```
' PROGRAM: Auto-Tuner_code.bsx      Revision 04.04.2012
' Roland Mensch DK3GI                Revision 24.11.2012 changed L-limit from lower to upper limit
'                                                        and L-range to 4150

' {$STAMP BS2sx}          'Specify Basic Stamp version -- required statement

' ============================================================
'   Program controls automatic LC-tuner. This version uses stepper motors to control a roller inductor and a vacuum
'   capacitor. In auto mode, tuner measures TX frequency, looks up corresponding settings in memory, and moves
'   steppers to the settings. The vacuum capacitor and roller inductor can also be manually controlled with manual
'   up/down buttons or Preset buttons. The tuner shifts to manual mode whenever a manual or preset button is pressed.
'   The manual up/down buttons include a ramp-up function to allow precise settings.
'   The tuner powers up by initializing the stepper motors, switching to auto mode, and returning to the last-used
'   memory settings. There is 1 dual-function pushbutton, with functions as follows:

'   STORE-RESET Button:  One press stores current settings in memory. If held down, initializes the steppers to
'                        home positions.

'   The display shows a 2 sec startup message and then displays the L and C settings (top line), and the memory
'   frequency (second line). During movement of the stepper motors the flashing cursor is displayed.

'      Port Connections & INPUT/OUTPUT Status

'      P0:   IN   manual button1
'      P1:   IN   manual button2
'      P2:   IN   manual button3
'      P3:   IN   manual button4
'      P4:   IN   DOWN button
'            OUT  Speaker
'      P5:   IN   UP button
'      P6:   OUT  Auto Mode LED
'      P7:   IN   IN/OUT-RESET button
'      P8:   IN   MODE/STORE button
'      P9    IN   Motor Limit
'      P10:  OUT  Ext.capacitor relay
'      P11:  OUT  Gear-reduced Sw-stepper step
'      P12:  OUT  Stepper1 step
'      P13:  OUT  Stepper2 step
'      P14:  OUT  Stepper rotation CW=0,CCW=1
'      P15:  IN   TX frequency detect
'      P16:  OUT  LCD serial data

'==================================DEFAULT TUNER SETTINGS=====================================

'Default L and C Data in EEPROM (144 bytes). These values match a 50 ohm load. If updating
'program, insert (') before each DATA statement to avoid overwriting existing settings.

' L - Data (144 byte-values /72 word-values = 72 settings) stored in EEPROM addresses 0-143 ($000-$08F)
```

```
   DATA Word 160,Word 61,Word 62,Word 63,Word 64,Word 65,Word 66,Word 67,Word 68,Word 69 '160meters (0-19) 20kHz steps
   DATA Word 80, Word 81,Word 82,Word 83,Word 84,Word 85,Word 86,Word 87,Word 88,Word 89 '80 meters(20-49) 20kHz steps
   DATA Word 110,Word 111,Word 112,Word 113,Word 114
   DATA Word 400,Word 41, Word 900,Word 43, Word 44, Word 45, Word 46, Word 47          '40 meters(50-65) 25kHz steps
   DATA Word 30, Word 31                                                               '30 meters(66-69) 25kHz steps
   DATA Word 20, Word 21, Word 22, Word 23, Word 24, Word 25, Word 26, Word 27, Word 28  '20 meters(70-87) 40kHz steps
   DATA Word 17                                                                        '17 meters(88-89) 100kHz steps
   DATA Word 15, Word 151,Word 152,Word 153,Word 154,Word 155,Word 156,Word 157,Word 158 '15 meters(90-107) 50kHz steps
   DATA Word 12                                                                        '12 meters(108-109)100kHz steps
   DATA Word 10, Word 11,Word 12,Word 13,Word 14,Word 15,Word 16,Word 17,Word 18,Word 19 '10 meters(110-143)100kHz steps
   DATA Word 110,Word 111,Word 112,Word 113,Word 114,Word 115,Word 116

' C - Data (144 byte-values /72 word-values = 72 settings)) stored in EEPROM addresses 144-287 ($090-$11F)

   DATA Word 160,Word 61,Word 62,Word 63,Word 64,Word 65,Word 66,Word 67,Word 68,Word 69 '160meters(144-163) 20kHz steps
