... the process attributes are available on a per-process basis ...
... each entry in the process attributes data structure includes the following information:
... Id  the Kernel-generated process Id ...
... state  an indication of whether the process (Id) is:  running, suspended ...
... bucketed, or dead ...
... time state change ...
... the elapsed time when the process (Id) entered the above state ...
... current priority ...
... the priority level at which the process (Id) is currently ...
... registered ...
... current preemption ...
... indication of whether or not the process (Id) is preemptable ...
... alarm ...
... indication of whether or not an alarm is set for the process (Id)
... primitive identity ...
... name of the Kernel primitive invocation by the process (Id)
... primitive return status ...
... indication of the status of the above Kernel primitive call by ...
... the process (Id)
...
... REQ: 14.1.6, 14.1.9 ...

...type Process_Attributes_Entry is record
...  Id: Pb.Process_Identifier;
...  State: Si.Process_State;
...  Time State Change: Tg.Epoch_Time;
...  Current Priority: St.Priority;
...  Current Preemption: St.Preemption;
...  Alarm Pending: Boolean;
...  Primitive Identity: Kernel_Primitive_Name_Type;
...  Primitive Return Status: Ke.Kernel_Exceptions;
... end record;
...
... This type and unchecked conversion function allow the user of the ...
... tool interface to cast the message buffer address used by the tool ...
... interface process into a pointer to a record containing the data.
...
...type Process_Attributes_Entry_Pointer is access Process_Attributes_Entry;
function To_Process_Attributes_Is new Unchecked_Conversion
(Source => Hw_Address,
Target => Process_Attributes_Entry_Pointer);
pragma (page):
this type represents the full process id as it would appear on the
processor on which the process is sited

there are two parts to this information:
- sited process id
- the bus address of the processor on which the process is sited
- process id
- the process id of the process on the processor on which it is
  sited (i.e., its local process id on its owning processor)

REQ: 14.1.10

- Type Full Process Id la record
  Process Id:  PId; Process Identifier;
  Sited Process Id: PId; Bus Address,
end record.

- Each entry in the message attributes data structure includes the
  following information:
  - Sender process id
  - the full id of the sending process
  - Receiver process id
  - the full id of the receiving process
  - Time Kernel got message
    - for a sent message, this is time when Kernel SEND gets control
    - for a received message, this is time when Kernel receives that
      message
  - NOT when the kernel RECEIVE is called
  - Message length
    - the number of bytes in the message text itself
  - Message tag
    - the application specified tag indicating the kind of message this
     is

REQ: 14.1.10, 14.1.12

- Type Message Attributes Entry la record
  Sender Process Id: Full Process Id;
  Receiver Process Id: Full Process Id;
  Message Length: Cg Message Length Type;
  Message Tag: Cg Message Tag Type;
  Time Kernel Got Message: Tg Epoch Time;
end record.

- This type and unchecked conversion function allow the user of the
  tool interface to cast the message buffer address used by the tool
  interface process into a pointer to a record containing the data

- Type Message Attributes Entry Pointer is access Message Attributes Entry;
function To Message Attributes Entry is new Unchecked Conversion
(Source => Hw Address)
type Process_Table is array (the_Natural range =>) of 
Ptb.Process_Information_Record;

-- the interrupt table records information about interrupts and is
-- available on a per-processor basis

-- REQ: 14.1.18

type Interrupt_Table is new Ptb.Interrupt_Table_Type;

Message_Length array (Tig.Collection_Type) of Cg.Message_Length_Type :=
(Tig.Process_Attributes =>
     Cg.Message_Length_Type(Process_Attributes_Entry.SizeTypeBits_Per_Byte),
     Tig.Message_Attributes =>
     Cg.Message_Length_Type(Message_Attributes.Entry.SizeTypeBits_Per_Byte),
     Tig.Message_Contents =>
     Cg.Message_Length_Type(Cg Maximum_Message_Length));

procedure Begin_Collection (Process_Id: In Ptb.Process_Identifier;
   Tool_Message_Tag: In Cg.Message_Tag_Type;
   Information_To_Collect: In Tig.Collection_Type);

putation: }
procedure Cease_Collection (Process_Id: in Ptb.Process_Identifer;
                 Information_Not_To_Collect: in Tgl.Collection_Type);

/* SUBPROGRAM NAME:
   */
   cease collection

/* DESCRIPTION:
   */
   disables the logging of information for the specified process on the
   specified type of collection

/*
   */
   --REQ. 14.1.8, 14.1.11; 14.1.14; 14.1.18
   --
   --PRIM. 23.1.4
   --
   --TIME: 14.2.2, 14.2.4
   --
   PARAMETERs:
   / process_id
   / the id of the process on which collection is to be disabled
   / default_value:
   / none
   / Information_Not_To_Collect
   / the kind of information for which collecting is to be disabled
   / for the specified process
   / values include:
   / process attributes
   / Information about the process identity and state
   / message attributes
   / the message "envelope" (e.g., sender, receiver, message tag,
   / message_length)
   / message contents
   / the actual message text
   / default_value:
   / none
   / PRECONDITIONS:
   / NONE
   / ACTIONS:
   / disables collection of the specified process attributes for the
   / specified process
   / POSTCONDITIONS:
   / disabled collection information
   / ERROR PROCESSING:
   / none: all erroneous collection requests are simply ignored

/*-----------------------------*/
function Size Of Process Table return Hw Natural;

procedure Read Process Table (The Process Table: out Process Table;
Last Entry: out Hw Natural);

pragma (page);
procedure Read Interrupt Table (The Interrupt Table out Interrupt Table);

/** ---------------------------------------------------------
 * SUBPROGRAM NAME:
 * read Interrupt table
 * /@
 * DESCRIPTION:
 * extracts information from the Kernel's interrupt table
 * /@
 * REQ 14 1 18
 * /@
 * PRIM 23 1 6
 * /@
 * TIME. N/A
 * /@
 * PARAMETERS:
 * the Interrupt Table
 * the location where the Kernel is to dump the interrupt table
 * information
 * /
 * default value:
 * none
 * /
 * PRECONDITIONS:
 * NONE
 * /
 * ACTIONS
 * reads the Interrupt table information into a user-provided buffer
 * /
 * POSTCONDITIONS:
 * the user-provided buffer contains as much of the interrupt table as
 * well fit into the user-provided buffer
 * no change in Kernel data structures
 * /
 * ERROR PROCESSING:
 * none
 * /
 * -------------------------------------------------------------
 * end (Generic Tool Interface);
 * pragma (page);
// Approved for public release, distribution is unlimited.
//
// Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
// PA. The Software Engineering Institute (SEI) is a federally
// funded research and development center established and operated
// by Carnegie Mellon University (CMU). Sponsored by the U.S.
// Department of Defense under contract F19628-85-C-0003, the SEI
// is supported by the services and defense agencies, with the U.S.
// Air Force as the executive contracting agent.
//
// Permission to use, copy, modify, or distribute this software and
// its documentation for any purpose and without fee is hereby
// granted, provided that the above copyright notice appear in all
// copies and that both that copyright notice and this permission
// notice appear in supporting documentation. Further, the names
// Software Engineering Institute or Carnegie Mellon University may
// not be used in advertising or publicity pertaining to
// distribution of the software without specific, written prior
// permission. CMU makes no claims or representations about the
// suitability of this software for any purpose. This software is
// provided "as is" and no warranty, express or implied, is made
// by the SEI or CMU, as to the accuracy and functioning of the
// program and related program material, nor shall the fact of
// distribution constitute any such warranty. No responsibility is
// assumed by the SEI or CMU in connection herewith.

passport shelf name TKO

package Tool Interface Output

is

... this flag is used to enable/disable conditional compilation of all tool
... Interface code

