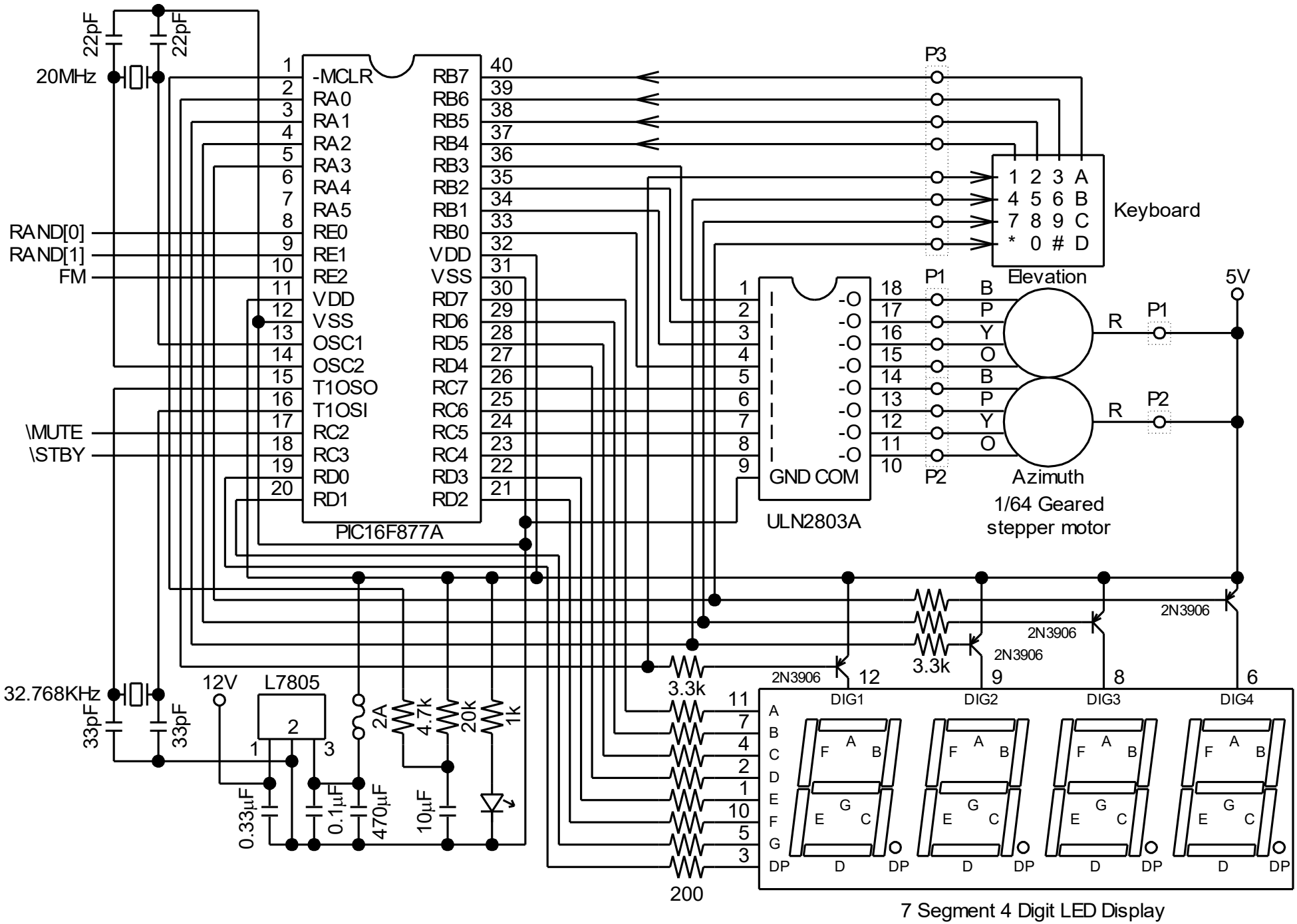
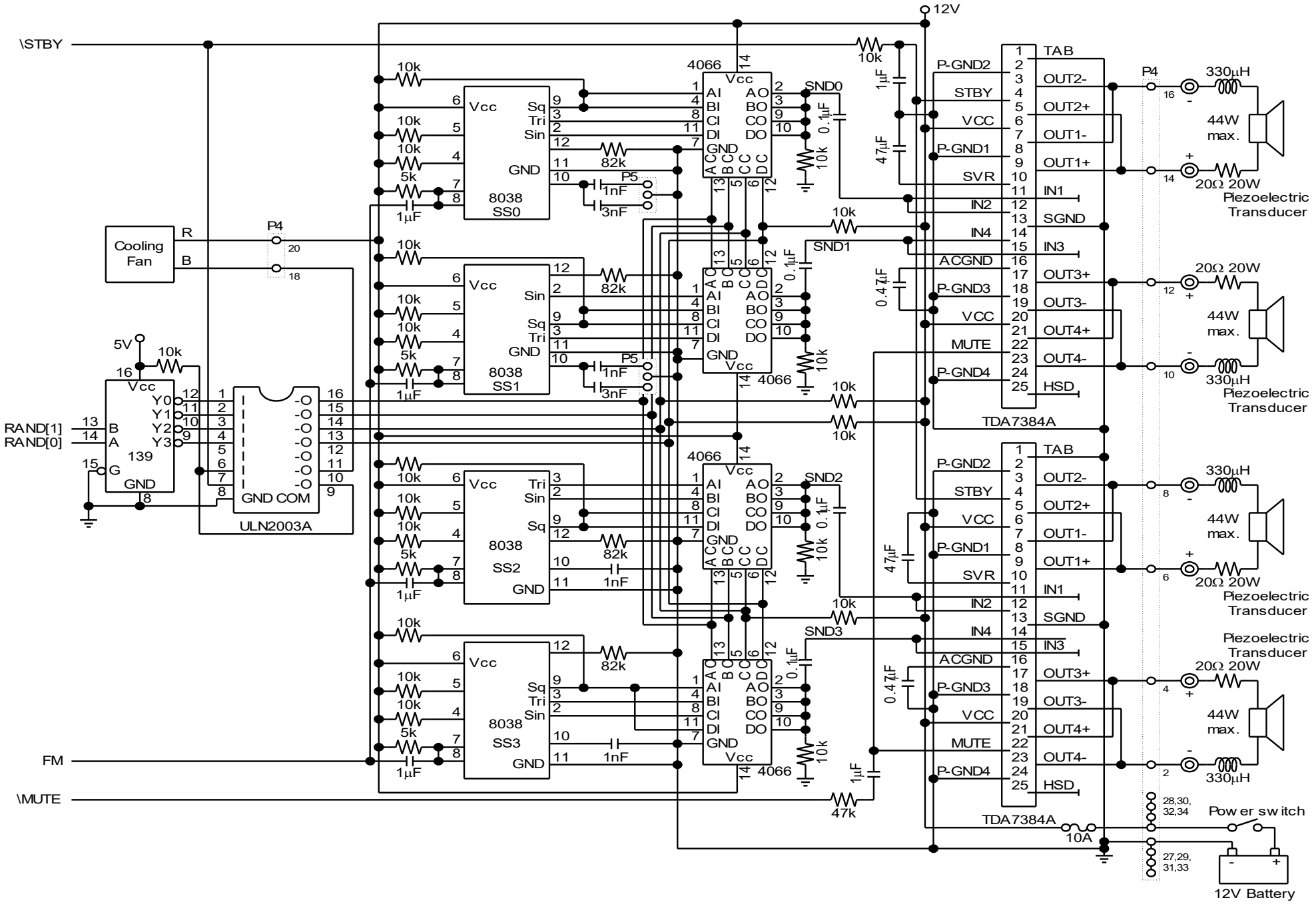
