

```

100 ;*****
101 ; Source code for Direct Digital Synthesis VFO controller
102 ;
103 ; Based on original program written for an AD9850 by
104 ; Curtis W. Preuss - WB2V
105 ; 11/08/96
106 ; Initial creation
107 ; 10/14/98
108 ; Ported source to Microchip MPASM -- Craig B. Johnson, AA0ZZ
109 ; Target Controller - PIC16f84a
110 ;
111 ; Further changes
112 ;
113 ; 10/02/05
114 ; Modified for an AD9851 with a 30MHz crystal oscillator and
115 ; x6 multiplier enabled. AD9851_4 contains the control bits
116 ; and the digit weights have been changed accordingly.
117 ;
118 ;*****
119 list p=16f84
120 radix hex
121
122 Indirect equ 0
123 Status equ 3
124 FSR equ 4
125 PortA equ 5
126 PortB equ 6
127 Enable equ 1 ; Enable x6 multiplier. If this is set to 0
128 ; the AD9851 will work as an AD9850
129
130 ;Assign names to IO pins
131
132 LCD_VDD equ 3 ;(PortA)Power to LCD module
133 AD9851_fqu equ 0 ;(PortB)Update pin on AD9851
134 LCD_rs equ 1 ;(PortB)0=instruction, 1=data
135 LCD_rw equ 2 ;(PortB)0=write, 1=read
136 LCD_e equ 3 ;(PortB)0=disable, 1=enable
137 AD9851_wc equ 5 ;(PortB)AD9851 write clock
138 AD9851_dat equ 7 ;(PortB)AD9851 serial data input
139
140 ;Allocate variables in general purpose register space
141
142 LCD_Dat equ 0x0C ;current Display data
143 AD9851 equ 0x13 ;5 byte control word for DDS chip
144 OP1 equ 0x18 ;Work Space for
145 OP2 equ 0x1C ; the multiply routine
146 temp1 equ 0x1D ;Temp storage
147 temp2 equ 0x1E ;Temp storage
148 temp3 equ 0x1F ;Temp storage
149 old equ 0x20 ;old encoder input data
150 new equ 0x21 ;new encoder input data
151 count1 equ 0x22 ;used for timing the shaft encoder
152 count2 equ 0x23 ;
153
154 ;Status Bits

```

```

155
156 C          equ 0          ;Carry
157 DC        equ 1          ;Digit (Half) Carry
158 Z          equ 2          ;Zero
159
160          org 0
161          goto Start
162
163          org 5
164 ;*****
165 ;BusyCheck Subroutine
166 ;
167 ;LCD read/write operations are sloooooow, this subroutine
168 ;polls the LCD busy flag to determine if previous operations are completed.
169 ;Note side effect that the LCD_rw and LCD_rs are left in write data mode.
170
171 BusyCheck
172     movlw 0xF0          ;Tristate the 4 data pins
173     tris  PortB          ;
174     bsf  PortB,LCD_rw   ;Raise the r/w bit to read the LCD
175 Busy
176     bsf  PortB,LCD_e    ;
177     movf PortB,w        ;Get first nibble of input data
178     movwf temp1        ; and save it
179     bcf  PortB,LCD_e    ;
180     nop                ;
181     bsf  PortB,LCD_e    ;Get next nibble to keep LCD happy
182     bcf  PortB,LCD_e    ;
183     btfsc temp1,7      ;Check for busy flag
184     goto Busy          ;It is busy
185     bcf  PortB,LCD_rw   ;Leave LCD in Write Data mode
186     bsf  PortB,LCD_rs   ;
187     return
188
189 ;*****
190 ;Send Character to LCD Subroutine
191 ;
192 ;The character to be sent must have been placed in "temp2" prior to calling
193 ;the subroutine. LCD_rw and LCD_rs must be set up prior to entry.
194
195 SendChar
196     movlw 0x00          ;
197     tris  PortB          ;Enable port B
198     movf  temp2,w        ;Save temp2
199     movwf temp1        ; in temp1
200     movlw 0x0F          ;
201     andwf PortB,f      ;
202     movlw 0xF0          ;
203     andwf temp2,f      ;
204     movf  temp2,w        ;
205     iorwf PortB,f      ;Load first nibble
206     bsf  PortB,LCD_e    ;Transfer first nibble
207     bcf  PortB,LCD_e    ;
208     swapf temp1,f      ;
209     movlw 0x0F          ;

