```
/*


################################################
COMANDI DA PIC A DSPIC

RD5             Chip Select DSPIC A
RD4             Chip Select DSPIC B


RD3-RD2-RD1-RD0
0   0   0   0             Nessuna operazione

RD3-RD2-RD1-RD0
Con gestione CS azione singolo asse
0   0   0   1             Avvia posizionamento in start stop
0   0   1   0             Avvia posizionamento singolo con rampa
0   0   1   1             Avvia posizionamento multiplo con rampa
0   1   0   0             Muovi in meno
0   1   0   1             Muovi in più
0   1   1   0             Azzeramento
0   1   1   1             ----------

RD3-RD2-RD1-RD0
senza gestione CS azione simultanea entrambi gli assi
1   0   0   0             Avvia posizionamento in start stop
1   0   0   1             Avvia posizionamento singolo con rampa
1   0   1   0             Avvia posizionamento multiplo con rampa
1   0   1   1             Azzeramento
1   1   0   0             ----------
1   1   1   0             ----------
1   1   1   1             ----------


************************************************
PORTE
RE3     -   INPUT     IN_2
RE2     -   INPUT     IN_3
RE1     -   INPUT     IN_4
RE0     -   INPUT     IN_1

PORTD
RD7     -   INPUT      ERRORE MOT B
RD6     -   INPUT      BUSY   MOT B
RD5     -   OUTPUT     Chip Select DSPIC A
RD4     -   OUTPUT     Chip Select DSPIC B
RD3     -   OUTPUT     Comando  a DSPIC COM3
RD2     -   OUTPUT     Comando  a DSPIC COM2
RD1     -   OUTPUT     Comando  a DSPIC COM1
RD0     -   OUTPUT     Comando  a DSPIC COM0

PORTC
RC7     -    RS232     RX
RC6     -    RS232     TX
RC5     -    INPUT     EMERGENZA
RC4     -    I2C       SDA
RC3     -    I2C       SCL
RC2     -    OUTPUT    SERVO PWM
RC1     -    OUTPUT    OUT2
RC0     -    OUTPUT    OUT1

PORTB
```

```
63: RB7     -    INPUT      ERRORE MOT A
64: RB6     -    INPUT      BUSY   MOT A
65: RB5     -    LCD        D7
66: RB4     -    LCD        D6
67: RB3     -    LCD        D5
68: RB2     -    LCD        D4
69: RB1     -    LCD        EN
70: RB0     -    LCD        RS
71:
72: PORTA
73: RA7     -    INPUT      IN_5
74: RA6     -    INPUT      IN_6
75: RA5     -    INPUT      PULSANTE STOP
76: RA4     -    INPUT      PULSANTE START
77: RA3     -    INPUT      PULSANTE INVIO
78: RA2     -    INPUT      PULSANTE ESC
79: RA1     -    INPUT      PULSANTE GIU
80: RA0     -    INPUT      PULSANTE SU
81:
82: CODICI ERRORE
83: -1      -    Calcolo errato, posizione impossibile
84: 1       -    Mancata risposta comunicazione seriale
85: 2       -    Errore carattere inviato comunicazione seriale
86: 3       -    Mancanza carattere di chiusura trasmissione comunicazione seriale
87: 4       -    Asse 1 occupato
88: 5       -    Asse 2 occupato
89: 6       -    Asse 1 in errore
90: 7       -    Asse 2 in errore
91: 8       -    Asse 1 errore azzeramento
92: 9       -    Asse 2 errore azzeramento
93: 10      -    Asse 1 comando non recepito
94: 11      -    Asse 2 comando non recepito
95: 12      -    Comando non recepito
96: 13      -    Errore azzeramento assi ricerca zero
97: 14      -    Errore azzeramento assi non azzerati
98: 15      -    Errore velocità A
99: 16      -    Errore velocità B
100: 18     -    Errore azzeramento asse 1 non azzerat0
101: 19     -    Errore azzeramento asse 2 non azzerato
102: 20     -    Errore passi impostati A errati
103: 21     -    Errore passi impostati B errati
104: 22     -    Posizionamento non eseguito
105: 23     -    Posizionamento errato
106: 24     -    Errore delay
107:
108: */
109:
110: #include <built_in.h>
111:
112: // DEFINIZIONE BIT LCD
113: sbit LCD_RS at RB0_bit;
114: sbit LCD_EN at RB1_bit;
115: sbit LCD_D4 at RB2_bit;
116: sbit LCD_D5 at RB3_bit;
117: sbit LCD_D6 at RB4_bit;
118: sbit LCD_D7 at RB5_bit;
119:
120: sbit LCD_RS_Direction at TRISB0_bit;
121: sbit LCD_EN_Direction at TRISB1_bit;
122: sbit LCD_D4_Direction at TRISB2_bit;
123: sbit LCD_D5_Direction at TRISB3_bit;
124: sbit LCD_D6_Direction at TRISB4_bit;
```

```
125: sbit LCD_D7_Direction at TRISB5_bit;
126: //softi2cinit
127: sbit Soft_I2C_Scl           at RC3_bit;
128: sbit Soft_I2C_Sda           at RC4_bit;
129: sbit Soft_I2C_Scl_Direction at TRISC3_bit;
130: sbit Soft_I2C_Sda_Direction at TRISC4_bit;
131:
132: //DEFINIZIONE BIT INPUT
133: sbit PULS_UP                at RA0_bit;
134: sbit PULS_DOWN              at RA1_bit;
135: sbit PULS_ESC               at RA2_bit;
136: sbit PULS_INVIO             at RA3_bit;
137: sbit PULS_START1            at RA4_bit;
138: sbit PULS_START2            at RA5_bit;
139: sbit ERR_MOT_1              at RB7_bit;
140: sbit BUSY_MOT_1             at RB6_bit;
141: sbit ERR_MOT_2              at RD7_bit;
142: sbit BUSY_MOT_2             at RD6_bit;
143: sbit EMERGENZA              at RC5_bit;
144: sbit ZERO_A                 at RE3_bit;
145: sbit IN_1                   at RE0_bit;
146: sbit IN_2                   at RE1_bit;
147: sbit ZERO_B                 at RE2_bit;
148: sbit IN_3                   at RA7_bit;
149: sbit POS_0                  at RA6_bit;
150:
151: //DEFINIZIONE BIT OUTPUT
152: sbit MAGNETE                at RC0_bit;
153: sbit PISTONE                at RC1_bit;
154: sbit SERVO                  at RC2_bit;
155: sbit COM0_DSPIC             at RD0_bit;
156: sbit COM1_DSPIC             at RD1_bit;
157: sbit COM2_DSPIC             at RD2_bit;
158: sbit COM3_DSPIC             at RD3_bit;
159: sbit CS_DSPIC_MOT_2         at RD4_bit;
160: sbit CS_DSPIC_MOT_1         at RD5_bit;
161:
162: char stato_asse_1;
163: sbit   ZERO_OK_1      at stato_asse_1.B0;
164:
165: char stato_asse_2;
166: sbit   ZERO_OK_2      at stato_asse_2.B0;
167:
168:
169:
170: //valore di PI greco
171: const float PI = 3.14159265358979323846;
172: //lunghezza dei bracci
173: const float Br1=600.0;
174: const float Br2=900.0;
175: //distanza assi di rotazione simmetrica al centro
176: const float Dist=600.0;
177: //coordinate cartesiane spigolo alto a sx
178: const float Xini=900.0;
179: const float Yini=150.0;
180: //posizone X e Y all'interno della casella
181: //partendo dallo spigolo in alto a sx
182: const float offset_X=50.0;
183: const float offset_Y=30.0;
184: //dimensioni spazio casella tavola periodica
185: const float larghezza=100.0;
186: const float altezza=120.0;
```

```
187: //distanza in Y tra le due parti della tavola periodica
188: const float Dist_y=0.0;
189: //angolo impulso motore
190: const float risoluzione=0.018;
191: //coordinate ed angoli formula Kinematics
192: double Xpunto,Ypunto,alfa1,alfa2;
193: float L1,L2,d1,d2,beta1,beta2,gamma1,gamma2;
194: //impulsi assoluti dei motori
195: long imp_alfa1,imp_alfa2;
196: //casella tavola periodica
197: unsigned short num_elemento,num_casella;
198: unsigned short num_sequenza;
199: unsigned char riga,colonna;
200: char esito,esito_menu;
201: //array degli angoli
202: //colonna_1=alfa1 (motore in X-)
203: //colonna_2=alfa2 (motore in X+)
204: float angoli[2][118];
205: int impulsi[2][118];
206:
207: //array elementi
208: char elementi[118]={1,18,19,20,31,32,33,34,35,36,37,38,49,50,51,52,53,54,55,56,
209: 57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,
210: 83,84,85,86,87,88,89,90,91,92,93,129,130,131,132,133,134,135,136,137,138,139,
211: 140,141,142,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,
212: 147,148,149,150,151,152,153,154,155,156,157,158,159,160,112,113,114,115,116,117,
213: 118,119,120,121,122,123,124,125,126};
214:
215: //array ordinato secondo la sequenza di posizionamento
216: char ordine_elementi[118]={1,6,82,50,26,29,80,47,79,7,8,25,3,5,11,12,
217: 19,20,38,56,53,4,9,13,14,15,16,17,22,23,28,30,33,34,35,37,39,40,41,42,44,
218: 45,46,48,49,51,52,55,57,58,65,68,73,74,76,77,78,81,83,90,92,63,2,10,18,21,
219: 24,27,31,32,36,43,54,59,60,61,62,64,66,67,69,70,71,72,75,84,85,86,87,88,89,
220: 91,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,
221: 112,113,114,115,116,117,118};
222: //gestione movimenti
223: /*
224: //array ordinato secondo la sequenza di posizionamento
225: char ordine_elementi[118]={1,32,104,86,62,65,102,83,101,63,33,34,61,19,31,37,38,
226: 55,56,74,92,89,20,35,49,50,51,52,53,58,59,60,64,66,69,70,71,73,75,76,77,78,80,
227: 81,82,84,85,87,88,91,93,129,136,139,95,96,98,99,100,103,105,147,149,134,18,36,54
     57,
228: 67,68,72,79,90,130,131,132,133,135,137,138,140,141,142,94,97,106,107,108,109,110
     111,
229: 148,150,151,152,153,154,155,156,157,158,159,160,112,113,114,115,116,117,118,119,
230: 120,121,122,123,124,125,126};
231: */
232:
233: //gestione movimenti
234: char robot_azzerato;
235: //gestione display LCD
236: char *riga_vuota="                ";
237: char *txt_montani="ITT MONTANI";
238: char *txt_manuale="MANUALE";
239: char *txt_azzera="AZZERA";
240: char *txt_errori="ERRORI";
241: char *txt_reset="RESET ALLARMI";
242: char *txt_offset="OFFSET ZERO";
243: char *txt_ciclo_rapido="CICLO RAPIDO";
244: char *txt_ciclo_lento="CICLO LENTO";
245: char *txt_stato_assi="STATO ASSI";
246: char *txt_valori="VALORI";
```

```
247: char *txt_emergenza="EMERGENZA";
248: char *txt_stato="Stato=   ";
249: char *txt_nozero="NO ZERO";
250: char *txt_okzero="OK ZERO";
251: char *txt_errore="ERRORE";
252: char *txt_motori="MOTORI";
253: char *txt_motore_A="MOTORE A";
254: char *txt_motore_B="MOTORE B";
255: char *txt_pistone="PISTONE";
256: char *txt_magnete="MAGNETE";
257: char *txt_riposo="POS.RIPOSO";
258: char *txt_90="POS_DIRITTO";
259: char *txt_8vuoti="        ";
260: char cont_rimbalzo=30;
261: char txt_byte[4];
262: char txt_int[7];
263: char txt_float[15];
264: char menu_ini_scelta,menu_man_scelta;
265: //GESTIONE SERIALE
266: char dato_rx[5];
267: char dato_tx[5];
268: int pos_attuale_A,pos_attuale_B;
269: int pos_richiesta_A,pos_richiesta_B;
270: char vel,vel_A,vel_B,vel_ini,num_asse,tipo_ciclo;
271: char codice_err_1,codice_err_2;
272: int offset_asse_1,offset_asse_2;
273: int del_A,del_B,delay_base;
274: //variabili generiche
275: int temp_int;
276: float temp_real;
277: float rapporto,temp_1,temp_2;
278: char motor_sel; //indica il motore selezionato
279: char debug;
280:
281: //#########################################################
282: //*********************************************************
283: //CONTROLLO BUSY MOT 2
284: //SE ATTIVO PER 100mS TORNA L'ERRORE
285: //*********************************************************
286: char CONTROLLO_BUSY_MOT_1()
287: {
288:   int controllo_busy;
289:
290:   controllo_busy=30000;
291:   while(BUSY_MOT_1)
292:    {
293:      delay_us(20);
294:      controllo_busy--;
295:      if(!controllo_busy) return(1);
296:    }//while(BUSY_MOT_1)
297:   return(0);
298: }//CONTROLLO_BUSY_MOT_1()
299:
300:
301: //*********************************************************
302: //CONTROLLO BUSY MOT 2
303: //SE ATTIVO PER 100mS TORNA L'ERRORE
304: //*********************************************************
305: char CONTROLLO_BUSY_MOT_2()
306: {
307:   int controllo_busy;
308:
```

```
309:   controllo_busy=30000;
310:   while(BUSY_MOT_2)
311:    {
312:      delay_us(20);
313:      controllo_busy--;
314:      if(!controllo_busy) return(1);
315:    }//while(BUSY_MOT_2)
316:   return(0);
317: }//CONTROLLO_BUSY_MOT_2()
318: //*******************************************************
319:
320:
321: //*******************************************************
322: //CONTROLLO ERR MOT 1
323: //SE ATTIVO PER 100mS TORNA L'ERRORE
324: //*******************************************************
325: char CONTROLLO_ERR_MOT_1()
326: {
327:   int controllo_err;
328:
329:   controllo_err=30000;
330:   while(ERR_MOT_1)
331:    {
332:      delay_us(20);
333:      controllo_err--;
334:      if(!controllo_err) return(1);
335:    }//while(ERR_MOT_1)
336:   return(0);
337: }//CONTROLLO_ERR_MOT_1()
338: //*******************************************************
339:
340:
341: //*******************************************************
342: //CONTROLLO ERR MOT 2
343: //SE ATTIVO PER 100mS TORNA L'ERRORE
344: //*******************************************************
345: char CONTROLLO_ERR_MOT_2()
346: {
347:   int controllo_err;
348:
349:   controllo_err=30000;
350:   while(ERR_MOT_2)
351:    {
352:      delay_us(20);
353:      controllo_err--;
354:      if(!controllo_err) return(1);
355:    }//while(ERR_MOT_2)
356:   return(0);
357: }//CONTROLLO_ERR_MOT_2()
358: //*******************************************************
359:
360: //#######################################################
361: //*******************************************************
362: //IMPOSTAZIONE COMANDI A DSPIC
363: //*******************************************************
364: //*******************************************************
365: //NO OPERATION
366: //*******************************************************
367: void no_comando()
368: {
369:   char temp_port;
370:
```

```
371:   temp_port=PORTD&0xC0;
372:   PORTD=temp_port;
373: }//no_comando()
374:
375: //*******************************************************
376: //SELEZIONA MOTORE 1
377: //*******************************************************
378: void SEL_MOT_1()
379: {
380:   CS_DSPIC_MOT_1=1;
381:   CS_DSPIC_MOT_2=0;
382:   motor_sel=1;
383:   delay_ms(1);
384: }//void SEL_MOT_1()
385:
386: //*******************************************************
387: //SELEZIONA MOTORE 2
388: //*******************************************************
389: void SEL_MOT_2()
390: {
391:   CS_DSPIC_MOT_1=0;
392:   CS_DSPIC_MOT_2=1;
393:   motor_sel=2;
394:   delay_ms(1);
395: }//void SEL_MOT_2()
396:
