```
------------------------------------------------------------------------------
--                                                                          --
--                            GNAT RAVENSCAR for NXT                        --
--                   Copyright (C) 2010-2012, AdaCore                       --
--                                                                          --
------------------------------------------------------------------------------

--  Low-level I/O driver for the AVR.

with Interfaces; use Interfaces;

package NXT.AVR_IO is

   --------------
   -- Outgoing --
   --------------

   procedure Send_Initialization;
   --  send the mandatory initialization message to the AVR

   type Outgoing_AVR_Message is private;
   --  Represents all messages sent to the AVR

   Null_Outgoing_Msg : constant Outgoing_AVR_Message;

   function Power_Down_Message return Outgoing_AVR_Message;

   function Set_Sensor_Power_Message
     (Port       : Sensor_Id;
      Power_Type : Sensor_Power)
      return Outgoing_AVR_Message;

   function Set_Motor_Message
     (Motor : Motor_Id;
      Power : PWM_Value;
      Brake : Boolean)
      return Outgoing_AVR_Message;

   procedure Send_To_AVR (This : Outgoing_AVR_Message);
   --  computes and appends checksum, sends to AVR
```

```
41
42    --------------
43    -- Incoming --
44    --------------
45
46    type Incoming_AVR_Message is private;
47    --  Represents all messages received from the AVR
48
49    type Raw_ADC_Inputs is array (Sensor_Id) of Unsigned_16;
50    pragma Atomic_Components (Raw_ADC_Inputs);
51    --  A/D converter values, each with a range of 0 .. 1023
52
53    function Raw_Inputs (From : Incoming_AVR_Message) return Raw_ADC_Inputs;
54    --  Returns the raw input readings from the message. Does no actual
55    --  "decoding", just provides access.
56    pragma Inline_Always (Raw_Inputs);  -- just an accessor function
57
58    function Buttons_Sample (From : Incoming_AVR_Message) return Unsigned_16;
59    --  Returns the raw buttons reading contained within From
60    pragma Inline_Always (Buttons_Sample);  -- just an accessor function
61
62    function Battery_Reading (From : Incoming_AVR_Message) return Unsigned_16;
63    --  Returns the battery reading contained within From
64    pragma Inline_Always (Battery_Reading);  -- just an accessor function
65
66    procedure Receive_From_AVR
67      (This  : out Incoming_AVR_Message;
68       Valid : out Boolean);
69    --  Receives the next message from the AVR and places into This iff the
70    --  checksum in the received message is correct.  If the checksum is not
71    --  correct, no update to This takes place.
72    --  On return, Valid will be True if checksum in received message is
73    --  correct, otherwise it will be False.
74
75 private
76
77    --  outgoing message representation, sent from ARM to AVR
78
79    type PWM_Output_Values is array (0 .. 3) of PWM_Value;
80    for PWM_Output_Values'Component_Size use 8;
```

```
81    for PWM_Output_Values'Size use 32;  -- confirming
82
83    type Port_Bits is array (Sensor_Id) of Boolean;
84    for Port_Bits'Component_Size use 1;
85    for Port_Bits'Size use 4;  -- confirming
86
87    type Sensor_Power_Control is
88       record
89          Pulsed_9V   : Port_Bits;  -- "active"
90          Constant_9V : Port_Bits;
91       end record;
92
93    for Sensor_Power_Control use
94       record
95          Pulsed_9V   at 0 range 0 .. 3;
96          Constant_9V at 0 range 4 .. 7;
97       end record;
98
99    for Sensor_Power_Control'Size use 8;  -- confirming
100
101   type Motor_Braking_Control is (No_Braking, Braking);
102   for Motor_Braking_Control use (No_Braking => 0, Braking => 1);
103   --  the above is confirming, but the values are essential so we make it
104   --  explicit to prevent accidental reordering later
105
106   type Motor_Braking_Modes is array (0 .. 7) of Motor_Braking_Control;
107   --  bit 0 is Motor A, 1 is Motor B, and 2 is Motor C; others are unused
108   for Motor_Braking_Modes'Component_Size use 1;
109   for Motor_Braking_Modes'Size use 8;  -- confirming
110
111   type Outgoing_AVR_Message is
112      record
113         Power_Command : Unsigned_8;
114         PWM_Frequency : Unsigned_8; --  in KHz units, range 1 .. 32
115         PWM_Values    : PWM_Output_Values;
116         Output_Mode   : Motor_Braking_Modes;
117         Input_Power   : Sensor_Power_Control;
118         Checksum      : Unsigned_8;
119      end record;
120
```

```
121    for Outgoing_AVR_Message use
122       record
123          Power_Command at 0 range 0 .. 7;
124          PWM_Frequency at 1 range 0 .. 7;
125          PWM_Values    at 2 range 0 .. 31;
126          Output_Mode   at 6 range 0 .. 7;
127          Input_Power   at 7 range 0 .. 7;
128          Checksum      at 8 range 0 .. 7;
129       end record;
130
131    Null_Outgoing_Msg : constant Outgoing_AVR_Message :=
132      (Power_Command => 0,
133       PWM_Frequency => 0,
134       PWM_Values    => (0, 0, 0, 0),
135       Output_Mode   => (others => No_Braking),
136       Input_Power   => ((others => False), (others => False)),
137       Checksum      => 0);
138
139    --  incoming message representation, received from AVR
140
141    type Incoming_AVR_Message is
142       record
143          Inputs   : Raw_ADC_Inputs;  --  Raw a/d converter values [0..1023]
144          Buttons  : Unsigned_16;     --  Raw a/d converter values [0..1023]
145          Battery  : Unsigned_16;     --  Raw a/d converter values [0..1023]
146          Checksum : Unsigned_8;
147       end record;
148
149    for Incoming_AVR_Message use
150       record
151          Inputs   at 0  range 0 .. 63;
152          Buttons  at 8  range 0 .. 15;
153          Battery  at 10 range 0 .. 15;  -- also contains firmware info
154          Checksum at 12 range 0 .. 7;
155       end record;
156
157 end NXT.AVR_IO;
158
```