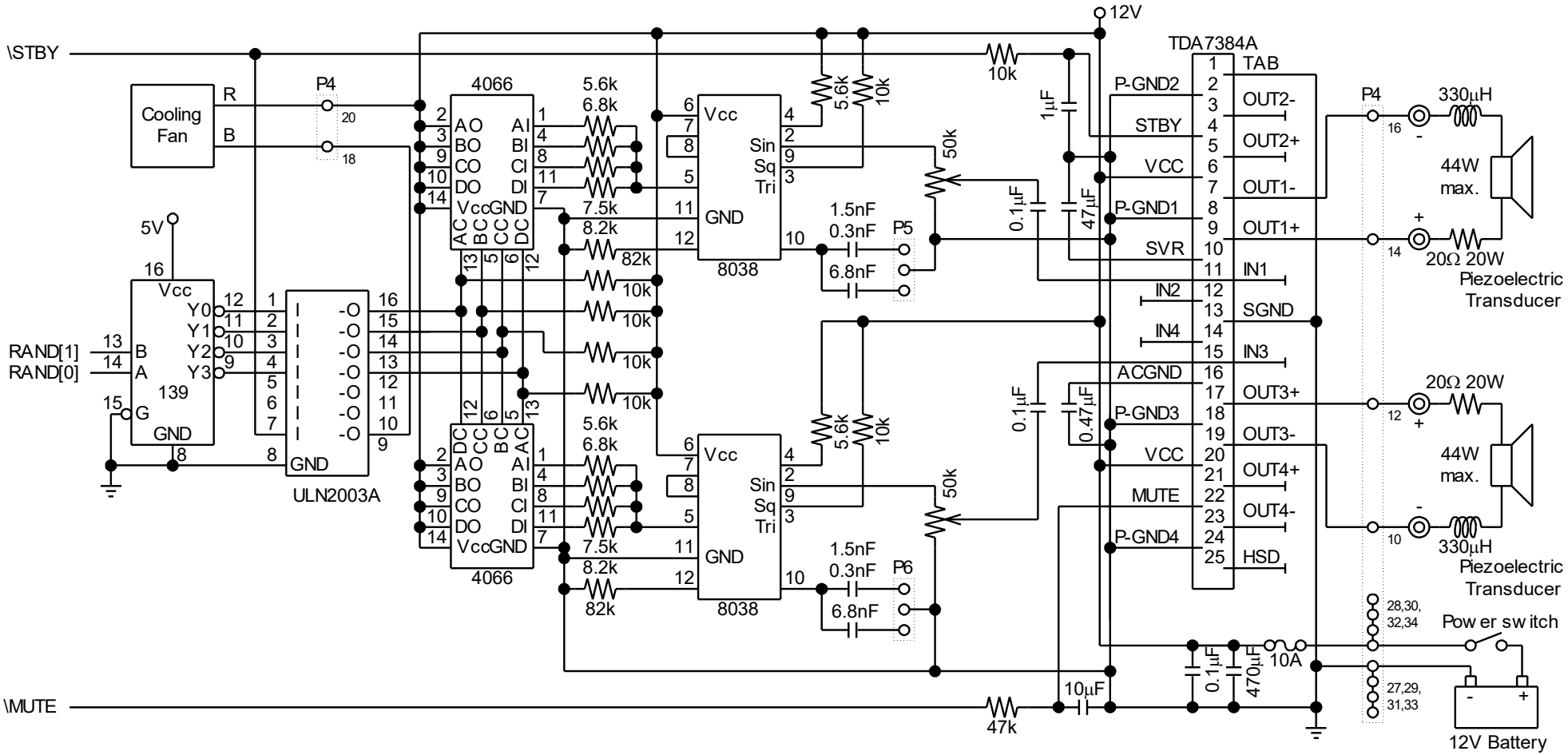


