```
1:
2: //=============================================================================
3: // Copyright (C) 2017, Active-Semi International
4: //
5: // THIS SOFTWARE IS SUBJECT TO A SOURCE CODE LICENSE AGREEMENT WHICH PROVIDES,
6: // AMONG OTHER THINGS:  (i) THAT IT CAN BE USED ONLY TO ADAPT THE LICENSEE'S
7: // APPLICATION TO PAC PROCESSORS SUPPLIED BY ACTIVE-SEMI INTERNATIONAL;
8: // (ii) THAT  IT IS PROVIDED "AS IS" WITHOUT WARRANTY;  (iii) THAT
9: // ACTIVE-SEMICONDUCTOR IS NOT LIABLE FOR ANY INDIRECT DAMAGES OR FOR DIRECT
10: // DAMAGES EXCEEDING US$1,500;  AND (iv) THAT IT CAN BE DISCLOSED TO AND USED
11: // ONLY BY CERTAIN AUTHORIZED PERSONS.
12: //
13: //=============================================================================
14:
15: #ifndef BLDC_APPLICATION_H
16: #define BLDC_APPLICATION_H
17:
18: #include "include.h"
19:
20: #define NUM_SPEED_SAMPLES                    6
20: // How many samples needed to accumulate for speed calculation
21: #define NUM_SPEED_SAMPLES_BLANKING           2
21: // Array length to hold motor speed used for automatic delay calculation
22:
23: typedef enum
24: {
25:     status_motor_enabled = 1,
26:     status_over_current = 2,
27:     status_closed_loop = 4,
28:     status_lose_angle = 8,
29:     status_motor_stall = 16,
30:     status_Vbus_overvoltage = 32,
31:     status_Vbus_undervoltage = 64,
32:     status_open_phase = 128,
33:     status_hall_error = 256,
34:     status_open_loop = 512,
35:     status_pwm_duty_loop = 1024,
36: } StatusStateBits;
37:
38:
39: typedef enum
40: {
41:     STATE_MAIN_STOP_STANDBY = 0,
42:     STATE_MAIN_OL,
43:     STATE_MAIN_OL_SWITCH_CL_WAIT,
44:     STATE_MAIN_CL_SPEED,
45:     STATE_MAIN_CL_CURRENT,
46:     STATE_MAIN_CL_PWM_DUTY,
47:     STATE_MAIN_SIM_DETECT,
48:     STATE_MAIN_STOP_BRAKING,
49:     STATE_MAIN_COASTING
50: }eBLDC_Main_Machine_State;
51:
52: typedef enum
53: {
54:     STATE_CONTROL_IDLE = 0,
55:     STATE_CONTROL_OL,
56:     STATE_CONTROL_OL_SWITCH_CL_WAIT,
57:     STATE_CONTROL_CL_SPEED,
58:     STATE_CONTROL_CL_CURRENT,
59:     STATE_CONTROL_CL_PWM_DUTY,
60:     STATE_CONTROL_SIM_DETECT,
61:     STATE_CONTROL_BRAKING_PWM_CONTROL,
62:     STATE_CONTROL_COASTING
63: }eSub_Control_Machine_State;
64:
65: typedef enum
66: {
67:     STATE_TIMERC_FIRST_SAMPLE = 0,
68:     STATE_TIMERC_GOOD_SAMPLES,
69:     STATE_TIMERC_ZERO_CROSS_POINT,
70:     STATE_TIMERC_WAIT_COMMUTATE,
71:     STATE_TIMERC_BLANKING_CYCLE_TIME
72: }eGet_Zero_Cross_Point_State;
73:
74: typedef enum
```

```
{
    STATE_HALL_STARTUP = 0,
    STATE_HALL_SWITCH_TO_BEMF
}eHall_Bemf_Switch_State;

typedef enum
{
    STATE_STARTUP_INIT = 0,
    STATE_STARTUP_ALIGN,
    STATE_STARTUP_ALIGN_DELAY,
    STATE_STARTUP_GO,
    STATE_STARTUP_SWITCH_TO_BEMF,
    STATE_STARTUP_DONE,
}eMotor_Align_Go_State;

typedef enum
{
    STATE_SIM_INIT = 0,
    STATE_SIM_DETECT,
    STATE_SIM_STOP,
    STATE_SIM_MOVE,
    STATE_SIM_SAME_DIRECTION,
    STATE_SIM_DIFF_DIRECTION,
    STATE_SIM_BRAKING,
}eSIM_State;









typedef struct
{
    eMotor_Align_Go_State align_and_go_state;
    uint16_t align_ticks;                           // Number of PWM ticks to keep in align mode
    uint16_t align_ticks_store;
    uint16_t align_ticks_temp;
// Temp variable to hold Number of PWM ticks in align mode
    uint16_t start_speed_ticks_period;