```

```

210     andwf    PortB,f           ;
211     movlw   0xF0              ;
212     andwf   temp1,f           ;Clear data nibble
213     movf    temp1,w           ;
214     iorwf   PortB,f           ;
215     bsf    PortB,LCD_e        ;Transfer second nibble
216     bcf    PortB,LCD_e        ;
217     return
218
219 ;*****
220 ;BCD increment
221 ;
222 ;This routine does a binary coded decimal increment to the values in LCD_Dat.
223 ;On entry the FSR register must point to the LCD_Dat digit to be incremented.
224 ;If the result is over 60MHz the value is reset back to 60MHz.
225
226 BCDInc
227     movf    Indirect,w         ;
228     movwf   temp2              ;
229     movlw   0x01              ;
230     addwf   temp2,f           ;
231     movf    temp2,w           ;
232     movwf   temp1              ;
233     movlw   0x06              ;
234     addwf   temp1,f           ;
235     btfsc   Status, DC         ;See if Digit Carry is clear
236     goto    BCDIncCarry       ;Digit Carry is set, so jump
237     movlw   0x0F              ;
238     andwf   temp2,f           ;
239     movf    temp2,w           ;
240     movwf   Indirect          ;
241     movlw   0x06              ;Max freq 60MHz
242     subwf   LCD_Dat+0,w        ;Test for max frequency
243     btfsc   Status, C         ;See if Carry is clear (LCD_Dat+0<6)
244     goto    SetMax            ;Carry is set, so jump
245     return
246 BCDIncCarry
247     clrf    temp2              ;
248     movf    temp2,w           ;
249     movwf   Indirect          ;
250     decf    FSR,f             ;
251     goto    BCDInc            ;
252 SetMax
253     movlw   0x12              ;Set the display to 60,000.00
254     movwf   FSR                ;Low digit
255     ;Point to it
256 SetMax1
257     movlw   0x00              ;
258     movwf   Indirect          ;Zero and repeat for five more digits
259     decf    FSR,f             ;
260     movlw   0xF3              ;
261     addwf   FSR,w             ;
262     btfsc   Status, C         ;See if Carry is clear
263     goto    SetMax1           ;Carry is set, so jump
264     movlw   0x06              ;Set digit to high x10MHz count
265     movwf   Indirect          ;Put in LCD_Dat+0

