```
with Ada.Interrupts.Names; use Ada.Interrupts;
with System; use System;
package body Adc_Device_Driver is
  Bits_In_Word : constant := 16;
  Word : constant := 2; -- bytes in word
  type Flag is (Down, Set);

  type Control_Register is record
    Ad_Start : Flag;
    IE       : Flag;
    Done     : Flag;
    Ch       : Channel;
    Error    : Flag;
  end record;

  for Control_Register use record
    -- specifies the layout of the control register
    Ad_Start at 0*Word range 0..0;
    IE       at 0*Word range 6..6;
    Done     at 0*Word range 7..7;
    Ch       at 0*Word range 8..13;
    Error    at 0*Word range 15..15;
  end record;

  for Control_Register'Size use Bits_In_Word;
    -- the register is 16 bits long
  for Control_Register'Alignment use Word;
    -- on a word boundary
  for Control_Register'Bit_Order use Low_Order_First;

  type Data_Register is range 0 .. Max_Measure;
  for Data_Register'Size use Bits_In_Word;
    -- the register is 16 bits long

  Contr_Reg_Addr : constant Address :=
          System.Storage_Elements.To_Address(8#150002#);
  Data_Reg_Addr : constant Address :=
          System.Storage_Elements.To_Address(8#150000\);
  Adc_Priority : constant Interrupt_Priority := 63;
  Control_Reg : aliased Control_Register;
  for Control_Reg'Address use Contr_Reg_Addr;
      -- specifies the address of the control register
  Data_Reg : aliased Data_Register;
  for Data_Reg'Address use Data_Reg_Addr;
      -- specifies the address of the data register

  protected type Interrupt_Interface(Int_Id : Interrupt_Id;
                 Cr : access Control_Register;
                 Dr : access Data_Register) is
    entry Read(Chan : Channel; M : out Measurement);
  private
    entry Done(Chan : Channel; M : out Measurement);
    procedure Handler;
    pragma Attach_Handler(Handler, Int_Id);
    pragma Interrupt_Priority(Adc_Priority);
      -- see Chapter 13 for discussion on priorities
    Interrupt_Occurred : Boolean := False;
    Next_Request : Boolean := True;
  end Interrupt_Interface;

  Adc_Interface : Interrupt_Interface(Names.Adc,
                  Control_Reg'Access, Data_Reg'Access);
  -- this assumes that 'Adc' is registered as an
  -- Interrupt_Id in Ada.Interrupts.Names

  protected body Interrupt_Interface is
    entry Read(Chan : Channel; M : out Measurement)
          when Next_Request is
      Shadow_Register : Control_Register;
    begin
      Shadow_Register := (Ad_Start => Set, IE => Set,
            Done => Down, Ch => Chan, Error => Down);
      Cr.all := Shadow_Register;
      Interrupt_Occurred := False;
      Next_Request := False;
      requeue Done;
    end Read;

    procedure Handler is
    begin
      Interrupt_Occurred := True;
    end Handler;

    entry Done(Chan : Channel; M : out Measurement)
                              when Interrupt_Occurred is
    begin
      Next_Request := True;
      if Cr.Done = Set and Cr. Error = Down then
         M := Measurement(Dr.all);
      else
        raise Conversion_Error;
      end if;
    end Done;
  end Interrupt_Interface;

  procedure Read(Ch : Channel; M : out Measurement) is
  begin
    for I in 1..3 loop
      begin
        Adc_Interface.Read(Ch,M);
        return;
      exception
        when Conversion_Error => null;
      end;
    end loop;
    raise Conversion_Error;
  end Read;
end Adc_Device_Driver;
```