```
/*

COMANDI DA DSPIC A PIC

RF6             Chip Select


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             ----------


************************************************
PORTF
RF6     -   INPUT      CHIP SELECT
RF5     -
RF4     -
RF3     -   RS232      RX
RF2     -   RS232      TX
RF1     -   OUTPUT     BUSY DSPIC
RF0     -   OUTPUT     ERRORE DSPIC

PORTE
RE8     -
RE5     -
RE4     -    INPUT     ALLARME
RE3     -    OUTPUT    RESET
RE2     -    OUTPUT    EN
RE1     -    OUTPUT    DIR
RE0     -    OUTPUT    CLK

PORTD
RD3     -   INPUT      Comando  da PIC
RD2     -   INPUT      Comando  da PIC
RD1     -   INPUT      Comando  da PIC
RD0     -   INPUT      Comando  da PIC

PORTC
RC15    -
RC14    -
RC13    -

PORTB
```

```
 63: RB8      -   INPUT      STATO ALTRO ASSE
 64: RB7      -   OUTPUT     IN MOTO
 65: RB6      -   INPUT      EMERGENZA
 66: RB5      -   INPUT      FASE B ENCODER
 67: RB4      -   INPUT      FASE A ENCODER
 68: RB3      -   INPUT      ZERO ENCODER
 69: RB2      -   INPUT
 70: RB1      -   INPUT
 71: RB0      -   INPUT      ZERO ASSE
 72:
 73: STATO_ASSE
 74: BIT7       -
 75: BIT6       -
 76: BIT5       -
 77: BIT4       -
 78: BIT3       -
 79: BIT2       -
 80: BIT1       -  1=ERRORE
 81: BIT0       -  1=AZZERATO
 82:
 83:
 84: STRUTTURA EEPROM INTERNA
 85: EEPROM  address iniziale 0x7FFC00
 86: 0      - offset low
 87: 2      - offset high
 88:
 89:
 90: */
 91:
 92:
 93: #include <built_in.h>
 94:
 95: sbit   ALTRO_ASSE    at RB8_bit;
 96: sbit   IN_MOTO       at RB7_bit;
 97: sbit   EMERGENZA     at RB6_bit;
 98: sbit   IN_2          at RB2_bit;
 99: sbit   IN_1          at RB1_bit;
100: sbit   FC_ZERO       at RB0_bit;
101:
102: sbit   COM4     at RD3_bit;
103: sbit   COM3     at RD2_bit;
104: sbit   COM2     at RD0_bit;
105: sbit   COM1     at RD1_bit;
106:
107: sbit   ALLARME    at RE4_bit;
108: sbit   RESET      at RE3_bit;
109: sbit   ENABLE     at RE2_bit;
110: sbit   DIREZIONE  at RE1_bit;
111: sbit   CLOCK      at RE0_bit;
112:
113: sbit   CHIP_SELECT   at RF6_bit;
114: sbit   RX_232        at RF3_bit;
115: sbit   TX_232        at RF2_bit;
116: sbit   BUSY          at LATF1_bit;
117: sbit   ERROR         at LATF0_bit;
118:
119: //stato_asse
120: char stato_asse;
121: sbit   ZERO_OK       at stato_asse.B0;
122:
123:
124:
```

```
125: unsigned int passi_rampa=540;
126: //cont_pos=96  0,8 secondi a ciclo  8kHz
127: unsigned short const rampa_passi [96] = {
128: 1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,
129: 4,4,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,7,7,7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,
130: 10,10,10,10,10,10,10,11,11,11,11,11,11,11,11,11,12};
131: //dt=4us
132: unsigned short const rampa_time [96] = {
133: 250,216,189,169,152,139,128,118,110,102,96,91,86,81,77,74,70,67,64,62,60,57,55,
134: 53,52,50,48,47,46,44,43,42,41,40,39,38,37,36,35,35,34,33,32,32,31,30,30,29,29,
135: 28,28,27,27,26,26,26,25,25,24,24,24,23,23,23,22,22,22,21,21,21,20,20,20,20,19,19,
136: 19,19,19,18,18,18,18,18,17,17,17,17,17,16,16,16,16,16,16,15};
137:
138: //angolo impulso motore
139: const float risoluzione=0.018;
140: //GESTIONE SERIALE
141: char dato_rx[5];
142: char attiva_seriale,disattiva_seriale;
143: //VARIABILI PER GESTIONE MOTORE
144: int passi_totali,conteggio_passi,passi_cost;
145: int cont_pos,pos_attuale,pos_richiesta,offset_zero;
146: unsigned short cont_vel;
147: char temp_rampa,cont_del;