```

```

265         return                ;
266 ;*****
267 ;BCD decrement
268 ;
269 ;This routine does a binary coded decimal decrement to the values in LCD_Dat.
270 ;On entry the FSR register must point to the LCD_Dat digit to be decremented,
271 ;(one of 10's, 100's, 1K, 10K or 100K digit)
272 ;If the result is under 100KHz the value is reset back to 100KHz.
273
274 BCDDec
275     movf    Indirect,w          ;
276     movwf   temp2              ;
277     movlw   0x01               ;
278     subwf   temp2,f            ;
279     btfss   Status, DC         ;See if Digit Carry is set
280     goto    BCDDecCarry       ;Digit Carry is clear, so jump
281     movf    temp2,w           ;
282     movwf   Indirect          ;
283     movlw   0xFF              ;
284     addwf   LCD_Dat+0,w       ;
285     btfsc   Status, C         ;See if Carry is clear
286     goto    BCDDecExit       ;Carry is set, so jump to exit
287     movlw   0xFF              ;
288     addwf   LCD_Dat+1,w       ;
289     btfsc   Status, C         ;See if Carry is clear
290     goto    BCDDecExit       ;Carry is set, so jump to exit
291     movlw   0xFF              ;
292     addwf   LCD_Dat+2,w       ;
293     btfsc   Status, C         ;See if Carry is clear
294     goto    BCDDecExit       ;Carry is set, so jump to exit
295     goto    SetMin           ;
296 BCDDecExit
297     return
298 BCDDecCarry
299     movlw   0x09              ;
300     movwf   temp2              ;
301     movf    temp2,w           ;
302     movwf   Indirect          ;
303     decf    FSR,f             ;
304     goto    BCDDec           ;
305 SetMin
306     movlw   0x12              ;
307     movwf   FSR               ;
308 SetMin1
309     movlw   0x00              ;
310     movwf   Indirect          ;
311     decf    FSR,f             ;
312     movlw   0xF1              ;
313     addwf   FSR,w             ;
314     btfsc   Status, C         ;See if Carry is clear
315     goto    SetMin1          ;Carry is set, so jump
316     movlw   0x01              ;
317     movwf   Indirect          ;
318     decf    FSR,f             ;
319     movlw   0x00              ;

```

```

320     movwf   Indirect      ;
321     decf   FSR,f         ;
322     movlw  0x00         ;
323     movwf   Indirect      ;
324     return                ;
325
326 ;*****
327 ; Delay 24ms subroutine
328 ;
329
330 Wait24ms
331     movlw  0x28         ; use 40 for 24ms wait if clock is 4Mhz
332     movwf   temp1       ;
333 Wait24Outer
334     movlw  0xC8         ; use 200 for 24 ms wait
335     movwf   temp2       ;
336 Wait24Inner
337     decfsz temp2,f      ;
338     goto   Wait24Inner  ;
339     decfsz temp1,f      ;
340     goto   Wait24Outer  ;
341     return
342
343 ;*****
344 ; Delay 60ms subroutine
345 ;
346 Wait60ms
347     movlw  0x64         ; use 100 for 60ms wait if clock is 4Mhz
348     movwf   temp1       ;
349 Wait60Outer
350     movlw  0xC8         ; use 200 for 60ms wait
351     movwf   temp2       ;
352 Wait60Inner
353     decfsz temp2,f      ;
354     goto   Wait60Inner  ;
355     decfsz temp1,f      ;
356     goto   Wait60Outer  ;
357     return
358
359 ;*****
360 ; 32 bit add
361 ;
362 ;The value in registers OP1 is added to the current 32 bit value in AD9851
363
364 Add32
365     movf   OP1+0,w      ;
366     addwf  AD9851+0,f   ;
367     btfss Status, C    ;See if Carry is set
368     goto   Add1         ;Carry is clear, so jump
369     incfsz AD9851+1,f   ;
370     goto   Add1         ;
371     incfsz AD9851+2,f   ;
372     goto   Add1         ;
373     incf  AD9851+3,f    ;
374 Add1