397: //*******************************************************
398: //NESSUN MOTORE SELEZIONATO
399: //*******************************************************
400: void NO_SEL()
401: {
402:   CS_DSPIC_MOT_1=0;
403:   CS_DSPIC_MOT_2=0;
404:   motor_sel=0;
405:   delay_ms(1);
406: }//NO_SEL())
407:
408: //*******************************************************
409: //COMANDI SINGOLO ASSE CON CHIP SELECT
410: //*******************************************************
411: //*******************************************************
412: //POSIZIONA MOTORE 1  IN START STOP
413: //RITORNA 0 SE OK
414: //*******************************************************
415: char POSIZIONA_SS_MOT_1()
416: {
417:   char temp_port1;
418:   char cont_attesa1;
419:   char cont_busy,cont_err;
420:
421:   if(CONTROLLO_BUSY_MOT_1()) return(4);
422:   if(CONTROLLO_ERR_MOT_1())  return(6);
423:   cont_attesa1=1000;
424:   temp_port1=PORTD&0xC0;
425:   temp_port1=temp_port1|0x21;
426:   PORTD=temp_port1;
427:   //attesa ricezione comando
428:   while(cont_attesa1)
429:     {
430:       delay_ms(1);
431:       cont_attesa1--;
432:       if(!cont_attesa1) {no_comando();return(10);}
```

```
433:       if(BUSY_MOT_1) cont_attesa1=0;
434:     }//while
435:    no_comando();
436:    return(0);
437: }//POSIZIONA_SS_MOT_1()
438:
439: //*****************************************************
440: //POSIZIONA MOTORE 2  IN START STOP
441: //RITORNA 0 SE OK
442: //*****************************************************
443: char POSIZIONA_SS_MOT_2()
444: {
445:   char temp_port2;
446:   char cont_attesa2;
447:
448:   if(CONTROLLO_BUSY_MOT_2()) return(5);
449:   if(CONTROLLO_ERR_MOT_2())  return(7);
450:   cont_attesa2=1000;
451:   temp_port2=PORTD&0xC0;
452:   temp_port2=temp_port2|0x11;
453:   PORTD=temp_port2;
454:   //attesa ricezione comando
455:   while(cont_attesa2)
456:     {
457:       delay_ms(1);
458:       cont_attesa2--;
459:       if(!cont_attesa2) {no_comando();return(11);}
460:       if(BUSY_MOT_2) cont_attesa2=0;
461:     }//while
462:    no_comando();
463:    return(0);
464: }//POSIZIONA_SS_MOT_2()
465:
466: //*****************************************************
467: //POSIZIONA MOTORE 1  IN VELOCITA' CON RAMPA
468: //RITORNA 0 SE OK
469: //*****************************************************
470: char POSIZIONA_MOT_1()
471: {
472:   char temp_port1;
473:   char cont_attesa1;
474:
475:   if(CONTROLLO_BUSY_MOT_1()) return(4);
476:   if(CONTROLLO_ERR_MOT_1())  return(6);
477:   cont_attesa1=1000;
478:   temp_port1=PORTD&0xC0;
479:   temp_port1=temp_port1|0x22;
480:   PORTD=temp_port1;
481:   //attesa ricezione comando
482:   while(cont_attesa1)
483:     {
484:       delay_ms(1);
485:       cont_attesa1--;
486:       if(!cont_attesa1) {no_comando();return(10);}
487:       if(BUSY_MOT_1) cont_attesa1=0;
488:     }//while
489:    no_comando();
490:    return(0);
491: }//POSIZIONA_MOT_1()
492:
493: //*****************************************************
494: //POSIZIONA MOTORE 2  IN VELOCITA' CON RAMPA
```

```
495: //RITORNA 0 SE OK
496: //******************************************************
497: char POSIZIONA_MOT_2()
498: {
499:   char temp_port2;
500:   char cont_attesa2;
501:
502:   if(CONTROLLO_BUSY_MOT_2()) return(5);
503:   if(CONTROLLO_ERR_MOT_2())  return(7);
504:   cont_attesa2=1000;
505:   temp_port2=PORTD&0xC0;
506:   temp_port2=temp_port2|0x12;
507:   PORTD=temp_port2;
508:   //attesa ricezione comando
509:   while(cont_attesa2)
510:     {
511:       delay_ms(1);
512:       cont_attesa2--;
513:       if(!cont_attesa2) {no_comando();return(11);}
514:       if(BUSY_MOT_2) cont_attesa2=0;
515:     }//while
516:    no_comando();
517:    return(0);
518: }//POSIZIONA_MOT_2()
519:
520: //******************************************************
521: //POSIZIONA MOTORE 1  POSIZIONI MULTIPLE
522: //RITORNA 0 SE OK
523: //******************************************************
524: char POSIZIONA_M_MOT_1()
525: {
526:   char temp_port1;
527:   char cont_attesa1;
528:
529:   if(CONTROLLO_BUSY_MOT_1()) return(4);
530:   if(CONTROLLO_ERR_MOT_1())  return(6);
531:   cont_attesa1=1000;
532:   temp_port1=PORTD&0xC0;
533:   temp_port1=temp_port1|0x23;
534:   PORTD=temp_port1;
535:   //attesa ricezione comando
536:   while(cont_attesa1)
537:     {
538:       delay_ms(1);
539:       cont_attesa1--;
540:       if(!cont_attesa1) {no_comando();return(10);}
541:       if(BUSY_MOT_1) cont_attesa1=0;
542:     }//while
543:    no_comando();
544:    return(0);
545: }//POSIZIONA_M_MOT_1()
546:
547: //******************************************************
548: //POSIZIONA MOTORE 2  POSIZIONI MULTIPLE
549: //RITORNA 0 SE OK
550: //******************************************************
551: char POSIZIONA_M_MOT_2()
552: {
553:   char temp_port2;
554:   char cont_attesa2;
555:
556:   if(CONTROLLO_BUSY_MOT_2()) return(5);
```

```
557:   if(CONTROLLO_ERR_MOT_2())  return(7);
558:   cont_attesa2=1000;
559:   temp_port2=PORTD&0xC0;
560:   temp_port2=temp_port2|0x13;
561:   PORTD=temp_port2;
562:   //attesa ricezione comando
563:   while(cont_attesa2)
564:     {
565:       delay_ms(1);
566:       cont_attesa2--;
567:       if(!cont_attesa2) {no_comando();return(11);}
568:       if(BUSY_MOT_2) cont_attesa2=0;
569:     }//while
570:    no_comando();
571:    return(0);
572: }//POSIZIONA_M_MOT_2()
573:
574: //*****************************************************
575: //MUOVI INDIETRO MOTORE 1
576: //RITORNA 0 SE OK
577: //*****************************************************
578: char MOT_1_INDIETRO()
579: {
580:   char temp_port1;
581:
582:   if(CONTROLLO_BUSY_MOT_1()) return(4);
583:   if(CONTROLLO_ERR_MOT_1())  return(6);
584:   temp_port1=PORTD&0xC0;
585:   temp_port1=temp_port1|0x24;
586:   PORTD=temp_port1;
587:   return(0);
588: }//MOT_1_INDIETRO()
589:
590: //*****************************************************
591: //MUOVI INDIETRO MOTORE 2
592: //RITORNA 0 SE OK
593: //*****************************************************
594: char MOT_2_INDIETRO()
595: {
596:   char temp_port2;
597:
598:   if(CONTROLLO_BUSY_MOT_2()) return(5);
599:   if(CONTROLLO_ERR_MOT_2())  return(7);
600:   temp_port2=PORTD&0xC0;
601:   temp_port2=temp_port2|0x14;
602:   PORTD=temp_port2;
603:   return(0);
604: }//MOT_2_INDIETRO()
605:
606: //*****************************************************
607: //MUOVI IN AVANTI MOTORE 1
608: //RITORNA 0 SE OK
609: //*****************************************************
610: char MOT_1_AVANTI()
611: {
612:   char temp_port1;
613:
614:   if(CONTROLLO_BUSY_MOT_1()) return(4);
615:   if(CONTROLLO_ERR_MOT_1())  return(6);
616:   temp_port1=PORTD&0xC0;
617:   temp_port1=temp_port1|0x25;
618:   PORTD=temp_port1;
```

```
619:   return(0);
620: }//MOT_1_AVANTI()
621:
622: //*****************************************************
623: //MUOVI IN AVANTI MOTORE 2
624: //RITORNA 0 SE OK
625: //*****************************************************
626: char MOT_2_AVANTI()
627: {
628:   char temp_port2;
629:
630:   if(CONTROLLO_BUSY_MOT_2()) return(5);
631:   if(CONTROLLO_ERR_MOT_2())  return(7);
632:   temp_port2=PORTD&0xC0;
633:   temp_port2=temp_port2|0x15;
634:   PORTD=temp_port2;
635:   return(0);
636: }//MOT_2_AVANTI()
637:
638: //*****************************************************
639: //AZZERA MOTORE 1 SINGOLARMENTE
640: //RITORNA 0 SE OK
641: //*****************************************************
642: char AZZERA_MOT_1()
643: {
644:   char temp_port1;
645:   char cont_attesa1;
646:
647:   if(CONTROLLO_BUSY_MOT_1()) return(4);
648:   if(CONTROLLO_ERR_MOT_1())  return(6);
649:   cont_attesa1=1000;
650:   temp_port1=PORTD&0xC0;
651:   temp_port1=temp_port1|0x26;
652:   PORTD=temp_port1;
653:   //attesa ricezione comando
654:   while(cont_attesa1)
655:     {
656:       delay_ms(1);
657:       cont_attesa1--;
658:       if(!cont_attesa1) {no_comando();return(10);}
659:       if(BUSY_MOT_1) cont_attesa1=0;
660:     }//while
661:    no_comando();
662:    return(0);
663: }//AZZERA_MOT_1()
664:
665: //*****************************************************
666: //AZZERA MOTORE 2 SINGOLARMENTE
667: //RITORNA 0 SE OK
668: //*****************************************************
669: char AZZERA_MOT_2()
670: {
671:   char temp_port2;
672:   char cont_attesa2;
673:
674:   if(CONTROLLO_BUSY_MOT_2()) return(5);
675:   if(CONTROLLO_ERR_MOT_2())  return(7);
676:   cont_attesa2=1000;
677:   temp_port2=PORTD&0xC0;
678:   temp_port2=temp_port2|0x16;
679:   PORTD=temp_port2;
680:   //attesa ricezione comando
```

```
681:   while(cont_attesa2)
682:     {
683:       delay_ms(1);
684:       cont_attesa2--;
685:       if(!cont_attesa2) {no_comando();return(11);}
686:       if(BUSY_MOT_2) cont_attesa2=0;
687:     }//while
688:    no_comando();
689:    return(0);
690: }//AZZERA_MOT_2()
691:
692:
693:
694: //*******************************************************
695: //MOVIMENTI SIMULTANEI ASSI
696: //*******************************************************
697: //*******************************************************
698: //POSIZIONA MOTORI  IN START STOP
699: //RITORNA 0 SE OK
700: //*******************************************************
701: char POSIZIONA_SS_MOT()
702: {
703:   char temp_port;
704:   char cont_attesa;
705:
706:   if(CONTROLLO_BUSY_MOT_1()) return(4);
707:   if(CONTROLLO_BUSY_MOT_2()) return(5);
708:   if(CONTROLLO_ERR_MOT_1())  return(6);
709:   if(CONTROLLO_ERR_MOT_2())  return(7);
710:   cont_attesa=1000;
711:   temp_port=PORTD&0xC0;
712:   temp_port=temp_port|0x08;
713:   PORTD=temp_port;
714:   //attesa ricezione comando
715:   while(cont_attesa)
716:     {
717:       delay_ms(1);
718:       cont_attesa--;
719:       if(!cont_attesa)
720:         {
721:           no_comando();
722:           if((!BUSY_MOT_1)&&(!BUSY_MOT_2)) return(12);
723:           if(!BUSY_MOT_1) return(10);
724:           if(!BUSY_MOT_2) return(11);
725:         }//if
726:       if((BUSY_MOT_1)&&(BUSY_MOT_2)) cont_attesa=0;
727:     }//while
728:    no_comando();
729:    return(0);
730: }//POSIZIONA_SS_MOT()
731:
732: //*******************************************************
733: //POSIZIONA MOTORI  IN VELOCITA' CON RAMPA
734: //RITORNA 0 SE OK
735: //*******************************************************
736: char POSIZIONA_MOT()
737: {
738:   char temp_port;
739:   char cont_attesa;
740:
741:   if(CONTROLLO_BUSY_MOT_1()) return(4);
742:   if(CONTROLLO_BUSY_MOT_2()) return(5);
```

```
743:   if(CONTROLLO_ERR_MOT_1())  return(6);
744:   if(CONTROLLO_ERR_MOT_2())  return(7);
745:   cont_attesa=1000;
746:   temp_port=PORTD&0xC0;
747:   temp_port=temp_port|0x09;
748:   PORTD=temp_port;
749:   //attesa ricezione comando
750:   while(cont_attesa)
751:     {
752:       delay_ms(1);
753:       cont_attesa--;
754:       if(!cont_attesa)
755:         {
756:           no_comando();
757:           if((!BUSY_MOT_1)&&(!BUSY_MOT_2)) return(12);
758:           if(!BUSY_MOT_1) return(10);
759:           if(!BUSY_MOT_2) return(11);
760:         }//if
761:       if((BUSY_MOT_1)&&(BUSY_MOT_2)) cont_attesa=0;
762:     }//while
763:    no_comando();
764:    return(0);
765: }//POSIZIONA_MOT()
766:
767: //*****************************************************
768: //POSIZIONA MOTORI  POSIZIONI MULTIPLE
769: //RITORNA 0 SE OK
770: //*****************************************************
771: char POSIZIONA_M_MOT()
772: {
773:   char temp_port;
774:   char cont_attesa;
775:
776:   if(CONTROLLO_BUSY_MOT_1()) return(4);
777:   if(CONTROLLO_BUSY_MOT_2()) return(5);
778:   if(CONTROLLO_ERR_MOT_1())  return(6);
779:   if(CONTROLLO_ERR_MOT_2())  return(7);
780:   cont_attesa=1000;
781:   temp_port=PORTD&0xC0;
782:   temp_port=temp_port|0x0A;
783:   PORTD=temp_port;
784:   //attesa ricezione comando
785:   while(cont_attesa)
786:     {
787:       delay_ms(1);
788:       cont_attesa--;
789:       if(!cont_attesa)
790:         {
791:           no_comando();
792:           if((!BUSY_MOT_1)&&(!BUSY_MOT_2)) return(12);
793:           if(!BUSY_MOT_1) return(10);
794:           if(!BUSY_MOT_2) return(11);
795:         }//if
796:       if((BUSY_MOT_1)&&(BUSY_MOT_2)) cont_attesa=0;
797:     }//while
798:    no_comando();
799:    return(0);
800: }//POSIZIONA_M_MOT()
801:
802: //*****************************************************
803: //AZZERAMENTO CONTEMPORANEO MOTORI
804: //RITORNA 0 SE OK
```

```
805: //*****************************************************
806: char AZZERA_MOT()
807: {
808:   char temp_port;
809:   char cont_attesa;
810:
811:   if(CONTROLLO_BUSY_MOT_1()) return(4);
812:   if(CONTROLLO_BUSY_MOT_2()) return(5);
813:   if(CONTROLLO_ERR_MOT_1())  return(6);
814:   if(CONTROLLO_ERR_MOT_2())  return(7);
815:   cont_attesa=1000;
816:   temp_port=PORTD&0xC0;
817:   temp_port=temp_port|0x0B;
818:   PORTD=temp_port;
819:   //attesa ricezione comando
820:   while(cont_attesa)
821:     {
822:       delay_ms(1);
823:       cont_attesa--;
824:       if(!cont_attesa)
825:         {
826:           no_comando();
827:           if((!BUSY_MOT_1)&&(!BUSY_MOT_2)) ;return(12);
828:           if(!BUSY_MOT_1) return(10);
829:           if(!BUSY_MOT_2) return(11);
830:         }//if
831:       if((BUSY_MOT_1)&&(BUSY_MOT_2)) cont_attesa=0;
832:     }//while
833:    no_comando();
834:    return(0);
835: }//AZZERA_MOT()
836: //#########################################################
837:
838:
839:
840:
841: //#########################################################
842: //*****************************************************
843: //FUNZIONI DI CALCOLO
844: //*****************************************************