Tool Interface Output Enabled: constant Boolean := True;

end Tool Interface Output;

pragma (page);
...MODULE NAME
   hardware interface
...
...MODULE TYPE
   package specification
...
...MODULE PURPOSE
   The Kernel package hardware interface
   provides an interface to compiler-specific primitive types.
   Within the Kernel itself, there are no references to the
   redefined types in Ada package Standard;
   all references to primitive types use names declared in
   package hardware interface.
   By doing this, certain implementation-dependent details are abstracted
   away from both the Kernel and the application in a uniform manner.
   It is recommended that applications avoid using package Standard entirely
   and use package hardware interface for ready compatibility with
   Kernel primitives.
...
...MODULE DESCRIPTION
   these types are defined in case the compiler's definition and/or
   implementation of them is not sufficient for the Kernel and/or the
   application, this allows Kernel/ application developers to provide
   their own version of types and operators without needing to modify
   Kernel or application code directly.
...
...MODULE CONTENTS
   hardware interface (package specification)
   bits per byte (constant)
   byte (constant)
   hw address (type)
   hw bits8 (type)
   hw bits8 ptr (type)
   hw byte (type)
   hw byte ptr (type)
   hw duration (type)
   hw integer (type)
   hw long integer (type)
   hw long natural (type)
   hw long positive (type)
   hw long unsigned (type)
   hw natural (type)
   hw positive (type)
   hw short integer (type) (CURRENTLY COMMENTED OUT)
   hw string (type)
   longword (constant)
   null hw address (constant)
   to hw address (function)
   to hw bits8 (function)
   to hw bits8 ptr (function)
   to hw byte ptr (function)
   word (constant)
...

...REFERENCES:
...
...DESIGN DOCUMENTS:
   Kernel Facilities Definition
   --REQ. 5.1.4
   --PRIM. N/A
   --TIME. N/A
...
...USER'S MANUAL:
   Kernel User's Manual
...
...TESTING AND VALIDATION:
   none
...
...NOTES:
...
...MODIFICATION HISTORY:
   18aug88 bamberg created (compiles clean)
   30aug88 bamberg correct type on bound of hw integer
   31aug88 dvk changed "byte" to "bits per byte"
   14sep88 dvk added constants byte, word, and longword
   17oct88 frith added hw. long natural
   14dec88 frith added hw. byte, byte. bits8, byte. ptr, hw. bits8, hw. ptr and conversions
   1nov88 frith added to hw address and null hw address
   12dec88 frith revised documentation, reorganised
   1jan89 frith final doc. check for beta release
   19jul89 dvk added hw. long unsigned
   25aug89 bamberg final doc. check for 01sep89 freeze
...
...VERSION: 3.0
...
...DISCLAIMER:
...
The following copyright must be included in this software and
all software utilizing this software.

...DISTRIBUTION:
...
...Approved for public release, distribution is unlimited.
...
Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
PA. The Software Engineering Institute (SEI) is a federally
funded research and development center established and operated
by Carnegie Mellon University (CMU), Sponsored by the U.S.
Department of Defense under contract F19628-85-C-0003, the SEI
is supported by the services and defense agencies, with the U.S.
Air Force as the executive contracting agent.
...
Permission to use, copy, modify, or distribute this software and
its documentation for any purpose without fee is hereby
granted, provided that the above copyright notice and this permission
with System;
   -- To get address
   -- unsigned longword

with Unchecked_Conversion;
   /************* ********************
   /* Package short name: HW */
   /************* ********************

package Hardware_Interface is

-- REQ: 5.1.4

-- The following definitions might need modification after any change to
d -- either compiler or hardware. The purpose of the definitions is to
d -- provide a coordinated and clean view of the underlying basic machine
d -- data types and addressing structure.

...

-- define the bit interface to the underlying system
...

Bits_Per_Byte : constant := 8;
...

-- define constants to be used as offsets in rep specs; these values
-- indicate how many bytes in the named storage unit (i.e., byte, word,
-- longword)
...

-- a byte contains 1 byte
-- a word contains 2 bytes
-- a longword contains 4 bytes
...

Byte : constant := 1;
Word : constant := 2;
Longword : constant := 4;

-- The type hw_duration must capture exactly the hardware
-- representation of the current Ada type DURATION
...

-- to be defined for hw_duration:
-- range
-- size
-- 'small' (i.e., granularity of measure)
...

-- In the event of any mismatch, the compiler should reject the
-- representation clauses

type Hw_Duration is new Duration range -86 400.0 .. 86 400.0;
for Hw_Duration use 2.0 ** 14;
for Hw_Duration use 4 * Bits_Per_Byte;

pragma (page);
... The types hw_byte and hw_bits8 are to be used to define essentially... untyped data formats, such as those found in device interfaces.

... hw_byte is intended to be an unsigned 8-bit byte.
... hw_bits8 is intended to be a byte considered as a sequence of bits.
... Note especially that the bit numbering in the type hw_bits8 should
... be whatever is most convenient for subsequent use of the type. If
... the bulk of the hardware and device documentation numbers the bits
... a certain way, that way should be used in the type declaration.

type hw_byte is range 0.255;
... for hw_byte size use 1 bits per byte,

type hw_bits8 is record
  Bits : array (0..7) of Boolean;
end record;

for hw_bits8 use record
  Bits_at : array (0..7) of 0 range 0..7;
end record;

pragma (page).
The following types and conversions are intended to allow the code to set individual bits in device registers. This is achieved by:

(a) declaring the register to be a hw_byte
(b) computing its Address
(c) casting that address into an access value designating an object of type hw_bits8
(d) accessing the individual bits of that object

These types are pointers to hw_byte or hw_bits8 objects and they are intended to allow unchecked conversion of values:

type Hw Byte Pnt is access Hw Byte;
type Hw Bits8 Pnt is access Hw Bits8;

-- SUBPROGRAM NAME:
-- to hw byte ptr
-- DESCRIPTION:
-- conversion routine from Ada address type to access type
-- -- REQ: 5.14
-- -- PRIM: N/A
-- -- TIME: N/A

-- PARAMETERS:
-- Source
-- the address of a declared object
-- WHICH MUST BE A HW_BYTE OR A HW_BITS8
-- default value: none
-- function return value
-- the same bit pattern, considered as an access value designating a hw_bits8
-- PRECONDITIONS:
-- none
-- ACTIONS:
-- return the same physical bit pattern as the parameter
-- POSTCONDITIONS:
-- none
-- ERROR PROCESSING:
-- none

-- EXAMPLES:
-- This conversion is to be used when it is necessary to change an individual bit of a device register or other hardware object.
-- Consider for instance a device status byte of which bit4 is the "enable" bit, to be set to enable the device. The code in Ada then looks like this.
-- In order to enable the device, we must set bit 4 of the status byte
-- to hw_bits8_ptr(status_byte_address).bit4 := True;
-- function To_Hw Bits8_Ptr is new Unchecked_Conversion(System.Address,Hw Bits8_Ptr)
-- The integer types are intended to capture the natural efficient
-- integer types of the target machine. Currently, these are
--
-- 8-bit signed : hw_short_integer
-- 16-bit signed : hw_integer
-- 32-bit signed : hw_long_integer
--
-- These types are defined in terms of explicit ranges, and a size
-- clause is provided as a check that the compiler has indeed mapped
-- them onto the correct hardware type.
--
-- The Ada types defined in package STANDARD should not be used, since
-- they are dependent on the compiler as well as the target
--
-- hw_integer is intended to be a 16-bit integer
--
-- to be defined for hw_integer:
--
-- range
-- size
--

type Hw_Integer is range -32768 .. 32767;
for Hw_Integer'Range use 2 ^ Bits_Per_Bit;  

-- hw_short_integer is intended to be an 8-bit integer
--
-- to be defined for hw_short_integer:
--
-- range
-- size

for Hw_Short_Integer'Range use 1 ^ Bits_Per_Bit;  

-- hw_long_integer is intended to be a 32-bit integer
--
-- to be defined for hw_long_integer:
--
-- range
-- size

for Hw_Long_Integer'Range use 4 ^ Bits_Per_Bit;  

pragma (page);
For additional security, the appropriate subsets of the machine
integer types are defined as full types, not as subtypes. The
two subsets are

