```
#include "display.h"
#include "to_7seg.h"
#include "range_finder.h"

// extended state -------------------------------------------------------
  // "basic" state
typedef enum {DP_IDLE, DP_MEAS, DP_MUX} dp_state_t;
static dp_state_t g_dp_state;

  // externally reachable objects
bool              gb_display_on_msg;          // start display
bool              gb_display_off_msg;         // stop display
bool              gb_display_update_msg;      // update display

     // parameter for gb_display_update_msg (and gb_display_on_msg)

uint16_t       g_display_segs;             // bits 6:0 are the segments of the right display
bool            gb_display_brightness_msg;  // update brightness

 // parameter for gb_display_brightness_msg (and gb_display_on_msg)
uint8_t        g_display_brightness;       // percentage (0-99) of brightness

uint8_t        high_izq;
uint8_t        low_der;
uint16_t       valor_anterior;
bool           sentido;

 // output messages
bool volatile  gb_display_can_sleep;       //this FSM can sleep


  // hardware resources
static PwmOut *g_dp_dsl;
static PwmOut *g_dp_dsr;
static BusOut *g_dp_seven_seg;

static Ticker tick_4ms;        // tiempo de multiplexacion

  // internal objects
static bool volatile  gb_dp_initd;            // true after call to dp_init()
static bool volatile  tick_4ms_evnt;          // timeout elapsed event


// end of extended state  -----------------------------------------------

// ISRs -----------------------------------------------------------------

  // timeout ISR
static void tick_4ms_isr(void) {
  tick_4ms_evnt = true;
  gb_display_can_sleep = false;
}


// FSM  -----------------------------------------------------------------
void display_fsm (void) {
  if (gb_dp_initd) {  // protect against calling dp_fsm() w/o a previous call to dp_init()

    switch (g_dp_state) {

      // complete this code to achieve the FSM functionality  ++++++++++
       case DP_IDLE :

        if(gb_display_on_msg){
          tick_4ms.attach_us(tick_4ms_isr,4000);
          gb_display_on_msg = false;
           g_dp_state = DP_MEAS;

        }else if (gb_display_off_msg){
              gb_display_off_msg = false;
              tick_4ms.detach();
               *g_dp_dsr = 0;
               *g_dp_dsl = 0;
                g_display_segs = 0;
              g_dp_state = DP_IDLE;
        }else{
        }

       case DP_MEAS:

          if(g_display_segs  == valor_anterior){
               g_dp_state = DP_MUX;
              gb_display_brightness_msg = true;

```

```
85              }else{
86                valor_anterior = g_display_segs;
87                gb_display_update_msg = true;
88                gb_display_brightness_msg = true;
89
90                g_dp_state = DP_MUX;
91              }
92
93           case DP_MUX:
94
95                 if(gb_display_update_msg){
96                    gb_display_update_msg = false;
97                      high_izq = g_display_segs >> 8;
98                      low_der = g_display_segs & 0xFFU;
99
100                }else  if(gb_display_brightness_msg){
101                      gb_display_brightness_msg = false;
102
103                        if(tick_4ms_evnt){
104                          tick_4ms_evnt = false;
105
106                          sentido = !sentido;
107
108                          if(sentido){
109                            *g_dp_dsr = 0;
110                            *g_dp_dsl = 1;
111                            *g_dp_seven_seg = high_izq;
112                            g_dp_dsl -> pulsewidth_us(g_display_brightness);
113
114                          }else{
115                            *g_dp_dsr = 1;
116                            *g_dp_dsl = 0;
117                            *g_dp_seven_seg = low_der;
118                            g_dp_dsr -> pulsewidth_us(g_display_brightness);
119                          }
120                       }
121                           g_dp_state = DP_IDLE;
122                      }else{
123
124                      }
125
126                   break;
127
128
129           //  ------------------------------------------------------------------
130       } // switch (rf_state)
131
132
133       __disable_irq();
134       if(!gb_display_on_msg && !gb_display_off_msg && !gb_display_update_msg &&
    !gb_display_brightness_msg && !tick_4ms_evnt ) {
135         gb_display_can_sleep = true;
136       }
137       __enable_irq();
138     } // if (gb_rf_initd)
139   }
140   // end of FSM ------------------------------------------------------------
141
142   // initialize FSM machinery  ---------------------------------------------
143   void display_init(PwmOut *dsl, PwmOut *dsr, BusOut *seven_seg) {
144     if (!gb_dp_initd) {
145       gb_dp_initd = true;    // protect against multiple calls to rf_init
146
147       // initialize state
148       g_dp_state = DP_IDLE;
149
150       // initial actions
151       g_dp_dsl = dsl;
152       g_dp_dsr = dsr;
153       g_dp_seven_seg = seven_seg;
154
155       valor_anterior = 0;
156
157        *g_dp_seven_seg = g_display_segs;
158         g_dp_dsl -> pulsewidth_us(g_display_brightness);
159         g_dp_dsr -> pulsewidth_us(g_display_brightness);
160     }
161   }
162   // end of FSM initialization  --------------------------------------------
163
164
165
166
167
```