   DATA Word 80, Word 81,Word 82,Word 83,Word 84,Word 85,Word 86,Word 87,Word 88,Word 89 '80 meters(164-193) 20kHz steps
   DATA Word 110,Word 111,Word 112,Word 113,Word 114
   DATA Word 400,Word 3900, Word 900,Word 43, Word 44, Word 45, Word 46, Word 47        '40 meters(194-209) 25kHz steps
   DATA Word 30, Word 31                                                               '30 meters(210-213) 25kHz steps
   DATA Word 20, Word 21, Word 22, Word 23, Word 24, Word 25, Word 26, Word 27, Word 28  '20 meters(214-231) 40kHz steps
   DATA Word 17                                                                        '17 meters(232-233) 100kHz steps
   DATA Word 15, Word 151,Word 152,Word 153,Word 154,Word 155,Word 156,Word 157,Word 158 '15 meters(234-251) 50kHz steps
   DATA Word 12                                                                        '12 meters(152-253)100kHz steps
   DATA Word 10, Word 11,Word 12,Word 13,Word 14,Word 15,Word 16,Word 17,Word 18,Word 19 '10 meters(154-287)100kHz steps
   DATA Word 110,Word 111,Word 112,Word 113,Word 114,Word 115,Word 116

' Startup addr1 288 ($120)

   DATA @$120,52,0,0,0

'  ================================================================================

'   SYMBOL TABLE
'   User Supplied Parameters

freqtrim   CON   322      'Parameter corrects for clock innacuracy in the BS2sx.
                         'The correction factor is 1/freqtrim, e.g., 1/322=.0031=.3% correction.
                         'Freqtrim varies from device to device and is determined experimentally.

'   ================================================================================

stpcnt1    VAR   Word     'stepper1 count
stpcnt2    VAR   Word     'stepper2 count
memval1    VAR   Word     'L memory variable
memval2    VAR   Word     'C memory variable
stp        VAR   Word     'step variable
btnwk1     VAR   Byte     'button workspace variable
```

```
btnwk2     VAR   Byte     'button workspace variable
btnwk3     VAR   Byte     'button workspace variable
btnwk4     VAR   Byte     'button workspace variable
x          VAR   Word     'gen purpose variable
y          VAR   Byte     'gen purpose variable
freq       VAR   Word     'frequency variable
freq1      VAR   Word     'frequency variable
addr1      VAR   Byte     'L EEPROM address variable (0-135)
addr2      VAR   Byte     'C EEPROM address variable (145-288)
loopcnt1   VAR   Byte     'loop counter
loopcnt2   VAR   Word     'loop counter
N96        CON   $40F0    'LCD 9600 baud
suspeed    CON   15000    'start up speed
speed      CON   6500     'pulse width corresponding speed
pau        CON   3        'motor pause
acc        CON   400      'acceleratio

DIRS=%0111110001000000   'P0-P5,P7-P9,P15 set to input, others to output

'================================================================================
'      PROGRAM BEGINS HERE

'Initialize tuner by zeroing stepper motors to home positions. Also called when
'STORE-RESET pushbutton is pushed FOr 1/2 second.

Reset:
          GOSUB longbeep             '3-tone beep
          OUT6 = 1                   'set Auto LED OFF
          DIR9 = 0                   'set P9 (limit detect) to input
          PAUSE 1000                 'wait 1sec for LCD to wake up
          SEROUT 16,n96,[12,27,67,0] 'clear LCD and delete cursor
          PAUSE 1                    'wait 1msec for LCD
          SEROUT 16,n96,["   LC - Tuner    Remote Control "]'display msg on LCD
          PAUSE 2000                 'wait 2 sec
          SEROUT 16,n96,[" Please Wait...  limit detect"]   'display msg on LCD
                                     'and start initialization sequence

'       ==================stepper 1 initialization=========

          OUT14=0                    'set stepper direction to CW
     reset1:
          PULSOUT 12,speed           'pulse stepper1
          PAUSE pau
          IF IN9=0 THEN upstep1      'is limit reached?