// Variable to hold Number of PWM ticks in go mode
    uint16_t start_speed_ticks_temp;
// Temp variable to hold number of PWM ticks to keep in go mode
    uint16_t align_go_duty_cycle;                   // Align and go duty cycle
    uint16_t align_go_duty_cycle_store;             // Align and go duty cycle
    uint16_t go_step;                               // Valid current go state for motor
    fix16_t  accel_rate_time_base;
// Number of 1 ms ticks in between speed changes
    fix16_t  start_speed_hz;
    fix16_t  current_speed_hz;
    fix16_t  switch_speed_hz;
    uint16_t switch_speed_ticks;
    uint16_t AccelRateTimeBaseTemp;
    fix16_t  accel_rate_factor;
    uint8_t  auto_acceleration_mode;

    uint16_t sine_wave_index;
    fix16_t  wave_pwm_duty_u;
    fix16_t  wave_pwm_duty_v;
    fix16_t  wave_pwm_duty_w;
} « end {anonMotor_Align_Go} » Motor_Align_Go;

typedef struct
{
    fix16_t Iq_ref;                                 // Iq_ref
    fix16_t Iq_ref_sp;                              // Iq_ref setpoint
    fix16_t Iq_ref_cl;                              // Iq_ref for close loop
    fix16_t Iq_fb;                                  // Iq_fb
    fix16_t Iq_previous;                            // Iq previous value use for digital filtering
    fix16_t Iq_prev_non_filtered;
// Iq previous value use for backup as good last non filtered value
    fix16_t Iq_filter_gain;
    PID_Data_Type Iq_pid;                           // PID controller for Iq
    PID_Data_Type speed_pid;
    fix16_t duty_percent_fix16;

```

```
147:     int16_t Ibus_ADC;                              // Stored Iu ADC value
148:     fix16_t Iq_ramp_rate;                          // Iq ref Ramp Rate
149:     uint8_t speed_pid_skips;                       // Number of times speed PID has been skipped
150:     uint8_t enable_current_control;                // Current control enable disable flag
151:     uint8_t num_of_good_zc_samples;
151: // Number of good zero cross samples, need to validate good zero crossing
152:     uint16_t blanking_sample_counts;
152: // Number of PWM cycles we blank out zero cross sensing after commutation
153:     fix16_t motor_close_loop_speed;
153: // Motor close loop speed in int format, 1HZ - 1 count
154:     fix16_t motor_close_loop_current;
155:     fix16_t motor_close_loop_speed_temp;
155: // Motor close loop speed temp variable use for speed ramp
156:     fix16_t motor_close_loop_speed_ramp;           // Motor close loop speed ramp rate variable
157:     fix16_t adc_zero_cross_offset_percentage;
157: // Used in ADC zero cross detection as an offset to move center tap point
158:     uint8_t auto_open_to_close_loop_sw_count;
158: // Use to check open loop speed number if cycles before switch over
158:                     /*!< Value comes from GUI as an percentage */
159:     fix16_t motor_auto_blank_percent;              // Blanking time comes in percentage from GUI
160:     fix16_t ratio_bw_pwmFreq_timerDFreq;
161:     fix16_t vbus_voltage_value;
162:
163:     uint8_t motor_enabled;
164:     uint8_t debug_buffer_command;
165:
166:     uint8_t control_mode_select_number;
167:     fix16_t closed_loop_current_command;
168:
169:     uint32_t coasting_timeout_counts;
170:     uint32_t coasting_timeout;
171:
172:
173:
174:
175:
176:     // configuration
177:     uint32_t app_status;
178:     uint32_t app_measured_speed;