Power Supply & Digital Processing Portion (5V)

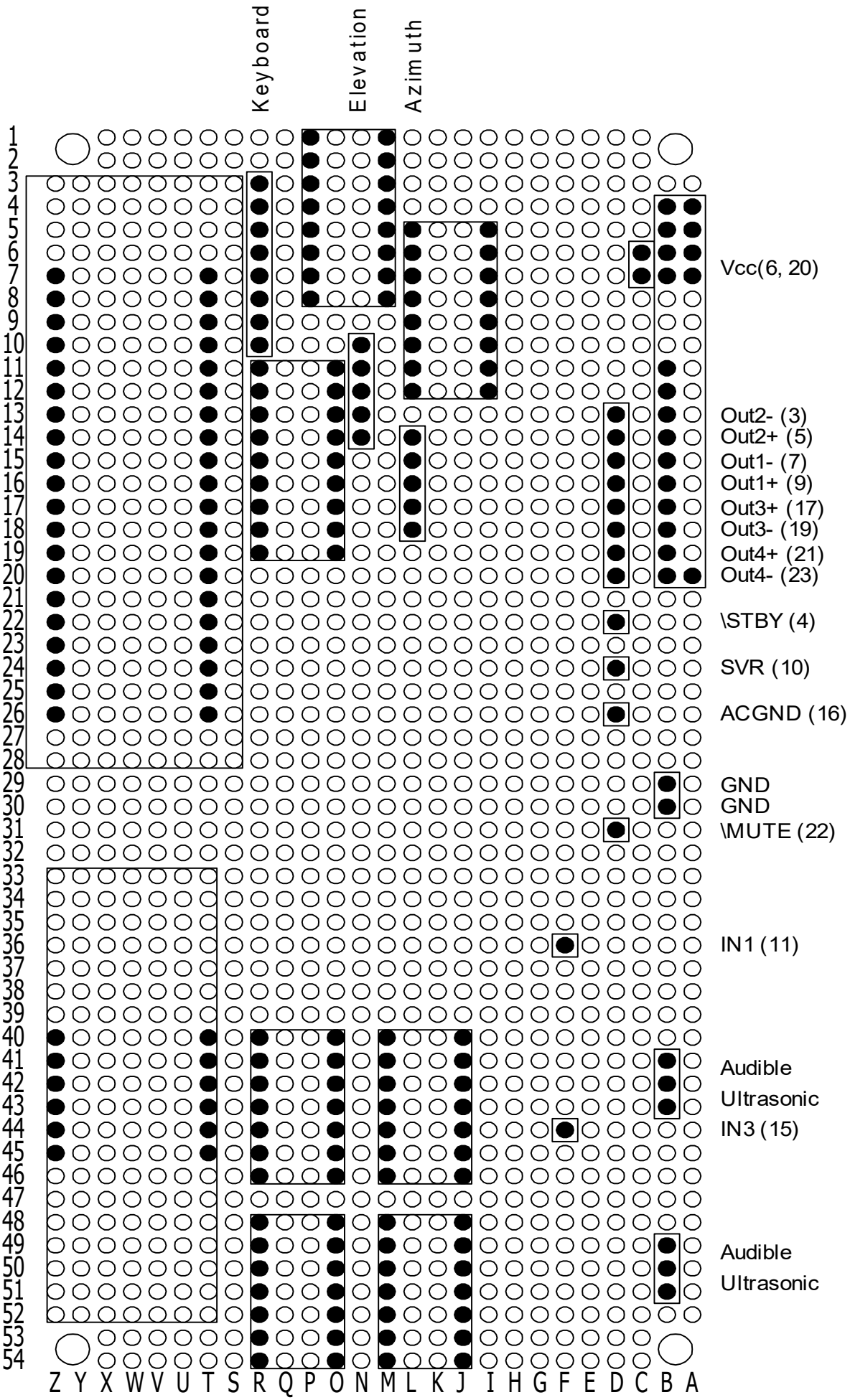


Ultra Sound Randomization & Amplifier (12V)