148: char flag_allarme,esito,esito_movimento,codice_errore;
149: unsigned long int address_eeprom;
150: int delay_SS;
151: int passi_offset;
152:
153:
154:
155: //------------------------------------------------------
156: //DIREZIONE AVANTI
157: void avanti()
158:  {
159:    DIREZIONE=0;
160:  }//avanti
161:
162: //DIREZIONE INDIETRO
163: void indietro()
164:  {
165:    DIREZIONE=1;
166:  }//indietro
167:
168: //------------------------------------------------------
169: //ABILITA_ASSE
170: void abilita_asse()
171:  {
172:    ENABLE=1;
173:  }//abilita_asse
174:
175: //DISABILITA_ASSE
176: void disabilita_asse()
177:  {
178:    ENABLE=0;
179:  }//disabilita_asse
180:
181:
182: //------------------------------------------------------
183: //CALCOLA PASSI PER RAMPE IN BASE ALLA VELOCITA' IMPOSTATA
184: void calcola_passi_rampa()
185: {
```

```
186:    char cont_1;
187:
188:    passi_rampa=0;
189:    cont_1=0;
190:    while(cont_1<cont_vel)
191:     {
192:       passi_rampa+=rampa_passi[cont_1];
193:       cont_1++;
194:     }//while(cont_vel<vel_imp)
195:    passi_rampa+=20;    //incremento i passi di 20
196:
197: }//calcola_passi_rampa
198:
199: //-------------------------------------------------------
200: //AZZERAMENTO ASSE
201: //torna 0 se ok 99 se EMERGENZA 1 se manca zero
202: char azzera_asse()
203: {
204:   int cont_zero;
205:   int imp_uscita,imp_entrata;
206:   float ang_uscita,ang_entrata;
207:
208:
209:   ang_uscita=5.0;
210:   ang_entrata=120.0;
211:   imp_uscita=ang_uscita/risoluzione;
212:   imp_entrata=ang_entrata/risoluzione;
213:   abilita_asse();
214:   IN_MOTO=1;
215:   if(FC_ZERO)
216:    {
217:     //esco dal finecorsa
218:     avanti();
219:     cont_zero=imp_uscita;
220:     while(cont_zero)
221:      {
222:        CLOCK=1;
223:        delay_ms(1);
224:        CLOCK=0;
225:        delay_ms(1);
226:        cont_zero--;
227:        if(!FC_ZERO) cont_zero=0;
228:        if(EMERGENZA)   {IN_MOTO=0;return(99);}  //emergenza
229:        if(ALLARME)    {IN_MOTO=0;return(1);}
230:      }//whie(cont_zero)
231:     cont_zero=imp_uscita;
232:     while(cont_zero)
233:      {
234:        CLOCK=1;
235:        delay_mS(1);
236:        CLOCK=0;
237:        delay_mS(1);
238:        cont_zero--;
239:        if(EMERGENZA)   {IN_MOTO=0;return(99);}  //emergenza
240:        if(ALLARME)    {IN_MOTO=0;return(1);}
241:      }//whie(cont_zero)
242:     }//if(FC_ZERO)
243:   indietro();
244:   cont_zero=imp_entrata;
245:   while(cont_zero)
246:    {
247:      CLOCK=1;
```

```
248:      delay_ms(1);
249:      CLOCK=0;
250:      delay_ms(1);
251:      cont_zero--;
252:      if(!cont_zero) {IN_MOTO=0;return(1);}
253:      if(FC_ZERO)
254:        {
255:          pos_attuale=offset_zero;
256:          IN_MOTO=0;
257:          return(0);
258:        }//if(FC_ZERO)
259:      if(EMERGENZA)   {IN_MOTO=0;return(99);}  //emergenza
260:      if(ALLARME)    {IN_MOTO=0;return(1);}
261:    }//whie(cont_zero)
262: }//char azzera_asse()
263:
264: //-------------------------------------------------------
265: //RITARDO VARIABILE IN MICRO SEC
266: void delay_micro(int valore)
267: {
268:   while(valore)
269:    {
270:      valore--;
271:      delay_us(1);      //1
272:    }//while(valore)
273: }//delay_micro(int valore)
274:
275:
276: //-------------------------------------------------------
277: //ESEGUO IL POSIZIONAMENTO IN START/STOP
278: //torna 0 se ok 99 se emergenza
279: char posiziona_SS()
280: {