- Natural: the non-negative half of the range [0, Last]
- Positive: the strictly positive subset (1..Last)

Again, a size clause is added as a check.

hw_long_natural is intended to be a 32-bit integer with natural range

to be defined for hw_long_natural:

- range
- size

```
type HW_Long_Natural is range 0 .. 2^147,483,647;
for HW_Long_Natural use 4 * Bits_Per_Bits;
```

hw_long_positive is intended to be a 32-bit integer with positive range

to be defined for hw_long_positive:

- range
- size

```
type HW_Long_Positive is range 1 .. 2^147,483,647;
for HW_Long_Positive use 4 * Bits_Per_Bits;
```

hw_long_unsigned is intended to be a full unsigned 32-bit integer

to be defined for hw_long_unsigned:

- range
- size

```
type HW_Long_Unsigned is new System.Unsigned_Longword;
for HW_Long_Unsigned use 4 * Bits_Per_Bits;
```

the range of hw_natural is to be a subset of hw_integer

(i.e., only 16-bits)

to be defined for hw_integer:

- range
- size

```
type HW_Natural is range 0 .. 32,767;
for HW_Natural use 2 * Bits_Per_Bits;
```

the range of hw_positive is to be a subset of hw_integer

(i.e., only 16-bits)

to be defined for hw_positive:

- range
- size

```
type HW_Positive is range 1 .. 32,767;
for HW_Positive use 2 * Bits_Per_Bits;
```
... hw address is included just in the off chance that the compiler's implementation of system address is not acceptable to the Kernel and/or the application, this allows kernel/application developers to provide their own version of address operators without needing to modify Kernel or application code.

... There is also a function to convert an integer of the appropriate size to an address, and a null value that cannot be a legal address.

... In the current implementation, it has been shown that the compiler provided address type is satisfactory.

type Hw_Address is new System.Address;

... -------------------------------------------------------------
... SUBPROGRAM NAME:
... to_hw_address
...
... DESCRIPTION:
... conversion routine from an integer type to type hw_address
...
... REQ. NONE
...
... PRIM. NONE
...
... TIME. NONE
...
... PARAMETERS:
... Source
... an integer value OF THE SAME SIZE AS AN ADDRESS
... the value MUST be a legal address
...
... default value:
... none
...
... function return value
... the same bit pattern, considered as an address
...
... PRECONDITIONS:
... NONE
...
... ACTIONS:
... return the same physical bit pattern as the parameter
...
... POSTCONDITIONS:
... NONE
...
... ERROR PROCESSING:
... NONE
...
... EXAMPLES:
... the most common use of this function is to construct a pointer
to a specific part of the machine's address space, for instance

... a part that corresponds to a device control register. Suppose
... that a device is controlled by a set of registers whose addresses
... start at 16#1000R8. A handler on this device can be created by
...
... device control block address: hw_address
...
... => to_hw_address(16#1000R8);
...
... -------------------------------------------------------------

function to_hw_address
... new Unchecked_Conversion(Hw_Long_Integer, Hw_Address);

null Hw_Address : constant Hw_Address
... => to_Hw_Address(Hw_Long_Integer(0));
...
... hw_string is included just in the off chance that the compiler's implementation of string is not acceptable to the Kernel and/or the application; this allows Kernel/application developers to provide their own version of string operators without needing to modify Kernel or application code.
...
... in this initial implementation, we assume that the compiler provided string type and operations are OK
...

type Hw_String is new String;

end Hardware_Interface;

pragma (page);
MODULE NAME: generic_Kernel_time

MODULE TYPE: generic package specification

MODULE PURPOSE: provide the abstraction of Kernel time

MODULE DESCRIPTION: Kernel_time abstract type and subprograms to manipulate it

MODULE CONTENTS:
- generic_Kernel_time (generic package specification)
  - integral_duration (type)
  - Kernel_time (type)
  - seconds (overloaded 2 function)
  - milliseconds (function)
  - microseconds (function)
  - ticks_per_second (constant)
  - ticks_per_second_value (generic formal parameter)
  - zero_Kernel_time (constant)
  - "、“ (function)
  - "*" (function)
  - "-" (function)
  - "-" (overloaded 2 function)
  - "*" (function)
  - "*" (function)
  - "-" (function)
  - "<" (function)
  - "<>" (function)
  - "=" (function)
  - "=" (function)

REFERENCES:
- Kernel Facilities Definition
- - REQ: 12.1.1, 12.1.3, 12.1.4, 12.1.6, 12.1.25
- - PRM: N/A
- - TIME: N/A

DESIGN DOCUMENTS:
- Kernel Architecture Manual

USER'S MANUAL:

TESTING AND VALIDATION:
- Unit tests and integration testing with Time_keeper

NOTES:
- this package must be instantiated and used in place of the compiler-provided Package Calendar
- this package provides an abstract data type representing Time

for use by the kernel. It is at the lowest level of a set of abstractions that jointly respond to Requirements Section 12.

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

MODIFICATION HISTORY:
- 0aug88 babang created (compiles clean)
- 1/8oct88 22th revised to interface to assembler
- 1/8oct88 22th changed type of ticks_per to long int
- 24oct88 22th added milliseconds' and microseconds'
- 0jan89 22th improved documentation
- 1/1jan89 22th final doc update for beta release
- 1/feb89 22th corrected error in documentation
- 2jan89 dk shifted to Vax/40 definitions
- 19/feb89 dk removed dependency on package system
- 20feb89 dk fixed interface spec for FLT_seconds

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

VERSION: 3.9

DISCLAIMER:
- The following copyright must be included in this software and all software utilizing this software.

DISTRIBUTION:
- Approved for public release; distribution is unlimited.

Copyright © 1989 by Carnegie Mellon University, Pittsburgh, PA. The Software Engineering Institute (SEI) is a federally funded research and development center established and operated by Carnegie Mellon University (CMU). Sponsored by the U.S. Department of Defense under contract F19628-86-D-0003, the SEI is supported by the services and defense agencies, with the U.S. Air Force as the executive contracting agent.

Permission to use, copy, modify, or distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Further, the names of the Software Engineering Institute or Carnegie Mellon University may not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. CMU makes no claims or representations about the suitability of this software for any purpose. This software is provided "as is" and no warranty, express or implied, is made by the SEI or CMU, as to the accuracy and functioning of the program and related program material, nor shall the fact of distribution constitute any such warranty. No responsibility is assumed by the SEI or CMU in connection herewith.

with Hardware Interface; use Hardware Interface;
- to get: hw duration
generic

-- the number of TICKs in a second (this value is never used within the
-- Kernel except for the initialization of the constant, nor should it be
-- used without the Kernel)

Ticks_Per_Second_Value : in Hw_Long_Integer;

package Generic_Kernel_Time

is

-- the constant number of TICKs per second (thus providing a mechanism of
-- converting from Kernel time to application time)

Ticks_Per_Second : Hw_Long_Integer := Ticks_Per_Second_Value;

-- type Kernel_time, the time on which elapsed time and epoch time
-- abstractions are built

-- this time representation allows applications beginning at time zero to
-- execute for some 15,000 years, so no overflow is monitored

-- the user should ensure that adjusting any times does not approach the
-- limit of this time representation; proper Kernel functioning IS NOT
-- guaranteed if time overflows

-- Kernel_Time is represented as a signed 64-bit binary integer,
-- representing a count of hundred-milliseconds. Hence, a kernel time
-- of 10,000,000 corresponds to one second.

-- For the purposes of Ada definition, kernel time is a record of
-- two components, being respectively the low-order and high-order
-- 32 bits. The high-order component can be correctly declared to
-- be a signed 32-bit integer, of type hw_long_integer. The low-order
-- component, however, is properly an UNSIGNED 32 bit integer, which
-- we have defined in HW_INTERFACE as hw_long_unsinged for VAX Ada

-- Applications should use ONLY the exported constructor functions to
-- create values of type kernel_time or of types derived from it.

type Kernel_Time is record
  Low : Hw_LongUnsigned := 0;

377 December 1990

378 December 1990
High: HW_Long_Integer := 0;
end record;

for Kernel_Time use record
  Low at 0 range 0..31;
  High at 4 range 0..31;
end record;

...-- the value of zero for the Kernel_time abstraction ...