Ultra Sound Randomization & Amplifier (12V) Ver.2

Layout of General Purpose Printed Circuit Board

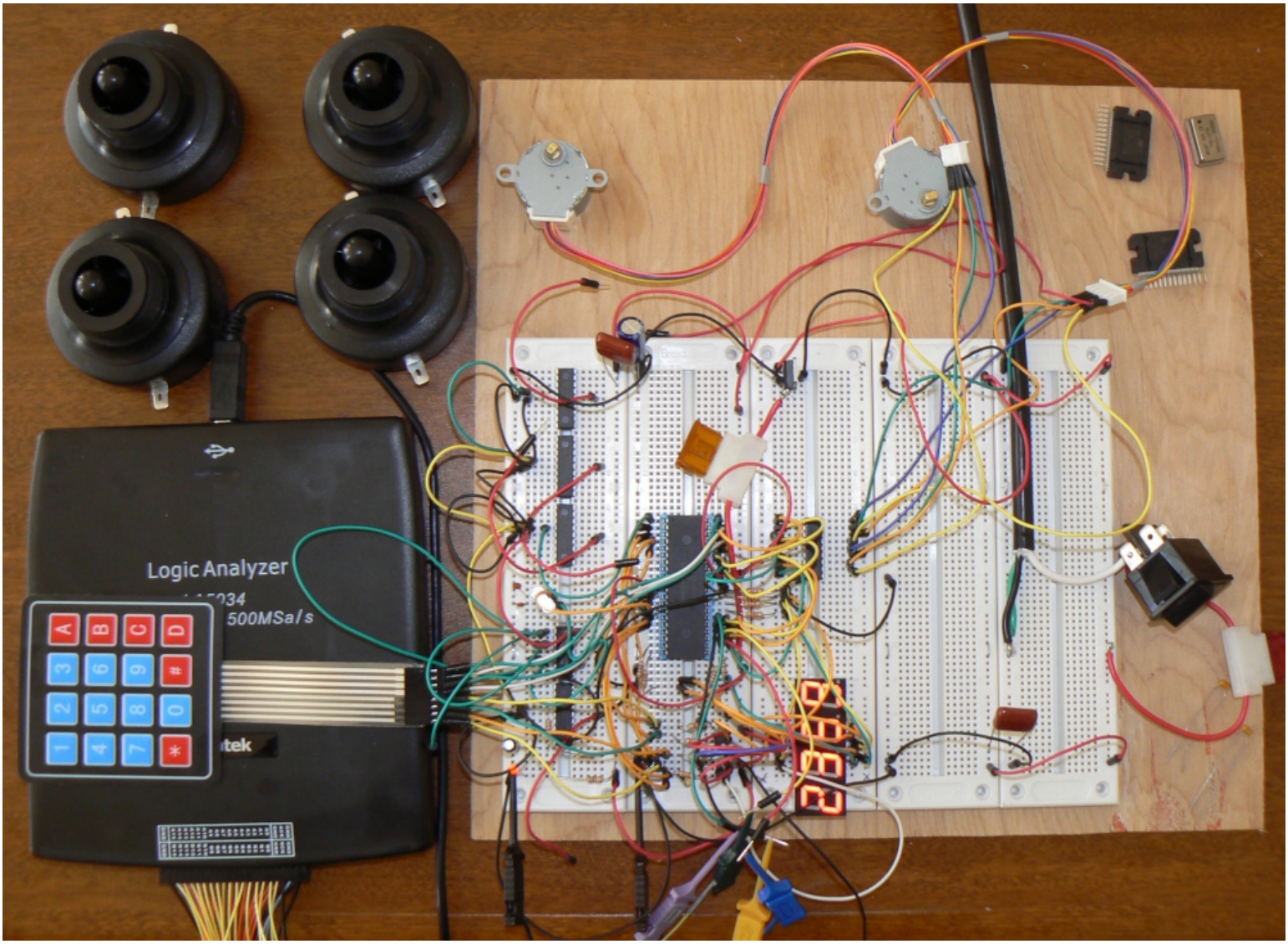


BOM (Bill Of Materials)

Name	Function	Piece(s)	Price ($)
PIC 16F877A	Stepper motor/Mute/Stand-by/Base time clock/Timer/Keyboard/Positioning	1	2.98
ULN2803A	500mA 8 Darlington Drivers for driving 2 stepper motors	1	0.46
28YBJ-48	4 phase 1/64 Geared Stepper Motor x 2 for azimuth & elevation sweep	2	10.00
TL084	Quad op-amp for inverters & 5V-to-12V shifters	1	0.60
8038	Waveform generators of sinusoidal/square/triangle waves with FM&duty control	4	3.00
4066	Quad analog switch for ultra sonic sound randomization	4	1.60
139	Dual 2 to 4 decoder	1	0.00
TDA7384A	4 channel 22W amplifier for sound main amplifier (Total output 174W)	2	12.29
Piezoelectric Transducer	50W output for ultrasonic sound generation	4	80.00
Keyboard	4 inputs x 4 outputs for system level debug and control	1	2.25
LED display unit	7 segment 4 digit LED display for system level debug and control	1	0.72
L7805	5V regulator	1	0.32
20MHz crystal	For processor clock	1	0.30
32.768KHz crystal	For base time clock	1	0.25
2N3906	PNP transistor for 4 digit LED display unit digit drive	4	0.00

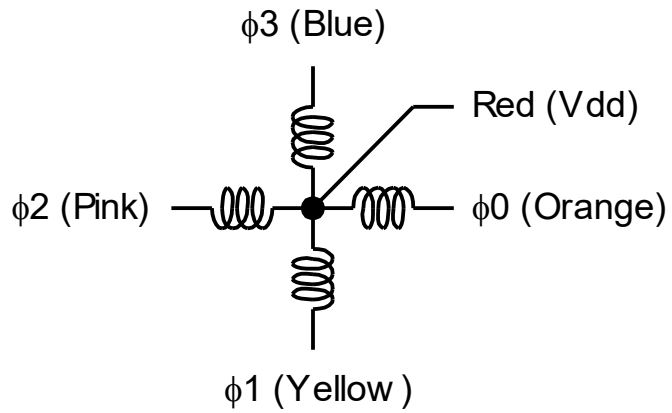
Total = $114.77

Firmware is written on a PIC16F877A using assembly language and programmed by a PIC programmer. Functional verification is done by firmware debugger along with system level debug aid done through keyboard & display as well as a logic analyzer assisted by PC through USB.
Ultra sound waveform, frequency, frequency modulation, duty control are observed by an oscilloscope and adjusted.

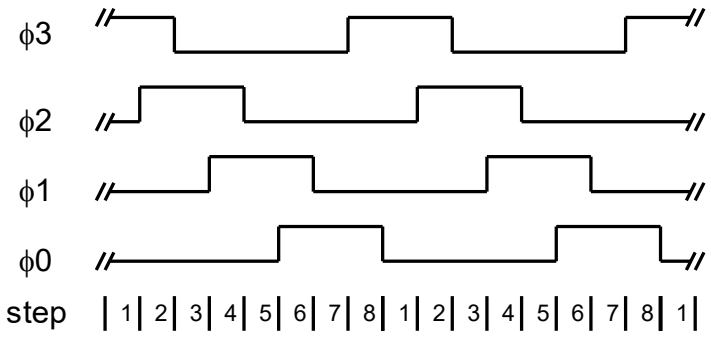


Breadboard connected with 500M samples per second Logic Analyzer

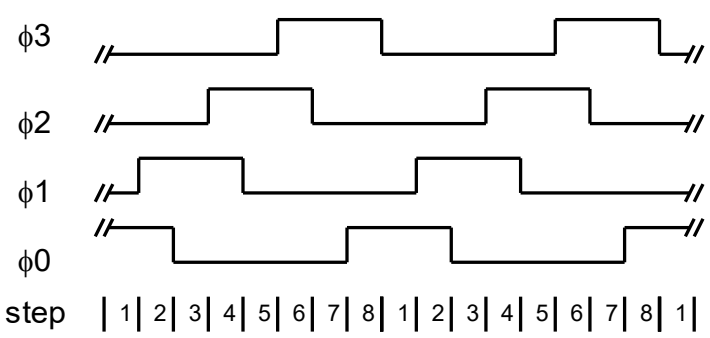
Stepper Motor Drive



28YBJ-48
1/64 Geared stepper motor



Turning clockwise
(8 cycle half steps)