          GOTO reset1                'if not,loop until limit reached
     upstep1:
          stpcnt1=4150               'set stpcnt1 to max range
          TOGGLE 14                  'set stepper direction to CW
```

```
      upstep11:
           PULSOUT 12,suspeed         'move to free up limit
           PAUSE pau
           IF IN9=0 THEN upstep11

'      ==================stepper 2 initialization===========

           OUT14=1                    'set stepper direction to CCW
      reset2:
           PULSOUT 13,3000            'pulse stepper2
'           PAUSE pau
           IF IN9=0 THEN upstep2      'limit reached?
           GOTO reset2                'if not,loop until limit reached
      upstep2:
           stpcnt2=4400               'set stpcnt2 to upper limit
           TOGGLE 14                  'set stepper direction to CW
      upstep21:
           PULSOUT 13,speed           'move to free up limit
           PAUSE pau
           IF IN9=0 THEN upstep21

'======================Initialize LCD and display opening message==============================

   SEROUT 16,n96,[27,79,1,2,"reload settings"]'disp. startup msg
   PAUSE 1000                      'wait 1 sec
   GOSUB Format_dis                'and format display

'======================Set tuner to ON-LINE and AUTO mode=====================================

   OUT9=1                          'set mode to ON-Line
   PUT 2,0                         'Set mode flag to AUTO
   OUT6=0                          'Light LED

'======================Retrieve settings from last turn-off===================================

'Input memory address stored at last turn-off, compute the frequency segment and move steppers.
'This routine used after power-on and when stepping through memories. It bypasses get_freq
'and band lookup routine.

   READ    $120,addr1                 'Get last-used L memory address from EEPROM

comp_freq:                            'Input addr1 and branch to corresponding band.

   LOOKDOWN addr1,<[20,50,66,70,88,90,108,110,144],x
   BRANCH x,[freq160,freq80,freq40,freq30,freq20,freq17,freq15,freq12,freq10]

'======================Measure transmit frequency=============================================
```

```
'Measure transmitter frequency at P15. First,poll P15 for 400uS to
'see if signal present. If no signal detected (x=0), skip to rd_buttons.
'Purpose is to avoid slowing down loop unless valid signal present, and to prevent
'errors by ensuring signal is present when freq-measuring gate opens.

'Second part of routine measures frequency twice, with gate of 250*0.4usec=0.1sec,
'and requries both measurements to agree. Reason for two measurements is to prevent errors caused by
'tracking of modulated SSB signals.

get_freq:
      COUNT 15,1,x                        'poll for signal, return 0 if NO,low integer if YES
      IF x=0 THEN rd_buttons              'skip if no signal present
      COUNT   15,250,freq                 'otherwise,measure frequency of signal
      COUNT 15,250,freq1                  'measure it again
      IF NOT freq = freq1 THEN rd_buttons  'skip if measurements not the same
      freq=freq+(freq/freqtrim)           'trim freq to correct for BS2sx clock innacuracy

                        'now go to freq_tune

'=============================================================================================

' Input transmit frequency from get_freq. Confirm if in amateur band, compute memory addresses and frequency
' segment corresponding to the addresses. Then retrieve settings from memory and move steppers to settings.
' If frequency not in amateur band, go back to get_freq and poll TX frequency.
' Display shows the lower end of the memory frequency segment, not the actual TX freq.
' Note that addr1 (address of C1) is computed from the get_freq TX frequency. Addr2 tracks addr1
' but is offset by $090 (addr2=addr1+$090).