Zero_Kernel_Time : constant Kernel_Time := (0,0);
...
-- the range of durations represented as integral values ...

type Integral_Duration is new HW_Long_Integer;

--------------------------------------------------------------------------
pragma (page);

function Seconds (Ada_Duration : Ada_Duration)
return Kernel_Time;
  -- The pragmas INTERFACE and IMPORT_FUNCTION can be found following
  -- the second declaration of the function (as per VAX RTL 13.9[notes])
  
/*---------------------------------------------------------------*/
/* SUBPROGRAM NAME:                                            */
/* seconds                                                      */
/*                                                            */
/* DESCRIPTION:                                                */
/* conversion routine from Ada duration type to Kernel_time    */
/*                                                            */
/* --REQ: 12.1.23, 12.1.24                                       */
/*                                                            */
/* --PRM: N/A                                                    */
/*                                                            */
/* --TIME: N/A                                                  */
/*                                                            */
/* --PARAMETERS:                                               */
/* -- Ada duration                                             */
/* -- the time to convert in Ada duration format (note that this limits */
/* -- legal values to 0 to 86400 seconds, 1 day)                */
/* -- default value:                                           */
/* -- none                                                     */
/*                                                                 */
/* -- function return value                                    */
/* -- the input duration in Kernel_time format (note that this contains */
/* -- time up to 1 day)                                        */
/* --                                                            */
/* --PRECONDITIONS:                                            */
/* -- NONE                                                     */
/* --                                                            */
/* --ACTIONS:                                                  */
/* -- returns the Ada duration time in Kernel_time format      */
/* --                                                            */
/* --POSTCONDITIONS:                                           */
/* -- NONE                                                     */
/* --                                                            */
/* --ERROR PROCESSING:                                         */
/* -- NONE                                                     */
/*                                                            */
/* --------------------------------------------------------------*/

pragma (page);
function Seconds (An Integral Duration : Integral_Duration)
    return Kernel_Time;
pragma Interface(Assembly, Seconds);
pragma Import Function (Internal => Seconds,
    External => "OKT_seconds!",
    Parameter_Types => (Hw, Duration),
    Mechanism => (Reference),
    Result_Type => Kernel_Time);
pragma Import Function (Internal => Seconds,
    External => "OKT_seconds2!",
    Parameter_Types => (Integral_Duration),
    Mechanism => (Value),
    Result_Type => Kernel_Time);

/* SUBPROGRAM NAME:
/ seconds
*/
/* DESCRIPTION:
/ conversion routine from integral elapsed times to Kernel time
*/
/* REQ: 12.1.23, 12.1.24
*/
/* PRIM: N/A
*/
/* TIME: N/A
*/
/* PARAMETERS:
/ an, integral, duration
/ the time to convert in integral format
/ default value:
/ none
/ function return value
/ the input integral duration in Kernel_time format
*/
/* PRECONDITIONS:
/ NONE
*/
/* ACTIONS:
/ returns the integral duration time in Kernel_time format
*/
/* POSTCONDITIONS:
/ NONE
*/
/* ERROR PROCESSING:
/ NONE
*/
/* */
/* */
pragma (page);

function Milliseconds (Integral_Milliseconds : Integral_Duration)
    return Kernel_Time;
pragma Interface(Assembly, Milliseconds);
pragma Import Function (Internal => Milliseconds,
    External => "OKT_milliseconds!",
    Parameter_Types => (Integral_Duration),
    Mechanism => (Value),
    Result_Type => Kernel_Time);

/* SUBPROGRAM NAME:
/ milliseconds
*/
/* DESCRIPTION:
/ conversion routine from integral elapsed times to Kernel time
*/
/* REQ: 12.1.23, 12.1.24
*/
/* PRIM: N/A
*/
/* TIME: N/A
*/
/* PARAMETERS:
/ Integral milliseconds
/ the duration in milliseconds to convert to kernel time
/ default value:
/ none
/ function return value
/ the input integral duration in Kernel_time format
*/
/* PRECONDITIONS:
/ NONE
*/
/* ACTIONS:
/ returns the duration time in Kernel_time format
*/
/* POSTCONDITIONS:
/ NONE
*/
/* ERROR PROCESSING:
/ NONE
*/
/* */
/* */
pragma (page);
function Microseconds(
  Integral_Microseconds : Integral_Duration)
return Kernel_Time;
pragma Interface(Assembly, Microseconds);
pragma Import Function (Internal => Microseconds,
  External => "GLK_micronosec",
  Parameter Types => (Integral_Duration),
  Mechanism => (Value),
  Result_Type => Kernel_Time);

-- SUBPROGRAM NAME:
-- seconds
--
-- DESCRIPTION:
-- conversion routine from integral elapsed times to Kernel_time
--
-- -- REQ: 12.1.23, 12.1.24
-- -- PRIM: N/A
-- -- TIME: N/A
--
-- PARAMETERS:
-- Integral_micronosec
-- the duration in microseconds to convert to Kernel_time
--
-- default value:
-- none
--
-- function return value
-- in = input integral duration in Kernel_time format
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- returns the integral duration time in Kernel_time format
--
-- POSTCONDITIONS:
-- NONE
--
-- ERROR PROCESSING:
-- NONE
--
--**************************************************************
pragma (page);

function "*" (
  Left : Kernel_Time;
  Right : Kernel_Time)
return Kernel_Time;

-- The pragma INTERFACE and IMPORT_FUNCTION can be found following
-- the second declaration of the function (as per VAX RTL 13.9(notes))
--
-- SUBPROGRAM NAME:
-- "*"
--
-- DESCRIPTION:
-- adds two Kernel_time values returning a Kernel_time value
--
-- -- REQ: 12.1.23, 12.1.24
-- -- PRIM: N/A
-- -- TIME: N/A
--
-- PARAMETERS:
-- left
-- left operand of the addition operator
--
-- default value:
-- none
--
-- right
-- right operand of the addition operator
--
-- default value:
-- none
--
-- function return value
-- result of the addition operator
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- takes the mathematical sum of the two Kernel_time values, returning a
-- properly formatted Kernel_time result
--
-- POSTCONDITIONS:
-- the sum of left and right is computed and returned
--
-- ERROR PROCESSING:
-- NONE
--
--**************************************************************
pragma (page);
function "" (  
Left : Kernel_Time;  
Right : Kernel_Time)  
return Kernel_Time;

/* The pragmas INTERFACE and IMPORT FUNCTION can be found following */
/* the second declaration of the function (as per VAX RTL 13.9) (notes) */

/* SUBPROGRAM NAME: */
""

/* DESCRIPTION: */
/* subtracts the right Kernel_time value from the left Kernel_time */
/* value returning a Kernel_time value */
/* --REQ: 12.1.23, 12.1.24 */
/* --PRIM: N/A */
/* --TIME: N/A */

/* PARAMETERS: */
/* left */
/* left operand of the subtraction operator (subtrahend) */
/* default value: */
/* none */
/* right */
/* right operand of the subtraction operator (minuend) */
/* default value: */
/* none */
/* function return value */
/* result of the subtraction operator */

/* PRECONDITIONS: */
/* NONE */

/* ACTIONS: */
/* takes the mathematical difference of the two Kernel_time values by */
/* subtracting the right operand from the left operand, returning a */
/* properly formatted Kernel_time result */
/* POSTCONDITIONS: */
/* the difference of left and right is computed and returned */
/* ERROR PROCESSING: */
/* NONE */

pragma (page);
function "**"( 
    Left : Hw_Long_Integer;
    Right : Kernel_Time)
return Kernel_Time;

pragma Interface(Assembly, "**");
pragma Import Function (Internal => ",
    External => "GKT multiply!",
    Parameter Types => (Kernel_Time,Hw_Long_Integer),
    Mechanism => (Reference ,Value),
    Result_Type => Kernel_Time);
pragma Import Function (Internal => "**",
    External => "GKT multiply?",
    Parameter Types => (Hw_Long_Integer,Kernel_Time),
    Mechanism => (Value ,Reference),
    Result_Type => Kernel_Time);

