```
/*****************************************************

    PA10 Control Program ( 1-Axis Only )

*****************************************************/


#include "include.h"
#include "main_cfg.h"
#include "cfg_arm.h"
#include "params_arm.h"
#include "./Arcnet/arc_pci.h"
#include "timer.h"

struct params motorp;
int brakeoff_joint;
int ctrlEndFlag = 0;
int ctrltrig = OFF;

const double inertia = INERTIA1;
const double Kd = KD1;
const double Kp = KP1;
const double joint_limit[2] = {175.*DEG2RAD, 85.*DEG2RAD};
const double max_torque    =  MAX_TORQUE1 ;
struct path    path_j;
struct status  cur_j, des_j;
double torque;

void init(void);
int initializeAll(void);
void initializeData(void);
void start(void);
int ctrlTask(struct params *motor);
void fin(void);
void jointCtrl(struct params *param, int trig);
void getCurrentPosition(struct status *Cur_j);
void pathInit_j(double *Start, double *Destination, double Time);
void pathGenerate_j(struct status *Des_j, unsigned long Time);
void pdCtrl(struct status *Des_j, struct status *Cur_j);
void allzero(void);
void allboff(void);
void allbon(void);
void boff1(void);
void boff2(void);
void boff3(void);
void boff4(void);
void boff5(void);
void boff6(void);
void boff7(void);
void control(struct params *motor);
void allbrakeoff(void);
void brakeoff(int joint);
void Nop(void);
int endTask(void);
void joint_moveto(double Angle,double Time);
void All_OFFBrake(void);
void OFFBrake(void);
void ONBrake(void);

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
extern thread_pool_t * cp (int argc, char **argv);

/*---------------------------------------------------------------------------*/

void
init(void)
```

```
{
    ThreadCtl(_NTO_TCTL_IO,0);

    if ((chid = ChannelCreate (0)) == -1){
        fprintf (stderr,"timer.c: couldn't create channel!\n");
        perror (NULL);
        exit (EXIT_FAILURE);
    }

    setupTimer();

    initializeAll();

}


int
initializeAll(void)
{
    char key[16];


    printf("\n\t***** Choose the Control Mode *****\n\n");
    printf("\tOperation or Simulation ? [o/s] \n\n");
    fgets(key, sizeof(key), stdin);

    if (*key == 'o') {
        CtrlMode = operation;
        arcInit();
        printf("\n\tControl Mode -> Operation \n\n");
    }
    else if (*key == 's') {
        CtrlMode = simulation;
        printf("\n\tControl Mode -> Simulation \n\n");
    }

    else {
        CtrlMode = simulation;
        printf("\n\tControl Mode -> Simulation \n\n");
    }


    initializeData();

    return OK;
}

void
initializeData(void)
{

    motorp.mode = nop;
    motorp.desPos = 0.;
    motorp.desTime = 0.;
    memset(&path_j, 0, sizeof(path_j));

}

void
start(void)
{
    arcnet_start();



    timer_settime(timerid, 0, &timer, NULL);
```

```
133     ctrltrig = 1;
134     ctrlEndFlag = 1;
135 
136     ctrlTask(&motorp);
137 
138 }
139 
140 int
141 ctrlTask(struct params *motor)
142 {
143     int rcvid=1;
144     MessageT msg;
145     static unsigned long ticks = 0;
146     pthread_attr_t attr;
147 
148     pthread_attr_init(&attr);
149     pthread_attr_setdetachstate(&attr,PTHREAD_CREATE_DETACHED );
150 
151     while(ctrltrig){
152         rcvid = MsgReceive(chid, &msg, sizeof(msg), NULL);
153         if(rcvid == 0){
154 
155 
156             if(!ctrlEndFlag){
157                 printf("ERROR : Control Task is out of time.\n        fin\n");
158                 fin();
159                 return (ERROR);
160             }
161             ctrlEndFlag = 0;
162             pthread_create(NULL,&attr,(void *)control,motor);
163             ticks++;
164 
165         }
166     }
167 
168     return (EXIT_SUCCESS);
169 
170 }
171 
172 void
173 control(struct params *motor)
174 {
175 
176     int stat1;
177     static struct params param;
178     static int mode = nop;
179     static int joint;
180     static int trig;
181 
182 
183     if( pthread_mutex_lock ( &mutex ) != EOK ){
184         printf(" ERROR : mutex cannnot lock.\n        fin\n");
185         fin();
186         return;
187     }
188 
189     switch(CtrlMode){
190     case operation:
191         stat1 = iSend_C();
192         break;
193     case simulation:
194         break;
195 
196     default:
197         break;
198     }
```

```
199
200     memcpy(&param, motor, sizeof(struct params));
201
202     getCurrentPosition(&cur_j);
203
204     if(param.trig){
205         trig = TRUE;
206         motor->trig = FALSE;
207     }
208     else
209         trig = FALSE;
210
211     mode = param.mode;