freq_tune:

      LOOKDOWN freq,<[1810,2001,3500,3801,7000,7201,10100,10151,14000,14351,18068,18169,21000,21451,24890,24991,28000,29701],x
      BRANCH x,[get_freq,band160,get_freq,band80,get_freq,band40,get_freq,band30,get_freq,band20,get_freq,band17,get_freq,band15,
                get_freq,band12,get_freq,band10]
      GOTO get_freq                       'if freq>29700, take no action & read the buttons

band160: addr1=freq/20*2-180              'addr1=0-19,addr2=144-163 for 160m band, 20kHz steps
                                          'addr1=(freq/steps)*2 - 180 ; division by 2 looses dec value
freq160: freq=addr1*10+1820               'compute freq corresponding to addr1
                                          '(freq=Steps(20kHz)*Adr Bandedge+Diff TO Bandedge)
                                          'Const = freq Bandedge - Steps * addr1
         GOTO mem_step                    'retrieve settings and move steppers

band80:  addr1=freq/20*2-330              'addr1=20-49,addr2=164-193 for 80m band, 20kHz steps
freq80:  freq=addr1*10+3310               'compute freq corresponding to address
         GOTO mem_step                    'retrieve settings and move steppers

band40:  addr1=freq/25*2-510              'addr1=50-65,addr2=194-209 for 40m band, 25kHz steps
freq40:  freq=(addr1*25)/2+6387           'compute freq corresponding to addr1
```

```
         GOTO mem_step                    'retrieve settings and move steppers

band30:  addr1=freq/25*2-742              'addr1=66-69, addr2=210-213 for 30m band, 25kHz steps
freq30:  freq=(addr1*25)/2+9282           'compute freq corresponding to addr1
         GOTO mem_step                    'retrieve settings and move steppers

band20:  addr1=freq/40*2-630              'addr1=70-87, addr2=214-231 for 20m band, 40kHz steps
freq20:  freq=addr1*20+12620              'compute freq corresponding to addr1
         GOTO mem_step                    'retrieve settings and move steppers

band17:  addr1=88                         'addr1=88-89, addr2=232-233 for 17m band, 100kHz steps
freq17:  freq=18068                       'compute freq corresponding to addr1
         GOTO mem_step                    'retrieve settings and move steppers

band15:  addr1=freq/50*2-750              'addr1=90-107, addr2=234-251 for 15m band, 50kHz steps
freq15:  freq=addr1*25+18775              'compute freq corresponding to addr1
         GOTO mem_step                    'retrieve settings and move steppers

band12:  addr1=108                        'addr1=108-109, addr2=252-253 for 12m band, 100kHz steps
freq12:  freq= 24890                      'compute freq corresponding to addr1
         GOTO mem_step                    'retrieve settings and move steppers

band10:  addr1=freq/100*2-450             'addr1=110-143, addr2=254-287 for 10m band, 100kHz steps
freq10:  freq=addr1*50+22550              'compute freq corresponding to addr1
         GOTO mem_step                    'retrieve settings and move steppers

'=================================================================================================

' Input addr1 and compute addr2,
' Then retrieve the settings from memory, and move both steppers to the memory settings.
' When done, go back and check for new frequency. Also engage ext. cap. if freq is in 160m band.

mem_step:
      addr2=addr1+$090                      'compute addr2
      GET 2,x                               'check if auto (0) or manual (1) mode
      BRANCH x,[ms1,manu1]

   ms1:
      READ addr1,  memval1.LOWBYTE          'retrieve memval for addr1
      READ addr1+1,memval1.HIGHBYTE
      IF memval1 > stpcnt1 THEN  CWmemstep1 'test for CW or CCW direction
      IF memval1 < stpcnt1 THEN CCWmemstep1 'and move stepper

   ms2:
      READ addr2,  memval2.LOWBYTE          'retrieve memval for addr2
      READ addr2+1,memval2.HIGHBYTE
      IF memval2 > stpcnt2 THEN CCWmemstep2
      IF memval2 < stpcnt2 THEN CWmemstep2
```

```
 msend: GOSUB display
        GOTO get_freq                       'if no change, check for new frequency

'=============================================================================================

' Stepper driver routines for CW and CCW rotation.