179:     uint32_t app_speed_ref;
180:     uint32_t app_pwm_freq_khz;
181:     fix16_t  app_control_freq_khz;
182:     fix16_t  app_speed_command;
183:     uint32_t app_over_current_limit;
184:     uint32_t app_over_current_resistor;
185:     uint32_t app_amplifier_gain;
186:     uint32_t app_amplifier_gain_register;
187:
188:
189:     // Global Variables
190:     uint8_t  motor_enable_cmd;
191:     fix16_t  I_total_factor;
192:     uint8_t  oc_reset_flag;
193:     uint8_t  reverse_tune_flag;
194:     fix16_t  main_machine_control_counts;
195:     uint16_t motor_lose_angle_count;
196:
197:
198:     // PWM duty cycle
199:     fix16_t  align_pwm_duty_max;
200:     fix16_t  pwm_control_duty_cycle;
201:     uint16_t pwm_control_duty_change_delay_counts;
202:     uint16_t final_pwm_duty;
203:     uint16_t pwm_duty_Fall_buf;
204:     uint16_t temp_pwm_duty;
205:     uint16_t temp_pwm_duty_u;
206:     uint16_t temp_pwm_duty_v;
207:     uint16_t temp_pwm_duty_w;
208:     uint16_t pwm_timer_ticks;
208: // Number of timer A ticks for PWM period
209:     fix16_t  one_div_pwm_timer_ticks_fix16;
210:
211:
212:     uint8_t  sample;
213:     uint8_t  samplecounter;
214:     uint8_t  target_sample;
214: // Last good commutation state sample store
```

```
215:     uint16_t speed_sample_index;
216:     uint16_t speed_sample_index_blanking;
217:     uint8_t  wait_30degree_enable;
217: // Force to invalid value to force commutation on startup
218:     uint8_t  sl_current_state;
218: // Force to invalid value to force commutation on startup
219:     uint8_t  sl_next_state;
219: // Force to invalid value to force commutation on startup
220:     int32_t  commutation_advanced_rise;
221:     int32_t  commutation_advanced_fall;
222:
223:     // Array to store speed for one electrical cycle
224:     uint16_t call_speed_loop_delay_count;
225:     uint16_t motorspeed;
226:     uint32_t motorspeed_sum;
227:     fix16_t  motor_speed_sum_fix16;
228:     fix16_t  motorspeed_sum_blanking_fix16;
229:     uint32_t motorspeed_sum_blanking;
230:     uint16_t motorspeed_buffer[NUM_SPEED_SAMPLES];
231:     uint16_t motorspeed_buffer_blanking[NUM_SPEED_SAMPLES_BLANKING];
232:     uint16_t timer_d_base_timer_first;
233:     uint16_t timer_d_base_timer_second;
234:     uint16_t timer_d_base_timer_offset;
235:     uint16_t average_speed;
236:     volatile uint16_t commutation_blanking_time;
237:     int32_t  comm_advanced_delay_temp;
238:     uint16_t comm_advanced_delay;
239:
240:     uint8_t timerA_isr_counts;
241:
242:     uint16_t charge_delay_count;
243:     uint16_t charge_done_flag;
244:
245: } « end {anonBLDC_Controller} »  BLDC_Controller;
246:
247:
248: typedef struct
249: {
250:     eSIM_State sim_process_state;
251:
252:     fix16_t bemf_u ;
253:     fix16_t bemf_v ;
254:     fix16_t bemf_w ;
255:
256:     fix16_t bemf_u_init;
257:     fix16_t bemf_v_init;
258:     fix16_t bemf_w_init;
259:     fix16_t bemf_uvw_init_center_tap_voltage;
260:
261:     fix16_t delta_uv_bemf;
262:     fix16_t delta_uw_bemf;
263:     fix16_t delta_vw_bemf;
264:     fix16_t uvw_center_tap_voltage ;
265:     uint8_t step_state;
266:     uint8_t step_state_p;
267:     uint8_t bemf_state;
268:     uint8_t bemf_state_p;
269:     uint8_t direction_flag;
270:     uint8_t direction_flag_P;
271:
272:     fix16_t bemf_voltage;