--/ SUBPROGRAM NAME:
--/ "**"
--/ DESCRIPTION:
--/ multiplies an integer value by a Kernel time value returning a
--/ Kernel time value
--/ -- REQ: 12.1.23, 12.1.24
--/ -- PREC: N/A
--/ -- TIME: N/A
--/ -- PARAMETERS:
--/ -- left
--/ -- left operand of the multiplication operator
--/ -- default value:
--/ -- none
--/ -- right
--/ -- right operand of the multiplication operator
--/ -- default value:
--/ -- none
--/ -- function return value
--/ -- result of the multiplication operator
--/ -- PRECONDITIONS:
--/ -- NONE
--/ -- ACTIONS:
--/ -- takes the mathematical product of the integer and the Kernel time
--/ -- value, returning a properly formatted Kernel time result
--/ -- POSTCONDITIONS:

--/ the product of left and right is computed and returned
--/
--/ ERROR PROCESSING:
--/ -- NONE
--/ -- 
--/ 
pragma (page);
function */
    Left : Kernel.Time;
    Right : hw_long Integer)
  return Kernel.Time;
pragma Interface(Assembly, "");
pragma Import Function (Internal => ",
    External => "DKT divide",
    Parameter Types => (Kernel.Time, hw_long Integer),
    Mechanism => (Reference.Value),
    Result_Type => Kernel.Time);

/*====================================================================*/
/* SUBPROGRAM NAME: */
/* */
/* DESCRIPTION: */
/* divides a Kernel time value by an Integer value returning a */
/* Kernel time value */
/* */
/* -REQ: 12.1.23, 12.1.24 */
/* */
/* -PRIM: N/A */
/* */
/* -TIME: N/A */
/* */
/* -PARAMETERS: */
/* left */
/* left operand of the division operator (dividend) */
/* */
/* default value: */
/* none */
/* */
/* right */
/* right operand of the division operator (divisor) */
/* */
/* default value: */
/* none */
/* */
/* function return value */
/* result of the division operator */
/* */
/* -PRECONDITIONS: */
/* -NONE */
/* */
/* -ACTIONS */
/* takes the mathematical quotient of the Kernel time value and the */
/* integer, truncating the result toward zero if necessary, returning */
/* a properly formatted Kernel time result */
/* */
/* -POSTCONDITIONS: */
/* the quotient of left divided by right is computed and returned */
/* */
/* -ERROR PROCESSING: */
/* -NONE */
IgE is a...
function "c" ( 
  Left : Kernel Time; 
  Right : Kernel Time) 
return Boolean;

pragma Interface(Assembly,"<");
pragma Import Function (Internal => "c", 
  External => "OKT Is", 
  Parameter Types => (Kernel Time, Kernel Time), 
  Mechanism => (Reference, Reference), 
  Result Type => Boolean);

/--------------------------------------------------------------------------/
/ \ SUBPROGRAM NAME: 
/  "c"
/--------------------------------------------------------------------------/
/ \ DESCRIPTION: 
/  compares two Kernel time values returning a Boolean true if left is 
/  less than right, false otherwise 
/--------------------------------------------------------------------------/
/  -REQ: 12 1 23, 12 1 24 
/--------------------------------------------------------------------------/
/  -T/R: N/A 
/--------------------------------------------------------------------------/
/ \ TIME: N/A 
/--------------------------------------------------------------------------/
/ \ PARAMETERS:  
/  left 
/  ... left operand of the less-than operator 
/  ... default value: none 
/  right 
/  ... right operand of the less-than operator 
/  ... default value: none 
/  ... function return value 
/  ... result of the less-than operator 
/--------------------------------------------------------------------------/
/ \ PRECONDITIONS: 
/  NONE 
/--------------------------------------------------------------------------/
/ \ ACTIONS 
/  compares the two Kernel time values, returning Boolean true if the 
/  left operand is less than the right, returning false otherwise 
/--------------------------------------------------------------------------/
/ \ POSTCONDITIONS 
/  the values of left and right are compared 
/--------------------------------------------------------------------------/
/ \ ERROR PROCESSING: 
/  NONE 
/--------------------------------------------------------------------------/
function "<"() {
    Left : Kernel Time;
    Right : Kernel Time
    return Boolean;
}

pragma Interface(Assembly, "<");
pragma Import Function (Internal
    External => "OK",
    Parameter Types => (Kernel Time, Kernel Time),
    Mechanism => (Reference, Reference),
    Result Type => Boolean);

/* SUBPROGRAM NAME: */
"<"

/* DESCRIPTION: */
// comp redefine two Kernel time values returning a Boolean true if left is
// less than or equal to right, false otherwise

// --REQ. 12.1.23, 12.1.24

// --PRIM. N/A

// --TIME: N/A

// --PARAMETERS:
// left
// left operand of the less-than-or-equal-to operator
//
// default value:
// none

// right
// right operand of the less-than-or-equal-to operator
//
// default value:
// none

// function return value
// result of the less-than-or-equal-to operator

// --PRECONDITIONS:
// --NONE

// --ACTIONS:
// compares the two Kernel time values, returning Boolean true if the
// left operand is less than or equal to the right, returning false
// otherwise

// --POSTCONDITIONS:
// the values of left and right are compared

// --ERROR PROCESSING:
// --NONE
function ">" (Left : Kernel.Time; Right : Kernel.Time) return Boolean;

pragma Interface(Assembly,’>’);
pragma Import Function (Internal) => ">’;

External => "OKT.gr”,
Parameter Types => (Kernel.Time,Kernel.Time),
Mechanism => (Reference,Reference),
Result_Type => Boolean);

-- SUBPROGRAM NAME:
-- ">
--
-- DESCRIPTION:
-- compares two Kernel time values returning a Boolean true if left is
-- greater than right, false otherwise
--
-- REQUIRED: 12.1.23, '2.1.24
--
-- PRIM: N/A
--
-- TIME: N/A
--
-- PARAMETERS:
-- left
-- left operand of the greater-than operator
-- default value:
-- none
--
-- right
-- right operand of the greater-than operator
-- default value:
-- none
--
-- function return value
-- result of the greater-than operator
--
-- PRECONDITIONS:
-- NONE
--
-- ACTIONS:
-- compares the two Kernel time values, returning Boolean true if the
-- left operand is greater than the right, returning false otherwise
--
-- POSTCONDITIONS:
-- the values of left and right are compared
--
-- ERROR PROCESSING:
-- NONE
--
function ">=" (Left Kernel Time, Right Kernel Time)
return Boolean;

pragma Interface (Assemble: ">=");
pragma Import Function (Internal, External: "GTK ge");
Parameter Types ➔ Kernel Time, Kernel Time,
Mechanism ➔ Reference, Reference,
Result_Type ➔ Boolean;