281:   char flag_posiziona;
282:   int valore;
283:
284:   flag_posiziona=1;
285:   if((flag_posiziona)&&(pos_richiesta>pos_attuale))
286:    {
287:      passi_totali=pos_richiesta-pos_attuale;
288:      valore=1;
289:      avanti();
290:      flag_posiziona=0;
291:    }//if
292:   if((flag_posiziona)&&(pos_richiesta<pos_attuale))
293:    {
294:      passi_totali=pos_attuale-pos_richiesta;
295:      valore=-1;
296:      indietro();
297:      flag_posiziona=0;
298:    }//if
299:   if((flag_posiziona)&&(pos_richiesta==pos_attuale))
300:     {
301:       return(0);
302:     }//if
303:
304:    if(delay_SS<=0) delay_SS=1000;
305:    flag_allarme=0;
306:    IN_MOTO=1;
307:    conteggio_passi=0;
308:    while(conteggio_passi<passi_totali)
309:     {
```

```
310:      CLOCK=1;
311:      delay_micro(delay_SS);
312:      CLOCK=0;
313:      delay_micro(delay_SS);
314:      conteggio_passi++;
315:      pos_attuale=pos_attuale+valore;
316:      if(EMERGENZA)    {IN_MOTO=0;return(99);}  //emergenza
317:      if(ALLARME)      {IN_MOTO=0;return(1);}
318:     }//while(conteggio_passi<passi_totali)
319:    IN_MOTO=0;
320:    return(0);
321: }//posiziona_SS
322:
323: //-----------------------------------------------------
324: //ESEGUO IL POSIZIONAMENTO
325: //torna 0 se ok 99 se emergenza
326: char posiziona_asse()
327: {
328:   char flag_posiziona;
329:   int valore;
330:
331:   calcola_passi_rampa();
332:   flag_posiziona=1;
333:   if((flag_posiziona)&&(pos_richiesta>pos_attuale))
334:    {
335:      passi_totali=pos_richiesta-pos_attuale;
336:      valore=1;
337:      avanti();
338:      flag_posiziona=0;
339:    }//if
340:   if((flag_posiziona)&&(pos_richiesta<pos_attuale))
341:    {
342:      passi_totali=pos_attuale-pos_richiesta;
343:      valore=-1;
344:      indietro();
345:      flag_posiziona=0;
346:    }//if
347:   if((flag_posiziona)&&(pos_richiesta==pos_attuale))
348:     {
349:       return(0);
350:     }//if
351:
352:    flag_allarme=0;
353:    IN_MOTO=1;
354:    //CONTROLLO SE SONO SUFFICIENTI I PASSI PER LA RAMPA
355:    //in caso non fossero sufficienti faccio la misurazione in START/STOP
356:    if(passi_totali<=(2*passi_rampa))
357:     {
358:       conteggio_passi=0;
359:       while(conteggio_passi<passi_totali)
360:        {
361:         CLOCK=1;
362:         delay_us(1000);
363:         CLOCK=0;
364:         delay_us(1000);
365:         conteggio_passi++;
366:         pos_attuale=pos_attuale+valore;
367:        }//while(conteggio_passi<passi_totali)
368:       IN_MOTO=0;
369:       return(0);
370:     }//if(passi_totali<=(2*passi_rampa))
371:
```

```
372:    //ESEGUO LA MISURAZIONE
373:    //rampa di salita
374:    cont_pos=0;
375:    conteggio_passi=0;
376:    while(cont_pos<cont_vel)
377:     {
378:       temp_rampa=rampa_passi[cont_pos];
379:       while(temp_rampa)
380:        {
381:          CLOCK=1;
382:          conteggio_passi++;
383:          pos_attuale=pos_attuale+valore;
384:          cont_del=rampa_time[cont_pos];
385:          while(cont_del)
386:           {
387:             Delay_us(5);//10
388:             cont_del--;
389:           }//while
390:          CLOCK=0;
391:          cont_del=rampa_time[cont_pos];
392:          while(cont_del)
393:           {
394:             Delay_us(5);//10
395:             cont_del--;
396:           }//while
397:          temp_rampa--;
398:        }//while
399:       cont_pos++;
400:       if(EMERGENZA)   {IN_MOTO=0;return(99);}  //emergenza
