```
1
2  /*********************************************/
3  /*                                           */
4  /* routines this module enk_12 - 24/3/13     */
5  /* Release version                           */
6  /* Modified routine polldla and dla9600 value*/
7  /*                                           */
8  /* 0) void setup                             */
9  /* 1) void loop                              */
10 /* 2) void pollenk - poll encoder            */
11 /* 3) int makbcd - convert int to BCD ascii str */
12 /* 4) int makhex - convert long to Hex ascii str*/
13 /* 5) void blowc - tx a char from second serial */
14 /* 6) void polldla - waits 9600 baud bit delay  */
15 /*                                           */
16 /*********************************************/
17
18 /*********************************************/
19 /* program enk_6                             */
20 /*                                           */
21 /* program reads encoder outputs             */
22 /*                                           */
23 /* direct Port i/o                           */
24 /* PORTB bits to arduino pin #:              */
25 /*  0 1  2  3  4  5  6 7 BIT#                */
26 /*  8 9 10 11 12 13 (xtl6,xtl7) -ino dig pin#  */
27 /*                                           */
28 /* PORTB masks:                              */
29 /* Pin 8 = 0x01                              */
30 /* Pin 9 = 0x02                              */
31 /* Pin10 = 0x04                              */
32 /* Pin11 = 0x08                              */
33 /* Pin12 = 0x10                              */
34 /* Pin13 = 0x20                              */
35 /*                                           */
36 /* PORTD maps to arduino dig pins 0 to 7     */
37 /* PORTD masks:                              */
38 /* Pin7 = 0x80                               */
39 /*                                           */
40 /*********************************************/
41
42 /* define digital input channels */
43 int chan1=9;
44 int getch1 = 0x02;
45 int chan2=8;
46 int getch2 = 0x01;
47
48 /* read channel 1/2 i/p patterns */
49 int ic1;
50 int ic2;
51
52 /* saved channel states */
53 int ch1state;
54 int ch2state;
55
56 /* chan 1/2 debounce counters */
57 int ic1up = 0;
58 int ic1dn = 0;
59 int ic2up = 0;
60 int ic2dn = 0;
61 int icdbnce = 0;
62
63 /* define digital o/p ports */
64 int clkout = 13; /* clock */
65 int clkpin = 1;
66 int setclk = 0x20;
67 int clrclk = 0xff-setclk;
68
69 int ch1out = 12; /*chan1 tick */
70 int setch1 = 0x10;
71 int clrch1 = 0xff-setch1;
72
73 int ch2out = 11; /*chan2 tick */
```



```
74 int setch2= 0x08;
75 int clrch2=0xff-setch2;
76
77 int dirout = 10; /*direction F(hi)/R(lo) */
78 int setdir=0x04;
79 int clrdir=0xff-setdir;
80 int dir1;
81 int dir2;
82 int dirf;
83 int cnt1;
84 int cnt2;
85
86 long tcount=0; /* tick count */
87
88 /* define test counts for dir change */
89 int fwdc=0;
90 int revc=0;
91 int dirlim=20;
92
93 /* char buffers for serial text */
94 char sout[10],tout[13];
95
96 /* 9600 baud timing delays */
97 int dla9600 = 95; /* 9600 bit time = 104us delay */
98 int t1_9600 = 0;  /* # of polling cycles in delay */
99 int t2_9600 = 0; /* remainder us in delay */
100
101 /* define digital serial out port */
102 int serout = 7;
103 int setser=0x80;
104 int clrser=0xff-0x80;
105
106 /*********************************************/
107 /*                                           */
108 /*********************************************/
109
110 void setup()
111 {
112 /*********************************************/
113 /*                                           */
114 /* setup - initialize analogue pins as input etc*/
115 /*                                           */
116 /*********************************************/
117
118     /* define autos */
119     unsigned long mus, mus2,diftl;
120     int difti;
121
122     pinMode(chan1, INPUT);
123     pinMode(chan2, INPUT);
124
125     pinMode(clkout,OUTPUT);
126     digitalWrite(clkout,HIGH);
127
128     pinMode(ch1out,OUTPUT);
129     digitalWrite(ch1out,HIGH);
130     ch1state=1;
131
132     pinMode(ch2out,OUTPUT);
133     digitalWrite(ch2out,HIGH);
134     ch2state=1;
135
136     pinMode(dirout,OUTPUT);
137     digitalWrite(dirout,HIGH);
138     dir1=1;
139     dir2=1;
140     dirf=1;
141
142     /* set up serial out */
143     pinMode(serout,OUTPUT);
144     digitalWrite(serout,HIGH);
145
146     /* now time ticker polling loop */
```

```
147     mus = micros();
148     pollenk();
149     mus2=micros();
150     diftl=mus2-mus;
151
152     /* check clock did not roll over */
153     if(diftl>1000) /* polling loop ca. 25us */
154     {