end Generic Kernel Time;
pragma (page);
/ / MODULE NAME:
/ / Kernel time
/ /
/ / MODULE TYPE
/ / package specification
/ /
/ / MODULE PURPOSE:
/ / provide the abstraction of Kernel time
/ /
/ / ------------------------------------------------------------------------
/ / MODULE DESCRIPTION:
/ / Kernel time is a hand instantiation of package generic kernel time
/ /
/ / MODULE CONTENTS:
/ / Kernel time (generic package specification)
/ / integral duration (type)
/ / Kernel time (type)
/ / seconds (overloaded (2) function)
/ / milliseconds (function)
/ / microseconds (function)
/ / ticks per second (constant)
/ / ticks per second value (generic formal parameter)
/ / zero Kernel time (constant)
/ / "*" (binary; function)
/ / "," (binary; function)
/ / "," (overloaded (2) function)
/ / "*" (function)
/ / "," (function)
/ / "*" (function)
/ / "," (function)
/ / "," (function)
/ /
/ / REFERENCES:
/ / DESIGN DOCUMENTS:
/ / Kernel Facilities Definition Document
/ / -REQ: 12.1.1, 12.1.5, 12.1.6, 12.1.25
/ / -PRIM: N/A
/ / -TIME: N/A
/ /
/ / USER'S MANUAL:
/ / Kernel User's Manual
/ /
/ / TESTING AND VALIDATION:
/ / Unit tests and integration testing with Time keeper
/ /
/ / NOTES:
/ / this package must be instantiated and used in place of the
/ / compiler provided Package Calendar
/ / this package provides an abstract data type representing time
/ / for use by the kernel. It is at the lowest level of a set of
/ / abstracts that jointly respond to Requirements Section 12
/ /
/ / MODIFICATION HISTORY:
/ / 18aug88 bamberg created (compiles clean)
/ / 18oct88 firth revised interface to assembler
/ / 24oct88 firth changed type of ticks per to long int
/ / 09jan89 firth added milliseconds' and microseconds
/ / 11jan89 firth added documentation
/ / 11jan89 firth final doc. update for beta release
/ / 16feb89 firth corrected error in documentation
/ / 20jun89 dve shifted to VMS definitions
/ / 18jul89 dve HAND INSTANTIATION due to DEC Ada limitation!!!
/ / generics MAY NOT have MACHINE CODE BODIES!
/ / 10aug89 dve removed dependency on package system
/ / fixed interface spec for GKT_seconds!
/ /
/ / VERSION: 3.0
/ /
/ / DISCLAIMER:
/ / The following copyright must be included in this software and
/ / all software utilizing this software.
/ /
/ / DISTRIBUTION:
/ / Approved for public release; distribution is unlimited.
/ /
/ / Copyright (c) 1989 by Carnegie Mellon University, Pittsburgh,
/ / PA. The Software Engineering Institute (SEI) is a federally
/ / funded research and development center established and operated
/ / by Carnegie Mellon University (CMU). Sponsored by the U.S.
/ / Department of Defense under contract F I96388-B-5-0003, the SEI
/ / is supported by the services and defense agencies, with the U.S.
/ / Air Force as the executive contracting agent.
/ / Permission to use, copy, modify, or distribute this software and
/ / its documentation for any purpose and without fee is hereby
/ / granted, provided that the above copyright notice appear in all
/ / copies and that both this copyright notice and this permission
/ / notice appear in supporting documentation. Further, the names
/ / Software Engineering Institute or Carnegie Mellon University may
/ / not be used in advertising or publicity pertaining to
/ / distribution of the software without specific, written prior
/ / permission. CMU makes no claims or representations about the
/ / suitability of this software for any purpose. This software is
/ / provided "as is" and no warranty, express or implied, is made
/ / by the SEI or CMU, as to the accuracy and functioning of the
/ / program and related program material, nor shall the fact of
/ / distribution constitute any such warranty. No responsibility is
/ / assumed by the SEI or CMU in connection herewith.
/ /
/ / with Hardware Interface, use Hardware Interface,
/ / to get: the duration

December 1989
pragma (page);

---

**package Kernel.Time**

is

---

```
```

---

```**BEGIN PSEUDO-GENERIC PARAMETERS**```

---

- the number of TICKs in a second (this value is never used within the Kernel except for the initialization of the constant, nor should it be used without the Kernel)

---

**Ticks_Per_Second_Value** : constant **HW_Long_Integer** := 500_000;

---

```**END PSEUDO-GENERIC PARAMETERS**```

---

```
```

---

- the constant number of TICKs per second (thus providing a mechanism of converting from Kernel time to application time)

---

**Ticks_Per_Second** : **HW_Long_Integer** := **Ticks_Per_Second_Value**;

---

- **Kernel.Time**, the time on which elapsed time and epoch time abstractions are built

- this time representation allows applications beginning at time zero to execute for some 15,000 years, so no overflow is monitored

- the user should ensure that adjusting any times does not approach the limit of this time representation, proper Kernel functioning is NOT guaranteed if time overflows

---

- **Kernel.Time** is represented as a signed 64-bit binary integer, representing a count of hundred-nanoseconds. Hence, a kernel time of 10,000,000 corresponds to one second.

- For the purposes of Ada definition, kernel.time is a record of two components, being respectively the low-order and high order 32 bits. The high order component can be correctly declared to be a signed 32-bit integer, of type **HW_Long_Integer**. The low order component, however, is properly an **UN SIGNED** 32-bit integer, which we have defined in **HW_INTERFACE** as **hw_long unsigned** for VAX Ada
Applications should use ONLY the exported constructor functions to
create values of type kernel_time or of types derived from it.

type Kernel Time is record
  Low: Hw_Long_Unsigned := 0;
  High: Hw_Long_Integer := 0;
end record;

for Kernel Time use record
  Low at 0 range 0..31;
  High at 4 range 0..31;
end record;

-- the value of zero for the Kernel_time abstraction

Zero Kernel Time: constant Kernel Time := (0,0);

-- the range of durations represented as integral values

type Integral Duration is new Hw_Long_Integer;

pragma (page);

function Seconds (   
  Ada_Duration : Hw_Duration) return Kernel_Time;

-- The pragmas INTERFACE and IMPORT FUNCTION can be found following
-- the second declaration of the function (as per VAX RTI 13.9[note])

:// SUBPROGRAM NAMP:
:// seconds
:// DESCRIPTION:
:// conversion routine from Ada duration type to Kernel_time
:// /REQ: 12.1.23, 12.1.24
:// /PRIM: N/A
:// /TIME: N/A
:// PARAMETERS:
:// Ada duration
:// the time to convert in Ada duration format (note that this limits
:// legal values to 0 to 86400 seconds, 1 day)
:// default value:
:// none
:// function return value
:// the input duration in Kernel_time format (note that this contains
:// time up to 1 day)
:// PRECONDITIONS:
:// NONE
:// ACTIONS
:// returns the Ada duration time in Kernel_time format
:// POSTCONDITIONS:
:// NONE
:// ERROR PROCESSING:
:// NONE

pragma (page);
function Seconds (  
    An Integral Duration : Integral_Duration)  
return Kernel_Time;  
pragma Interface(Assembly, Seconds);  
pragma Import Function (Internal => Seconds,  
    External => "GKT_seconds",  
    Parameter Types => (Int Dur),  
    Mechanism => (Ref),  
    Result Type => Kernel_Time);  
pragma Import Function (Internal => Seconds,  
    External => "GKT_seconds2",  
    Parameter Types => (Int Dur),  
    Mechanism => (Ref),  
    Result Type => Kernel_Time);  

::: SUBPROGRAM NAME:  
::: Seconds  
::: DESCRIPTION:  
::: conversion routine from integral elapsed times to Kernel time  
::: - REQ: 12.1.23, 12.1.24  
::: - PRIM: N/A  
::: - TIME: N/A  
::: - PARAMETERS:  
::: - Duration: integral  
::: - the time to convert in integer format  
::: - default value: none  
::: function return value  
::: the input integral duration in Kernel_time format  
::: - PRECONDITIONS:  
::: - NONE  
::: - ACTIONS  
::: returns the integral duration time in Kernel_time format  
::: - POSTCONDITIONS:  
::: - NONE  
::: - ERROR_PROCESSING:  
::: - NONE  
:::  
::: SUBPROGRAM NAME:  
::: Milliseconds  
::: DESCRIPTION:  
::: conversion routine from integral elapsed times to Kernel time  
::: - REQ: 12.1.23, 12.1.24  
::: - PRIM: N/A  
::: - TIME: N/A  
::: - PARAMETERS:  
::: - Duration: integral milliseconds  
::: - the duration in milliseconds to convert to Kernel_time  
::: - default value: none  
::: function return value  
::: the input integral duration in Kernel_time format  
::: - PRECONDITIONS:  
::: - NONE  
::: - ACTIONS  
::: returns the duration time in Kernel_time format  
::: - POSTCONDITIONS:  
::: - NONE  
::: - ERROR_PROCESSING:  
::: - NONE  
:::  
::: SUBPROGRAM NAME:  
::: (page);  
::: (page);  
::: (page);  
::: (page);
function Microseconds (
    Integral Microseconds : Integral_Duration)
return Kernel_Time;
pragma Interface(Assembly, Microseconds);
pragma Import Function (Internal = Microseconds,
    External = "GKT microsseconds",
    Parameter Types = (Integral_Duration),
    Mechanism = (Value),
    Result_Type = Kernel_Time);