212     joint = param.joint;
213
214     switch(mode){
215
216     case joint_mode:
217         jointCtrl(&param, trig);
218         break;
219
220     case allbrakeoff_mode:
221         allbrakeoff();
222         break;
223     case brakeoff_mode:
224         brakeoff(joint);
225         break;
226     default:
227         Nop();
228
229         break;
230     }
231
232     switch(CtrlMode){
233     case operation:
234         stat1 = RecData();
235         break;
236     case simulation:
237         break;
238
239     default:
240         break;
241     }
242
243     endTask();
244     return;
245
246 }
247
248 void
249 jointCtrl(struct params *param, int trig)
250 {
251
252     static unsigned long tick;
253     static double desPos,desTime;
254
255
256     if(trig){
257
258         desPos = param->desPos;
259     }
260
261     desTime = param->desTime;
262     tick = 0;
263
264     pathInit_j(des_j.pos, desPos, desTime);
```

```
265
266     return;
267 }
268
269 pathGenerate_j(&des_j, tick);
270
271 tick++;
272 pdCtrl(&des_j, &cur_j);
273
274
275 }
276
277 void
278 getCurrentPosition(struct status *Cur_j)
279 {
280
281 double angle = 0.;
282 static double pre_j_pos = 0.;
283 static double pre_j_vel = 0.;
284
285 /*--- get joint angles ---*/
286
287 GetPosition(angle);
288
289 Cur_j->pos = angle;
290
291 Cur_j->vel = (Cur_j->pos - pre_j_pos) / TICKS;
292 Cur_j->acc = (Cur_j->vel - pre_j_vel) / TICKS;
293
294 pre_j_pos = Cur_j->pos;
295 pre_j_vel = Cur_j->vel;
296 }
297
298 /*--- Selection of control mode ---*/
299
300 switch(CtrlMode){
301 case operation:
302 break;
303 case simulation:
304
305
306 break;
307
308 default:
309 break;
310 }
311 }
312
313 void
314 pathInit_j(double *Start, double *Destination, double Time)
315 {
316 int jnt;
317 double t3, t4, t5;
318 double diff;
319
320 t3 = Time * Time * Time;
321 t4 = t3 * Time;
322 t5 = t4 * Time;
323
324
325 diff = Destination - Start;
326
327 path_j.pos[0] = Start;
328 path_j.pos[3] =  10. * diff / t3;
329 path_j.pos[4] = -15. * diff / t4;
330 path_j.pos[5] =   6. * diff / t5;
```

```
331
332  path_j.vel[2] = 3. * path_j.pos[3];
333  path_j.vel[3] = 4. * path_j.pos[4];
334  path_j.vel[4] = 5. * path_j.pos[5];
335
336  path_j.acc[1] =  6. * path_j.pos[3];
337  path_j.acc[2] = 12. * path_j.pos[4];
338  path_j.acc[3] = 20. * path_j.pos[5];
339  }
340  path_j.time = Time;
341  }
342
343  void
344  pathGenerate_j(struct status *Des_j, unsigned long Time)
345  {
346
347  double t;
348
349
350  t = (Time * TICKS);
351  t = (t > path_j.time) ? path_j.time : t;
352
353
354  Des_j->pos = path_j.pos[0]
355  + t * t * t * (path_j.pos[3]
356  + t * (path_j.pos[4] + t * path_j.pos[5]));
357
358  Des_j->vel = t * t * (path_j.vel[2]
359  + t * (path_j.vel[3] + t * path_j.vel[4]));
360
361  Des_j->acc = t * (path_j.acc[1] + t * (path_j.acc[2]
362  + t * path_j.acc[3]));
363  }
364  }
365
366  void
367  pdCtrl(struct status *Des_j, struct status *Cur_j)
368  {
369
370  double accel;
371
372  accel = Des_j->acc
373  + Kd * (Des_j->vel - Cur_j->vel)
374      + Kp * (Des_j->pos - Cur_j->pos);
375
376      torque = inertia * accel;
377
378      if(torque > max_torque)
379      torque = max_torque;
380      else if(torque < - max_torque)
381      torque = - max_torque;
382      }
383
384  switch(CtrlMode){
385  case operation:
386  SetTorq(torque);
387  break;
388  case simulation:
389  NoTorq();
390  break;
391
392  default:
393  NoTorq();
394  break;
395  }
396  }
```

```
397
398 int
399 endTask( void )
400 {
401 ctrlEndFlag = 1;
402 pthread_mutex_unlock ( &mutex );
403
404 return (EXIT_SUCCESS);
405
406 }
407
408 void
409 allbrakeoff(void)
410 {
411     AllBrakeOFF();
412 }
413
414 void
415 brakeoff(int joint)
416 {
417     BrakeOFF(joint);
418
419 }
420
421 void
422 Nop(void)
423 {
424     NoTorq();
425 }
426
427 void
428 joint_moveto(double Angle, double Time)
429 {
430     motorp.desPos = Angle;
431     motorp.desTime = Time;
432     motorp.trig    = TRUE;
433     motorp.mode    = joint_mode;
434 }
435
436 void
437 All_OFFBrake(void)
438 {
439
440     motorp.mode = allbrakeoff_mode;
441 }
442
443 OFFBrake(void)
444 {
445
446     motorp.mode = brakeoff_mode;
447     motorp.joint = brakeoff_joint;
448 }
449
450 void
451 ONBrake(void)
452 {
453     motorp.mode = nop;
454 }
455
456 void
457 fin(void)
458 {
459     ctrltrig = 0;
460     ONBrake();
461     arcFin();
462     timer_delete(timerid);
```

```
}

int
main(void)
{

    cp(NULL, NULL);
    return(OK);

}
```