CWmemstep1:   OUT14=0                       'set stepper1 to CW
              GOTO pulse1

CWmemstep2:   OUT14=0                       'set stepper2 to CW
              GOTO pulse2

'      ====================================================

CCWmemstep1:  OUT14=1                       'set stepper1 to CCW
              GOTO pulse1

CCWmemstep2:  OUT14=1                       'set stepper2 to CCW
              GOTO pulse2

'      ====================================================

pulse1:       stp = ABS(memval1-stpcnt1)    'calculate no. of steps needed
              GOSUB flash                   'set cursor flashing
              FOR x = 1 TO stp
              PULSOUT 12,speed              'send stepper to destination
              PAUSE pau                     'wait for stepper to finish
              NEXT
              stpcnt1=memval1               'update stpcnt1
              GOTO ms2

pulse2:       stp = ABS(memval2-stpcnt2)    'calculate no. of steps needed
              GOSUB flash                   'set cursor flashing
              FOR x = 1 TO stp
              PULSOUT 13,3000                'send stepper to destination
              PAUSE pau                     'wait for stepper to finish
              NEXT
              stpcnt2=memval2               'update stpcnt2
              GOTO msend                    'and return

'=============================================================================================

' Read the UP, DOWN, MODE/STORE, and IN-OUT/RESET buttons. If pressed, take appropriate action.
' and if not pressed, read the manual buttons

rd_buttons:
```

```
      BUTTON 5,0,254,1,btnwk1,1,upbutton       'UP button pressed?
      BUTTON 4,0,254,1,btnwk2,1,dwnbutton      'DOWN button pressed?
      BUTTON 8,0,254,1,btnwk3,1,automode       'set AUTO MODE
stbtn:BUTTON 7,0,254,255,btnwk4,1,store_reset  'STORE if 1 press, RESET if held down

' The variable loopcnt1 is used to count loops for the STORE/RESET button, which has
' a second function if held down.

      loopcnt1=loopcnt1+1 MAX 255              'count loops for STORE & RESET buttons
      BRANCH OUT9,[stbtn,rd_manbutt]           'if OFF-LINE read I/O button until changed

'=============================================================================================

      automode:   PUT 2,0                     'change mode flag to auto
                  OUT6 = 0                    'AUTO LED ON
                  GOSUB beep                  'short beep to confirm press
                  GOTO comp_freq

'=============================================================================================

'The IO_reset BUTTON instruction checks status of port 7. Normally,the loop counter = 255
'when the button is first pressed, the routine stores the current settings and resets loopcnt to 0.
'After a delay of 254 loops, the button status is checked again, and if still pressed the routine
'branches To RESET.

store_reset:IF loopcnt1=253 THEN reset          'reset if loop counter=254, otherwise store current settings
            loopcnt1=0                          'reset loop counter

'================== Long Press - store settings ======================================

store:      WRITE addr1,  stpcnt1.LOWBYTE    'save settings
            WRITE addr1+1,stpcnt1.HIGHBYTE
            WRITE addr2,  stpcnt2.LOWBYTE
            WRITE addr2+1,stpcnt2.HIGHBYTE
            WRITE $120,addr1
            GOSUB longbeep                   'long beep to confirm store (3 tones)
            PAUSE 500                        'wait 1/2 sec for button to be released
            GOTO  rd_manbutt                 'and go read the manual buttons
'======================================== Read manual buttons and move steppers ====================

rd_manbutt:
  butt1:
         IF IN0=1 THEN butt2      'read port P0, if no read next manual button
         PUT 2,1                  'if pressed, set mode flag to Manual
         OUT6=1                   'turn off LED
         GOSUB flash              'set cursor flashing
         OUT14=0                  'set rotation for CW
         x=suspeed                'set startup speed
```

```
    pulse1_cw:
         IF stpcnt1=4150 THEN pulse1_cw_end    'upper limit
         x = x-acc MIN speed      'ramp up speed
         stpcnt1 = stpcnt1 + 1    'increment Stepcounter
         PULSOUT 12,x             'move stepper 1
         PAUSE pau
         IF IN0=0 THEN pulse1_cw  'Button pressed?