/* SUBPROGRAM NAME:
   seconds
   */
/* DESCRIPTION:
   conversion routine from integral elapsed times to Kernel time
   */
/* -REQ: 12.1.23, 12.1.24
   */
/* -PRIM: N/A
   */
/* -TIME: N/A
   */
/* PARAMETERS:
   integral microsseconds
   the duration in microseconds to convert to kernel time
   default value:
   none
   function return value
   the input integral duration in Kernel time format
   */
/* PRECONDITIONS:
   NONE
   */
/* ACTIONS
   returns the integral duration time in Kernel time format
   */
/* POSTCONDITIONS:
   NONE
   */
/* ERROR PROCESSING:
   NONE
   */
/* SUBPROGRAM NAME:
   */
/* DESCRIPTION:
   adds two Kernel time values returning a Kernel time value
   */
/* -REQ: 12.1.23, 12.1.24
   */
/* -PRIM: N/A
   */
/* -TIME: N/A
   */
/* PARAMETERS:
   left
   left operand of the addition operator
   default value:
   none
   right
   right operand of the addition operator
   default value:
   none
   function return value
   result of the addition operator
   */
/* PRECONDITIONS:
   NONE
   */
/* ACTIONS
   takes the mathematical sum of the two Kernel time values, returning a
   properly formatted Kernel time result
   */
/* POSTCONDITIONS:
   the sum of left and right is computed and returned
   */
/* ERROR PROCESSING:
   NONE
   */
function ** {  
Left Kernel Time;  
Right: Kernel Time)  
return Kernel Time;  

-- The pragmas INTERFACE and IMPORT FUNCTION can be found following  
the second declaration of the function (as per VAX RTL 13.9)(notes))

/**/SUBPROGRAM NAME:  
/** """"  
/**  
/** DESCRIPTION  
/** subtracts the right Kernel time value from the left Kernel time  
/** value returning a Kernel time value  
/**  
/** --REQ:  12.1.23, 12.1.24  
/**  
/** --PRIM: N/A  
/**  
/** --TIME: N/A  
/**  
/** PARAMETERS:  
/**  
/** left  
/**  
/** left operand of the subtraction operator (subtrahend)  
/**  
/** default value:  
/**  
/** none  
/**  
/** right  
/**  
/** right operand of the subtraction operator (minuend)  
/**  
/** default value:  
/**  
/** none  
/**  
/** function return value  
/**  
/** result of the subtraction operator  
/**  
/** PRECONDITIONS:  
/**  
/** NONE  
/**  
/** ACTIONS  
/** takes the mathematical difference of the two Kernel time values by  
/** subtracting the right operand from the left operand, returning a  
/** properly formatted Kernel time result  
/**  
/** POSTCONDITIONS:  
/**  
/** the difference of left and right is computed and returned  
/**  
/** ERROR PROCESSING:  
/**  
/** NONE  
/**  
/**  
/** */}
function """
  Left : Hw Long Integer;
  Right : Kernel Time)
  return Kernel_Time;

pragma Interface(Assembly, "");
pragma Import Function (Internal => "",
  External => "GKT multiply1",
  Parameter Types => (Kernel Time, Hw Long Integer),
  Mechanism => (Reference, Value),
  Result Type => Kernel Time);
pragma Import Function (Internal => "",
  External => "GKT multiply2",
  Parameter Types => (Hw Long Integer, Kernel Time),
  Mechanism => (Value, Reference),
  Result Type => Kernel Time);

"""

-- the product of left and right is computed and returned
--
-- ERROR PROCESSING:
-- none
--
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

pragma (page);

-- SUBPROGRAM NAME:
-- ""
--
-- DESCRIPTION:
-- multiplies an integer value by a Kernel time value returning a
-- Kernel time value
--
-- --REQ 12.1.23, 12.1.24
--
-- --PRM N/A
--
-- --TIME N/A
--
-- PARAMETERS:
-- left
-- left operand of the multiplication operator
--
-- default value
-- none
--
-- right
-- right operand of the multiplication operator
--
-- default value
-- none
--
-- function return value
-- result of the multiplication operator
--
-- PRECONDITIONS:
-- none
--
-- ACTIONS:
-- takes the mathematical product of the integer and the Kernel time
-- value, returning a properly formatted Kernel time result
--
-- POSTCONDITIONS:
function "i"( 
Left : Kernel.Time;
Right : hw Long Integer;
return Kernel.Time;

pragma interface(Assembly, "i");
pragma import function (Internal => "i");
\ External => "GKT divide",
\ Parameter Types => (Kernel.Time, hw Long Integer),
\ Mechanism => (Reference, Value),
\ Result_Type => Kernel.Time);

/* ******************************************************************
*/
/* DESCRIPTION
*/
/* divides a Kernel time value by an integer value returning a
*/
/* Kernel time value
*/
/*
*/
/* --REQ: 12.1.23, 12.1.24
*/
/*
*/
/* --PRIM: N/A
*/
/*
*/
/* --PARAMETERS:
*/
/* -- Left
*/
/* left operand of the division operator (dividend)
*/
/*
*/
/* -- default value
*/
/* none
*/
/*
*/
/* -- Right
*/
/* right operand of the division operator (divisor)
*/
/*
*/
/* -- default value
*/
/* none
*/
/*
*/
/* -- function return value
*/
/* result of the division operator
*/
/*
*/
/* -- PRECONDITIONS:
*/
/* none
*/
/*
*/
/* -- ACTIONS
*/
/* takes the mathematical quotient of the Kernel time value and the
*/
/* integer, truncating the result toward zero if necessary, returning
*/
/* a properly formatted Kernel time result
*/
/*
*/
/* --POSTCONDITIONS
*/
/* the quotient of left divided by right is computed and returned
*/
/*
*/
/* --ERROR PROCESSING:
*/
/* none
*/

/*----------------------------------------*/
pragma (page):
function "*"(
  left Kernel Time)
return Kernel Time;
pragma Interface(Assembly, "*"),
pragma Import Function (Internal  => "*", 
  External => "GKT add", 
  Parameter Types => (Kernel Time, Kernel Time), 
  Mechanism => (Reference, Reference), 
  Result Type => Kernel Time);
pragma Import Function (Internal  => "*", 
  External => "GKT unary plus", 
  Parameter Types => (Kernel Time), 
  Mechanism => (Reference), 
  Result Type => Kernel Time);

------------------------------------------------------------------------
// SUBPROGRAM NAME:  
// "*"
// DESCRIPTION: 
// unary plus of a Kernel time value returning a Kernel time value
// 
// REQUIREMENTS:  
// 12.1.23, 12.1.24
// 
// PARAMETERS:  
// left
//   left operand of the unary plus operator
//   default value: none
//   function return value
//     result of the unary plus operator
// 
// PRECONDITIONS: 
// NONE
// 
// ACTIONS:  
// takes the mathematical unary plus of the Kernel time value, returning 
// a properly formatted Kernel time result
// 
// POSTCONDITIONS:  
// the unary plus of left is computed and returned
// 
// ERROR PROCESSING: 
// NONE
// 
//--------------------------------------------------------------------------------

pragma (page);

function "-"(
  left Kernel Time)
return Kernel Time;
pragma Interface(Assembly, ",")
pragma Import Function (Internal  => ",", 
  External => "GKT subtract", 
  Parameter Types => (Kernel Time, Kernel Time), 
  Mechanism => (Reference, Reference), 
  Result Type => Kernel Time);
pragma Import Function (Internal  => ",", 
  External => "GKT unary minus", 
  Parameter Types => (Kernel Time), 
  Mechanism => (Reference), 
  Result Type => Kernel Time);