401:       if(ALLARME)    {IN_MOTO=0;return(1);}
402:     }//while
403:    cont_pos--;
404:    passi_cost=passi_totali-2*conteggio_passi;
405:    passi_cost++;
406:    //segmento a velocità fissa
407:    while(passi_cost)
408:      {
409:       CLOCK=1;
410:       conteggio_passi++;
411:       pos_attuale=pos_attuale+valore;
412:       cont_del=rampa_time[cont_pos];
413:       while(cont_del)
414:        {
415:          Delay_us(5);//10
416:          cont_del--;
417:        }//while
418:       CLOCK=0;
419:       cont_del=rampa_time[cont_pos];
420:       while(cont_del)
421:        {
422:          Delay_us(5);//10
423:          cont_del--;
424:        }//while
425:       passi_cost--;
426:       if(EMERGENZA)   {IN_MOTO=0;return(99);}  //emergenza
427:       if(ALLARME)    {passi_cost=0;IN_MOTO=0;flag_allarme=1;}
428:      }//while
429:    //rampa di discesa
430:    while(cont_pos)
431:     {
432:       temp_rampa=rampa_passi[cont_pos];
433:       while(temp_rampa)
```

```
434:        {
435:          CLOCK=1;
436:          conteggio_passi++;
437:          pos_attuale=pos_attuale+valore;
438:          cont_del=rampa_time[cont_pos];
439:          while(cont_del)
440:           {
441:             Delay_us(5);//10
442:             cont_del--;
443:           }//while
444:          CLOCK=0;
445:          cont_del=rampa_time[cont_pos];
446:          while(cont_del)
447:           {
448:             Delay_us(5);//10
449:             cont_del--;
450:           }//while
451:          temp_rampa--;
452:        }//while
453:       cont_pos--;
454:       if(EMERGENZA)   {IN_MOTO=0;return(99);}  //emergenza
455:      }//while
456:     if(flag_allarme) {IN_MOTO=0;return(1);}
457:     IN_MOTO=0;
458:     return(0);
459: }//posiziona_asse
460:
461: //*****************************************************
462: //INVIA SINGOLO BYTE
463: //*****************************************************
464: void invia_byte(char valore)
465: {
466:   char cont_bit;
467:
468:   //INVIO BYTE
469:   cont_bit=8;
470:   TX_232=0;   //start bit
471:   Delay_us(104);
472:   while(cont_bit)
473:     {
474:       if(valore.F0)     TX_232=1;
475:       else              TX_232=0;
476:       Delay_us(104);
477:       valore=valore>>1;
478:       cont_bit--;
479:     }//while(cont_bit)
480:   TX_232=1;
481:   delay_us(104);
482: }//invia_byte(char valore)
483:
484: //*****************************************************
485: //LEGGI SINGOLO BYTE
486: //*****************************************************
487: char leggi_byte()
488:  {
489:   char cont_bit,dato_leggi;
490:
491:   cont_bit=8;
492:   //attesa start bit
493:   Delay_us(100);     //attesa start bit
494:   //ricezione byte
495:   Delay_us(50);      //attesa metà bit
```

```
496:   dato_leggi=0;
497:   while(cont_bit)
498:    {
499:      dato_leggi=dato_leggi>>1;
500:      if(RX_232)   {dato_leggi.F7=1;}
501:      else         {dato_leggi.F7=0;}
502:      Delay_us(100);
503:      cont_bit--;
504:    }//while(cont_bit)
505:   Delay_us(150);  //stop bit + mezzo bit
506:   return(dato_leggi);
507: }//leggi_byte()
508:
509: //******************************************************
510: //INVIA VALORE INTERO XX#
511: //torna 1 se ERRORE tramissione
512: //******************************************************
513: char invia_valore(int valore)
514: {
515:   char valore_byte[2];
516:   char cont_char,dato_invia,cont_bit;
517:   int cont_time;
518:
519:   valore_byte[0]=hi(valore);