273:     uint16_t input_pwm_duty_count;
274:     uint8_t motor_stop_bemf_flag;
275:     uint16_t detect_diff_position_counts;
276:
277:     uint16_t braking_vbus_voltage_ref;
278:     uint16_t braking_vbus_voltage_fdb;
279:     uint16_t braking_depth_pwm_duty_counts;
280:     uint16_t braking_bemf_u;
281:     uint16_t braking_bemf_v;
282:     uint16_t braking_bemf_w;
283:     uint16_t braking_bemf_uvw_center_tap_voltage;
284:     uint8_t  braking_detect_motor_stop_flag;
285:     uint8_t  braking_stop_counts;
286:     uint8_t  braking_restart_disable_driver_counts;
287:     uint8_t  motor_in_motion_flag;
288:     uint32_t in_braking_state_counts;
289:
```

```
290:     fix16_t  bemf_ratio;
291:     fix16_t  bemf_adc_scal_fator;
292:     uint16_t each_phs_bemf_delta_threshold;
293:     uint8_t  keep_detect_step_times;
294:     fix16_t  ratio_timerc_timerd_freq;
295: } « end {anonSim_Detect} »  Sim_Detect;
296:
297:
298: typedef struct
299: {
300:     uint8_t  hall_value;
301:     uint8_t  sensored_current_state;
302:     uint8_t  sensored_current_state_pre;
303:     uint16_t hall_bemf_switch_speed;
304:     uint32_t hall_to_mos_state_detect;
305:     uint8_t  commutate_flag;
306: } Hall_Sensor;
307:
308: typedef struct
309: {
310:     fix16_t app_measured_speed_fix16;
311:     fix16_t app_speed_ref_fix16;
312:     fix16_t ibus_adc_fix16;
313:
314:     fix16_t sl_current_hall_fix16;
315:     fix16_t sl_current_state_fix16;
316:     uint8_t sl_current_step_position;
317:     uint8_t sl_current_step_position_pre;
318:
319:     fix16_t aio7_adc_value_fix16;
320:     fix16_t aio8_adc_value_fix16;
321:     fix16_t aio9_adc_value_fix16;
322:     fix16_t centre_adc_value_fix16;
323:     uint8_t phase_comparator_output;
324:
325: } Pwm_Dac_Debug;
326:
327:
328: // structural and enumerate body
329: extern Motor_Align_Go               bldc_align_go;
330: extern BLDC_Controller              bldc_m1;
331: extern eBLDC_Main_Machine_State     main_machine_state;
332: extern eSub_Control_Machine_State   control_machine_state;
333: extern eGet_Zero_Cross_Point_State get_zero_cross_point_state;
334: extern eHall_Bemf_Switch_State      hall_bemf_switch_state;
335:
336:
337: //three phase sine wave table for motor start-up
338: extern const fix16_t sine_wave_3phase[60][3];
339: // Commutation state table with Analog MAUX setting
340: extern const uint8_t slcomp_mux[7] ;
341: extern const uint8_t slcomp_next_state[7] ;
342: extern const uint8_t slcomp_last_state[7];
343: extern const uint8_t slcomp_cross_polarity[7] ;                //POLx = 0,active High
344: extern const uint8_t slcomp_next_state_rev[7] ;                //reverse
345: extern const uint8_t slcomp_last_state_rev[7];                     //reverse
346: extern const uint8_t slcomp_cross_polarity_rev[7] ;            //reverse
347:
348:
349:
350:
351:
352:
353:
354:
355:
356: #define ENABLE                          (1)
357: #define DISABLE                         (0)
358:
359: #define DELAY_CHARGE_COUNTS               5
360: #define DELAY_SWITCH_TO_SPEED_LOOP_COUNTS  12
361: #define OVER_CURRENT_FLAG                 0x02
362:
363: //  PWM DAC
364: #define PWMDAC_PWM_PIN_PORT_A_DEBUG     0x38
364: // Pin mask for LS gate drive PWM for Port A
365:
```

```
366:
367: // Definitions
368: #define TIMERA_PERIOD_TICKS             781             // Number of timer a ticks for PWM
