

trackforce.vivado\trackforce.srsc\sources_1\imports\imports\source\gpsdecode.sv

```
1 // Macros for message ASCII
2 `define LF 10
3 `define CR 13
4 `define DOLLAR 36
5 `define COMMA 44
6 `define STAR 42
7
8
9 // Macro for message bits
10 `define CHAR_LENGTH 8
11
12 // gpsdecode
13 // -----
14 // Receives a NEMA message from the UART interface and decode it to signals
15 // used by other modules
16 // Only cares about GPVTG message
17 // Ready Valid& interface
18 //
19 // Inputs
20 // -----
21 // clk_i - Clock
22 // rst_i - Reset
23 // [7:0] data_i - ASCII char read from FIFO
24 // valid_i - Input message is valid
25 // ready_i - Next module is ready to receive data
26 //
27 // Outputs
28 // -----
29 //
30 // ready_o - Decoder is ready to receive a new message
31 // vkmh_o - Speed in km/h
32 //
33 // Example data from GPS
34 // $GPVTG,0.00,T,,M,0.00,N,0.00,K,N*32\r\n
35 //
36 // TalkerID, course, Reference, Course, Reference, Speed, Unit (knot), Speed, Unit (km/h),
37 // Mode, Checksum, CR, LF
38 // should get Speed before Unit (km/h)
39
40 module gpsdecode(
41     input [0:0] clk_i,
42     input [0:0] rst_i,
43     input [7:0] data_i,
44     // data from FIFO is valid
45     input [0:0] valid_i,
46     // next module is ready to receive data
47     input [0:0] ready_i,
48     // data is ready to be read
49     output [0:0] ready_o,
50     output [0:0] valid_o,
51     output [7:0] vkmh_o
52 );
```

```

53
54
55 logic [0:0] ready_r;
56 logic [0:0] valid_r;
57 logic [7:0] vkmh_r;
58
59 //
60 // State machine
61 //
62 // 0 - Idle      // Wait for $
63 // 1 - MsgID     // Parse message ID, go to Continue if parsed as GPVTG. When msgidindex_r ==
// 3 and msgidmatch_r == 1, stay in msgID and go to field when msgidindex_r == 5. Else go to
// ignore
64 // 2 - Continue  // Parse message until 7th comma
65 // 3 - Speed     // Parse Speed
66 // 4 - Done      // Done parsing speed, set valid_o to 1.
67
68 typedef enum logic [2:0] {
69     IDLE,
70     MSGID,
71     FIELD,
72     IGNORE,
73     DONE
74 } state_t;
75
76 state_t state_r, state_n;
77
78 logic [7:0] msgidindex_r;
79 logic [7:0] commaindex_r;
80 logic [7:0] speedindex_r;
81
82 logic [0:0] msgidmatch_r;
83
84 always_ff @(posedge clk_i) begin
85     if (rst_i) begin
86         state_r <= IDLE;
87     end
88     else begin
89         state_r <= state_n;
90     end
91 end
92
93 // state transitions
94
95 always_comb begin
96     if (rst_i) begin
97         state_n = IDLE;
98     end
99     else begin
100         case (state_r)
101             IDLE: begin
102                 if (data_i == "$") begin
103                     state_n = MSGID;
104                 end
105                 else begin
106                     state_n = IDLE;
107                 end

```

```

108         end
109         MSGID: begin
110             if (msgidxmatch_r == 1 && msgidxindex_r == 5) begin
111                 state_n = FIELD;
112             end
113             else if (msgidxmatch_r == 0 && msgidxindex_r > 3) begin
114                 state_n = IGNORE;
115             end
116             else begin
117                 state_n = MSGID;
118             end
119         end
120         FIELD: begin
121             if (speedindex_r == 3) begin
122                 state_n = DONE;
123             end
124             else begin
125                 state_n = FIELD;
126             end
127         end
128         IGNORE: begin
129             if (data_i == "$") begin
130                 state_n = MSGID;
131             end
132             else begin
133                 state_n = IGNORE;
134             end
135         end
136         DONE: begin
137             state_n = IDLE;
138         end
139         default : begin
140             state_n = IDLE;
141         end
142     endcase
143 end
144
145 end
146
147 end
148
149
150 // msgindex logic
151 // increment once whenever a new char is read (pulse of valid_i) when in msgid state
152
153 always_ff @(posedge clk_i) begin
154     if (rst_i) begin
155         msgidxindex_r <= 0;
156     end
157     else begin
158         case (state_r)
159             IDLE: begin
160                 msgidxindex_r <= 0;
161             end
162             MSGID: begin
163                 if (valid_i) begin