```

```

375     movf     OP1+1,w           ;
376     addwf   AD9851+1,f       ;
377     btfss   Status, C        ;See if Carry is set
378     goto    Add2             ;Carry is clear, so jump
379     incfsz  AD9851+2,f       ;
380     goto    Add2             ;
381     incf    AD9851+3,f       ;
382 Add2
383     movf     OP1+2,w           ;
384     addwf   AD9851+2,f       ;
385     btfss   Status, C        ;See if Carry is set
386     goto    Add3             ;Carry is clear, so jump
387     incf    AD9851+3,f       ;
388 Add3
389     movf     OP1+3,w           ;
390     addwf   AD9851+3,f       ;
391     return                    ;
392
393 ;*****
394 ;Write Data
395 ;
396 ;This subroutine does two things:
397 ; (1) it sends the contents LCD_dat to the LCD,
398 ;     (LCD_Dat must contain BCD digits)
399 ; (2) the contents of AD9851 (40 bits) are serial shifted
400 ;     into the AD9851 module
401
402 WriteData
403     movlw   0x83               ; LCD position for first digit
404     movwf   temp2             ;
405     call    BusyCheck         ;
406     bcf     PortB,LCD_rs      ; Set LCD for instruction write
407     call    SendChar          ;
408     movf    LCD_Dat+0,w       ;
409     movwf   temp2             ;
410     movlw   0x20              ;Send a space if first digit is zero
411     iorwf   temp2,f           ;
412     movlw   0x20              ;Set up for space check
413     subwf   temp2,w           ;Subtract to compare
414     btfsc   Status, Z        ;See if Zero is clear (not a space)
415     goto    Wd2              ;Zero is set, so go send the space
416     movlw   0x10              ;Zero is clear, so convert BCD
417     iorwf   temp2,f           ; to ASCII
418 Wd2
419     call    BusyCheck         ;
420     call    SendChar          ;Send 10Mhz digit
421     movf    LCD_Dat+1,w       ;
422     movwf   temp2             ;
423     movlw   0x30              ;
424     iorwf   temp2,f           ;
425     call    BusyCheck         ;
426     call    SendChar          ;Send 1Mhz digit
427     movlw   0x2C              ;
428     movwf   temp2             ;
429     call    BusyCheck         ;

```

```

430     call    SendChar           ;Send a comma
431     movf   LCD_Dat+2,w         ;
432     movwf  temp2              ;
433     movlw  0x30               ;
434     iorwf  temp2,f            ;
435     call   BusyCheck          ;
436     call   SendChar           ;Send a 100KHz digit
437     movf   LCD_Dat+3,w         ;
438     movwf  temp2              ;
439     movlw  0x30               ;
440     iorwf  temp2,f            ;
441     call   BusyCheck          ;Send 10 KHz digit
442     call   SendChar           ;
443     movlw  0xC0               ;Point LCD at digit#9 (Gap in LCD address)
444     movwf  temp2              ;
445     call   BusyCheck          ;
446     bcf   PortB,LCD_rs        ;Set LCD for instruction write
447     call   SendChar           ;
448     movf   LCD_Dat+4,w         ;
449     movwf  temp2              ;
450     movlw  0x30               ;
451     iorwf  temp2,f            ;
452     call   BusyCheck          ;
453     call   SendChar           ; Send 1Khz digit
454     movlw  0x2E               ;
455     movwf  temp2              ;
456     call   BusyCheck          ;
457     call   SendChar           ; Send a period
458     movf   LCD_Dat+5,w         ;
459     movwf  temp2              ;
460     movlw  0x30               ;
461     iorwf  temp2,f            ;
462     call   BusyCheck          ;
463     call   SendChar           ; Send 100Hz digit
464     movf   LCD_Dat+6,w         ;
465     movwf  temp2              ;
466     movlw  0x30               ;
467     iorwf  temp2,f            ;
468     call   BusyCheck          ;
469     call   SendChar           ; Send 10Hz digit
470 WriteAD9851
471     clrf   PortB              ; Clear PortB
472     movlw  0x05               ; Send 5 bytes
473     movwf  temp3              ; to AD9851
474     movlw  0x00               ;
475     tris   PortB              ; Enable PortB
476     movlw  0x13               ; Point at AD9851+0
477     movwf  FSR                ;
478 NextByte
479     movf   Indirect,w         ;
480     movwf  temp1              ;
481     movlw  0x08               ; set bit counter to 8
482     movwf  temp2              ;
483 NextBit
484     rrf    temp1,f            ;