845: //*****************************************************
846: //ARROTONDA UN NUMERO REALE
847: //*****************************************************
848: int arrotonda(float numero_real)
849: {
850:   float temp_numero;
851:   int numero_arrotondato;
852:
853:   temp_numero=floor(numero_real);
854:   numero_arrotondato=temp_numero;
855:   temp_numero=numero_real-temp_numero;
856:   if(temp_numero>0.5) numero_arrotondato++;
857:   return(numero_arrotondato);
858: }//arrotonda
859:
860: //*****************************************************
861: //CALCOLO ANGOLI
862: //*****************************************************
863: //calcola gli angoli partendo dalle coordinate
864: //formula Kinematics inversa
865: //ritorna 0 s ok 1 se posizione non possibile
866: char calcola_ang(float Xc,float Yc)
```

```
867: {
868:    float temp1,temp2;
869:
870:    //calcolo gamma1  angolo tra L1 e asse ascisse
871:    if(Xc>=(-Dist/2))
872:      {
873:        d1=Dist/2+Xc;            //base del triangolo rettangolo con ipotenusa L1
874:        temp1=d1*d1+Yc*Yc;
875:        L1=sqrt(temp1);       //calcolo L1
876:        temp1=Yc/L1;
877:        if(temp1>1.0) return(-1);
878:        gamma1=asin(temp1);   //angolo di L1
879:      }//if
880:    else
881:     {
882:        d1=-Dist/2-Xc;           //base del triangolo rettangolo con ipotenusa L1
883:        temp1=d1*d1+Yc*Yc;
884:        L1=sqrt(temp1);       //calcolo L1
885:        temp1=Yc/L1;
886:        if(temp1>1.0) return(-1);
887:        gamma1=asin(temp1);
888:        gamma1=PI-gamma1;  //angolo di L1
889:     }//else
890:    //calcolo gamma2  angolo tra L2 e asse ascisse
891:    if(Xc<=(Dist/2))
892:      {
893:        d2=Dist/2-Xc;             //base del triangolo rettangolo con ipotenusa L2
894:        temp1=d2*d2+Yc*Yc;
895:        L2=sqrt(temp1);        //calcolo L2
896:        temp1=Yc/L2;
897:        if(temp1>1.0) return(-1);
898:        gamma2=asin(temp1);    //angolo di L2
899:      }//if
900:    else
901:     {
902:        d2=-Dist/2+Xc;            //base del triangolo rettangolo con ipotenusa L2
903:        temp1=d2*d2+Yc*Yc;
904:        L2=sqrt(temp1);
905:        temp1=Yc/L2;           //calcolo L2
906:        if(temp1>1.0) return(-1);
907:        gamma2=asin(temp1);
908:        gamma2=PI-gamma2;   //angolo di L2
909:     }//else
910:     //calcolo beta1 angolo tra L1 e B1
911:     temp1=Br1*Br1-Br2*Br2;
912:     temp2=L1*L1;
913:     temp2=temp2+temp1;
914:     temp2=temp2/2.0;
915:     temp2=temp2/Br1;
916:     temp2=temp2/L1;
917:     if(temp1>1.0) return(-1);
918:     beta1=acos(temp2);
919:     //calcolo beta2 angolo tra L2 e B1
920:     temp2=L2*L2;
921:     temp2=temp2+temp1;
922:     temp2=temp2/2.0;
923:     temp2=temp2/Br1;
924:     temp2=temp2/L2;
925:     if(temp1>1.0) return(-1);
926:     beta2=acos(temp2);
```

```
927:     //calcolo di alfa1 e alf2
928:     alfa1=PI-beta1-gamma1;
929:     alfa2=PI-beta2-gamma2;
930:    //converto gli angoli in gradi
931:    temp1=180.0/PI;
932:    temp2=alfa1;
933:    alfa1=temp1*temp2;
934:    temp1=180.0/PI;
935:    temp2=alfa2;
936:    alfa2=temp1*temp2;
937:    return(0);
938: }//calcola_ang(Xc,Yc)
939:
940: //******************************************************
941: //CALCOLA COORDINATE
942: //******************************************************
943: //trova le coordinate dal numero della casella
944: //ritorna 0 s ok 1 se posizione non possibile
945: char calcola_coord(char numero)
946: {
947:   float temp;
948:   char esito_calcola;
949:
950:   //leggo il numero della casella corrispondente al numero dell'elemento
951:   num_casella=elementi[numero-1];
952:   //ottengo riga e colonna
953:   //riga da 0 a 8 e colonna da 0 a 17
954:   //colonna
955:   colonna = num_casella % 18;     //calcolo il resto
956:   if(colonna==0)  colonna=17;
957:   else            colonna--;
958:   //riga
959:   riga = num_casella / 18;        //calcolo il quoziente
960:   if(colonna==17) riga--;
961:   //trovo le coordinate cartesiane del punto di presa
962:   temp=larghezza*colonna;
963:   Xpunto=Xini-offset_X-temp;
964:   temp=altezza*riga;
965:   Ypunto=Yini+offset_Y+temp;
966:   //aggiungo la distanza in Y per la seconda parte della tavola
967:   if(numero>126) Ypunto+=Dist_y;
968:   //calcolo gli angoli dalle coordinate
969:   esito_calcola=calcola_ang(Xpunto,Ypunto);
970:   if(esito_calcola) return(esito_calcola);
971:   else              return(0);
972: }//calcola_coord(char numero)
973: //#######################################################
974:
975:
976:
977:
978:
979: //#######################################################
980: //******************************************************
981: //COMUNICAZIONE SERIALE CON DSCPIC
982: //******************************************************
983: //******************************************************
984: //CANCELLA GLI ARRAY PER LA COMUNICAZIONE SERIALE
985: //******************************************************
986: void vuota_array_seriale()
987: {
988:    char cont=0;
```

```
 989:
 990:    while(cont<5)
 991:     {
 992:       dato_rx[cont]=0;
 993:       dato_tx[cont]=0;
 994:       cont++;
 995:     }//while(cont)
 996: }//vuota_array_seriale
 997:
 998: //******************************************************
 999: //COMUNICAZIONE PIC-DSPIC
1000: //VX#         -      INVIO VELOCITA' X
1001: //DXX#        -      INVIO DELAY IN uSEC PER POSIZIONAMENTO IN START STOP
1002: //PXX#        -      INVIO POSIZIONE ASSOLUTA IN IMPULSI
1003: //A#          -      LEGGI IL NUMERO ASSE
1004: //C#          -      LEGGI STATO ASSE
1005: //S#          -      LEGGI CODICE ERRORE
1006: //s#          -      RESETTA ERRORE
1007: //OXX#        -      INVIA OFFSET IN IMPULSI
1008: //o#          -      LEGGI OFFSET IN IMPULSI
1009: //v#          -      LEGGI LA VELOCITA' IMPOSTATA
1010: //d#          -      LEGGI I DELAY IMPOSTATO
1011: //p#          -      LEGGI LA POSIZIONE IMPOSTATA
1012: //L#          -      LEGGI POSIZIONE ASSOLUTA IN IMPULSI
1013: //M1XX#       -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP1
1014: //M2XX#       -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP2
1015: //M3XX#       -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP3
1016: //M4XX#       -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP4
1017: //M5XX#       -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP5
1018: //******************************************************
1019: char invia_dati()
1020: {
1021:   char cont_char;
1022:   int cont_time;
1023:
1024:   if(CONTROLLO_BUSY_MOT_1()) return(4);
1025:   if(CONTROLLO_BUSY_MOT_2()) return(5);
1026:   while(UART1_data_ready())
1027:    {
1028:      cont_char=UART1_Read();  //svuoto il buffer di ricezione
1029:      delay_ms(1);
1030:    }//while(UART1_data_ready())
1031:
1032:   cont_char=0;
1033:   while(cont_char<5)
1034:    {
1035:      //invio carattere dal vettore dati
1036:      UART1_Write(dato_tx[cont_char]);
1037:
1038:      cont_time=30000;
1039:      //attesa risposta
1040:      while(!UART1_Data_Ready())
1041:       {
1042:          delay_us(10);
1043:          cont_time--;
1044:          if(!cont_time) return(1);
1045:       }//while(!UART1_Data_Ready())
1046:      dato_rx[cont_char]=UART1_Read();
1047:      if(dato_rx[cont_char]!=dato_tx[cont_char]) return(2);
1048:      if(dato_rx[cont_char]=='#') return(0);
1049:      cont_char++;
1050:    }//while(cont_char<5)
```

```
1051:    return(3);
1052: }//invia_dati
1053:
1054: //*****************************************************
1055: //LEGGI UN VALORE INTERO DUE BYTE  XX  POSIZIONE ASSE
1056: //TORNA 1 SE ERRORE
1057: //tipo=0 valore in quota attuale
1058: //tipo=1 valore in quota richiesta
1059: //tipo=2 offset impostato
1060: //tipo=3 delay Start Stop impostato
1061: //*****************************************************
1062: char lettura_intero(char tipo)
1063: {
1064:   char cont_char;
1065:   char valore[2];
1066:   int cont_time;
1067:
1068:   cont_char=0;
1069:   while(cont_char<2)
1070:    {
1071:      cont_time=30000;
1072:      //attesa risposta
1073:      while(!UART1_Data_Ready())
1074:       {
1075:          delay_us(10);
1076:          cont_time--;
1077:          if(!cont_time) return(1);
1078:       }//while(!UART1_Data_Ready())
1079:     valore[cont_char]=UART1_Read();
1080:     cont_char++;
1081:     delay_us(100);
1082:     UART1_Write('+');
1083:    }//while(cont_char<2)
1084:   if(motor_sel==1)
1085:     {
1086:       if(tipo==0)
1087:        {
1088:          pos_attuale_A=valore[0];
1089:          pos_attuale_A=pos_attuale_A<<8;
1090:          pos_attuale_A=pos_attuale_A|valore[1];
1091:        }//if(tipo==0)
1092:       if(tipo==1)
1093:        {
1094:          pos_richiesta_A=valore[0];
1095:          pos_richiesta_A=pos_richiesta_A<<8;
1096:          pos_richiesta_A=pos_richiesta_A|valore[1];
1097:        }//if(tipo==1)
1098:       if(tipo==2)
1099:        {
1100:          offset_asse_1=valore[0];
1101:          offset_asse_1=offset_asse_1<<8;
1102:          offset_asse_1=offset_asse_1|valore[1];
1103:        }//if(tipo==2)
1104:       if(tipo==3)
1105:        {
1106:          del_A=valore[0];
1107:          del_A=del_A<<8;
1108:          del_A=del_A|valore[1];
1109:        }//if(tipo==3)
1110:     }//if
1111:   if(motor_sel==2)
1112:     {
```

```
1113:       if(!tipo)
1114:        {
1115:          pos_attuale_B=valore[0];
1116:          pos_attuale_B=pos_attuale_B<<8;
1117:          pos_attuale_B=pos_attuale_B|valore[1];
1118:        }//if(tipo==0)
1119:       if(tipo==1)
1120:        {
1121:          pos_richiesta_B=valore[0];
1122:          pos_richiesta_B=pos_richiesta_B<<8;
1123:          pos_richiesta_B=pos_richiesta_B|valore[1];
1124:        }//if(tipo==1)
1125:       if(tipo==2)
1126:        {
1127:          offset_asse_2=valore[0];
1128:          offset_asse_2=offset_asse_2<<8;
1129:          offset_asse_2=offset_asse_2|valore[1];
1130:        }//if(tipo==2)
1131:       if(tipo==3)
1132:        {
1133:          del_B=valore[0];
1134:          del_B=del_B<<8;
1135:          del_B=del_B|valore[1];
1136:        }//if(tipo==3)
1137:     }//if
1138:   return(0);
1139: }//lettura_intero()
1140:
1141: //*******************************************************
1142: //LEGGI UN BYTE - VELOCITA' IMPOSTATA
1143: //TORNA 1 SE ERRORE
1144: //TIPO 0 VELOCITA' ASSE
1145: //TIPO 1 NUMERO ASSE
1146: //TIPO 2 CODICE ERRORE
1147: //TIPO 3 STATO ASSE
1148: //*******************************************************
1149: char lettura_byte(char tipo)
1150: {
1151:   int cont_time;
1152:   char ricevuto;
1153:
1154:   cont_time=30000;
1155:   //attesa risposta
1156:   while(!UART1_Data_Ready())
1157:    {
1158:      delay_us(10);
1159:      cont_time--;
1160:      if(!cont_time) return(1);
1161:    }//while(!UART1_Data_Ready())
1162:   ricevuto=UART1_Read();
1163:   if(motor_sel==1)
1164:     {
1165:       if(tipo==0)   vel_A=ricevuto;
1166:       if(tipo==1)   num_asse=ricevuto;
1167:       if(tipo==2)   codice_err_1=ricevuto;
1168:       if(tipo==3)   stato_asse_1=ricevuto;
1169:     }//if
1170:   if(motor_sel==2)
1171:     {
1172:       if(tipo==0)   vel_B=ricevuto;
1173:       if(tipo==1)   num_asse=ricevuto;
1174:       if(tipo==2)   codice_err_2=ricevuto;
```

```
1175:       if(tipo==3)   stato_asse_2=ricevuto;
1176:     }//if
1177:   return(0);
1178: }//lettura_byte()
1179:
1180: //*****************************************************
1181: //INVIA IL VALORE DI OFFSET
1182: //TORNA 0 SE OK
1183: //*****************************************************
1184: char invia_offset(char asse)
1185: {
1186:   char esito_offset;
1187:   unsigned short offset_lo,offset_hi;
1188:
1189:   //imposta offset motore 1
1190:   if(asse==1)
1191:    {
1192:      if(CONTROLLO_BUSY_MOT_1()) return(4);
1193:      SEL_MOT_1();
1194:      offset_lo=lo(offset_asse_1);
1195:      offset_hi=hi(offset_asse_1);
1196:      vuota_array_seriale();
1197:      dato_tx[0]='O';
1198:      dato_tx[1]=offset_hi;
1199:      dato_tx[2]=offset_lo;
1200:      dato_tx[3]='#';
1201:      esito_offset=invia_dati();
1202:      if(esito_offset) return(esito_offset);
1203:    }//if(asse==1)
1204:   //imposta offset motore 2
1205:   if(asse==2)
1206:    {
1207:      if(CONTROLLO_BUSY_MOT_2()) return(5);
1208:      SEL_MOT_2();;
1209:      offset_lo=lo(offset_asse_2);
1210:      offset_hi=hi(offset_asse_2);
1211:      vuota_array_seriale();
1212:      dato_tx[0]='O';
1213:      dato_tx[1]=offset_hi;
1214:      dato_tx[2]=offset_lo;
1215:      dato_tx[3]='#';
1216:      esito_offset=invia_dati();
1217:      if(esito_offset) return(esito_offset);
1218:    }//if(asse==2)
1219:   return(0);
1220: }//invia_offset()
1221:
1222: //*****************************************************
1223: //LEGGI L'OFFSET DELL'ASSE
1224: //TORNA 0 SE OK
1225: //*****************************************************
1226: char leggi_offset()
1227: {
1228:   char esito_offset;
1229:
1230:   if(CONTROLLO_BUSY_MOT_1()) return(4);
1231:   if(CONTROLLO_BUSY_MOT_2()) return(5);
1232:   //leggi offset asse 1
1233:   SEL_MOT_1();
1234:   vuota_array_seriale();
1235:   dato_tx[0]='o';
1236:   dato_tx[1]='#';
```

```
1237:   esito_offset=invia_dati();
1238:   if(esito_offset) return(esito_offset);
1239:   esito_offset=lettura_intero(2);
1240:   if(esito_offset) return(esito_offset);
1241:   if(CONTROLLO_BUSY_MOT_1()) return(4);
1242:   if(CONTROLLO_BUSY_MOT_2()) return(5);
1243:   delay_ms(100);