```

```

164         msgidindex_r <= msgidindex_r + 1;
165     end
166     else begin
167         msgidindex_r <= msgidindex_r;
168     end
169 end
170 FIELD: begin
171     msgidindex_r <= 0;
172 end
173 IGNORE: begin
174     msgidindex_r <= 0;
175 end
176 DONE: begin
177     msgidindex_r <= 0;
178 end
179 default : begin
180     msgidindex_r <= 0;
181 end
182 endcase
183 end
184 end
185
186
187 // msgid match logic
188 //
189 // check at index 3 if the char is V
190 // if it is, msgidmatch_r = 1
191 // else msgidmatch_r = 0
192
193 always_ff @(posedge clk_i) begin
194     if (rst_i) begin
195         msgidmatch_r <= 0;
196     end
197     else begin
198         case (state_r)
199             IDLE: begin
200                 msgidmatch_r <= 0;
201             end
202             MSGID: begin
203                 if (msgidindex_r == 3) begin
204                     if (data_i == "V") begin
205                         msgidmatch_r <= 1;
206                     end
207                     else begin
208                         msgidmatch_r <= msgidmatch_r;
209                     end
210                 end
211                 else begin
212                     msgidmatch_r <= msgidmatch_r;
213                 end
214             end
215             FIELD: begin
216                 msgidmatch_r <= 0;
217             end
218             IGNORE: begin
219                 msgidmatch_r <= 0;

```

```

220         end
221         DONE: begin
222             msgidmatch_r <= 0;
223         end
224         default : begin
225             msgidmatch_r <= 0;
226         end
227     endcase
228 end
229 end
230
231
232
233
234
235
236 // commaindex logic
237
238 always_ff @(posedge clk_i) begin
239     if (rst_i) begin
240         commaindex_r <= 0;
241     end
242     else begin
243         case (state_r)
244             IDLE: begin
245                 commaindex_r <= 0;
246             end
247             MSGID: begin
248                 commaindex_r <= 0;
249             end
250             FIELD: begin
251                 if (valid_i) begin
252                     if (data_i == ",")
253                         commaindex_r <= commaindex_r + 1;
254                     end
255                     else begin
256                         commaindex_r <= commaindex_r;
257                     end
258                 end
259             IGNORE: begin
260                 commaindex_r <= 0;
261             end
262             DONE: begin
263                 commaindex_r <= 0;
264             end
265             default : begin
266                 commaindex_r <= 0;
267             end
268         endcase
269     end
270 end
271
272
273 // speedindex logic
274
275 always_ff @(posedge clk_i) begin

```



```

332         "2" : speed_r <= 20;
333         "3" : speed_r <= 30;
334         "4" : speed_r <= 40;
335         "5" : speed_r <= 50;
336         "6" : speed_r <= 60;
337         "7" : speed_r <= 70;
338         "8" : speed_r <= 80;
339         "9" : speed_r <= 90;
340         default: speed_r <= 0;
341     endcase
342 end
343 else if (speedindex_r == 1) begin
344     case (data_i)
345         "0" : speed_r <= speed_r + 0;
346         "1" : speed_r <= speed_r + 1;
347         "2" : speed_r <= speed_r + 2;
348         "3" : speed_r <= speed_r + 3;
349         "4" : speed_r <= speed_r + 4;
350         "5" : speed_r <= speed_r + 5;
351         "6" : speed_r <= speed_r + 6;
352         "7" : speed_r <= speed_r + 7;
353         "8" : speed_r <= speed_r + 8;
354         "9" : speed_r <= speed_r + 9;
355         // this should handle empty msg
356         default: speed_r <= speed_r + 0;
357     endcase
358 end
359 end
360 end
361 else begin
362     speed_r <= speed_r;
363 end
364 end
365 IGNORE: begin
366     speed_r <= 0;
367 end
368 DONE: begin
369     speed_r <= 0;
370 end
371 default : begin
372     speed_r <= 0;
373 end
374 endcase
375 end
376 end
377
378
379
380
381 //
382
383
384 always_ff @(posedge clk_i) begin
385     if (rst_i) begin
386         ready_r <= 0;
387         valid_r <= 0;

```

```

388     end
389     else begin
390         case (state_r)
391             IDLE: begin
392                 ready_r <= 1;
393                 valid_r <= 0;
394             end
395             MSGID: begin
396                 ready_r <= 1;
397                 valid_r <= 0;
398             end
399             FIELD: begin
400                 if (speedindex_r == 3) begin
401                     ready_r <= 0;
402                     valid_r <= 1;
403                 end
404                 else begin
405                     ready_r <= 1;
406                     valid_r <= 0;
407                 end
408             end
409             IGNORE: begin
410                 ready_r <= 1;
411                 valid_r <= 0;
412             end
413             DONE: begin
414                 ready_r <= 0;
415                 valid_r <= 0;
416             end
417             default : begin
418                 ready_r <= 0;
419                 valid_r <= 0;
420             end
421         endcase
422     end
423 end
424
425
426 // outputs
427
428 assign ready_o = ready_r;
429 assign valid_o = valid_r;
430 assign vkmh_r = speed_r;
431 assign vkmh_o = vkmh_r;
432
433
434
435 endmodule

```