520:   valore_byte[1]=lo(valore);
521:
522:   cont_char=0;
523:   while(cont_char<2)
524:    {
525:     invia_byte(valore_byte[cont_char]);
526:     cont_char++;
527:     cont_time=30000;
528:     while(RX_232)
529:      {
530:        delay_us(1);
531:        cont_time--;
532:        if(!cont_time) return(1);
533:      }//while(cont_time)
534:     dato_invia=leggi_byte();
535:     if(dato_invia=!'+') return(1);
536:    }//while(cont_char<2)
537:   return(0);
538: }//invia_valore(int_valore)
539:
540:
541:
542: //******************************************************
543: //LETTURA COMANDO DA SERIALE
544: //COMUNICAZIONE PIC-DSPIC
545: //VX#         -      INVIO VELOCITA' X
546: //DXX#        -      INVIO DELAY IN uSEC PER POSIZIONAMENTO IN START STOP
547: //PXX#        -      INVIO POSIZIONE ASSOLUTA IN IMPULSI
548: //A#          -      LEGGI IL NUMERO ASSE
549: //C#          -      LEGGI STATO ASSE
550: //S#          -      LEGGI CODICE ERRORE
551: //s#          -      RESETTA ERRORE
552: //OXX#        -      INVIA OFFSET IN IMPULSI
553: //o#          -      LEGGI OFFSET IN IMPULSI
554: //v#          -      LEGGI LA VELOCITA' IMPOSTATA
555: //d#          -      LEGGI I DELAY IMPOSTATO
556: //p#          -      LEGGI LA POSIZIONE IMPOSTATA
557: //L#          -      LEGGI POSIZIONE ASSOLUTA IN IMPULSI
```

```
558: //M1XX#      -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP1
559: //M2XX#      -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP2
560: //M3XX#      -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP3
561: //M4XX#      -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP4
562: //M5XX#      -      INVIO POSIZIONAMENTO ASSOLUTO MULTIPLO IN IMPULSI STEP5
563: //*****************************************************
564: char lettura_seriale()
565:  {
566:   char cont_char,cont_bit;
567:   unsigned int cont_time;
568:
569:   //attesa risposta
570:   cont_char=0;
571:   while(cont_char<5)
572:    {
573:     dato_rx[cont_char]=leggi_byte();
574:     delay_us(100);
575:     invia_byte(dato_rx[cont_char]); //invio eco carattere ricevuto
576:     if(dato_rx[cont_char]=='#') return(0);
577:     cont_char++;
578:     cont_time=30000;
579:     while(RX_232)
580:      {
581:        delay_us(1);
582:        cont_time--;
583:        if(!cont_time) return(1);
584:      }//while(cont_time)
585:    }//while(cont_char<5)
586:   return(1);
587:  }//lettura_seriale
588:
589:
590:
591:
592: void main() {
593:   ADPCFG=0xFFFF;       //inizializza i pin di PORTB come digitali
594:   C1CTRL=0x3200;       //disabilitare il CANBUS
595:   TRISB=0xFF7F;
596:   TRISD=0xFFFF;
597:   TRISE=0xFFF0;
598:   TRISF=0xFFFC;
599:  // QEICON=0x0740;       //moltiplicatore per 4 sull'ingresso encoder
600:  // DFLTCON=0x1D0;       //filtro attivo sulla lettura encoder 1/64 Clock
601:   INTCON1=0x0000;
602:   INTCON2=0x0000;
603:   IFS0=0x0000;
604:   IFS1=0x0000;
605:   IEC0=0x0200;
606:   IEC1=0x0000;
607:   address_eeprom=0x7FFC00;
608:
609:   ERROR=0;
610:   BUSY=0;
611:   codice_errore=0;
612:   stato_asse=0;
613:   esito=0;
614:   esito_movimento=0;
615:   disattiva_seriale=0;
616:   attiva_seriale=1;
617:   abilita_asse();
618:   cont_vel=20;
619:
```

```
620:   offset_zero=EEPROM_Read(address_eeprom+2);
621:   offset_zero=offset_zero<<8;
622:   offset_zero=offset_zero|EEPROM_Read(address_eeprom+0);
623:
624:
625:   while(1)
626:    {
627:      //CONTROLLA SE ATTIVARE LA SERIALE