1244:   //leggi offset asse 2
1245:   SEL_MOT_2();
1246:   vuota_array_seriale();
1247:   dato_tx[0]='o';
1248:   dato_tx[1]='#';
1249:   esito_offset=invia_dati();
1250:   if(esito_offset) return(esito_offset);
1251:   esito_offset=lettura_intero(2);
1252:   if(esito_offset) return(esito_offset);
1253:   return(0);
1254: }//leggi_offset(char asse)
1255:
1256:
1257: //*****************************************************
1258: //LEGGI LIL DELAY DELL'ASSE PER I MOVIMENTI IN START/STOP
1259: //TORNA 0 SE OK
1260: //*****************************************************
1261: char leggi_delay()
1262: {
1263:   char esito_delay;
1264:
1265:   if(CONTROLLO_BUSY_MOT_1()) return(4);
1266:   if(CONTROLLO_BUSY_MOT_2()) return(5);
1267:   //leggi offset asse 1
1268:   SEL_MOT_1();
1269:   vuota_array_seriale();
1270:   dato_tx[0]='d';
1271:   dato_tx[1]='#';
1272:   esito_delay=invia_dati();
1273:   if(esito_delay) return(esito_delay);
1274:   esito_delay=lettura_intero(3);
1275:   if(esito_delay) return(esito_delay);
1276:   if(CONTROLLO_BUSY_MOT_1()) return(4);
1277:   if(CONTROLLO_BUSY_MOT_2()) return(5);
1278:   delay_ms(10);
1279:   //leggi offset asse 2
1280:   SEL_MOT_2();
1281:   vuota_array_seriale();
1282:   dato_tx[0]='d';
1283:   dato_tx[1]='#';
1284:   esito_delay=invia_dati();
1285:   if(esito_delay) return(esito_delay);
1286:   esito_delay=lettura_intero(3);
1287:   if(esito_delay) return(esito_delay);
1288:   return(0);
1289: }//leggi_delay(char asse)
1290:
1291:
1292: //*****************************************************
1293: //INVIA IL VALORE DI VELOCITA' DI UN ASSE
1294: //TORNA 0 SE OK
1295: //*****************************************************
1296: char invia_vel(char asse, char vel)
1297: {
1298:    char esito_vel;
```

```
1299:
1300:    vuota_array_seriale();
1301:    if(asse==1)
1302:     {
1303:       if(CONTROLLO_BUSY_MOT_1()) return(4);
1304:       SEL_MOT_1();
1305:       vuota_array_seriale();
1306:       dato_tx[0]='V';
1307:       dato_tx[1]=vel;
1308:       dato_tx[2]='#';
1309:       esito_vel=invia_dati();
1310:       if(esito_vel) return(esito_vel);
1311:       //controlla velocità inviata
1312:       vuota_array_seriale();
1313:       dato_tx[0]='v';
1314:       dato_tx[1]='#';
1315:       esito_vel=invia_dati();
1316:       if(esito_vel) return(esito_vel);
1317:       esito_vel=lettura_byte(0);
1318:       if(esito_vel) return(esito_vel);
1319:       if(vel_A!=vel) return(15);
1320:     }//if(asse==1)
1321:    if(asse==2)
1322:     {
1323:       if(CONTROLLO_BUSY_MOT_2()) return(5);
1324:       SEL_MOT_2();
1325:       vuota_array_seriale();
1326:       dato_tx[0]='V';
1327:       dato_tx[1]=vel;
1328:       dato_tx[2]='#';
1329:       esito_vel=invia_dati();
1330:       if(esito_vel) return(esito_vel);
1331:       //controlla velocità inviata
1332:       vuota_array_seriale();
1333:       dato_tx[0]='v';
1334:       dato_tx[1]='#';
1335:       esito_vel=invia_dati();
1336:       if(esito_vel) return(esito_vel);
1337:       esito_vel=lettura_byte(0);
1338:       if(esito_vel) return(esito_vel);
1339:       if(vel_B!=vel) return(15);
1340:     }//if(asse==2)
1341:    return(0);
1342: }//invia_vel(char asse)
1343:
1344:
1345: //*****************************************************
1346: //INVIA IL VALORE DI RITARDO IN uS PER IL MOVIMENTI IN START STOP
1347: //DEI DUE ASSI
1348: //TORNA 0 SE OK
1349: //*****************************************************
1350: char invia_del_SS(int del_mot_1, int del_mot_2)
1351: {
1352:    char esito_vel_SS;
1353:    unsigned short valore_lo,valore_hi;
1354:
1355:
1356:    vuota_array_seriale();
1357:    if(CONTROLLO_BUSY_MOT_1()) return(4);
1358:    SEL_MOT_1();
1359:    valore_lo=lo(del_mot_1);
1360:    valore_hi=hi(del_mot_1);
```

```
1361:    vuota_array_seriale();
1362:    dato_tx[0]='D';
1363:    dato_tx[1]=valore_hi;
1364:    dato_tx[2]=valore_lo;
1365:    dato_tx[3]='#';
1366:    esito_vel_SS=invia_dati();
1367:    if(esito_vel_SS) return(esito_vel_SS);
1368:    delay_ms(5);
1369:    vuota_array_seriale();
1370:    if(CONTROLLO_BUSY_MOT_2()) return(5);
1371:    SEL_MOT_2();
1372:    valore_lo=lo(del_mot_2);
1373:    valore_hi=hi(del_mot_2);
1374:    vuota_array_seriale();
1375:    dato_tx[0]='D';
1376:    dato_tx[1]=valore_hi;
1377:    dato_tx[2]=valore_lo;
1378:    dato_tx[3]='#';
1379:    esito_vel_SS=invia_dati();
1380:    if(esito_vel_SS) return(esito_vel_SS);
1381:    delay_ms(5);
1382:    esito_vel_SS=leggi_delay();
1383:    if(esito_vel_SS) return(esito_vel_SS);
1384:    if((del_mot_1==del_A)&&(del_mot_2==del_B)) return(0);
1385:    else                                       return(24);
1386: }//invia_del_SS(int del_mot_1, int del_mot_2)
1387:
1388:
1389: //*****************************************************
1390: //INVIA IL VALORE DI VELOCITA' DEI DUE ASSI
1391: //TORNA 0 SE OK
1392: //*****************************************************
1393: char invia_vel_assi(char vel_1, char vel_2)
1394: {
1395:    char esito_vel_1;
1396:
1397:    vuota_array_seriale();
1398:    if(CONTROLLO_BUSY_MOT_1()) return(4);
1399:    SEL_MOT_1();
1400:    vuota_array_seriale();
1401:    dato_tx[0]='V';
1402:    dato_tx[1]=vel_1;
1403:    dato_tx[2]='#';
1404:    esito_vel_1=invia_dati();
1405:    if(esito_vel_1) return(esito_vel_1);
1406:    //controlla velocità inviata
1407:    vuota_array_seriale();
1408:    dato_tx[0]='v';
1409:    dato_tx[1]='#';
1410:    esito_vel_1=invia_dati();
1411:    if(esito_vel_1) return(esito_vel_1);
1412:    esito_vel_1=lettura_byte(0);
1413:    if(esito_vel_1) return(esito_vel_1);
1414:    if(vel_A!=vel_1) return(15);
1415:    delay_ms(100);
1416:    if(CONTROLLO_BUSY_MOT_2()) return(5);
1417:    SEL_MOT_2();
1418:    vuota_array_seriale();
1419:    dato_tx[0]='V';
1420:    dato_tx[1]=vel_2;
1421:    dato_tx[2]='#';
1422:    esito_vel_1=invia_dati();
```

```
1423:    if(esito_vel_1) return(esito_vel_1);
1424:    //controlla velocità inviata
1425:    vuota_array_seriale();
1426:    dato_tx[0]='v';
1427:    dato_tx[1]='#';
1428:    esito_vel_1=invia_dati();
1429:    if(esito_vel_1) return(esito_vel_1);
1430:    esito_vel_1=lettura_byte(0);
1431:    if(esito_vel_1) return(esito_vel_1);
1432:    if(vel_B!=vel_2) return(15);
1433:    return(0);
1434: }//invia_vel_assi(char vel_1, char vel_2)
1435:
1436: //*****************************************************
1437: //Leggi posiziona attuale asse
1438: //torna 0 se ok
1439: //*****************************************************
1440: char leggi_pos_attuale(char asse)
1441: {
1442:   char esito_leggi_att;
1443:
1444:   if(asse==1)
1445:    {
1446:      if(CONTROLLO_BUSY_MOT_1()) return(4);
1447:      SEL_MOT_1();
1448:      delay_ms(100);
1449:      vuota_array_seriale();
1450:      dato_tx[0]='L';
1451:      dato_tx[1]='#';
1452:      esito_leggi_att=invia_dati();
1453:      if(esito_leggi_att) return(esito_leggi_att);
1454:      esito_leggi_att=lettura_intero(0);
1455:      if(esito_leggi_att) return(esito_leggi_att);
1456:    }//if(asse==1)
1457:   if(asse==2)
1458:    {
1459:      if(CONTROLLO_BUSY_MOT_2()) return(5);
1460:      SEL_MOT_2();
1461:      delay_ms(100);
1462:      vuota_array_seriale();
1463:      dato_tx[0]='L';
1464:      dato_tx[1]='#';
1465:      esito_leggi_att=invia_dati();
1466:      if(esito_leggi_att) return(esito_leggi_att);
1467:      esito_leggi_att=lettura_intero(0);
1468:      if(esito_leggi_att) return(esito_leggi_att);
1469:    }//if(asse==2)
1470: }//leggi_pos_attuale(char asse)
1471:
1472:
1473: //*****************************************************
1474: //Leggi posiziona impostata asse
1475: //torna 0 se ok
1476: //*****************************************************
1477: char leggi_pos_impostata(char asse)
1478: {
1479:   char esito_leggi_imp;
1480:
1481:   if(asse==1)
1482:    {
1483:      if(CONTROLLO_BUSY_MOT_1()) return(4);
1484:      SEL_MOT_1();
```

```
1485:       delay_ms(10);
1486:       vuota_array_seriale();
1487:       dato_tx[0]='p';
1488:       dato_tx[1]='#';
1489:       esito_leggi_imp=invia_dati();
1490:       if(esito_leggi_imp) return(esito_leggi_imp);
1491:       esito_leggi_imp=lettura_intero(1);
1492:       if(esito_leggi_imp) return(esito_leggi_imp);
1493:     }//if(asse==1)
1494:    if(asse==2)
1495:     {
1496:       if(CONTROLLO_BUSY_MOT_2()) return(5);
1497:       SEL_MOT_2();
1498:       delay_ms(10);
1499:       vuota_array_seriale();
1500:       dato_tx[0]='p';
1501:       dato_tx[1]='#';
1502:       esito_leggi_imp=invia_dati();
1503:       if(esito_leggi_imp) return(esito_leggi_imp);
1504:       esito_leggi_imp=lettura_intero(1);
1505:       if(esito_leggi_imp) return(esito_leggi_imp);
1506:     }//if(asse==2)
1507: }//leggi_pos_impostata(char asse)
1508:
1509:
1510: //*******************************************************
1511: //TEST DELLA SERIALE ESC PER USCIRE
1512: //*******************************************************
1513: void test_seriale()
1514: {
1515:   char esito_test,esito_test_1;
1516: //TEST SERIALE
1517: SEL_MOT_1();
1518: while(PULS_ESC)
1519:  {
1520:    vuota_array_seriale();
1521:    dato_tx[0]='A';
1522:    dato_tx[1]='#';
1523:    esito_test=invia_dati();
1524:    esito_test_1=lettura_byte(1);
1525:    Lcd_Cmd(_LCD_CLEAR);
1526:    ByteToStr(num_asse,txt_byte);
1527:    Lcd_Out(2,1,txt_byte);
1528:    Lcd_Out(1,1,"A#");
1529:    ByteToStr(esito_test,txt_byte);
1530:    Lcd_Out(1,5,txt_byte);
1531:    ByteToStr(esito_test_1,txt_byte);
1532:    Lcd_Out(1,10,txt_byte);
1533:    delay_ms(1000);
1534:
1535:    vuota_array_seriale();
1536:    dato_tx[0]='V';
1537:    dato_tx[1]='1';
1538:    dato_tx[2]='#';
1539:    esito_test=invia_dati();
1540:    Lcd_Cmd(_LCD_CLEAR);
1541:    Lcd_Out(1,1,"V1#");
1542:    ByteToStr(esito_test,txt_byte);
1543:    Lcd_Out(1,8,txt_byte);
1544:    Lcd_Chr(2,1,dato_rx[0]);
1545:    Lcd_Chr(2,2,dato_rx[1]);
1546:    Lcd_Chr(2,3,dato_rx[2]);
```

```
1547:     delay_ms(1000);
1548:
1549:     vuota_array_seriale();
1550:     dato_tx[0]='P';
1551:     dato_tx[1]='2';
1552:     dato_tx[2]='2';
1553:     dato_tx[3]='#';
1554:     esito_test=invia_dati();
1555:     Lcd_Cmd(_LCD_CLEAR);
1556:     Lcd_Out(1,1,"P22#");
1557:     ByteToStr(esito_test,txt_byte);
1558:     Lcd_Out(1,8,txt_byte);
1559:     Lcd_Chr(2,1,dato_rx[0]);
1560:     Lcd_Chr(2,2,dato_rx[1]);
1561:     Lcd_Chr(2,3,dato_rx[2]);
1562:     Lcd_Chr(2,4,dato_rx[3]);
1563:     delay_ms(1000);
1564:
1565:     vuota_array_seriale();
1566:     dato_tx[0]='M';
1567:     dato_tx[1]='1';
1568:     dato_tx[2]='3';
1569:     dato_tx[3]='3';
1570:     dato_tx[4]='#';
1571:     esito_test=invia_dati();
1572:     Lcd_Cmd(_LCD_CLEAR);
1573:     Lcd_Out(1,1,"M133#");
1574:     ByteToStr(esito_test,txt_byte);
1575:     Lcd_Out(1,8,txt_byte);
1576:     Lcd_Chr(2,1,dato_rx[0]);
1577:     Lcd_Chr(2,2,dato_rx[1]);
1578:     Lcd_Chr(2,3,dato_rx[2]);
1579:     Lcd_Chr(2,4,dato_rx[3]);
1580:     Lcd_Chr(2,5,dato_rx[4]);
1581:     delay_ms(1000);
1582:
1583:     vuota_array_seriale();
1584:     dato_tx[0]='L';
1585:     dato_tx[1]='#';
1586:     esito_test=invia_dati();
1587:     esito_test_1=lettura_intero(0);
1588:     if(motor_sel==1)
1589:      {
1590:        IntToStr(pos_attuale_A,txt_int);
1591:      }//if
1592:     if(motor_sel==2)
1593:      {
1594:        IntToStr(pos_attuale_B,txt_int);
1595:      }//if
1596:     Lcd_Cmd(_LCD_CLEAR);
1597:     Lcd_Out(1,1,"L#");
1598:     ByteToStr(esito_test,txt_byte);
1599:     Lcd_Out(1,5,txt_byte);
1600:     ByteToStr(esito_test_1,txt_byte);
1601:     Lcd_Out(1,10,txt_byte);
1602:     Lcd_Out(2,1,txt_int);
1603:     delay_ms(1000);
1604:
1605:     vuota_array_seriale();
1606:     dato_tx[0]='p';
1607:     dato_tx[1]='#';
1608:     esito_test=invia_dati();
```

```
1609:    esito_test_1=lettura_intero(1);
1610:    if(motor_sel==1)
1611:     {
1612:       IntToStr(pos_richiesta_A,txt_int);;
1613:     }//if
1614:    if(motor_sel==2)
1615:     {
1616:       IntToStr(pos_richiesta_B,txt_int);;
1617:     }//if
1618:    Lcd_Cmd(_LCD_CLEAR);
1619:    Lcd_Out(1,1,"p#");
1620:    ByteToStr(esito_test,txt_byte);
1621:    Lcd_Out(1,5,txt_byte);
1622:    ByteToStr(esito_test_1,txt_byte);
1623:    Lcd_Out(1,10,txt_byte);
1624:    Lcd_Out(2,1,txt_int);
1625:    delay_ms(1000);
1626:
1627:    vuota_array_seriale();
1628:    dato_tx[0]='v';
1629:    dato_tx[1]='#';
1630:    esito_test=invia_dati();
1631:    esito_test_1=lettura_byte(0);
1632:    Lcd_Cmd(_LCD_CLEAR);
1633:    if(motor_sel==1)
1634:     {
1635:       ByteToStr(vel_A,txt_byte);
1636:       Lcd_Out(2,1,txt_byte);
1637:     }//if
1638:    if(motor_sel==2)
1639:     {
1640:       ByteToStr(vel_B,txt_byte);
1641:       Lcd_Out(2,1,txt_byte);
1642:     }//if
1643:    Lcd_Out(1,1,"v#");
1644:    ByteToStr(esito_test,txt_byte);
1645:    Lcd_Out(1,5,txt_byte);
1646:    ByteToStr(esito_test_1,txt_byte);
1647:    Lcd_Out(1,10,txt_byte);
1648:    delay_ms(1000);
1649:
1650:   if(motor_sel==1)
1651:    {
1652:       SEL_MOT_2();
1653:    }
1654:   else
1655:    {
1656:       SEL_MOT_1();
1657:    }
1658:  }
1659:  Lcd_Cmd(_LCD_CLEAR);
1660: }//test_seriale
1661: //#########################################################
1662:
1663:
1664:
1665: //*****************************************************
1666: //LEGGI ERRORE
1667: //RITORNA 0 SE OK
1668: //*****************************************************
1669: char leggi_errore()
1670: {
```

```
1671:   char esito_leggi;
1672:
1673:   codice_err_1=0;
1674:   codice_err_2=0;
1675:
1676:   Lcd_Cmd(_LCD_CLEAR);
