SECABB_python_target_v4_1_3_2

```
/*
 * File:        Python control prototype
 *
 * Author:      Bruce Land, Group 06
 */

#include "config_1_3_2.h"
// threading library
#include "pt_cornell_1_3_2_python.h"
#include <math.h>

////////////////////////////////////
// graphics libraries
// SPI channel 1 connections to TFT
#include "tft_master.h"
#include "tft_gfx.h"

//#include "i2c_helper.h"
#include "VL53L1X_helper.h"

//== Timer 2 interrupt handler ==========================================
// direct digital synthesis of sine wave
#define two32 4294967296.0 // 2^32
#define Fs 50 //PWM frequency for chained timers23 PS=1
#define Fs4 1 //delay function frequqency, PS=1
#define Fs5 1 // scanning interrupt frequency, PS=64
#define WAIT {}
#define clock_speed 40000000 //40MHz clock speed
int TIMEOUT = clock_speed/Fs;
int TIMEOUT4 = 0xFFFF;
int TIMEOUT5 = 0xFFFF;
// DAC ISR
// A-channel, 1x, active
#define DAC_config_chan_A 0b0011000000000000
// B-channel, 1x, active
#define DAC_config_chan_B 0b1011000000000000
//
volatile unsigned int DAC_data ;// output value
volatile SpiChannel spiChn = SPI_CHANNEL2 ; // the SPI channel to use
volatile int spiClkDiv = 2 ; // 20 MHz max speed for DAC!!
// the DDS units:
// volatile unsigned int phase_accum_main, phase_incr_main=440.0*two32/Fs ;//
// DDS sine table
#define sine_table_size 256
volatile int sin_table[sine_table_size][3] ;
// the dds state controlled by python interface
volatile int dds_state = 1;
// the voltage specifed from python
volatile float V_data = 0;
// sine, sq, tri
volatile char wave_type = 0 ;
```

```
48
49  volatile int is_full;
50  volatile int toggle_led;
51
52  /* Robot scanning and save data variables */
53  volatile int isr_counter = 0;
54  int pan_location = 19000-2800;
55  int direction = 1; //direction1 == counterclock, direction0=clock
56  #define degree 2800 // degree of rotation = 78-16 alpha 180
57  uint16_t distance;
58  #define num_samples 20 //number of sample points we are scanning in the room
59  uint16_t distance_row[num_samples];
60  int row_index = 0;
61  volatile uint8_t is_ready;
62  uint8_t range_status;
63
64  void __ISR(_TIMER_3_VECTOR, ipl2) Timer3Handler(void)
65  {
66      // you MUST clear the ISR flag
67      mT3ClearIntFlag();
68  }
69
70  ////////////////////////////////////
71
72  // === outputs from python handler ==============================================
73  // signals from the python handler thread to other threads
74  // These will be used with the prototreads PT_YIELD_UNTIL(pt, condition);
75  // to act as semaphores to the processing threads
76  char new_string = 0;
77  char new_button = 0;
78  char new_toggle = 0;
79  char new_slider = 0;
80  char new_list = 0 ;
81  char new_radio = 0 ;
82  // identifiers and values of controls
83  // current button
84  char button_id, button_value ;
85  // current toggle switch/ check box
86  char toggle_id, toggle_value ;
87  // current radio-group ID and button-member ID
88  char radio_group_id, radio_member_id ;
89  // current slider
90  int slider_id;
91  float slider_value ; // value could be large
92  // current listbox
93  int list_id, list_value ;
94  // current string
95  char receive_string[64];
96
97  //==== Flags for Spy Robot ====
98  volatile int is_full;
99
100 int toggle = 0;
101
102 //timer thread ================================================================
103 //Sends data via serial communication to the python back end
```

```
104 static PT_THREAD (protothread_timer(struct pt *pt)) {
105 
106     PT_BEGIN(pt);
107 
108     while(1){
109         PT_YIELD_UNTIL(pt, row_index==num_samples);
110         int i;
111         for(i=0; i<num_samples-1; i++){
112             printf("%d,", distance_row[i]);
113         }
114         printf("%d\n", distance_row[num_samples-1]);
115         row_index=0;
116     }
117 
118     PT_END(pt);
119 }
120 //dataread thread ================================================================
121 //Pans robot and collects distance data for radar plot
122 static PT_THREAD (protothread_dataread (struct pt *pt)) {
123     PT_BEGIN(pt);
124 
125       if (direction == 1) { //if scanning clockwise
126         pan_location+=degree;
127         SetDCOC3PWM(pan_location);
128         if (pan_location >= 75000) {
129             direction = 0;
130             PT_YIELD_TIME_msec(100);
131         }
132     }
133       else  { //if scanning counterclockwise