```

```

485      btfss   Status, C           ;See if carry is set
486      goto    Send0             ;Carry is clear, so jump
487      bsf    PortB,AD9851_dat
488      bsf    PortB,AD9851_wc    ; toggle write clock
489      bcf    PortB,AD9851_wc    ; (repeated 40 times)
490      goto    Break
491  Send0
492      bcf    PortB,AD9851_dat
493      bsf    PortB,AD9851_wc
494      bcf    PortB,AD9851_wc
495  Break
496      decfsz temp2,f
497      goto    NextBit
498      incf   FSR,f
499      decfsz temp3,f
500      goto    NextByte
501      bsf    PortB,AD9851_fqu    ; send load signal to AD9851
502      bcf    PortB,AD9851_fqu    ; Clear after last data dbit is written
503      return
504
505  ;*****
506  ;7 Digit BCD by 32 bit binary mulitply
507  ;
508  ;The 7 BCD digits stored in LCD_dat are each multiplied by a precalculated digit
509  ;weight. Digit weigths assume a 30MHz input clock with the x6 multiplier enabled
510  ;The sum of all 7 digits times their weigth is stored in the AD9851 registers.
511  ;
512
513  Mult
514      clrf   temp1
515      clrf   AD9851+0           ; Clear all five bytes of
516      clrf   AD9851+1           ; AD9851 control word
517      clrf   AD9851+2           ; W0 thru W4
518      clrf   AD9851+3
519      clrf   AD9851+4           ; First 3 bits contain control
520      movlw  Enable             ; Enable the x6 multiplier
521      movwf  AD9851+4
522  Mult10
523      movlw  0xEF               ;EF
524      movwf  OP1+0
525      clrf   OP1+1
526      clrf   OP1+2
527      clrf   OP1+3
528      movf   LCD_Dat+6,w        ;
529      movwf  OP2                ;
530      goto   Go
531  Mult100
532      movlw  0x52               ;0952
533      movwf  OP1+0
534      movlw  0x09               ;
535      movwf  OP1+1
536      clrf   OP1+2
537      clrf   OP1+3
538      movf   LCD_Dat+5,w        ;
539      movwf  OP2

```

```

540         goto      Go
541 Mult1K
542         movlw    0x35                ;5D35
543         movwf    OP1+0              ;
544         movlw    0x5D                ;
545         movwf    OP1+1
546         clrf     OP1+2              ;
547         clrf     OP1+3
548         movf     LCD_Dat+4,w        ;
549         movwf    OP2
550         goto      Go
551 Mult10K
552         movlw    0x11                ;03A411
553         movwf    OP1+0
554         movlw    0xA4                ;
555         movwf    OP1+1
556         movlw    0x03                ;
557         movwf    OP1+2
558         clrf     OP1+3              ;
559         movf     LCD_Dat+3,w        ;
560         movwf    OP2
561         goto      Go
562 Mult100K
563         movlw    0xAA                ;2468AA
564         movwf    OP1+0
565         movlw    0x68                ;
566         movwf    OP1+1
567         movlw    0x24                ;
568         movwf    OP1+2
569         clrf     OP1+3
570         movf     LCD_Dat+2,w        ;
571         movwf    OP2                ;
572         goto      Go
573 Mult1M
574         movlw    0xA3                ;016C16A3
575         movwf    OP1+0
576         movlw    0x16                ;
577         movwf    OP1+1
578         movlw    0x6C                ;
579         movwf    OP1+2
580         movlw    0x01                ;
581         movwf    OP1+3
582         movf     LCD_Dat+1,w        ;
583         movwf    OP2                ;
584         goto      Go
585 Mult10M
586         movlw    0x5D                ;0E38E25D
587         movwf    OP1+0
588         movlw    0xE2                ;
589         movwf    OP1+1
590         movlw    0x38                ;
591         movwf    OP1+2
592         movlw    0x0E                ;
593         movwf    OP1+3
594         movf     LCD_Dat+0,w        ;