628:      if((CHIP_SELECT)&&(attiva_seriale))
629:        {
630:          attiva_seriale=0;
631:          disattiva_seriale=1;
632:          TRISF.F2=0;
633:          TX_232=1;
634:        }//if((CHIP_SELECT)&&(attiva_seriale))
635:      if((!CHIP_SELECT)&&(disattiva_seriale))
636:        {
637:          attiva_seriale=1;
638:          disattiva_seriale=0;
639:          TRISF.F2=1;
640:        }//if((!CHIP_SELECT)&&(diattiva_seriale))
641:
642:      //CONTROLLA SE C'E' UN DATO SULLA SERIALE
643:      if(CHIP_SELECT)
644:       {
645:         esito=99;
646:         if((!RX_232)&&(!BUSY))   esito=lettura_seriale();
647:         if(!esito)
648:          {
649:            switch(dato_rx[0])
650:             {
651:               case 'A':
652:                 {
653:                   invia_byte(1);
654:                   break;
655:                 }
656:               case 'C':
657:                 {
658:                   invia_byte(stato_asse);
659:                   break;
660:                 }
661:               case 'L':
662:                 {
663:                   invia_valore(pos_attuale);
664:                   break;
665:                 }
666:               case 'S':
667:                 {
668:                   invia_byte(codice_errore);
669:                   break;
670:                 }
671:               case 's':
672:                 {
673:                   codice_errore=0;
674:                   ERROR=0;
675:                   break;
676:                 }
677:               case 'p':
678:                 {
679:                   invia_valore(pos_richiesta);
680:                   break;
681:                 }
```

```
682:              case 'v':
683:                {
684:                  invia_byte(cont_vel);
685:                  break;
686:                }
687:              case 'V':
688:                {
689:                  cont_vel=dato_rx[1];
690:                  break;
691:                }
692:              case 'D':
693:                {
694:                  delay_SS=dato_rx[1];
695:                  delay_SS=delay_SS<<8;
696:                  delay_SS=delay_SS|dato_rx[2];
697:                  break;
698:                }
699:              case 'P':
700:                {
701:                  pos_richiesta=dato_rx[1];
702:                  pos_richiesta=pos_richiesta<<8;
703:                  pos_richiesta=pos_richiesta|dato_rx[2];
704:                  break;
705:                }
706:              case 'O':
707:                {
708:                  EEPROM_Write((address_eeprom+0), dato_rx[2]);
709:                  while(WR_bit);
710:                  EEPROM_Write((address_eeprom+2), dato_rx[1]);
711:                  while(WR_bit);;
712:                  offset_zero=dato_rx[1];
713:                  offset_zero=offset_zero<<8;
714:                  offset_zero=offset_zero|dato_rx[2];
715:                  break;
716:                }
717:              case 'o':
718:                {
719:                  invia_valore(offset_zero);
720:                  break;
721:                }
722:              case 'd':
723:                {
724:                  invia_valore(delay_SS);
725:                  break;
726:                }
727:              case 'M':
728:                {
729:
730:                  break;
731:                }
732:            }//switch
733:        }//if(!esito)
734:       }//if(CHIP_SELECT)
735:
736:      //*****************************