1677:   Lcd_Out(1, 1, "ERR.M1  ERR.M2");
1678:   //leggi errore mot 1
1679:   if(ERR_MOT_1)
1680:    {
1681:      SEL_MOT_1();
1682:      vuota_array_seriale();
1683:      dato_tx[0]='S';
1684:      dato_tx[1]='#';
1685:      esito_leggi=invia_dati();
1686:      if(esito_leggi) return(esito_leggi);
1687:      esito_leggi=lettura_byte(2);
1688:      if(esito_leggi) return(esito_leggi);
1689:      ByteToStr(codice_err_1,txt_byte);
1690:      Lcd_Out(2,2,txt_byte);
1691:    }//if(ERR_MOT_1)
1692:   //leggi errore mot 2
1693:   if(ERR_MOT_2)
1694:    {
1695:      SEL_MOT_2();
1696:      vuota_array_seriale();
1697:      dato_tx[0]='S';
1698:      dato_tx[1]='#';
1699:      esito_leggi=invia_dati();
1700:      if(esito_leggi) return(esito_leggi);
1701:      esito_leggi=lettura_byte(2);
1702:      if(esito_leggi) return(esito_leggi);
1703:      ByteToStr(codice_err_2,txt_byte);
1704:      Lcd_Out(2,12,txt_byte);
1705:    }//if(ERR_MOT_1)
1706:    delay_ms(200);
1707:    while(PULS_ESC){}  //attesa pulsante invio
1708:    delay_ms(200);
1709:    return(0);
1710: }//leggi_errore()
1711: //#######################################################
1712:
1713: //*******************************************************
1714: //LEGGI STATO
1715: //RITORNA 0 SE OK
1716: //*******************************************************
1717: char leggi_stato(char numero_asse)
1718: {
1719:   char esito_leggi;
1720:
1721:   //leggi stato asse mot 1
1722:   if(numero_asse==1)
1723:    {
1724:      SEL_MOT_1();
1725:      vuota_array_seriale();
1726:      dato_tx[0]='C';
1727:      dato_tx[1]='#';
1728:      esito_leggi=invia_dati();
1729:      if(esito_leggi) return(esito_leggi);
1730:      esito_leggi=lettura_byte(3);
1731:      if(esito_leggi) return(esito_leggi);
1732:    }//if(numero_asse==1)
```

```
1733:   //leggi stato asse mot 2
1734:   if(numero_asse==2)
1735:    {
1736:      SEL_MOT_2();
1737:      vuota_array_seriale();
1738:      dato_tx[0]='C';
1739:      dato_tx[1]='#';
1740:      esito_leggi=invia_dati();
1741:      if(esito_leggi) return(esito_leggi);
1742:      esito_leggi=lettura_byte(3);
1743:      if(esito_leggi) return(esito_leggi);
1744:    }//if(numero_asse==2)
1745:    return(0);
1746: }//leggi_stato()
1747: //#########################################################
1748:
1749: //#########################################################
1750: //*********************************************************
1751: //FUNZIONI DI CONTROLLO CICLO E MOVIMENTI
1752: //*********************************************************
1753:
1754:
1755: //*********************************************************
1756: //CALCOLA I DELAY PER IL MOVIMENTO IN START STOP
1757: //CONSIDERO UNA VELOCITA' DI START/STOP DI 1000Hz
1758: //DELAY DI 1000 uSEC
1759: //1/4 di giro al secondo
1760: //torna 0 se ok
1761: //*********************************************************
1762: char calcola_delay(int passi_1, int passi_2)
1763: {
1764:   int passiA,passiB;
1765:   char esito_delay;
1766:
1767:   esito_delay=leggi_pos_attuale(1);
1768:   if(esito_delay) return(esito_delay);
1769:   delay_ms(5);
1770:   esito_delay=leggi_pos_attuale(2);
1771:   if(esito_delay) return(esito_delay);
1772:   delay_ms(5);
1773:
1774:   passiA=0;
1775:   if(passi_1>pos_attuale_A)  passiA=passi_1-pos_attuale_A;
1776:   if(passi_1<pos_attuale_A)  passiA=pos_attuale_A-passi_1;
1777:   passiB=0;
1778:   if(passi_2>pos_attuale_B)  passiB=passi_2-pos_attuale_B;
1779:   if(passi_2<pos_attuale_B)  passiB=pos_attuale_B-passi_2;
1780:
1781:   if(passiA==passiB)
1782:    {
1783:      del_A=delay_base;
1784:      del_B=delay_base;
1785:    }//if(passiA==passiB)
1786:   if(passiA>passiB)
1787:    {
1788:      if(passiB!=0)
1789:       {
1790:         temp_1=passiA;
1791:         temp_2=passiB;
1792:         rapporto=temp_1/temp_2;
1793:         del_A=delay_base;
1794:         temp_1=del_A;
```

```
1795:         temp_2=temp_1*rapporto;
1796:         del_B=temp_2;
1797:       }//if(passiB!=0)
1798:      else
1799:       {
1800:         del_A=delay_base;
1801:         del_B=delay_base;
1802:       }//else
1803:    }//if(passiA>passiB)
1804:   if(passiB>passiA)
1805:    {
1806:      if(passiA!=0)
1807:       {
1808:         temp_1=passiA;
1809:         temp_2=passiB;
1810:         rapporto=temp_2/temp_1;
1811:         del_B=delay_base;
1812:         temp_2=del_B;
1813:         temp_1=temp_2*rapporto;
1814:         del_A=temp_1;
1815:       }//if(passiA!=0)
1816:      else
1817:       {
1818:         del_A=delay_base;
1819:         del_B=delay_base;
1820:       }//else
1821:    }//if(passiB>passiA)
1822: /*
1823:   Lcd_Cmd(_LCD_CLEAR);
1824:   inttostr(passiA,txt_int);
1825:   Lcd_Out(1,1,txt_int);
1826:   inttostr(del_A,txt_int);
1827:   Lcd_Out(1,8,txt_int);
1828:   inttostr(passiB,txt_int);
1829:   Lcd_Out(2,1,txt_int);
1830:   inttostr(del_B,txt_int);
1831:   Lcd_Out(2,8,txt_int);
1832: */
1833: }//void calcola delay()
1834:
1835:
1836:
1837: //******************************************************
1838: //POSIZIONA ALLA QUOTA INDICATA
1839: //RITORNA 0 SE OK
1840: //******************************************************
1841: char posiziona_quota(int val1, int val2)
1842: {
1843:   char esito_pos_quota;
1844:   unsigned short valore_lo,valore_hi;
1845:   int cont_attesa,temp_del_A,temp_del_B;
1846:
1847:   //carica impulsi angolo 1
1848:   SEL_MOT_1();
1849:   valore_lo=lo(val1);
1850:   valore_hi=hi(val1);
1851:   vuota_array_seriale();
1852:   dato_tx[0]='P';
1853:   dato_tx[1]=valore_hi;
1854:   dato_tx[2]=valore_lo;
1855:   dato_tx[3]='#';
1856:   debug=2;
```

```
1857:   esito_pos_quota=invia_dati();
1858:   if(esito_pos_quota) return(esito_pos_quota);
1859:   delay_ms(5);
1860:   //carica impulsi angolo 2
1861:   SEL_MOT_2();
1862:   valore_lo=lo(val2);
1863:   valore_hi=hi(val2);
1864:   vuota_array_seriale();
1865:   dato_tx[0]='P';
1866:   dato_tx[1]=valore_hi;
1867:   dato_tx[2]=valore_lo;
1868:   dato_tx[3]='#';
1869:   debug=3;
1870:   esito_pos_quota=invia_dati();
1871:   if(esito_pos_quota) return(esito_pos_quota);
1872:   delay_ms(5);
1873:   //verifica la quota impostata
1874:   debug=4;
1875:   esito_pos_quota=leggi_pos_impostata(1);
1876:   if(esito_pos_quota) return(esito_pos_quota);
1877:   delay_ms(5);
1878:   debug=5;
1879:   esito_pos_quota=leggi_pos_impostata(2);
1880:   if(esito_pos_quota) return(esito_pos_quota);
1881:   if((pos_richiesta_A)!=(val1)) return(20);
1882:   if((pos_richiesta_B)!=(val2)) return(21);
1883:   delay_ms(5);
1884:   //verifico se è il ciclo lento
1885:   if(tipo_ciclo==1)
1886:    {
1887:     //calcolo i delay per i due motori
1888:     calcola_delay(val1,val2);
1889:     //invio i delay per i due motori
1890:     esito_pos_quota=invia_del_SS(del_A,del_B);
1891:     if(esito_pos_quota) return(esito_pos_quota);
1892:     delay_ms(5);
1893:    }//if(tipo_ciclo==1)
1894:   //avvia il posizionamento
1895:   debug=6;
1896:   if(tipo_ciclo==0)  esito_pos_quota=POSIZIONA_MOT();
1897:   if(tipo_ciclo==1)  esito_pos_quota=POSIZIONA_SS_MOT();
1898:   if(esito_pos_quota) return(esito_pos_quota);
1899:   if(CONTROLLO_ERR_MOT_1())  return(6);
1900:   if(CONTROLLO_ERR_MOT_2())  return(7);
1901:   //attesa fine posizionamento
1902:   cont_attesa=20000;
1903:   while(cont_attesa)
1904:    {
1905:      delay_ms(1);
1906:      cont_attesa--;
1907:      if(!cont_attesa) return(22);
1908:      if(EMERGENZA) return(99);
1909:      if((ERR_MOT_1)||(ERR_MOT_2)) return(0);
1910:      if((!BUSY_MOT_1)&&(!BUSY_MOT_2)) cont_attesa=0;
1911:    }//while(cont_attesa)
1912:   debug=7;
1913:   esito_pos_quota=leggi_pos_attuale(1);
1914:   if(esito_pos_quota) return(esito_pos_quota);
1915:   delay_ms(5);
1916:   debug=8;
1917:   esito_pos_quota=leggi_pos_attuale(2);
1918:   if(esito_pos_quota) return(esito_pos_quota);
```

```
1919:   if((pos_attuale_A==pos_richiesta_A)&&(pos_attuale_B)==(pos_richiesta_B)) return
      (0);
1920:   else                                                                    return
      (1);
1921: }//posiziona_quota(float val1, float val2)
1922:
1923:
1924: //*****************************************************
1925: //POSIZIONA ALLA CASELLA INDICATA
1926: //RITORNA 0 SE OK
1927: //*****************************************************
1928: char posiziona(char casella)
1929: {
1930:   char esito_posiziona;
1931:   int impulsi_1,impulsi_2;
1932:
1933:   impulsi_1=impulsi[0][casella-1];
1934:   impulsi_2=impulsi[1][casella-1];
1935:   esito_posiziona=posiziona_quota(impulsi_1,impulsi_2);
1936:   return(esito_posiziona);
1937: }//posiziona(char casella)
1938:
1939:
1940: //*****************************************************
1941: //VAI IN ZONA PRELIEVO
1942: //RITORNA 0 SE OK
1943: //*****************************************************
1944: char prelievo()
1945: {
1946:   float X_prelievo,Y_prelievo;
1947:   int ang1_imp_pre,ang2_imp_pre;
1948:   char esito_pre;
1949:
1950:   //coordinate punto di prelievo
1951:   X_prelievo=Xini+offset_x+larghezza;
1952:   Y_prelievo=Yini+offset_Y;
1953:   esito_pre=calcola_ang(X_prelievo,Y_prelievo);
1954:   if(esito_pre) return(esito_pre);
1955:   //impulsi punto di prelievo
1956:   ang1_imp_pre=arrotonda(alfa1/risoluzione);
1957:   ang2_imp_pre=arrotonda(alfa2/risoluzione);
1958:   esito_pre=posiziona_quota(ang1_imp_pre,ang2_imp_pre);
1959:   return(esito_pre);
1960: }//prelievo()
1961:
1962:
1963: //*****************************************************
1964: //VAI IN POSIZIONE ZERO
1965: //RITORNA 0 SE OK
1966: //*****************************************************
1967: char posiziona_riposo()
1968: {
1969:   char esito_pos_riposo;
1970:   float ang_riposo_1,ang_riposo_2;
1971:
1972:   ang_riposo_1=0;
1973:   ang_riposo_2=0;
1974:
1975:   esito_pos_riposo=posiziona_quota(ang_riposo_1,ang_riposo_2);
1976:   return(esito_pos_riposo);
1977: }//posiziona_riposo()
1978:
```

```
1979:
1980: //*****************************************************
1981: //VAI IN POSIZIONE DIRITTA IN BASSO
1982: //RITORNA 0 SE OK
1983: //*****************************************************
1984: char posiziona_diritto()
1985: {
1986:   char esito_pos_diritto;
1987:   float ang_diritto_1,ang_diritto_2;
1988:
1989:   ang_diritto_1=5000;
1990:   ang_diritto_2=5000;
1991:
1992:   esito_pos_diritto=posiziona_quota(ang_diritto_1,ang_diritto_2);
1993:   return(esito_pos_diritto);
1994: }//posiziona_diritto()
1995:
1996:
1997:
1998: //*****************************************************
1999: //VAI PRIMA DEGLI OFFSET
2000: //RITORNA 0 SE OK
2001: //*****************************************************
2002: char posiziona_offset()
2003: {
2004:   char esito_pos_riposo;
2005:
2006:   esito_pos_riposo=posiziona_quota((offset_asse_1+100),(offset_asse_2+100));
2007:   return(esito_pos_riposo);
2008: }//posiziona_offset()
2009:
2010:
2011: //#####################################################
2012:
2013:
2014: //#####################################################
2015: //*****************************************************
2016: //COMANDI DA MENU
2017: //AZZERA
2018: //ERRORI
2019: //RESET ALLARMI
2020: //START CICLO
2021: //MANUALE
2022: //OFFSET ZERO
2023: //VALORI
2024: //*****************************************************
2025:
2026: //*****************************************************
2027: //AZZERA IL BRACCIO
2028: //RITORNA 0 SE ASSI AZZERATI SENZA ERRORE
2029: //*****************************************************
2030: char azzera_robot()
2031: {
2032:   char esito_azzera;
2033:   int attesa_azzera;
2034:   char cont_attesa;
2035:
2036:   Lcd_Cmd(_LCD_CLEAR);
2037:   Lcd_Out(1,1,"AZZERAMENTO");
2038:   Lcd_Out(2,1,"IN CORSO");
2039:   delay_ms(1000);
2040:
```

```
2041:   robot_azzerato=0;
2042:   esito_azzera=AZZERA_MOT();
2043:   no_comando();
2044:   if(esito_azzera) return(esito_azzera);
2045:
2046:   //attesa azzeramento assi
2047:   cont_attesa=5;
2048:   attesa_azzera=1200;
2049:   while(attesa_azzera)
2050:    {
2051:      InttoStr(attesa_azzera,txt_int);
2052:      Lcd_Out(2,10,txt_int);
2053:      attesa_azzera--;
2054:      delay_ms(100);
2055:      if(!attesa_azzera)
2056:       {
2057:         if((BUSY_MOT_1)&&(BUSY_MOT_2)) return(13);
2058:         if(BUSY_MOT_1) return(8);
2059:         if(BUSY_MOT_2) return(9);
2060:         if((ERR_MOT_1)||(ERR_MOT_2))  return(0);
2061:       }//if
2062:      if((!BUSY_MOT_1)&&(!BUSY_MOT_2)) cont_attesa--;
2063:      else                             cont_attesa=5;
2064:      if(!cont_attesa)  attesa_azzera=0;
2065:    }//while
2066:   esito_azzera=leggi_stato(1);
2067:   if(esito_azzera) return(esito_azzera);
2068:   esito_azzera=leggi_stato(2);
2069:   if(esito_azzera) return(esito_azzera);
2070:   if((!ZERO_OK_1)&&(!ZERO_OK_2)) return(14);
2071:   if(!ZERO_OK_1) return(18);
2072:   if(!ZERO_OK_2) return(19);
2073:   robot_azzerato=1;
2074:   return(0);
2075: }//azzera_robot
2076:
2077: //*****************************************************
2078: //AZZERA IL BRACCIO
2079: //RITORNA 0 SE ASSI AZZERATI SENZA ERRORE
2080: //*****************************************************
2081: char controllo_zero()
2082: {
2083:   char esito_azzera;
2084:   int attesa_azzera;
2085:   char cont_attesa;
2086:
2087:   esito_azzera=AZZERA_MOT();
2088:   no_comando();
2089:   if(esito_azzera) return(esito_azzera);
2090:
2091:   //attesa azzeramento assi
2092:   cont_attesa=5;
2093:   attesa_azzera=1200;
2094:   while(attesa_azzera)
2095:    {
2096:      InttoStr(attesa_azzera,txt_int);
2097:      Lcd_Out(2,10,txt_int);