155         mus = micros();
156         pollenk();
157         mus2=micros();
158         diftl=mus2-mus;
159     } /* end redo pollenk timing */
160
161     /* now see how long serial delay time is */
162     difti=int(diftl);
163     t1_9600=int(dla9600/difti); /* number of polling cycles in the delay */
164     t2_9600 = dla9600 % difti; /* # of remaining us */
165
166     /* debug - set COMx serial */
167     /*Serial.begin(9600);*/
168
169     /* wait a bit */
170     /*delay(1000);*/
171 }
172
173 /***********************************************/
174 /*                                             */
175 /***********************************************/
176
177 void loop()
178 {
179 /***********************************************/
180 /*                                             */
181 /* loop - loop reading encoder channels        */
182 /*                                             */
183 /***********************************************/
184
185     /* declare autos */
186     int slen,i,j;
187     char *s;
188
189     for (j=1;;)
190     {
191         /* poll encoder */
192         pollenk();
193
194         /* write out count every 6msec or so */
195         if(++j > 500)
196         {
197             j=0;
198
199             /* now convert the count to a string */
200             /*slen=makbcd();*/
201             slen=makhex();
202
203             /* now write tick count chars */
204             for (s = tout, i=0; i < slen; i++, s++)
205             {
206                 /* write char to serial */
207                 /* *s=0x55; /* debug bit pattern - 0101 0101 (lsb first) */
208                 blowc(*s);
209
210                 /* debug - write to COMx serial */
211                 /*Serial.write(*s);*/
212
213             } /* end write data */
214
215             /* terminate COMx serial */
216             *s='\n';
217             /*Serial.write(*s);*/
218
219         } /* end if a write output cycle */
```



```
220
221     } /* end loop forever */
222
223 } /* end main */
224
225 /***********************************************/
226 /*                                             */
227 /***********************************************/
228
229 void pollenk()
230 {
231
232 /***********************************************/
233 /*                                             */
234 /* routine reads encoder outputs               */
235 /*                                             */
236 /* direct Port i/o                             */
237 /* PORTB bits to arduino pin #:                */
238 /*  0 1  2  3  4  5  6 7 BIT#                  */
239 /*  8 9 10 11 12 13 (xtl6,xtl7) -ino dig pin#  */
240 /*                                             */
241 /* PORTB masks:                                */
242 /* Pin 8 = 0x01                                */
243 /* Pin 9 = 0x02                                */
244 /* Pin10 = 0x04                                */
245 /* Pin11 = 0x08                                */
246 /* Pin12 = 0x10                                */
247 /* Pin13 = 0x20                                */
248 /*                                             */
249 /***********************************************/
250     /* define autos */
251
252     /* toggle clock */
253     clkpin *=-1;
254     cnt1=0;
255     cnt2=0;
256
257     /*digitalWrite(clkout,LOW);*/
258     PORTB &= clrclk; /* put pin low*/
259
260     /* read data chan 1 and 2*/
261     /*ic1=digitalRead(chan1);*/
262     ic1 = PINB & getch1;
263     ic2 = PINB & getch2;
264     if(ic1>0)
265     {
266         /* debounce */
267         ic1up++;
268         ic1dn=0;
269     }
270     else
271     {
272         /* debounce */
273         ic1dn++;
274         ic1up=0;
275     }
276     if(ic1up>icdbnce)
277     {
278         /* see if A rising edge  */
279         if(ch1state<0)
280         {
281             /* if B hi then REV */
282             if(ch2state >0)
283             {
284                     dir1=-1;
285             } /* end set chan 1 direction Rev */
286             else
287             {
288                 /* if B lo then FWD */
289                 dir1=1;
290             } /* end if B lo */
291
292             /* inc tick count */
```

```
            cnt1=1;
        } /* end A rising edge */

        /* put ic1 pin hi */
        //digitalWrite(ch1out,HIGH);
        PORTB |= setch1;
        ch1state=1;
    }
    if(ic1dn>icdbnce)
    {
        /* see if A falling edge  */
        if(ch1state>0)
        {
            /* if B hi then FWD */
            if(ch2state >0)
            {
                dir1=1;
            } /* end set chan 1 direction Fwd */
            else