```

```

595     movwf  OP2           ;
596     goto   Go
597 Go
598     movlw  0x04
599     movwf  temp2
600 DigitLoop
601     bcf    Status, C      ;Clear Carry bit
602     rrf    OP2,f         ;Rotate right (through carry)
603     btfss  Status, C     ;See if Carry is set
604     goto   NoCarry      ;Carry is clear, so jump
605     call   Add32
606 NoCarry
607     bcf    Status, C      ;Clear Carry bit
608     rlf    OP1+0,f       ;Rotate left through carry
609     rlf    OP1+1,f       ;
610     rlf    OP1+2,f       ;
611     rlf    OP1+3,f       ;
612     decfsz temp2,f
613     goto   DigitLoop
614     incf   temp1,f
615     movlw  0x01
616     subwf  temp1,w
617     btfsc  Status, Z     ;See if Zero is clear
618     goto   Mult100      ;Zero is set, so jump
619     movlw  0x02
620     subwf  temp1,w
621     btfsc  Status, Z     ;See if Zero is clear
622     goto   Mult1K      ;Zero is set, so jump
623     movlw  0x03
624     subwf  temp1,w
625     btfsc  Status, Z     ;See if Zero is clear
626     goto   Mult10K     ;Zero is set, so jump
627     movlw  0x04
628     subwf  temp1,w
629     btfsc  Status, Z     ;See if Zero is clear
630     goto   Mult100K    ;Zero is set, so jump
631     movlw  0x05
632     subwf  temp1,w
633     btfsc  Status, Z     ;See if Zero is clear
634     goto   Mult1M      ;Zero is set, so jump
635     movlw  0x06
636     subwf  temp1,w
637     btfsc  Status, Z     ;See if Zero is clear
638     goto   Mult10M     ;Zero is set, so jump
639     return
640
641 ;*****
642 ;Initialize PIC controller
643 ;
644 ; Start here after reset or power on
645 ; Set DDS initially to 30,000.00
646
647 Start
648     clrf   PortA         ; Clear data Port A
649     clrf   PortB         ; Clear data Port B

```

```

650      movlw    0x00          ;
651      tris    PortB        ; Enable port B drivers
652      movlw    0xFF        ;
653      tris    PortA        ; Tristate Port A drivers
654      movlw    0x03        ; Set initial LCD data for 35MHz
655      movwf   LCD_Dat+0
656      movlw    0x05
657      movwf   LCD_Dat+1
658      movlw    0x00
659      movwf   LCD_Dat+2
660      movlw    0x00
661      movwf   LCD_Dat+3
662      movlw    0x00
663      movwf   LCD_Dat+4
664      movlw    0x00
665      movwf   LCD_Dat+5
666      movlw    0x00
667      movwf   LCD_Dat+6
668      clrf    count1
669      clrf    count2
670
671      ;*****
672      ;Power on initialization of Liquid Crystal Display.
673      ; The LCD controller chip must be equivalent to an Hitachi 44780.
674      ; The LCD is assumed to be a 16 X 1 display.
675
676      movlw    0x07          ; Enable LCD_VDD pin
677      tris    PortA        ; power up LCD
678      bsf     PortA,LCD_VDD
679      call    Wait60ms      ; Wait for LCD to power up
680      bsf     PortB,LCD_e   ;
681      movlw    0x38          ; func set + enable to LCD
682      movwf   PortB
683      bcf     PortB,LCD_e
684      call    Wait60ms      ;
685      bsf     PortB,3
686      movlw    0x38          ; func set + enable to LCD
687      movwf   PortB
688      bcf     PortB,LCD_e   ;
689      call    Wait60ms      ;
690      bsf     PortB,LCD_e
691      movlw    0x38          ; func set + enable to LCD
692      movwf   PortB
693      bcf     PortB,LCD_e   ;
694      call    Wait60ms      ;
695      bsf     PortB,LCD_e
696      movlw    0x28          ; 4bit_mode instruction + enable
697      movwf   PortB
698      bcf     PortB,LCD_e
699      call    Wait60ms
700      movlw    0x01          ; Clear and reset cursor
701      movwf   temp2
702      call    BusyCheck
703      bcf     PortB,LCD_rs
704      call    SendChar