2098:      attesa_azzera--;
2099:      delay_ms(100);
2100:      if(!attesa_azzera)
2101:       {
2102:         if((BUSY_MOT_1)&&(BUSY_MOT_2)) return(13);
```

```
2103:         if(BUSY_MOT_1) return(8);
2104:         if(BUSY_MOT_2) return(9);
2105:         if((ERR_MOT_1)||(ERR_MOT_2))  return(0);
2106:       }//if
2107:      if((!BUSY_MOT_1)&&(!BUSY_MOT_2)) cont_attesa--;
2108:      else                             cont_attesa=5;
2109:      if(!cont_attesa)  attesa_azzera=0;
2110:    }//while
2111:   esito_azzera=leggi_stato(1);
2112:   if(esito_azzera) return(esito_azzera);
2113:   esito_azzera=leggi_stato(2);
2114:   if(esito_azzera) return(esito_azzera);
2115:   if((!ZERO_OK_1)&&(!ZERO_OK_2)) return(14);
2116:   if(!ZERO_OK_1) return(18);
2117:   if(!ZERO_OK_2) return(19);
2118:   robot_azzerato=1;
2119:   return(0);
2120: }//controllo_zero
2121:
2122: //*****************************************************
2123: //RESET ERRORI ASSI
2124: //RITORNA 0 SE OK TRASMISSIONE
2125: //*****************************************************
2126: char reset_errori_assi()
2127: {
2128:    char esito_reset;
2129:
2130:    if(CONTROLLO_BUSY_MOT_1()) return(4);
2131:    SEL_MOT_1();
2132:    vuota_array_seriale();
2133:    dato_tx[0]='s';
2134:    dato_tx[1]='#';
2135:    esito_reset=invia_dati();
2136:    if(esito_reset) return(esito_reset);
2137:    delay_ms(100);
2138:    if(CONTROLLO_BUSY_MOT_2()) return(5);
2139:    SEL_MOT_2();
2140:    vuota_array_seriale();
2141:    dato_tx[0]='s';
2142:    dato_tx[1]='#';
2143:    esito_reset=invia_dati();
2144:    if(esito_reset) return(esito_reset);
2145:    delay_ms(100);
2146:    return(0);
2147: }//reset_errori_assi
2148:
2149: //*****************************************************
2150: //CICLO AUTOMATICO
2151: //tipo=0 rapido tipo=1 lento
2152: //RITORNA 0 SE ASSI AZZERATI SENZA ERRORE
2153: //*****************************************************
2154: char ciclo_automatico(char tipo)
2155: {
2156:   char esito_auto,conferma_num;
2157:   char cont_esc;
2158:   char tipo_puls,fase;
2159:
2160:   tipo_ciclo=tipo;
2161:
2162:   if(!robot_azzerato)
2163:    {
2164:      Lcd_Cmd(_LCD_CLEAR);
```

```
2165:       Lcd_Out(1, 1, "AZZERARE");
2166:       delay_ms(2000);
2167:       Lcd_Cmd(_LCD_CLEAR);
2168:       return(0);
2169:     }//if(!robot_azzerato)
2170:
2171:     //scegli velocità di lavoro
2172:     conferma_num=1;
2173:     Lcd_Cmd(_LCD_CLEAR);
2174:     Lcd_Out(1, 1, "VELOCITA'=");
2175:     bytetostr(vel,txt_byte);
2176:     Lcd_Out(1,11,txt_byte);
2177:     Lcd_Out(2, 1, "CONFERMI?");
2178:     while(conferma_num)
2179:      {
2180:        if(!PULS_UP)
2181:          {
2182:            if(vel<90) vel++;
2183:            bytetostr(vel,txt_byte);
2184:            Lcd_Out(1,11,txt_byte);
2185:            delay_ms(100);
2186:          }// if(!PULS_UP)
2187:        if(!PULS_DOWN)
2188:          {
2189:            if(vel>1) vel--;
2190:            bytetostr(vel,txt_byte);
2191:            Lcd_Out(1,11,txt_byte);
2192:            delay_ms(100);
2193:          }// if(!PULS_DOWN)
2194:        if(!PULS_INVIO) conferma_num=0;
2195:        if(!PULS_ESC) {vel=vel_ini;conferma_num=0;}
2196:      }//while(conferma_num)
2197:     while((!PULS_INVIO)||(!PULS_ESC)) {}  //attesa rilascio
2198:
2199:   if(tipo==0)
2200:     {
2201:       //invio la velocità selezionata
2202:       vel_A=vel;
2203:       vel_B=vel;
2204:       esito_auto=invia_vel_assi(vel_A,vel_B);
2205:       if(esito_auto) return(esito_auto);
2206:       delay_ms(100);
2207:     }//if(tipo==0)
2208:   if(tipo==1)
2209:     {
2210:       //inposto il delay base
2211:       delay_base=(100-vel)*10;
2212:     }//if(tipo==1)
2213:
2214:   //scegli l'elemento da cui cominciare
2215:   conferma_num=1;
2216:   Lcd_Cmd(_LCD_CLEAR);
2217:   Lcd_Out(1, 1, "ELEMENTO");
2218:   bytetostr(ordine_elementi[num_sequenza-1],txt_byte);
2219:   Lcd_Out(1,10,txt_byte);
2220:   Lcd_Out(2, 1, "CONFERMI?");
2221:   bytetostr(num_sequenza,txt_byte);
2222:   Lcd_Out(1,14,txt_byte);
2223:   while(conferma_num)
2224:     {
2225:       if(!PULS_UP)
2226:         {
```

```
2227:           if(num_sequenza<118) num_sequenza++;
2228:           bytetostr(ordine_elementi[num_sequenza-1],txt_byte);
2229:           Lcd_Out(1,10,txt_byte);
2230:           bytetostr(num_sequenza,txt_byte);
2231:           Lcd_Out(1,14,txt_byte);
2232:           delay_ms(100);
2233:         }// if(!PULS_UP)
2234:       if(!PULS_DOWN)
2235:         {
2236:           if(num_sequenza>1) num_sequenza--;
2237:           bytetostr(ordine_elementi[num_sequenza-1],txt_byte);
2238:           Lcd_Out(1,10,txt_byte);
2239:           bytetostr(num_sequenza,txt_byte);
2240:           Lcd_Out(1,14,txt_byte);
2241:           delay_ms(100);
2242:         }// if(!PULS_DOWN)
2243:       if(!PULS_INVIO) conferma_num=0;
2244:       if(!PULS_ESC)       //esc per iniziare dal primo elemento
2245:        {
2246:          num_sequenza=1;
2247:          bytetostr(ordine_elementi[num_sequenza-1],txt_byte);
2248:          Lcd_Out(1,10,txt_byte);
2249:          bytetostr(num_sequenza,txt_byte);
2250:          Lcd_Out(1,14,txt_byte);
2251:          delay_ms(2000);
2252:          conferma_num=0;
2253:        }//if(!PULS_ESC)
2254:     }//while(conferma_num)
2255:   while((!PULS_INVIO)||(!PULS_ESC)) {}  //attesa rilascio
2256:
2257:   Lcd_Cmd(_LCD_CLEAR);
2258:   Lcd_Out(1, 1, "CICLO IN CORSO");
2259:   Lcd_Out(2, 1,riga_vuota);
2260:   Lcd_Out(2, 1, "Pos.Riposo");
2261:   esito_auto=posiziona_riposo();
2262:   if(esito_auto) return(esito_auto);
2263:   //esito_auto=posiziona_offset();
2264:   //if(esito_auto) return(esito_auto);
2265:   //esito_auto=controllo_zero();
2266:   //if(esito_auto) return(esito_auto);
2267:
2268:   fase=0; //0 in zero 1 in deposito 2 in prelievo
2269:   while(num_sequenza<119)
2270:     {
2271:       tipo_puls=0;
2272:       //ATTESA PULSANTE DI START1 O START2
2273:       while(tipo_puls==0)
2274:         {
2275:            if(PULS_START1) tipo_puls=1;
2276:            if(PULS_START2) tipo_puls=2;
2277:            //controllo pulsante ESC
2278:           cont_esc=1000;
2279:           while(!PULS_ESC)
2280:             {
2281:               cont_esc--;
2282:               delay_ms(1);
2283:               if(cont_esc==0)
2284:                {
2285:                   if(fase!=0)
2286:                     {
2287:                        //vai in posizione zero
2288:                        esito_auto=posiziona_riposo();
```

```
2289:                       if(esito_auto) return(esito_auto);
2290:                       //esito_auto=posiziona_offset();
2291:                       //if(esito_auto) return(esito_auto);
2292:                       //esito_auto=controllo_zero();
2293:                       //if(esito_auto) return(esito_auto);
2294:                     }//if(fase!=0)
2295:                   return(0);
2296:                 }//if
2297:               }//while(!PULS_ESC)
2298:        }//while
2299:
2300:       //vai in deposito
2301:       fase=1;
2302:       Lcd_Out(2, 1,riga_vuota);
2303:       Lcd_Out(2, 1, "Deposito");
2304:       bytetostr(ordine_elementi[num_sequenza-1],txt_byte);
2305:       Lcd_Out(2,10,txt_byte);
2306:       bytetostr(num_sequenza,txt_byte);
2307:       Lcd_Out(2,14,txt_byte);
2308:       esito_auto=posiziona(ordine_elementi[num_sequenza-1]);
2309:       if(esito_auto) return(esito_auto);
2310:       num_sequenza++;
2311:
2312:       //se ho premuto il primo pulsante
2313:       if(tipo_puls==1)
2314:        {
2315:          //ATTESA PULSANTE DI START
2316:          while((!PULS_START1)&&(!PULS_START2))
2317:            {
2318:               PISTONE=1;
2319:               delay_ms(200);
2320:               PISTONE=0;
2321:               delay_ms(200);
2322:            }//while
2323:          PISTONE=0;
2324:          fase=0;
2325:          Lcd_Out(2, 1,riga_vuota);
2326:          Lcd_Out(2, 1, "Pos.Riposo");
2327:          esito_auto=posiziona_riposo();
2328:          if(esito_auto) return(esito_auto);
2329:        }//if(tipo_puls==1)
2330:
2331:       //se ho premuto il secondo pulsante
2332:       if(tipo_puls==2)
2333:        {
2334:          //ATTESA PULSANTE DI START
2335:          while(tipo_puls==2)
2336:            {
2337:               PISTONE=1;
2338:               delay_ms(200);
2339:               PISTONE=0;
2340:               delay_ms(200);
2341:               if(PULS_START2) tipo_puls=0;
2342:               if(PULS_START1) tipo_puls=1;
2343:            }//while
2344:          PISTONE=0;
2345:          if(tipo_puls==1)
2346:           {
2347:              fase=0;
2348:              Lcd_Out(2, 1,riga_vuota);
2349:              Lcd_Out(2, 1, "Pos.Riposo");
2350:              esito_auto=posiziona_riposo();
```

```
2351:             if(esito_auto) return(esito_auto);
2352:           }//if(tipo_puls==1)
2353:         }//if(tipo_puls==2)
2354:       }//while(num_sequenza<119)
2355:
2356:     //vai in posizione zero
2357:     if(num_sequenza==119) num_sequenza=1;
2358:     delay_ms(100);
2359:     esito_auto=posiziona_riposo();
2360:     if(esito_auto) return(esito_auto);
2361:     return(0);
2362: }//ciclo_automatico
2363:
2364:
2365: //******************************************************
2366: //IMPOSTA OFFSET ZERO ASSI
2367: //RITORNA 0 SE OK TRASMISSIONE
2368: //******************************************************
2369: char imposta_offset()
2370: {
2371:   char esito_offset,varia_offset,esci_offset;
2372:   int offset_max,offset_min;
2373:   int temp_offset;
2374:
2375:   offset_max=180.0/risoluzione;
2376:   offset_min=offset_max*-1.0;
2377:   //leggo l'offset attuale dei due assi
2378:   esito_offset=leggi_offset();
2379:   if(esito_offset) return(esito_offset);
2380:   //imposto l'offset del primo asse
2381:   Lcd_Cmd(_LCD_CLEAR);
2382:   Lcd_Out(1, 1,"IMPOSTA OFFSET 1");
2383:   IntToStr(offset_asse_1,txt_int);
2384:   Lcd_Out(2,1,txt_int);
2385:   temp_offset=offset_asse_1;
2386:   varia_offset=0;
2387:   esci_offset=1;
2388:   while(esci_offset)
2389:     {
2390:       if(!PULS_ESC)
2391:         {
2392:           offset_asse_1=0;
2393:           varia_offset=1;
2394:         }//if(!PULS_INVIO)
2395:       if(!PULS_UP)
2396:         {
2397:           if(offset_asse_1<offset_max) offset_asse_1++;
2398:           varia_offset=1;
2399:         }//if(!PULS_UP)
2400:       if(!PULS_DOWN)
2401:         {
2402:           if(offset_asse_1>offset_min) offset_asse_1--;
2403:           varia_offset=1;
2404:         }//if(!PULS_DOWN)
2405:       if(varia_offset)
2406:         {
2407:           Lcd_Out(2,1,riga_vuota);
2408:           IntToStr(offset_asse_1,txt_int);
2409:           Lcd_Out(2,1,txt_int);
2410:           varia_offset=0;
2411:           delay_ms(50);
2412:         }//if(varia_offset)
```

```
2413:       if(!PULS_INVIO)  esci_offset=0;
2414:     }//while(esci_offset)
2415:    //attesa rilascio pulsante ESC
2416:    while(!PULS_INVIO) {}
2417:    //se l'offset è cambiato lo invio al DSPIC
2418:    if(temp_offset!=offset_asse_1)
2419:     {
2420:       esito_offset=invia_offset(1);
2421:       if(esito_offset) return(esito_offset);
2422:     }//if(temp_offset!=offset_asse_1)
2423:
2424:    //imposto l'offset del secondo asse
2425:    Lcd_Cmd(_LCD_CLEAR);
2426:    Lcd_Out(1, 1,"IMPOSTA OFFSET 2");
2427:    IntToStr(offset_asse_2,txt_int);
2428:    Lcd_Out(2,1,txt_int);
2429:    temp_offset=offset_asse_2;
2430:    varia_offset=0;
2431:    esci_offset=1;
2432:    delay_ms(300);
2433:    while(esci_offset)
2434:     {
2435:       if(!PULS_ESC)
2436:        {
2437:          offset_asse_2=0;
2438:          varia_offset=1;
2439:        }//if(!PULS_INVIO)
2440:       if(!PULS_UP)
2441:        {
2442:          if(offset_asse_2<offset_max) offset_asse_2++;
2443:          varia_offset=1;
2444:        }//if(!PULS_UP)
2445:       if(!PULS_DOWN)
2446:        {
2447:          if(offset_asse_2>offset_min) offset_asse_2--;
2448:          varia_offset=1;
2449:        }//if(!PULS_DOWN)
2450:       if(varia_offset)
2451:        {
2452:          Lcd_Out(2,1,riga_vuota);
2453:          IntToStr(offset_asse_2,txt_int);
2454:          Lcd_Out(2,1,txt_int);
2455:          varia_offset=0;
2456:          delay_ms(50);
2457:        }//if(varia_offset)
2458:       if(!PULS_INVIO)  esci_offset=0;
2459:     }//while(esci_offset)
2460:    //attesa rilascio pulsante ESC
2461:    while(!PULS_INVIO) {}
2462:    //se l'offset è cambiato lo invio al DSPIC
2463:    if(temp_offset!=offset_asse_2)
2464:     {
2465:       esito_offset=invia_offset(2);
2466:       if(esito_offset) return(esito_offset);
2467:     }//if(temp_offset!=offset_asse_2)
2468:    delay_ms(300);
2469:    return(0);
2470: }//imposta_offset
2471:
2472: //*****************************************************
2473: //VISUALIZZA VALORI
2474: //*****************************************************
```

```
2475: void visualizza_valori()
2476: {
2477:   char attesa,tipo;
2478:   int cont_time;
2479:
2480:   attesa=1;
2481:   Lcd_Cmd(_LCD_CLEAR);
2482:   Lcd_Out(1,1,"INVIO PER SCEGLIERE");
2483:   while(attesa)
2484:    {
2485:      if(attesa)
2486:        {
2487:          Lcd_Out(2,1,riga_vuota);
2488:          Lcd_Out(2,1,"IMPULSI");
2489:          cont_time=1000;
2490:        }//if(attesa)
2491:      while((cont_time)&&(attesa))
2492:       {
2493:         cont_time--;
2494:         delay_ms(1);
2495:         if(!PULS_INVIO)
2496:          {
2497:           attesa=0;
2498:           cont_time=0;
2499:           tipo=0;