            {
                /*if B lo then REV */
                dir1=-1;
            } /* end if B lo */

            /* inc tick count */
            cnt1=1;
        } /* end A falling edge */

        /* put ic1 pin lo */
        //digitalWrite(ch1out,LOW);
        PORTB &= clrch1;
        ch1state=-1;
    }

    /* read data chan 2 */
    /*ic2=digitalRead(chan2);*/
    /*ic2 = PINB & getch2; */
    if(ic2>0)
    {
        /* debounce */
        ic2up++;
        ic2dn=0;
    }
    else
    {
        /* debounce */
        ic2dn++;
        ic2up=0;
    }
    if(ic2up > icdbnce)
    {
        /* see if B rising edge  */
        if(ch2state<0)
        {
            /* if A hi then FWD */
            if(ch1state >0)
            {
                dir2=1;
            } /* end set chan 1 direction Fwd */
            else
            {
                /* if A lo then REV */
                dir2=-1;
            } /* end if A lo */

            /* inc tick count */
            cnt2=1;
        } /* end B rising edge */

        /* put ic2 pin hi */
        //digitalWrite(ch2out,HIGH);
        PORTB |= setch2;
        ch2state=1;
```



```
    }
    if(ic2dn > icdbnce)
    {
        /* see if B falling edge  */
        if(ch2state>0)
        {
            /* if A hi then REV */
            if(ch1state >0)
            {
                dir2=-1;
            } /* end set chan 1 direction Rev */
            else
            {
                /* if A lo then FWD */
                dir2=1;
            } /* end if A lo */

            /* inc tick count */
            cnt2=1;
        } /* end B falling edge */

        /* put ic1 pin lo */
        //digitalWrite(ch2out,LOW);
        PORTB &= clrch2;
        ch2state=-1;
    }

    /* ch2 update - now write direction pin */
    if(dir1>0 && dir2>0)
    {
        revc=0;
        fwdc++;
        if (fwdc>dirlim)
        {
            //digitalWrite(dirout,HIGH);
            PORTB |= setdir;
            dirf=1;
        }
    }
    if(dir1<0 && dir2<0)
    {
        fwdc=0;
        revc++;
        if(revc>dirlim)
        {
            //digitalWrite(dirout,LOW);
            PORTB &= clrdir;
            dirf=-1;
        }
    }

    /* now add/sub position cnt */
    if(dirf>0)
    {
        tcount += cnt1;
        tcount += cnt2;
    }
    if(dirf<0)
    {
        tcount -= cnt1;
        tcount -= cnt2;
    } /* end inc the counts

    /*digitalWrite(clkout,HIGH);*/
    PORTB |= setclk; /* put pin high */

    return;
} /* end sub pollenk */

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

```
439 int makbcd()
440 {
441
442 /*********************************************/
443 /*                                           */
444 /* makbcd - coverts long count to a decimal ascii*/
445 /* string, returns length of string          */
446 /*                                           */
447 /*********************************************/
448     /* define autos */
449     /*long lnum,subn; */
450     int lnum,subn;
451     int dig,i,dig_cnt;
452     char *s,*t;
453
454     /* pick up current value of tick count */
455     lnum=tcount;
456
457     /* setup output string */
458     t=tout;
459     *t='+';
460     dig_cnt=0;
461     if(lnum<0)
462     {
463         *t='-';
464         lnum*= -1;
465     }
466     t++;
467
468     /* test if case zero */
469     if(lnum==0)
470     {
471         *(t++)='0';
472         dig_cnt=1;
473     } /* end fix case = 0 */
474
475     /*poll encoder */
476     pollenk();
477
478     for (s=sout,i=0;lnum>0;i++)
479     {
480         /* loop extracting lsf digits */
481         /*subn=long(lnum/10l);*/
482         subn=int(lnum/10);
483
484         /*poll encoder */
485         pollenk();
486
487         /*dig=int(lnum-subn*10l);*/
488         dig=int(lnum-subn*10);
489         *s=dig+0x30;
490         s++;
491         lnum=subn;
492
493         /*poll encoder */
494         pollenk();
495     }
496
497     /*poll encoder */
498     pollenk();
499
500     /* now put digits msd first */
501     dig_cnt +=i;     /*save # of digits */
502     for(s--;i>0;i--)
503     {
504         *t++=*s--;
505     } /*end sort into msd order */
506
507     /*poll encoder */
508     pollenk();
509
510     /* finish string */
511     *(t++)=',';
```



```
512     *t='\0';
513
514     /*set length of number string, incl sign and delimiter */
515     /*dig=strlen(tout);*/
516     dig_cnt +=2;
517
518     /* return # valid chars in string */
519     return dig_cnt;
520
521 } /* end routine makbcd */
522
523
524 /*********************************************/
525 /*                                           */
526 /*********************************************/