         GOSUB DISPLAY            'update L on display
         GOTO butt2

    pulse1_cw_end
         SEROUT 16,n96,[27,79,3,1," >> "]   'display L upper limit

         '==================================

  butt2:
         IF IN1=1 THEN butt3       'read port P1, L>CCW if no read 3rd manual button
         PUT 2,1                   'if changed, set mode flag to Manual
         OUT6=1                    'turn off LED
         GOSUB flash               'set cursor flashing
         OUT14=1                   'set rotation to CCW
         x=suspeed                 'set startup speed
   pulse1_CCW:
         IF stpcnt1=0 THEN pulse1_ccw_end         'lower limit
         x = x-acc MIN speed       'ramp up speed
         stpcnt1 = stpcnt1 - 1     'decrement Stepcounter 1
         PULSOUT 12,x              'move stepper1
         PAUSE pau
         IF IN1=0 THEN pulse1_ccw  'Button pressed?
         GOSUB DISPLAY
         GOTO butt3
  pulse1_ccw_end:
         SEROUT 16,n96,[27,79,3,1," << "]   'display L lower limit
         GOTO butt3

'=============================================================================================

    butt3:
         IF IN2=1 THEN butt4       'read port P3, decrease capacity if no read next button
         PUT 2,1                   'if changed, set mode flag to Manual
         OUT6=1                    'turn off LED
         GOSUB flash               'set cursor flashing
         OUT14=1                   'set rotation to CCW
         x=suspeed                 'set startup speed
   pulse2_CCW:
         IF stpcnt2=4400 THEN pulse2_ccw_end         'lower limit limit
         x = x-acc MIN speed       'ramp up
         stpcnt2 = stpcnt2 + 1     'increment Stepcounter 2
```

```
         PULSOUT 13,x              'move stepper2
         PAUSE pau
         IF IN2=0 THEN pulse2_ccw  'Button pressed?
         GOSUB DISPLAY
         GOTO butt4
  pulse2_ccw_end:
        SEROUT 16,n96,[27,79,11,1," >> "]   'display C lower limit

        '=========================================================
   butt4:
         IF IN3=1 THEN ck_mode    'read ports P2, increase capacity if no loop in manual mode
         PUT 2,1                  'if pressed, set mode flag to Manual
         OUT6=1                   'turn off LED
         GOSUB flash              'set cursor flashing
         OUT14=0                  'set rotation for CW
         x=suspeed                'set start-up speed
   pulse2_cw:
         IF stpcnt2= 0 THEN pulse2_cw_end      'upper limit
         x = x-acc MIN speed      'ramp up
         stpcnt2 = stpcnt2 - 1    'decrement Stepcounter 2
         PULSOUT 13,x             'move stepper 2
         PAUSE pau
         IF IN3=0 THEN pulse2_cw  'Button pressed?
         GOSUB DISPLAY
         GOTO rd_buttons          'loop in MANUAL mode
    pulse2_cw_end:
        SEROUT 16,n96,[27,79,11,1," << "]   'display C upper limit
        GOTO rd_buttons           'loop in MANUAL mode

'=============================================================================================
' Check for AUTO or MANUAL mode, and after 5000 loops of no activity (to avoid excessive
' writing to EEPROM) store updated L address (addr1) in EEPROM.

ck_mode: loopcnt2=loopcnt2 + 1              'increment loop counter
         IF loopcnt2=5000 THEN store_last   'store L address after 5000 loops of no activity
ck_md1:  GET 2,x                           'retrieve mode (0=auto,1=manual)
         BRANCH x,[get_freq,rd_buttons]    'and branch accordingly.