2500:          }//if(!PULS_INVIO)
2501:       }//while(cont_time)
2502:
2503:      if(attesa)
2504:        {
2505:          Lcd_Out(2,1,riga_vuota);
2506:          Lcd_Out(2,1,"ANGOLI");
2507:          cont_time=1000;
2508:        }//if(attesa)
2509:      while((cont_time)&&(attesa))
2510:       {
2511:         cont_time--;
2512:         delay_ms(1);
2513:         if(!PULS_INVIO)
2514:          {
2515:           attesa=0;
2516:           cont_time=0;
2517:           tipo=1;
2518:          }//if(!PULS_INVIO)
2519:       }//while(cont_time)
2520:    }//while(scelta);
2521:
2522:   while(!PULS_INVIO){}  //attesa rilascio pulsante invio
2523:   delay_ms(500);
2524:
2525:   Lcd_Cmd(_LCD_CLEAR);
2526:   if(tipo==0)
2527:    {
2528:      Lcd_Out(1,1,"VISUALIZZA");
2529:      Lcd_Out(2,1,"IMPULSI");
2530:    }//if(tipo==0)
2531:   else
2532:    {
2533:      Lcd_Out(1,1,"VISUALIZZA");
2534:      Lcd_Out(2,1,"ANGOLI");
2535:    }//else
2536:   delay_ms(2000);
```

```
2537:   //inizio visualizzazione
2538:   Lcd_Cmd(_LCD_CLEAR);
2539:   Lcd_Out(1,1,"ELEMENTO NUM.");
2540:   num_elemento=1;
2541:   while(num_elemento<119)
2542:    {
2543:      ByteToStr(num_elemento,txt_byte);
2544:      Lcd_Out(1,14,txt_byte);
2545:      if(tipo==0)
2546:       {
2547:        IntToStr((impulsi[0][num_elemento-1]),txt_int);
2548:        Lcd_Out(2,1,txt_int);
2549:        IntToStr((impulsi[1][num_elemento-1]),txt_int);
2550:        Lcd_Out(2,9,txt_int);
2551:       }//if(tipo==0)
2552:      else
2553:       {
2554:        FloatToStr_FixLen((angoli[0][num_elemento-1]),txt_float,7);
2555:        Lcd_Out(2,1,txt_float);
2556:        FloatToStr_FixLen((angoli[1][num_elemento-1]),txt_float,7);
2557:        Lcd_Out(2,9,txt_float);
2558:       }//else
2559:     num_elemento++;
2560:     delay_ms(300);
2561:     attesa=1;
2562:     while(attesa)
2563:      {
2564:       if(!PULS_INVIO) attesa=0;  //per avanzare
2565:       if(!PULS_ESC)              //per uscire
2566:         {
2567:          attesa=0;
2568:          num_elemento=119;
2569:         }//if(!PULS_ESC)
2570:      }//while(attesa)
2571:    }//while(num_elemento<119)
2572:    while(!PULS_INVIO){}//attesa rilascio
2573:    delay_ms(200);
2574: }//visualizza_valori
2575: //#######################################################
2576:
2577:
2578: char visualizza_info_assi()
2579: {
2580:   char esito_info;
2581:
2582:   if(CONTROLLO_BUSY_MOT_1()) return(4);
2583:   if(CONTROLLO_BUSY_MOT_2()) return(5);
2584:   //leggo e visualizzo lo stato dei due motori
2585:   Lcd_Cmd(_LCD_CLEAR);
2586:   Lcd_Out(1, 1, "Mot1    Mot2");
2587:   esito_info=leggi_stato(1);
2588:   if(esito_info) return(esito_info);
2589:   esito_info=leggi_stato(2);
2590:   if(esito_info) return(esito_info);
2591:   //visualizza i valori
2592:   ByteToStr(stato_asse_1,txt_byte);
2593:   Lcd_Out(2,1,txt_byte);
2594:   ByteToStr(stato_asse_2,txt_byte);
2595:   Lcd_Out(2,8,txt_byte);
2596:   delay_ms(200);
2597:   while(PULS_ESC){}  //attesa pulsante invio
2598:   delay_ms(200);
```

```
2599:   //leggo e visualizzo l'offset dei due motori
2600:   Lcd_Cmd(_LCD_CLEAR);
2601:   Lcd_Out(1, 1, "Offset1 Offset2");
2602:   esito_info=leggi_offset();
2603:   if(esito_info) return(esito_info);
2604:   //visualizza i valori
2605:   IntToStr(offset_asse_1,txt_int);
2606:   Lcd_Out(2,1,txt_int);
2607:   IntToStr(offset_asse_2,txt_int);
2608:   Lcd_Out(2,8,txt_int);
2609:   delay_ms(200);
2610:   while(PULS_ESC){}  //attesa pulsante invio
2611:   delay_ms(200);
2612:   //leggo e visualizzo la posizione attuale dei due motori
2613:   Lcd_Cmd(_LCD_CLEAR);
2614:   Lcd_Out(1, 1, "Pos1    Pos2");
2615:   esito_info=leggi_pos_attuale(1);
2616:   if(esito_info) return(esito_info);
2617:   delay_ms(100);
2618:   esito_info=leggi_pos_attuale(2);
2619:   if(esito_info) return(esito_info);
2620:   delay_ms(100);
2621:   //visualizza i valori
2622:   IntToStr(pos_attuale_A,txt_int);
2623:   Lcd_Out(2,1,txt_int);
2624:   IntToStr(pos_attuale_B,txt_int);
2625:   Lcd_Out(2,8,txt_int);
2626:   delay_ms(200);
2627:   while(PULS_ESC){}  //attesa pulsante invio
2628:   delay_ms(200);
2629:   //leggo e visualizzo la velocità impostata dei due motori
2630:   Lcd_Cmd(_LCD_CLEAR);
2631:   Lcd_Out(1, 1, "Vel1    Vel2");
2632:   if(CONTROLLO_BUSY_MOT_1()) return(4);
2633:   SEL_MOT_1();
2634:   delay_ms(100);
2635:   vuota_array_seriale();
2636:   dato_tx[0]='v';
2637:   dato_tx[1]='#';
2638:   esito_info=invia_dati();
2639:   if(esito_info) return(esito_info);
2640:   esito_info=lettura_byte(0);
2641:   if(esito_info) return(esito_info);
2642:   if(CONTROLLO_BUSY_MOT_2()) return(5);
2643:   SEL_MOT_2();
2644:   delay_ms(100);
2645:   vuota_array_seriale();
2646:   dato_tx[0]='v';
2647:   dato_tx[1]='#';
2648:   esito_info=invia_dati();
2649:   if(esito_info) return(esito_info);
2650:   esito_info=lettura_byte(0);
2651:   //visualizza i valori
2652:   if(esito_info) return(esito_info);
2653:   ByteToStr(vel_A,txt_byte);
2654:   Lcd_Out(2,1,txt_byte);
2655:   ByteToStr(vel_B,txt_byte);
2656:   Lcd_Out(2,8,txt_byte);
2657:   delay_ms(200);
2658:   while(PULS_ESC){}  //attesa pulsante invio
2659:   delay_ms(200);
2660:   //leggo e visualizzo il delay dei due motori
```

```
2661:   Lcd_Cmd(_LCD_CLEAR);
2662:   Lcd_Out(1, 1, "Del1    Del2");
2663:   esito_info=leggi_delay();
2664:   if(esito_info) return(esito_info);
2665:   delay_ms(100);
2666:   //visualizza i valori
2667:   IntToStr(del_A,txt_int);
2668:   Lcd_Out(2,1,txt_int);
2669:   IntToStr(del_B,txt_int);
2670:   Lcd_Out(2,8,txt_int);
2671:   delay_ms(200);
2672:   while(PULS_ESC){}  //attesa pulsante invio
2673:   return(0);
2674: }//visualizza_info_assi()
2675:
2676:
2677:
2678: //******************************************************
2679: //VISUALIZZA IL MENU' MANUALE
2680: //RITORNA DA 1 A 5 SECONDO L'OPZIONE RICHIESTA
2681: //******************************************************
2682: char menu_manuale()
2683: {
2684:    char man_down,man_up,man_invio,man_esc;
2685:    char man_cambio;
2686:    unsigned int cont_visu=15000;
2687:
2688:    man_down=cont_rimbalzo;
2689:    man_up=cont_rimbalzo;
2690:    man_invio=cont_rimbalzo;
2691:    man_esc=cont_rimbalzo;
2692:
2693:    man_cambio=1;
2694:    while(1)
2695:     {
2696:      if(!PULS_INVIO) man_invio--;
2697:      else            man_invio=cont_rimbalzo;
2698:      if(!PULS_UP)    man_up--;
2699:      else            man_up=cont_rimbalzo;
2700:      if(!PULS_DOWN)  man_down--;
2701:      else            man_down=cont_rimbalzo;
2702:
2703:      if(EMERGENZA) return(99);
2704:      if((ERR_MOT_1)||(ERR_MOT_2)) return(0);
2705:      if(!PULS_ESC) return(0);
2706:
2707:
2708:      //pulsante invio
2709:      if(!man_invio)
2710:       {
2711:         while(!PULS_INVIO){}//attesa rilascio
2712:         return(menu_man_scelta);
2713:       }//(!man_invio)
2714:
2715:      //pulsante sù
2716:      if(!man_up)
2717:       {
2718:        if(menu_man_scelta>1)
2719:         {
2720:          menu_man_scelta--;
2721:         }
2722:        man_up=cont_rimbalzo;
```

```
2723:         man_cambio=1;
2724:        }//if(!man_up)
2725:
2726:       //pulsante giù
2727:       if(!man_down)
2728:        {
2729:         if(menu_man_scelta<4)
2730:          {
2731:           menu_man_scelta++;
2732:          }
2733:         man_down=cont_rimbalzo;
2734:         man_cambio=1;
2735:        }//if(!man_down)
2736:
2737:       if(man_cambio==1)
2738:         {
2739:           man_cambio=0;
2740:           Lcd_Out(2,1, riga_vuota);
2741:           switch(menu_man_scelta)
2742:            {
2743:              case 1:
2744:                {
2745:                  Lcd_Out(2,1,"1-");
2746:                  Lcd_Out(2,3,txt_motore_A);
2747:                  break;
2748:                }
2749:              case 2:
2750:                {
2751:                  Lcd_Out(2,1,"2-");
2752:                  Lcd_Out(2,3,txt_motore_B);
2753:                  break;
2754:                }
2755:              case 3:
2756:                {
2757:                  Lcd_Out(2,1,"3-");
2758:                  Lcd_Out(2,3,txt_pistone);
2759:                  break;
2760:                }
2761:              case 4:
2762:                {
2763:                  Lcd_Out(2,1,"4-");
2764:                  Lcd_Out(2,3,txt_magnete);
2765:                  break;
2766:                }
2767:            }//switch
2768:           while((!PULS_DOWN)||(!PULS_UP)){} //attesa rilascio
2769:           delay_ms(50);
2770:         }//if(man_cambio)
2771:      }//while
2772: }//menu_manuale
2773:
2774:
2775: //*****************************************************
2776: //MOVIMENTI MANUALI
2777: //RITORNA 0 SE ASSI AZZERATI SENZA ERRORE
2778: //*****************************************************
2779: char movimenti_manuali()
2780: {
2781:   char esito_manuale;
2782:   char attesa;
2783:
2784:   Lcd_Cmd(_LCD_CLEAR);
```

```
2785:   Lcd_Out(1, 5, "MANUALE");
2786:
2787:   while(1)
2788:    {
2789:       esito_menu=menu_manuale();
2790:       switch(esito_menu)
2791:         {
2792:           case 1:
2793:             {
2794:               Lcd_Out(2,1, riga_vuota);
2795:               Lcd_Out(2,1, "M1  -Muovi o ESC");
2796:               while(PULS_ESC)
2797:                {
2798:                  if((ERR_MOT_1)||(ERR_MOT_2)) return(0);
2799:                  if(EMERGENZA) return(99);
2800:                  if(!PULS_UP)
2801:                    {
2802:                      esito_manuale=MOT_1_INDIETRO();
2803:                      esito_manuale=0;
2804:                      attesa=100;
2805:                      while(attesa)
2806:                        {
2807:                          delay_ms(1);
2808:                          if(PULS_UP) attesa--;
2809:                          else        attesa=100;
2810:                        }//while(attesa)
2811:                      no_comando();
2812:                    }//if(!PULS_UP)
2813:                  if(!PULS_DOWN)
2814:                    {
2815:                      esito_manuale=MOT_1_AVANTI();
2816:                      esito_manuale=0;
2817:                      attesa=100;
2818:                      while(attesa)
2819:                        {
2820:                          delay_ms(1);
2821:                          if(PULS_DOWN) attesa--;
2822:                          else        attesa=100;
2823:                        }//while(attesa)
2824:                      no_comando();
2825:                    }//if(!PULS_DOWN)
2826:                }//while(PULS_ESC)
2827:               while(!PULS_ESC){}
2828:               delay_ms(200);
2829:               break;
2830:             }
2831:           case 2:
2832:             {
2833:               Lcd_Out(2,1, riga_vuota);
2834:               Lcd_Out(2,1, "M2  -Muovi o ESC");
2835:               while(PULS_ESC)
2836:                {
2837:                  if((ERR_MOT_1)||(ERR_MOT_2)) return(0);
2838:                  if(EMERGENZA) return(99);
2839:                  if(!PULS_UP)
2840:                    {
2841:                      esito_manuale=MOT_2_INDIETRO();
2842:                      esito_manuale=0;
2843:                      attesa=100;
2844:                      while(attesa)
2845:                        {
2846:                          delay_ms(1);
```

```
2847:                         if(PULS_UP) attesa--;
2848:                         else        attesa=100;
2849:                       }//while(attesa)
2850:                     no_comando();
2851:                   }//if(!PULS_UP)
2852:                 if(!PULS_DOWN)
2853:                   {
2854:                     esito_manuale=MOT_2_AVANTI();
2855:                     esito_manuale=0;
2856:                     attesa=100;
2857:                     while(attesa)
2858:                       {
2859:                         delay_ms(1);
2860:                         if(PULS_DOWN) attesa--;
2861:                         else        attesa=100;
2862:                       }//while(attesa)
2863:                     no_comando();
2864:                   }//if(!PULS_DOWN)
2865:                }//while(PULS_ESC)
2866:               while(!PULS_ESC){}
2867:               delay_ms(200);
2868:               break;
2869:               break;
2870:             }
2871:           case 3:
2872:             {
2873:               Lcd_Out(2,1, riga_vuota);
2874:               Lcd_Out(2,1, "Pistone");
2875:               if(PISTONE)  Lcd_Out(2,9, "ON ");
2876:               else         Lcd_Out(2,9, "OFF");
2877:               if((ERR_MOT_1)||(ERR_MOT_2)) return(0);
2878:               if(EMERGENZA) return(99);
2879:               while(PULS_ESC)
2880:                {
2881:                  if(!PULS_UP)
2882:                    {
2883:                      PISTONE=1;
2884:                      Lcd_Out(2,9, "ON ");
2885:                      while(!PULS_UP){}
2886:                    }//if(!PULS_UP)
2887:                  if(!PULS_DOWN)
2888:                    {
2889:                      PISTONE=0;
2890:                      Lcd_Out(2,9, "OFF");
2891:                      while(!PULS_DOWN){}
2892:                    }//if(!PULS_DOWN)
2893:                }//while(PULS_ESC)
2894:               while(!PULS_ESC){}
2895:               delay_ms(200);
2896:               break;
2897:             }
2898:           case 4:
2899:             {
2900:               Lcd_Out(2,1, riga_vuota);
2901:               Lcd_Out(2,1, "Magnete");
2902:               if(MAGNETE)  Lcd_Out(2,9, "ON ");