Turning counterclockwise
(8 cycle half steps)

Step	ϕ3	ϕ2	ϕ1	ϕ0	ϕ[3:0]
1	1	1	0	0	c
2	0	1	0	0	4
3	0	1	1	0	6
4	0	0	1	0	2
5	0	0	1	1	3
6	0	0	0	1	1
7	1	0	0	1	9
8	1	0	0	0	8

Initial value = 0000b

Stepper motor operation mode (Step_mode[7:0])

7	6	5	4	3	2	1	0	Operation
-	-	1/0	-	-	-	-	-	Elevation Up/Down
-	-	-	1/0	-	-	-	-	Elevation On/Off
-	-	-	-	-	-	1/0	-	Azimuth Right/Left
-	-	-	-	-	-	-	1/0	Azimuth On/Off

Key input mode (Keyin_mode[7:0])

7	6	5	4	3	2	1	0	Operation
-	-	cnt_1	cnt_0	-	-	-	-	Keyin count
-	-	-	-	1/0	-	-	-	travel_cnt
-	-	-	-	-	1/0			End timer
-	-	-	-	-	-	1/0		Start timer
-	-	-	-	-	-	-	1/0	clock

Key prefix (Key_prefix[7:0])

7	6	5	4	3	2	1	0	Operation
-	-	-	-	-	-	-	1/0	A
-	-	-	-	-	-	1/0	-	B
-	-	-	-	-	1/0	-	-	C
-	-	-	-	1/0	-	-	-	D
-	-	-	1/0	-	-	-	-	*
-	-	1/0	-	-	-	-	-	#

Display mode (Disp_mode[7:0])

7	6	5	4	3	2	1	0	Operation
1/0-	-	-	-	-	-	-	-	On/Off
-	-	-	1/0	-	-	-	-	Travel counter
-	-	-	-	1/0	-	-	-	Key input
-	-	-	-	-	1/0	-	-	End timer
-	-	-	-	-	-	1/0	-	Start timer
-	-	-	-	-	-	-	1/0	clock

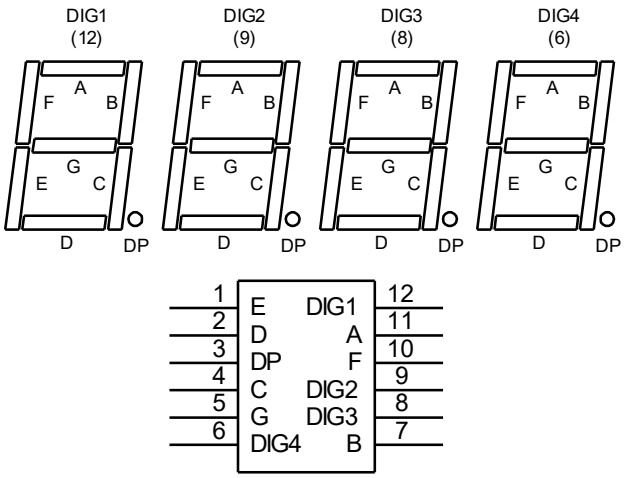
Pin Function of 16F877A

Port name	Pin name	Pin functions	I/O functions	Initial value	Interrupt
RA[5:4]		<No use>	I	-	-
RA[3:0]	~DT[3:0]	Digit timing / Key scan	O	1111	-
RB[7:4]	~KEY[3:0]	Keyboard	IPU	-	
RB[3:0]	E_DRV[3:0]	Elevation motor drive	O	0000	-
RC[7:4]	A_DRV[3:0]	Azimuth motor drive	O	0000	-
RC[3]*	STBY	Stand-by	O	1	-
RC[2]*	MUTE	Mute	O	1	-
RC[1]	T1OSI	32.768 KHz oscillator	CLK	-	TMR1 overflow
RC[0]	T1OSO	32.768 KHz oscillator	CLK	-	TMR1 overflow
RD[7:0]	~SEG[7:0]	Segment drive	O	11111111	-
RE[2]*	FM	Frequency modulation	O	1	-
RE[1:0]*	RAND[1:0]	Randomization	O	11	-

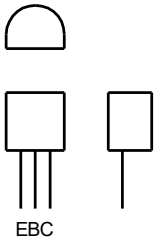
Interrupts (Global interrupt enable = INTCON[7], Peripheral interrupt enable = INTCON[6])

Timer 0 overflow	
Source	Fosc/4 ((20 / 4) MHz)
Prescaler	1/64
Initial value	TMR0[7:0] = 0
Period	(20,000,000 / 4) / 64 / 256 = 305.2 Hz (3.2 mSec)
Purpose	Transition timing for ~DT[3:0] & ~SEG[7:0], V_DRV[3:0] & H_DRV[3:0]
Interrupt enable	INTCON[5]
Interrupt flag	INTCON[2]
Timer 1 overflow	
Source	T1OSI (32.768 kHz)
Prescaler	1/1
Initial value	TMR1H[7:0] = 0 & TMR1L[7:0] = 0
Period	2 seconds
Purpose	Clock & Timer
Interrupt enable	PIE1[0]
Interrupt flag	PIR1[0]