134         pan_location-=degree;
135         SetDCOC3PWM(pan_location);
136         if (pan_location <= 19000) {
137             direction = 1;
138             PT_YIELD_TIME_msec(100);
139         }
140     }
141     VL53L1X_StartRanging();
142 
143     while (is_ready == 0){
144         VL53L1X_CheckForDataReady(&is_ready);
145     }
146     is_ready = 0;  //reset the flag
147     VL53L1X_GetDistance(&distance); //Get the result of the measurement from the sensor
148     VL53L1X_ClearInterrupt();
149     VL53L1X_StopRanging();
150     VL53L1X_GetRangeStatus(&range_status);
151     if(row_index<num_samples){
152         distance_row[row_index] = (uint16_t) ((float)distance)/10;  //in cm, rounded off
153         row_index++;
154     }
155     PT_YIELD(pt);
156 
157     PT_END(pt);
158 }
159 
```

```
160
161 // === Main  ======================================================
162
163 void main(void) {
164
165     // === set up DAC on big board ========
166     OpenTimer23(T2_ON | T2_SOURCE_INT | T2_PS_1_1 | T2_32BIT_MODE_ON, TIMEOUT);
167 //    // set up the timer interrupt with a priority of 2
168     ConfigIntTimer23(T23_INT_ON | T23_INT_PRIOR_2);
169     mT3ClearIntFlag(); // and clear the interrupt flag for servos
170
171     // set up the timer interrupt with a priority of 2
172     ConfigIntTimer2(T2_INT_ON | T2_INT_PRIOR_2);
173     mT2ClearIntFlag(); // and clear the interrupt flag
174
175     //setup timer 4 for delay timing
176     OpenTimer4(T4_ON | T4_SOURCE_INT | T4_PS_1_256, TIMEOUT4);
177
178     // Set up Timer 5 for ISR scanning the room and acquiring distance
179 //       //setup timer 4 for scanning
180     OpenTimer5(T5_ON | T5_SOURCE_INT | T5_PS_1_256, TIMEOUT5);
181 //
182     // 16 bit transfer CKP=1 CKE=1
183     // possibles SPI_OPEN_CKP_HIGH;   SPI_OPEN_SMP_END;  SPI_OPEN_CKE_REV
184     // For any given peripherial, you will need to match these
185
186     //================ I2C setup ====================//
187     // Enable channel
188     // BRG computed value for the baud rate generator. The value is
189     // calculated as follows: BRG = (Fpb / 2 / baudrate) - 2.
190     // (40Mhz / 2 / 400khz) - 2 = 42
191     OpenI2C1( I2C_ON, 0x02C );
192
193     // === setup system wide interrupts  ========
194     INTEnableSystemMultiVectoredInt();
195
196     // === TFT setup ============================
197     // init the display in main since more than one thread uses it.
198     // NOTE that this init assumes SPI channel 1 connections
199     tft_init_hw();
200     tft_begin();
201     tft_fillScreen(ILI9340_BLACK);
202     //240x320 vertical display
203     tft_setRotation(1); // Use tft_setRotation(1) for 320x240
204
205     // ===      config threads ======================
206     PT_setup();
207
208     //========== configure sensor ====================//
209     volatile int status = 1;
210     status = VL53L1X_SensorInit(); //Initialize Sensor
211     IdleI2C1();
212     if(status == 1){
213         printf("BAD INIT\n");
214     }
215     else if(status == 0){
```

```
216         printf("GOOD INIT\n");
217     }
218     /* ======= Set up Output Compare for Servo */
219     OpenOC3(OC_ON | OC_TIMER_MODE32 | OC_TIMER2_SRC | OC_PWM_FAULT_PIN_DISABLE, 0,0);
220     OpenOC4(OC_ON | OC_TIMER_MODE32 | OC_TIMER2_SRC | OC_PWM_FAULT_PIN_DISABLE, 0,0);
221     PPSOutput(4,RPA3,OC3); //configure OC33 to RPA33
222     PPSOutput(3,RPA2,OC4); //configure OC4 to RPA2
223 
224     //Initialize sensor to point outward
225     SetDCOC4PWM(80000);
226 
227     // === identify the threads to the scheduler =====
228     // add the thread function pointers to be scheduled
229     // --- Two parameters: function_name and rate. ---
230     // rate=0 fastest, rate=1 half, rate=2 quarter, rate=3 eighth, rate=4 sixteenth,
231     // rate=5 or greater DISABLE thread!
232 
233     pt_add(protothread_timer, 0);
234     pt_add(protothread_dataread, 0);
235     // === initalize the scheduler ====================
236     PT_INIT(&pt_sched) ;
237 
238     pt_sched_method = SCHED_ROUND_ROBIN;
239 
240     // === scheduler thread ======================
241     // scheduler never exits
242     PT_SCHEDULE(protothread_sched(&pt_sched));
243     // ============================================
244 
245 
246 } // main
247 // === end  ======================================================
248 
249 
```

PDF document made with CodePrint using Prism