527
528 int makhex()
529 {
530
531 /*********************************************/
532 /*                                           */
533 /* makhex - coverts long count to a hex ascii   */
534 /* string, returns length of string          */
535 /*                                           */
536 /*********************************************/
537     /* define autos */
538     long lnum;
539     int i,dig_cnt;
540     char *s,*t, *ovlay,cwsl,cwsr;
541
542     /* pick up current value of tick count */
543     lnum=tcount;
544
545     /* setup output string */
546     t=tout;
547     *t++='0';
548     *t++='x';
549
550     /*poll encoder */
551     pollenk();
552
553     /* now print hex byte-wise version - 32 bit int*/
554     for (ovlay = (char *) (&lnum),s=sout,i = 0; i<4; i++)
555     {
556         cwsr =*ovlay;
557         cwsl =*(ovlay++);
558
559         /* get high 4 bits */
560         cwsl=cwsl >> 4; /*move high bits right */
561         cwsl=cwsl & 0x0f;
562         cwsr=cwsr & 0x0f;
563
564         /*poll encoder */
565         pollenk();
566
567         /* save low(R) nibble as hex */
568         *s=cwsr+0x30;
569         if(*s > 0x39)
570         {
571             *s += 0x27 ; /* letters to lower case */
572         } /* end  out out A to F */
573
574         /*poll encoder */
575         pollenk();
576
577         /* save high(L) nibble as hex */
578         *(++s)=cwsl+0x30;
579         if(*s > 0x39)
580         {
581             *s += 0x27 ; /* letters to lower case */
582         } /* end  out out A to F */
583
584         s++;
```

```
585
586         /*poll encoder */
587         pollenk();
588     } /* end loop over 8 nibbles */
589
590     /*poll encoder */
591     pollenk();
592
593     /* now reverse order string */
594     for (s--, i=0; i<8; i++)
595     {
596         *t++=*s--;
597     } /* end format string */
598
599     /*poll encoder */
600     pollenk();
601
602     /* finish string */
603     *(t++)=',';
604     *t='\0';
605     dig_cnt=11;
606
607     /* return # valid chars in string */
608     return dig_cnt;
609
610 } /* end routine makhex */
611
612 /*********************************************/
613 /*                                           */
614 /*********************************************/
615
616 void blowc(char s)
617 {
618
619 /*********************************************/
620 /*                                           */
621 /* blowc - puts a char out the hand rolled   */
622 /* serial link                               */
623 /*                                           */
624 /*********************************************/
625     /* define autos */
626     byte mask;
627
628     /* put out startbit */
629     PORTD &= clrser; /* send a lo */
630
631     /* wait in polling delay */
632     polldla();
633
634     /* shift bits out - lsb first */
635     for (mask = 0x01; mask>0; mask <<= 1)
636     {
637         /* select bit */
638         if (s & mask)
639         {
640             PORTD |= setser; /* send a hi */
641         }
642         else
643         {
644             PORTD &= clrser; /* send a lo */
645         }
646
647         /* wait in polling delay */
648         polldla();
649     }
650
651     /* send stop bit */
652     PORTD |= setser; /* send a hi */
653
654       /* wait in polling delay */
655     polldla();
656
657     /* now wait out another byte-time or so */
```



```
658     for (mask = 0; mask <15; mask++)
659     {
660         polldla();
661     } /* end long pause */
662
663 } /* end routine blowc */
664
665 /*********************************************/
666 /*                                           */
667 /*********************************************/
668
669 void polldla()
670 {
671
672 /*********************************************/
673 /*                                           */
674 /* polldla - waits 9600 baud delay by polling   */
675 /*                                           */
676 /*********************************************/
677     /* define autos */
678     unsigned long mus, mus2;
679     int i;
680
681     /* note start time */
682     mus=micros();
683
684     for(i=0;i<=t1_9600;i++)
685     {
686         pollenk();
687     } /* end polling delay */
688
689     /* pick up time now */
690     mus2=micros();
691     mus2 -= mus;
692     i = dla9600-int(mus2);
693
694     /* now remainder us */
695     if(i>0)
696     {
697         delayMicroseconds(i);
698     }
699
700 } /* end routine polldla */
701
702 /*********************************************/
703 /*                                           */
704 /*********************************************/
705
```