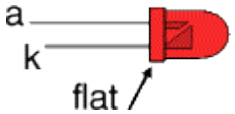
7 Segment Digital Display



HSX-410561K-D30

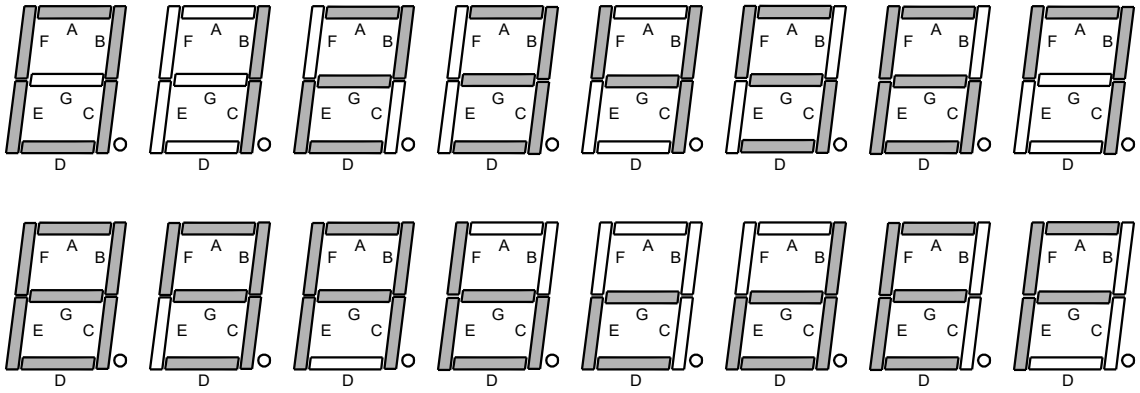


2N3906
pnp transistor

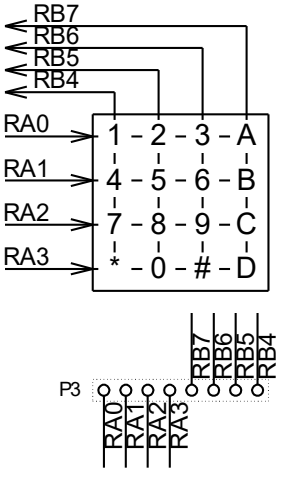


Infra red LED

Segment Drive



16 Key Keyboard



Key Encoding Scheme

		RB[7:4] = Key_image			
Dt_image	RA[3:0]	1110	1101	1011	0111
0	1110	1	2	3	0ah
1	1101	4	5	6	0bh
2	1011	7	8	9	0ch
3	0111	0eh	0	0fh	0dh

Key Functions

Function	Key sequence	Display mode	Key-in mode	Motor mode	Register
System reset	A - 0	On, Clock			
Display clock time	A - 1				
Display start timer	A - 2	On, Start timer			
Display end timer	A - 3	On, End timer			
Specify clock time	A - 4 - HHMM	On, Key input	Clock	--	{Hour[1:0], Minute[1:0]}
Specify start timer	A - 5 - HHMM	On, Key input	Start timer		{Hour_st[1:0], Minute_st[1:0]}
Specify end timer	A - 6 - HHMM	On, Key input	End Timer		{Hour_et[1:0], Minute_et[1:0]}
	A - 7				
	A - 8				
Display off	A - 9	Off, Clock			

Function	Key sequence	Display mode	Key-in mode	Motor mode	Register
Stepper motors off	B - 0	On, Travel count		Off, left	
Azimuth left swing	B - 1			Azimuth on, left	
Azimuth right swing	B - 2			Azimuth on, right	
Elevation up swing	B - 3			Elevation on, up	
Elevation down swing	B - 4			Elevation on, down	
Specify travel count of Swing - 1	B - 5 - 0TTT	On, Key input	Travel_cnt	--	Travel_cnt_0
Recall travel count of Swing - 1	B - 5 - 1				
Specify travel count of Swing - 2	B - 6 - 0TTT				Travel_cnt_1
Recall travel count of Swing - 2	B - 6 - 1				
Specify travel count of Swing - 3	B - 7 - 0TTT				Travel_cnt_2
Recall travel count of Swing - 3	B - 7 - 1				
Specify travel count of Swing - 4	B - 8 - 0TTT				Travel_cnt_3
Recall travel count of Swing - 4	B - 8 - 1				
Specify travel count of Swing - 5	B - 9 - 0TTT				Travel_cnt_4
Recall travel count of Swing - 5	B - 9 - 1				

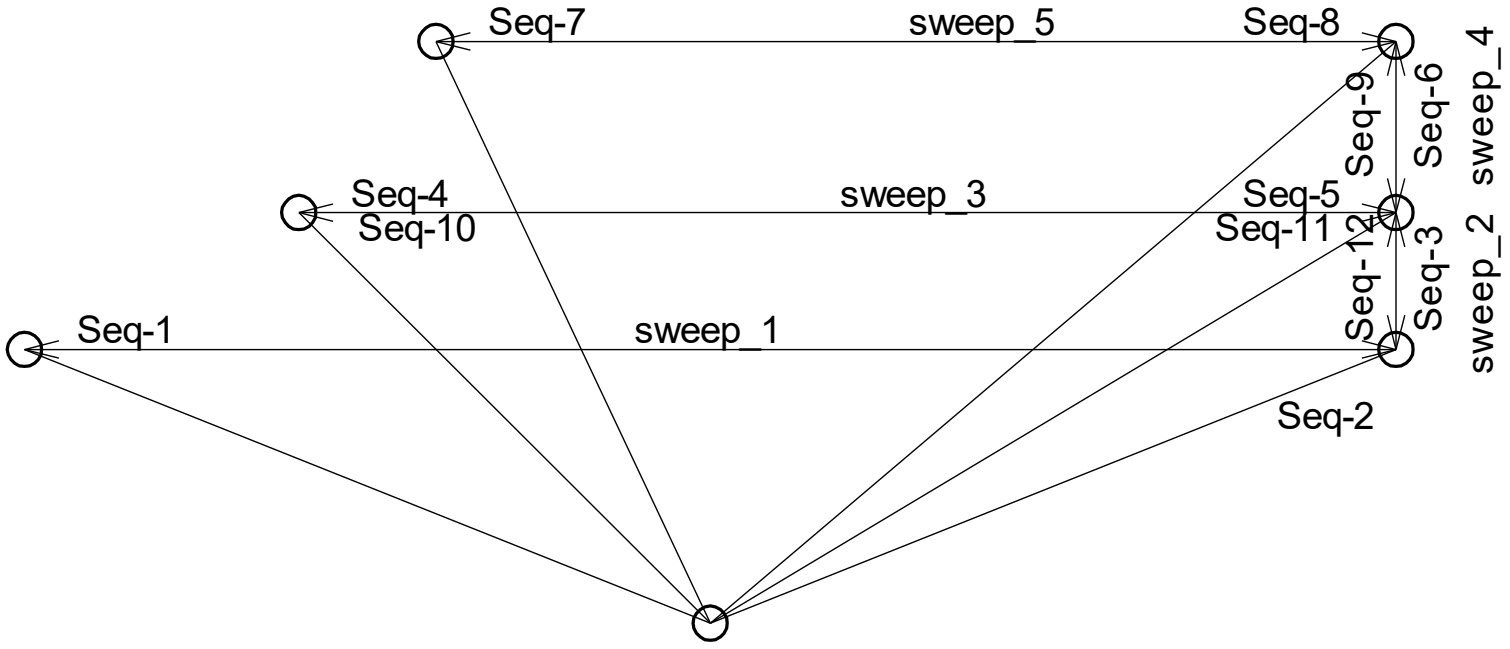
Function	Key sequence	Display mode	Motor mode	Register
Sweep - 1 (left)	C - 0	On, Travel count	Azimuth on to off, left	Travel_cnt_0
Sweep - 1 (right)	C - 1		Azimuth on to off, right	
Sweep - 2 (up)	C - 2		Elevation on to off, up	Travel_cnt_1
Sweep - 2 (down)	C - 3		Elevation on to off, down	
Sweep - 3 (left)	C - 4		Azimuth on to off, left	Travel_cnt_2
Sweep - 3 (right)	C - 5		Azimuth on to off, right	
Sweep - 4 (up)	C - 6		Elevation on to off, up	Travel_cnt_3
Sweep - 4 (down)	C - 7		Elevation on to off, down	
Sweep - 5 (left)	C - 8		Azimuth on to off, left	Travel_cnt_4
Sweep - 5 (right)	C - 9		Azimuth on to off, right	