2903:               else         Lcd_Out(2,9, "OFF");
2904:               if((ERR_MOT_1)||(ERR_MOT_2)) return(0);
2905:               if(EMERGENZA) return(99);
2906:               while(PULS_ESC)
2907:                {
2908:                  if(!PULS_UP)
```

```
2909:                     {
2910:                       MAGNETE=1;
2911:                       Lcd_Out(2,9, "ON ");
2912:                       while(!PULS_UP){}
2913:                     }//if(!PULS_UP)
2914:                   if(!PULS_DOWN)
2915:                     {
2916:                       MAGNETE=0;
2917:                       Lcd_Out(2,9, "OFF");
2918:                       while(!PULS_DOWN){}
2919:                     }//if(!PULS_DOWN)
2920:                  }//while(PULS_ESC)
2921:                 while(!PULS_ESC){}
2922:                 delay_ms(200);
2923:                 break;
2924:                 break;
2925:               }
2926:             case 0:
2927:               {
2928:                 return(0);
2929:                 break;
2930:               }
2931:             case 99:
2932:               {
2933:                 return(99);
2934:                 break;
2935:               }
2936:           }//switch
2937:
2938:       }//while(1)
2939:
2940: }//movimenti_manuali
2941:
2942:
2943:
2944: //*****************************************************
2945: //VISUALIZZA IL MENU' INIZIALE
2946: //RITORNA DA 1 A 8 SECONDO L'OPZIONE RICHIESTA
2947: //char txt_montani="ITT MONTANI";
2948: //char txt_azzera="AZZERA";           return(1)
2949: //char txt_errori="ERRORI";           return(2)
2950: //char txt_reset="RESET ALLARMI";     return(3)
2951: //char txt_ciclo_rapido=CICLO RAPIDO";      return(4)
2952: //char txt_ciclo_lento="CICLO_LENTO";         return(5)
2953: //char txt_manuale="MANUALE";         return(6)
2954: //char txt_manuale="OFFSET ZERO";     return(7)
2955: //char txt_manuale="VALORI";          return(8)
2956: //*****************************************************
2957: char menu_iniziale()
2958: {
2959:
2960:    char cont_down,cont_up,cont_invio,cont_esc;
2961:    char flag_cambio;
2962:    unsigned int cont_visu=15000;
2963:    char flag_emergenza,flag_zero_si,flag_zero_no,flag_errore,errori;
2964:
2965:    cont_down=cont_rimbalzo;
2966:    cont_up=cont_rimbalzo;
2967:    cont_invio=cont_rimbalzo;
2968:    cont_esc=cont_rimbalzo;
2969:
2970:    Lcd_Cmd(_LCD_CLEAR);
```

```
2971:     Lcd_Cmd(_LCD_CURSOR_OFF);
2972:     flag_emergenza=0;
2973:     flag_zero_si=0;
2974:     flag_zero_no=0;
2975:     flag_errore=0;
2976:     errori=0;
2977:     flag_cambio=1;
2978:     while(1)
2979:      {
2980:       //EMERGENZA ATTIVA
2981:       if((EMERGENZA)&&(!flag_emergenza))
2982:         {
2983:           Lcd_Out(1, 1, riga_vuota);
2984:           Lcd_Out(1, 1, txt_emergenza);
2985:           robot_azzerato=0;
2986:           flag_emergenza=1;
2987:           flag_zero_si=0;
2988:           flag_zero_no=0;
2989:           flag_errore=0;
2990:         }//if((EMERGENZA)&&(!flag_emergenza))
2991:       //EMERGENZA DISATTIVA
2992:       if(!EMERGENZA)
2993:         {
2994:           flag_emergenza=0;
2995:           //ERRORI ASSI
2996:           if((ERR_MOT_1)||(ERR_MOT_2)) errori=1;
2997:           else                         errori=0;
2998:           if((errori)&&(!flag_errore))
2999:            {
3000:              Lcd_Out(1, 1, riga_vuota);
3001:              Lcd_Out(1, 1, txt_errore);
3002:              robot_azzerato=0;
3003:              flag_errore=1;
3004:              flag_zero_si=0;
3005:              flag_zero_no=0;
3006:            }//if
3007:           //NO ERRORI
3008:           if(!errori)
3009:            {
3010:             flag_errore=0;
3011:             //ROBOT AZZERATO
3012:             if((!robot_azzerato)&&(!flag_zero_no))
3013:              {
3014:                flag_zero_no=1;
3015:                flag_zero_si=0;
3016:                Lcd_Out(1, 1, txt_stato);
3017:                Lcd_Out(1, 10, txt_nozero);
3018:               }//if
3019:             //ROBOT NON AZZERATO
3020:             if((robot_azzerato)&&(!flag_zero_si))
3021:              {
3022:                flag_zero_no=0;
3023:                flag_zero_si=1;
3024:                Lcd_Out(1, 1, txt_stato);
3025:                Lcd_Out(1, 10, txt_okzero);
3026:               }//if
3027:            }//if(!errori)
3028:         }//if((EMERGENZA)
3029:
3030:       if(!PULS_INVIO) cont_invio--;
3031:       else            cont_invio=cont_rimbalzo;
3032:       if(!PULS_UP)    cont_up--;
```

```
3033:       else            cont_up=cont_rimbalzo;
3034:       if(!PULS_DOWN)  cont_down--;
3035:       else            cont_down=cont_rimbalzo;
3036:
3037:       //pulsante invio
3038:       if(!cont_invio)
3039:        {
3040:          while(!PULS_INVIO){}//attesa rilascio
3041:          if(menu_ini_scelta==2) flag_errore=0;
3042:          return(menu_ini_scelta);
3043:        }//if(!cont_invio)
3044:
3045:       //pulsante sù
3046:       if(!cont_up)
3047:        {
3048:         if(menu_ini_scelta>1)
3049:          {
3050:           menu_ini_scelta--;
3051:          }
3052:         cont_up=cont_rimbalzo;
3053:         flag_cambio=1;
3054:        }//if(!cont_sù)
3055:
3056:       //pulsante giù
3057:       if(!cont_down)
3058:        {
3059:         if(menu_ini_scelta<11)
3060:          {
3061:           menu_ini_scelta++;
3062:          }
3063:         cont_down=cont_rimbalzo;
3064:         flag_cambio=1;
3065:        }//if(!cont_down)
3066:
3067:       if(flag_cambio==1)
3068:         {
3069:           flag_cambio=0;
3070:           Lcd_Out(2,1, riga_vuota);
3071:           switch(menu_ini_scelta)
3072:            {
3073:              case 1:
3074:                {
3075:                  Lcd_Out(2,1,"1-");
3076:                  Lcd_Out(2,3,txt_azzera);
3077:                  break;
3078:                }
3079:              case 2:
3080:                {
3081:                  Lcd_Out(2,1,"2-");
3082:                  Lcd_Out(2,3,txt_errori);
3083:                  break;
3084:                }
3085:              case 3:
3086:                {
3087:                  Lcd_Out(2,1,"3-");
3088:                  Lcd_Out(2,3,txt_reset);
3089:                  break;
3090:                }
3091:              case 4:
3092:                {
3093:                  Lcd_Out(2,1,"4-");
3094:                  Lcd_Out(2,3,txt_ciclo_rapido);
```

```
3095:                 break;
3096:               }
3097:             case 5:
3098:               {
3099:                 Lcd_Out(2,1,"5-");
3100:                 Lcd_Out(2,3,txt_ciclo_lento);
3101:                 break;
3102:               }
3103:             case 6:
3104:               {
3105:                 Lcd_Out(2,1,"6-");
3106:                 Lcd_Out(2,3,txt_manuale);
3107:                 break;
3108:               }
3109:             case 7:
3110:               {
3111:                 Lcd_Out(2,1,"7-");
3112:                 Lcd_Out(2,3,txt_offset);
3113:                 break;
3114:               }
3115:             case 8:
3116:               {
3117:                 Lcd_Out(2,1,"8-");
3118:                 Lcd_Out(2,3,txt_valori);
3119:                 break;
3120:               }
3121:             case 9:
3122:               {
3123:                 Lcd_Out(2,1,"9-");
3124:                 Lcd_Out(2,3,txt_stato_assi);
3125:                 break;
3126:               }
3127:             case 10:
3128:               {
3129:                 Lcd_Out(2,1,"10-");
3130:                 Lcd_Out(2,4,txt_riposo);
3131:                 break;
3132:               }
3133:             case 11:
3134:               {
3135:                 Lcd_Out(2,1,"11-");
3136:                 Lcd_Out(2,4,txt_90);
3137:                 break;
3138:               }
3139:           }//switch
3140:         while((!PULS_DOWN)||(!PULS_UP)){} //attesa rilascio
3141:         delay_ms(50);
3142:       }//if(flag_cambio)
3143:     }//while
3144: }//menu_iniziale
3145:
3146:
3147: //******************************************************
3148: //MAIN
3149: //******************************************************
3150: void main() {
3151:
3152:
3153:  OSCCON=0xF0;        //clock interno a 8MHz
3154:  CMCON=0x07;         //disabilito i comparatori
3155:  ADCON1=0x0F;        //disabilito i convertitori
3156:  TRISA=0xFF;
```

```
3157:  TRISB=0xC0;
3158:  TRISC=0xA0;
3159:  TRISD=0xC0;
3160:  TRISE.F0=1;
3161:  TRISE.F1=1;
3162:  TRISE.F2=1;
3163:  TRISE.F3=1;
3164:  TRISE.F4=0;         //disabilito la gestione parallela su PORTD
3165:  INTCON=0;           //disabilito gli interrupt
3166:  RCON.IPEN=0;        //disabilito la priorità sugli interrupt
3167:  EECON1.WREN=1;      //abilito la scrittura sulla eeprom
3168:  UART1_Init(9600);
3169:  Lcd_Init();
3170:  //Soft_I2C_Init();    //I2C software
3171:  Lcd_Cmd(_LCD_CLEAR);
3172:  Lcd_Cmd(_LCD_CURSOR_OFF);
3173:  robot_azzerato=0;
3174:  MAGNETE=0;
3175:  PISTONE=0;
3176:  pos_attuale_A=0;
3177:  pos_attuale_B=0;
3178:  pos_richiesta_A=0;
3179:  pos_richiesta_B=0;
3180:  NO_SEL();
3181:  menu_ini_scelta=1;
3182:  num_sequenza=1;
3183:  debug=0;
3184:  vel_ini=20;
3185:  delay_base=100;
3186:
3187:  //test comunicazione seriale
3188:  if((!PULS_DOWN)&&(!PULS_UP)) test_seriale();
3189:
3190:  //calcola angoli ed impulsi
3191:  Lcd_Out(1, 3, txt_montani);
3192:  Lcd_Out(2, 1, "ATTENDERE..");
3193:
3194:  num_elemento=1;
3195:  while(num_elemento<119)
3196:   {
3197:     esito=calcola_coord(num_elemento);
3198:     if(esito==0)
3199:      {
3200:        angoli[0][num_elemento-1]=alfa1;
3201:        angoli[1][num_elemento-1]=alfa2;
3202:        impulsi[0][num_elemento-1]=arrotonda(alfa1/risoluzione);
3203:        impulsi[1][num_elemento-1]=arrotonda(alfa2/risoluzione);
3204:      }//if
3205:     else
3206:      {
3207:        angoli[0][num_elemento-1]=0;
3208:        angoli[1][num_elemento-1]=0;
3209:        impulsi[0][num_elemento-1]=0;
3210:        impulsi[0][num_elemento-1]=0;
3211:      }//else
3212:     num_elemento++;
3213:     Lcd_Out(2, 14, "     ");
3214:     ByteToStr(num_elemento,txt_byte);
3215:     Lcd_Out(2,14,txt_byte);
3216:    }//while
3217:
3218:  vel_A=vel_ini;
```

```
3219:  vel_B=vel_ini;
3220:  vel=vel_ini;
3221:  esito=invia_vel_assi(vel_A,vel_B);
3222:  while(1)
3223:    {
3224:       esito_menu=menu_iniziale();
3225:       switch(esito_menu)
3226:         {
3227:           case 1:
3228:             {
3229:               vel_A=vel_ini;
3230:               vel_B=vel_ini;
3231:               esito=invia_vel_assi(vel_A,vel_B);
3232:               if((!EMERGENZA)&&(!esito))  esito=azzera_robot();
3233:               NO_SEL();
3234:               break;
3235:             }
3236:           case 2:
3237:             {
3238:               esito=leggi_errore();
3239:               NO_SEL();
3240:               break;
3241:             }
3242:           case 3:
3243:             {
3244:               esito=reset_errori_assi();
3245:               NO_SEL();
3246:               break;
3247:             }
3248:           case 4:
3249:             {
3250:               if(!EMERGENZA)  esito=ciclo_automatico(0);
3251:               PISTONE=0;
3252:               MAGNETE=0;
3253:               NO_SEL();
3254:               vel_A=20;
3255:               vel_B=20;
3256:               break;
3257:             }
3258:           case 5:
3259:             {
3260:               if(!EMERGENZA)  esito=ciclo_automatico(1);
3261:               PISTONE=0;
3262:               MAGNETE=0;
3263:               NO_SEL();
3264:               vel_A=20;
3265:               vel_B=20;
3266:               break;
3267:             }
3268:           case 6:
3269:             {
3270:               menu_man_scelta=1;
3271:               if(!EMERGENZA)  esito=movimenti_manuali();
3272:               PISTONE=0;
3273:               MAGNETE=0;
3274:               NO_SEL();
3275:               break;
3276:             }
3277:           case 7:
3278:             {
3279:               imposta_offset();
3280:               NO_SEL();
```

```
3281:                 break;
3282:               }
3283:             case 8:
3284:               {
3285:                 visualizza_valori();
3286:                 NO_SEL();
3287:                 break;
3288:               }
3289:             case 9:
3290:               {
3291:                 esito=visualizza_info_assi();
3292:                 NO_SEL();
3293:                 break;
3294:               }
3295:             case 10:
3296:               {
3297:                 esito=posiziona_riposo();
3298:                 NO_SEL();
3299:                 break;
3300:               }
3301:             case 11:
3302:               {
3303:                 esito=posiziona_diritto();
3304:                 NO_SEL();
3305:                 break;
3306:               }
3307:          }//switch
3308:          if(esito)
3309:           {
3310:              Lcd_Cmd(_LCD_CLEAR);
3311:              ByteToStr(esito,txt_byte);
3312:              Lcd_Out(1,1,"ERRORE COMANDO");
3313:              Lcd_Out(2,1,"CODICE=");
3314:              Lcd_Out(2,8,txt_byte);
3315:              if(debug>0)
3316:               {
3317:                 ByteToStr(debug,txt_byte);
3318:                 Lcd_Out(2,13,txt_byte);
3319:               }//if
3320:              delay_ms(200);
3321:              while(PULS_ESC){}  //attesa pulsante invio
3322:              delay_ms(200);
3323:              esito=0;
3324:              debug=0;
3325:           }//if(esito)
3326:     }//while(1)
3327: }
```