------------------------------------------------------------------------
// SUBPROGRAM NAME:  
// ","
// DESCRIPTION:  
// unary negation of a Kernel time value returning a Kernel time value
// 
// REQUIREMENTS: 
// 12.1.23, 12.1.24
// 
// PARAMETERS:  
// left
//   left operand of the unary negation operator
//   default value: none
//   function return value
//     result of the negation operator
// 
// PRECONDITIONS: 
// NONE
// 
// ACTIONS:  
// takes the mathematical unary negation of the Kernel time value, returning 
// a properly formatted Kernel time result
// 
// POSTCONDITIONS: 
// the unary negation of left is computed and returned
// 
// ERROR PROCESSING: 
// NONE
// 
//--------------------------------------------------------------------------------

pragma (page);
function "<" (  
Left : Kernel Time; 
Right : Kernel Time) 
return Boolean; 

pragma Interface (Assembly, "<"); 
pragma Import Function (Internal => "<"); 
External => "KGT.Is", 
Parameter Types => (Kernel Time, Kernel Time), 
Mechanism => (Reference, Reference), 
Result Type => Boolean); 

/* SUBPROGRAM NAME: */ 
"<" /* */ 
/* */ 
/* DESCRIPTION: */ 
"comparing two Kernel time values returning a Boolean true if the left is 
less than right, false otherwise */ 
/* */ 
/* PARAMETERS: */ 
left 
left operand of the less-than operator 
default value: none 
right 
right operand of the less-than operator 
default value: none 
function return value 
result of the less-than operator 
/* */ 
/* PRECONDITIONS: */ 
NONE /* */ 
/* */ 
/* ACTIONS: */ 
comparing the two Kernel time values, returning Boolean true if the 
left operand is less than the right, returning false otherwise 
/* */ 
/* POSTCONDITIONS */ 
the values of left and right are compared 
/* */ 
/* ERROR PROCESSING: */ 
NONE /* */ 
/* */ 
pragma (page); 

function "<=" (  
Left : Kernel Time; 
Right : Kernel Time) 
return Boolean; 

pragma Interface (Assembly, "<="); 
pragma Import Function (Internal => "<="); 
External => "KGT.Is", 
Parameter Types => (Kernel Time, Kernel Time), 
Mechanism => (Reference, Reference), 
Result Type => Boolean); 

/* SUBPROGRAM NAME: */ 
"<=" /* */ 
/* */ 
/* DESCRIPTION: */ 
"comparing two Kernel time values returning a Boolean true if the left is 
less than or equal to the right, false otherwise */ 
/* */ 
/* REQ: */ 12.1.23, 12.1.24 
/* */ 
/* PRIM: */ N/A 
/* */ 
/* TIME: */ N/A 
/* */ 
/* PARAMETERS: */ 
left 
left operand of the less-than-or-equal-to operator 
default value: none 
right 
right operand of the less-than-or-equal-to operator 
default value: none 
function return value 
result of the less-than-or-equal operator 
/* */ 
/* PRECONDITIONS: */ 
NONE /* */ 
/* */ 
/* ACTIONS: */ 
comparing the two Kernel time values, returning Boolean true if the 
left operand is less than or equal to the right, returning false 
otherwise 
/* */ 
/* POSTCONDITIONS: */ 
the values of left and right are compared 
/* */ 
/* ERROR PROCESSING: */ 
NONE /* */ 
/* */ 
pragma (page); 

- 419  420  December 1989  December 1989
```
function "\(\text{\textgreater}\)" (
    Left : Kernel.Time;
    Right : Kernel.Time)
return Boolean;

pragma Interface(Assembly, ":>");
pragma Import Function (Internal, External, "GKT gr", Parameter Types => (Kernel.Time, Kernel.Time), Mechanism => (Reference, Reference), Result Type => Boolean);

`/` SUBPROGRAM NAME:
`/` "\(\text{\textgreater}\)"
`/`
`/` DESCRIPTION:
`/` compares two Kernel time values returning a Boolean true if left is
`/` greater than right, false otherwise
`/`
`/` -REQ: 12.1.23, 12.1.24
`/` -PRM: N/A
`/` -TIME: N/A
`/` PARAMETERS:
`/` left
`/` left operand of the greater-than operator
`/`
`/` default value:
`/` none
`/`
`/` right
`/` right operand of the greater-than operator
`/`
`/` default value:
`/` none
`/`
`/` function return value
`/` result of the greater-than operator
`/`
`/` PRECONDITIONS:
`/` NONE
`/`
`/` ACTIONS:
`/` compares the two Kernel time values, returning Boolean true if the
`/` left operand is greater than the right, returning false otherwise
`/`
`/` POSTCONDITIONS:
`/` the values of left and right are compared
`/`
`/` ERROR PROCESSING:
`/` NONE
```

421  December 1989

422  December 1989
function "=>" (
    left : Kernel.Time;
    right : Kernel.Time)
return Boolean;

pragma Interface(Assembly, "=>");
pragma Import Function (Internal "=>" => "=>");
   External "=>" "KRT.ge",
   Parameter Types => (Kernel.Time, Kernel.Time),
   Mechanism => (Reference, Reference),
   Result_Type => Boolean);

-- SUBPROGRAM NAME:
-- "=>"

-- DESCRIPTION:
-- compares two Kernel time values returning a Boolean true if left is
-- greater than or equal to right, false otherwise

-- REQ: 12.1.23, 12.1.24
-- PRIM: N/A
-- TIME: N/A

-- PARAMETERS:
-- left
-- left operand of the greater-than-or-equal-to operator
-- default value:
-- none

-- right
-- right operand of the greater-than-or-equal-to operator
-- default value:
-- none

-- function return value
-- result of the greater-than-or-equal-to operator

-- PRE-CONDITIONS:
-- NONE

-- ACTIONS:
-- compares the two Kernel time values, returning Boolean true if the
-- left operand is greater than or equal to the right, returning
-- false otherwise

-- POSTCONDITIONS:
-- the values of left and right are compared

-- ERROR PROCESSING:
-- NONE
Process Name Type 132
Process State 88
Process Table 349
Process Table Entry 331
Process Type 313
Processor Identifier 66
Schedule Attributes Information 318
Semaphore 310
Semaphore Attributes Information 327
Semaphore Head 310
Semaphore Head Pr 310
Tool, Interface Attributes Information 329

Unary Minus, function 35
Unary Plus, function 34
Unchecked Conversion, with 6, 344, 364

Wait, procedure 201, 203
Who Am I, function 213

With
Communication Globals 161, 303, 344
Context Stack Area 303
Context Switcher Globals 303
Datagram Globals 303
Generic Alarm Management 266
Generic Communication Globals 157
Generic Communication Management 191
Generic Interunit Globals 327
Generic Interrupt Management 241
Generic Kernel Time 45
Generic Network Globals 99
Generic Process Attribute Modifiers 208
Generic Process Attribute Readers 217
Generic Process Managers 151
Generic Process Managers, Global 134
Generic Process Table 340
Generic Processor Management 128
Generic Queue Manager 303
Generic Schedule Types 90
Generic Semaphore Management 255
Generic Time, Global 82
Generic Time Management 281
Generic Time source Management 290
Generic Tool Interface 357
Hardware Interface 17, 40, 85, 94, 102, 110,
118, 131, 139, 153, 155, 161, 210, 220, 230, 290,
303, 344, 378, 402
Interrupt Globals 290, 344
Kernel Exceptions 85, 94, 118, 139, 161, 196,
210, 220, 244, 258, 269, 284, 303, 344
Kernel Time 40, 284, 290, 303
Network Globals 102, 110, 161, 303, 344
Process Managers, Global 120, 303
Process Table 102, 110, 139, 161, 195, 210,
220, 244, 344
Schedule Types 130, 161, 195, 210, 244, 258,
269, 303, 344
System 6, 364
Time, Global 118, 161, 195, 244, 258, 269,
284, 290, 344