```

```

705     movlw    0x06                ;increment no_shift mode
706     movwf    temp2
707     call     BusyCheck
708     bcf      PortB,LCD_rs
709     call     SendChar
710     movlw    0x0C                ; Display on
711     movwf    temp2
712     call     BusyCheck
713     bcf      PortB,LCD_rs
714     call     SendChar
715
716 ;Send initial display value to LCD
717
718     call     Mult
719     call     Wait24ms            ; allow AD9851 clock time to start
720     call     Wait24ms            ;
721     call     WriteData           ; side effect reset power down bits of AD9851
722     call     WriteData           ; load the initial display value
723
724 ;Get the power on encoder value
725
726     movf     PortA,w             ; get encoder value
727     movwf    temp2
728     movlw    0x03
729     andwf    temp2,w            ; isolate direction bits
730     movwf    old                 ; save initial values
731
732
733 ;*****
734 ;Check encoder
735 ; PortA.0 shaft encoder output A
736 ; PortA.1 shaft encoder output B
737 ; PortA.2 shaft encoder push button switch
738 ; Timing checks assume a 4MHz cpu clock and that the encoder
739 ; has 32 detents
740
741 ChkEncoder
742     incf     count1,f
743     movlw    0x75                ;Check encoder loop takes 17 us
744     subwf    count1,w
745     btfss   Status,C            ;See if Carry is set
746     goto    Ce                  ;Carry is clear, so jump
747     incf     count2,f
748     clrf    count1              ;0x75 * 17us = 2ms
749 Ce
750     movf     PortA,w             ;Get encoder value
751     movwf    temp1
752     movf     temp1,w
753     movwf    temp2              ;Double latch the input
754     movlw    0x03
755     andwf    temp2,w
756     movwf    new
757     xorwf    old,w
758     btfsc   Status,Z            ;See if Zero is clear
759     goto    ChkEncoder          ;Zero is set, so jump

```

```

760      movlw    0x0E                ;Pointer set to 100KHz digit
761      movwf    FSR
762      btfs    temp2,2             ;If pb switch is closed (active low)
763      goto    CeGo                ;
764      movlw    0x12                ;Pointer set to 10's
765      movwf    FSR
766      movlw    0x0F
767      subwf    count2,w           ;If speed < 1.0 RPM
768      btfs    Status, C           ;See if Carry is clear
769      goto    CeGo                ;Carry is set, so jump
770      movlw    0x11                ;Pointer set to 100's
771      movwf    FSR
772      movlw    0x05
773      subwf    count2,w           ;If (1.0RPM < Speed < 3.0RPM)
774      btfs    Status, C           ;See if Carry is clear
775      goto    CeGo                ;Carry is set , so jump
776      movlw    0x10                ;Pointer set to 1K
777      movwf    FSR
778  CeGo
779      clrf     count2              ;Add time to write LCD etc to count1
780      movlw    0x46                ; (about 1.2 ms)
781      movwf    count1
782      bcf     Status, C           ;Clear Carry bit
783      rlf     old,f                ;Rotate left
784      movf    new,w
785      xorwf    old,f                ;Determine direction
786      btfs    old, DC              ;See if Digit Carry is clear
787      goto    CeUp                ;Digit Carry is set, so jump
788      call    BCDDec              ;Calculate new LCD display
789      call    Mult                 ;Calculate new AD9851 control word
790      goto    CeWrite
791  CeUp
792      call    BCDInc               ; calculate new LCD display
793      call    Mult                 ; calculate new AD9851 control word
794  CeWrite
795      call    WriteData
796      movf    new,w
797      movwf    old
798      goto    ChkEncoder           ; continue polling the encoder
799
800 ; ID
801 ; ID
802 ; ID
803 ; ID
804 ; Fuses (CP=Off, PWRTE=Enabled, WDTE=Disabled, OSC=XT)
805 ; 0001
806
807      END
808
809
810
811
812
813
814

```

815
816
817
818
819