369: #define TIMERB_TICKS_DEBUG_DAC_45KHZ    1111
370: #define TIMERB_PERIOD_TICKS_PPM         65535
371: #define TIMERC_PERIOD_TICKS             500
371: // Number of timer c ticks for 100Khz timer (up timer, HCLK@50MHz, /1)
372: #define TIMERD_TICKS_30DEG_PLL              65535
372: // Number of timer ticks for 30 degree timer (max 335ms per 60 degrees)
373:
374: #define FIX16_1DOT_024                  0x10624
375: #define RATIO_BW_TIMERC_FREQ_TIMERD_FREQ   16777
375: // TimerC 100Khz:  0x4189 = 65536*(100Kz/(50M/128))  TimerC 160Khz:  26843 = 65536*(160Kz/(50M/128)
376: #define TIMER_D_FREQ_F16                    (0x0186A000)
376: // TMRD Freq = 50MHz / 128 = 195,312.5 Hz (This number is divided by 1000 so it can be represented
377:
378:
379: #define SPEED_RAMP_COUNTS               5
380: #define CURRENT_RAMP_COUNTS             10
381:
382: #define LED_PIN_MASK                        0x01                    // Pin mask for LED output PE0
383: #define LED_PIN_NUM                     0
383: // Pin number for LED output (active high) PE0
384: #define LED1_TOGGLE                     (pac5xxx_gpio_out_toggle_e(LED_PIN_MASK))
384: // Toggle LED state
385: #define LED1_ON                         (pac5xxx_gpio_out_pin_set_e(LED_PIN_NUM))
385: // Set LED to on
386: #define LED1_OFF                            (pac5xxx_gpio_out_pin_clear_e(LED_PIN_NUM))
386: // Set LED to off
387:
388:
389: #define NIRQ1_PIN_MASK                      0x01
390: #define NIRQ2_PIN_MASK                      0x80
391: #define SLCOMP7                         0x10
392: #define SLCOMP8                         0x20
393: #define SLCOMP9                         0x30
394:
395:
396: #define AIO7_LOW2HIGH                       0
397: #define AIO7_HIGH2LOW                       1
398: #define AIO8_LOW2HIGH                       0
399: #define AIO8_HIGH2LOW                       1
400: #define AIO9_LOW2HIGH                       0
401: #define AIO9_HIGH2LOW                       1
402:
403:
404: #define HALF_DEGREE_ADV_DLY             (0x00000222)
404: // 0.5 degree advance delay factor = 1/2 Comm Adv Delay * 1/2 degree * 1/30 degrees
405:
406:
407:
408: #define COMMUTATION_BLANKING_COUNT          10
409:
410: #define IQ_RAMP_RATE                         0x65536                // 0.01 A, convert into fix16
411: #define IQ_REFERENCE                        0x00008000             // 0.5 A, convert to fix16
412: #define TD_FOR_IQ_PID                       0x000020C5
412: // 8KHZ = 125 usec * 1024 = 0.128* 65536, converted into fix16 = 0x000020CA
413:
414: #define TRUE   (1)
415: #define FALSE  (0)
416:
417: #define DEFAULT_GOOD_ZERO_CROSS_SAMPLES            3
418: #define DEFAULT_CLOSE_LOOP_SPEED              50
419:
420: #define OPEN_TO_CLOSE_LOOP_GOOD_SPEED_COUNT       1
421: #define SPEED_RAMP_RATE_1HZ_SEC                   0x00001999
422:
423: #define DEFAULT_AUTO_BLANKING_PERCENT             0x00008000        // 50% of commutation time
424:
425: #define MATH_ZERO_DOT_ONE_FIX16                    0x1999
426: #define MATH_ONE_DIV_SIX_FIX16                     0x2AAA
427: #define MATH_ONE_DIV_THREE_FIX16                0x5555
428:
429: #define SINE_WAVE_ARRAY_INDEX_MAX                  59
430: #define SINE_WAVE_ARRAY_INDEX_MIN                 0
431:
432: #define COASTING_DUTY_REDUCE_RATE                  5
433: #endif
```