737:      //CONTROLLO COMANDI
738:      //MOVIMENTI SINGOLO CONTROLLO CS
739:      //*****************************
740:      //Avvia posizionamento in start stop
741:      if((COM4==0)&&(COM3==0)&&(COM2==0)&&(COM1==1))
742:        {
743:          if(CHIP_SELECT)
```

```
744:            {
745:             BUSY=1;
746:             delay_ms(50);
747:             esito_movimento=posiziona_SS();
748:            }//if
749:           BUSY=0;
750:         }//if
751:       //*****************************
752:       //Avvia posizionamento singolo con rampa
753:       if((COM4==0)&&(COM3==0)&&(COM2==1)&&(COM1==0))
754:         {
755:           if(CHIP_SELECT)
756:             {
757:               BUSY=1;
758:               delay_ms(50);
759:               esito_movimento=posiziona_asse();
760:             }//if
761:           BUSY=0;
762:         }//if
763:       //*****************************
764:       //Avvia posizionamento multiplo con rampa
765:       if((COM4==0)&&(COM3==0)&&(COM2==1)&&(COM1==1))
766:         {
767:           if(CHIP_SELECT)
768:            {
769:              BUSY=1;
770:              delay_ms(100);
771:              BUSY=0;
772:            }//if
773:         }//if
774:       //*****************************
775:       //Muovi in meno
776:       if((COM4==0)&&(COM3==1)&&(COM2==0)&&(COM1==0))
777:         {
778:           if(CHIP_SELECT)
779:            {
780:              BUSY=1;
781:              abilita_asse();
782:              indietro();
783:              IN_MOTO=1;
784:              while((COM4==0)&&(COM3==1)&&(COM2==0)&&(COM1==0))
785:               {
786:                 CLOCK=1;
787:                 delay_ms(2);
788:                 CLOCK=0;
789:                 delay_ms(2);
790:                 pos_attuale=pos_attuale-1;
791:               }//while
792:              delay_ms(100);
793:              IN_MOTO=0;
794:              BUSY=0;
795:            }//if
796:         }//if
797:       //*****************************
798:       //Muovi in piu
799:       if((COM4==0)&&(COM3==1)&&(COM2==0)&&(COM1==1))
800:         {
801:           if(CHIP_SELECT)
802:            {
803:              BUSY=1;
804:              abilita_asse();
805:              avanti();
```

```
806:              IN_MOTO=1;
807:              while((COM4==0)&&(COM3==1)&&(COM2==0)&&(COM1==1))
808:               {
809:                 CLOCK=1;
810:                 delay_ms(2);
811:                 CLOCK=0;
812:                 delay_ms(2);
813:                 pos_attuale=pos_attuale+1;
814:               }//while
815:              delay_ms(100);
816:              IN_MOTO=0;
817:              BUSY=0;
818:           }//if
819:         }//if
820:       //*****************************
821:       //Azzeramento
822:       if((COM4==0)&&(COM3==1)&&(COM2==1)&&(COM1==0))
823:         {
824:           if(CHIP_SELECT)
825:            {
826:              BUSY=1;
827:              ZERO_OK=0;
828:              delay_ms(50);
829:              esito_movimento=azzera_asse();
830:              if(!esito_movimento)
831:               {
832:                 ZERO_OK=1;
833:                 delay_ms(200);
834:                 pos_richiesta=0;
835:                 esito_movimento=posiziona_asse();
836:                 if(!esito_movimento)   {pos_attuale=0;ZERO_OK=1;}
837:               }//if(!esito_movimento)
838:              else
839:               {
840:                 avanti();
841:                 passi_offset=50;