'=============================================================================================

'Store current L address (addr1) in EEPROM at $120 after 5000 loops and if
'addr1 has changed.

store_last:
         READ $120,memval1                 'get last stored addr1
         IF memval1=addr1 THEN resetlc2    'skip if no change
         WRITE $120,addr1                  'if changed, store new addr1
```

```
resetlc2:
         loopcnt2=0                        'reset loopcnt2 after store
         GOTO ck_md1                       'go back to ck_mode

'=============================================================================================

'Increment or decrement indices and prepare update of stepper settings if UP or DOWN button pressed.
'Change to MANUAL mode and step through frequency segments.
'Settings are activated by pressing AUTO-button

upbutton:
         PUT 2,1                           'if pressed, set mode flag to Manual
         OUT6=1                            'turn off LED
         addr1 = addr1 + 2 MAX 142
         GOTO comp_freq

dwnbutton:
         PUT 2,1                           'if pressed, set mode flag to Manual
         OUT6=1                            'turn off LED

         IF addr1=0 THEN comp_freq         'exit if addr1 is zero
         addr1 = addr1 - 2
         GOTO comp_freq

manu1:
        GOSUB display                      'update display
        GOTO rd_buttons


'=============================================================================================

'Display stepper counts and frequency segments

' calculate L:
' Lo  =  2,5uH
' Lmax= 19,1uH
' calculate value*10 for decimal point
' 1 turn = 200 steps = 0,8uH

' calculate C:
' Co   =  30pF
' Cmax =1500pF
' 1 turn = 200 steps = 65pF

display:
  'calculate inductivity
        memval1 = (stpcnt1 * 8/200)+25         '0.8uH per rotation
        Y       = memval1 DIG 0                'decimal digit
```

```
        memval1 = memval1/10

  'calculate capacity:
        memval2 = stpcnt2/10*68/20+20          'stpcnt2/10 * 68 / 200/10
                                               'value devided by 10 to avoid overflow)
                                               '68pF per rotation

'Format LCD for displaying "L:", "C:", and "Freq:" at following locations:
'L: C1_L1; C: C9_L1; Freq: C1_L2
'L integer-value = memval1 at location: C3_L1; decimal point: C5_L1; dec-value = Y: C6_L1; freq: C6_L2

SEROUT 16,n96,[27,67,0]                        'delete cursor
SEROUT 16,n96,[27,79,3,1,DEC2 memval1,27,79,5,1,$2E,27,79,6,1,DEC y,27,79,11,1,DEC4 memval2,27,79,6,2,DEC5 freq]

           IF memval1 < 10   THEN format1      'if leading zero for L
form2:     IF memval2 < 1000 THEN format2      'if leading zero for C
form3:     IF memval2 < 100  THEN format3      'if leading zero for C
form4:     IF freq    < 10000THEN format4      'if leading zero for freq
           RETURN

format1:   SEROUT 16,n96,[27,79,3,1," "]       'leading zero 10 uH
           GOTO form2
format2:   SEROUT 16,n96,[27,79,11,1," "]      'leading zero 1000 pF
           GOTO form3
format3:   SEROUT 16,n96,[27,79,12,1," "]      'leading zero 100 pF
           GOTO form4
format4:   SEROUT 16,n96,[27,79,6,2," "]       'leading zero for freq
           RETURN

Flash:     SEROUT 16,n96,[27,79,15,2,27,67,3]  'position cursor and set flashing
           RETURN

Format_Dis:SEROUT 16,n96,[27,79,1,1,"L:    u",27,79,9,1,"C:    pF",27,79,1,2,"Freq:      kHz "] 'format display
           RETURN
'================================================================================
beep:                                          'short beep
   FREQOUT 4,250,1000
   RETURN

'================================================================================
longbeep:                                      'long (3-tone) beep
   FREQOUT 4,300,800
   FREQOUT 4,300,1000
   FREQOUT 4,500,1200
   RETURN
'  END of Program
```