Function	Key sequence	Display mode	Motor mode	Register
Run (Repeat Seq - 1 to Seq - 12)	* - 0	On, Travel count	All combinations	Travel_cnt_N
Run (Seq - 1 to Seq - 12)	* - 1	On, Travel count	All combinations	Travel_cnt_N
	* - 2			
	* - 3			
	* - 4			
	* - 5			
	* - 6			
	* - 7			
	* - 8			
Stop (ends at Seq - 12)	* - 9	On, Travel count	All combinations	Travel_cnt_N

Sweep Sequence by Stepper Motor

Seq_cnt	Sweep_num	Azimuth stepper motor		Elevation stepper motor		Step_mode	Travel_cnt
0	0	On	Left	Off	-	0x01	_0
1			Right		-	0x03	
2	1	Off	-	On	Up	0x30	_1
3	2	On	Left	Off	-	0x01	_2
4			Right		-	0x03	
5	3	Off	-	On	Up	0x30	_3
6	4	On	Left	Off	-	0x01	_4
7			Right		-	0x03	
8	3	Off	-	On	Down	0x10	_3
9	2	On	Left	Off	-	0x01	_2
10			Right		-	0x03	
11	1	Off	-	On	Down	0x10	_1

Repeat Seq_cnt = 0 to 11.