842:                 while(passi_offset)
843:                   {
844:                      CLOCK=1;
845:                      delay_ms(1);
846:                      CLOCK=0;
847:                      delay_ms(1);
848:                      passi_offset--;
849:                   }
850:               }
851:            }//if
852:           BUSY=0;
853:         }//if
854:       //*****************************
855:       //non utilizzato
856:       if((COM4==0)&&(COM3==1)&&(COM2==1)&&(COM1==1))
857:         {
858:           if(CHIP_SELECT)
859:            {
860:              BUSY=1;
861:              delay_ms(100);
862:              BUSY=0;
863:            }//if
864:         }//if
865:       //*****************************
866:       //*****************************
867:       //MOVIMENTI SIMULANETO NO CONTROLLO CS
```

```
868:      //Avvia posizionamento in start stop
869:      if((COM4==1)&&(COM3==0)&&(COM2==0)&&(COM1==0))
870:        {
871:          BUSY=1;
872:          delay_ms(50);
873:          esito_movimento=posiziona_SS();
874:          BUSY=0;
875:        }//if
876:      //*****************************
877:      //Avvia posizionamento singolo con rampa
878:      if((COM4==1)&&(COM3==0)&&(COM2==0)&&(COM1==1))
879:        {
880:          BUSY=1;
881:          delay_ms(50);
882:          esito_movimento=posiziona_asse();
883:          BUSY=0;
884:        }//if
885:      //*****************************
886:      //Avvia posizionamento multiplo con rampa
887:      if((COM4==1)&&(COM3==0)&&(COM2==1)&&(COM1==0))
888:        {
889:          BUSY=1;
890:          delay_ms(100);
891:          BUSY=0;
892:        }//if
893:      //*****************************
894:      //Azzeramento
895:      if((COM4==1)&&(COM3==0)&&(COM2==1)&&(COM1==1))
896:        {
897:          BUSY=1;
898:          ZERO_OK=0;
899:          delay_ms(50);
900:          esito_movimento=azzera_asse();
901:          if(!esito_movimento)
902:           {
903:             ZERO_OK=1;
904:             pos_richiesta=0;
905:             esito_movimento=posiziona_asse();
906:             if(!esito_movimento)   {pos_attuale=0;ZERO_OK=1;}
907:           }//if(!esito_movimento)
908:          BUSY=0;
909:        }//if
910:      //*****************************
911:      //non utilizzato
912:      if((COM4==1)&&(COM3==1)&&(COM2==0)&&(COM1==0))
913:        {
914:          BUSY=1;
915:          delay_ms(100);
916:          BUSY=0;
917:        }//if
918:      //*****************************
919:      //non utilizzato
920:      if((COM4==1)&&(COM3==1)&&(COM2==0)&&(COM1==1))
921:        {
922:          BUSY=1;
923:          delay_ms(100);
924:          BUSY=0;
925:        }//if
926:      //*****************************
927:      //non utilizzato
928:      if((COM4==1)&&(COM3==1)&&(COM2==1)&&(COM1==0))
929:        {
```

```
930:           BUSY=1;
931:           delay_ms(100);
932:           BUSY=0;
933:         }//if
934:       //*****************************
935:       //non utilizzato
936:       if((COM4==1)&&(COM3==1)&&(COM2==1)&&(COM1==1))
937:         {
938:           BUSY=1;
939:           delay_ms(100);
940:           BUSY=0;
941:         }//if
942:       //*****************************
943:       //controllo errori
944:       if(esito_movimento)
945:         {
946:           codice_errore=esito_movimento;
947:           ERROR=1;
948:           esito_movimento=0;
949:         }//if(esito_movimento)
950:
951:     }//while(1)
952:
953: }//main
```