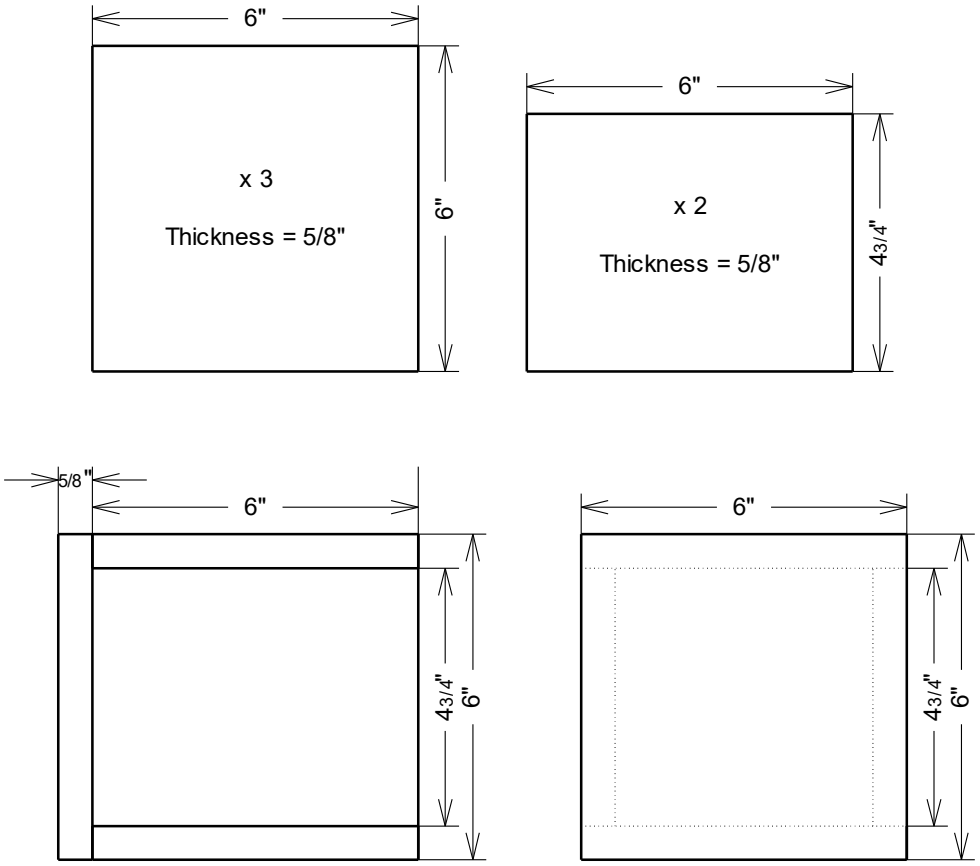
Source of Waveform

RAND[1:0]	SND0 (kHz)	SND1 (kHz)
00	33	18
01	26	22
10	22	26
11	18	33

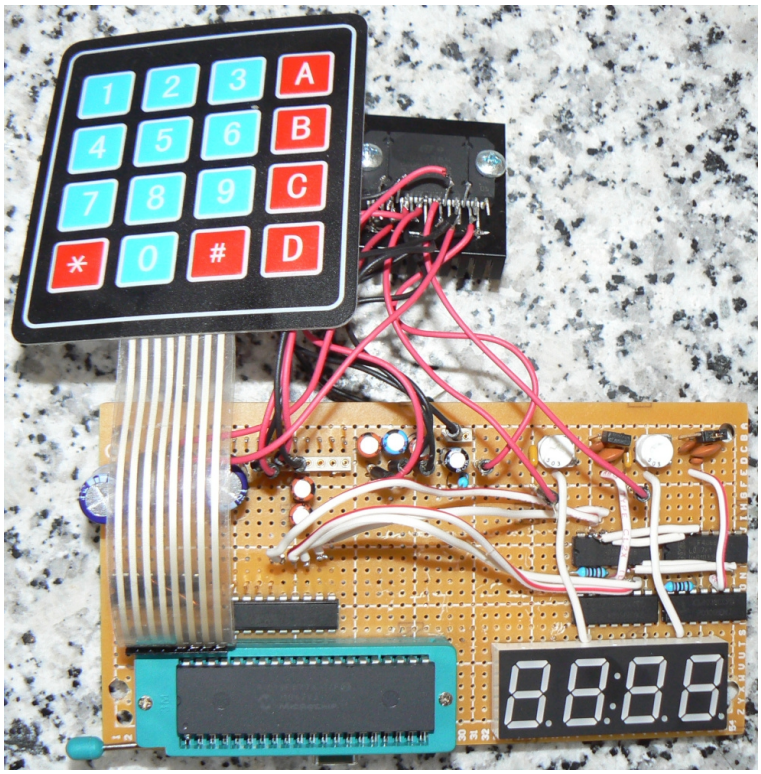
Oscillation Constants

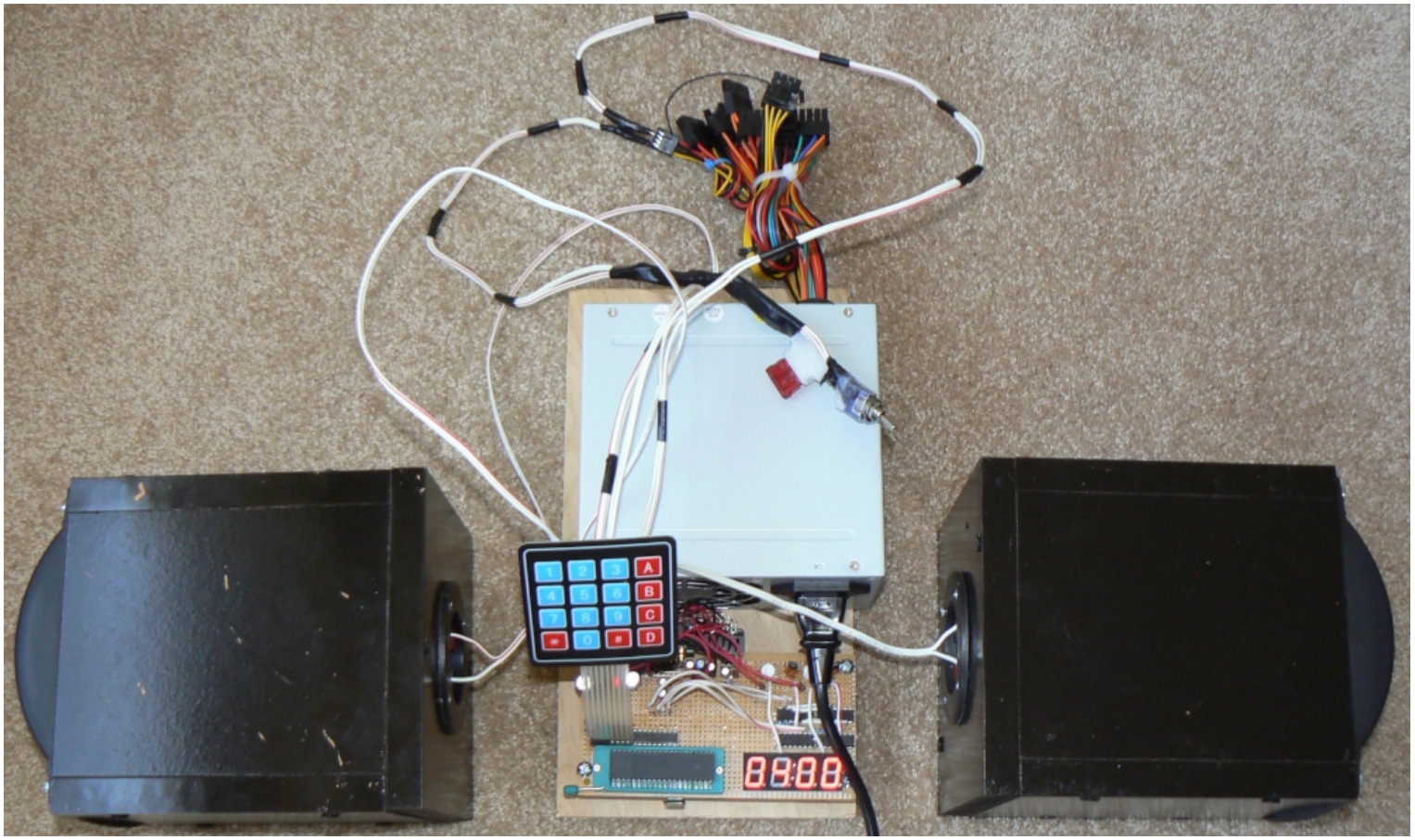
Frequency (kHz)	33	26	22	18
Capacitor (nF)	1.78			
Resistor (R1; kΩ)	5.6			
Resistor (R2; kΩ)	5.6	6.8	7.5	8.2

Speaker Box

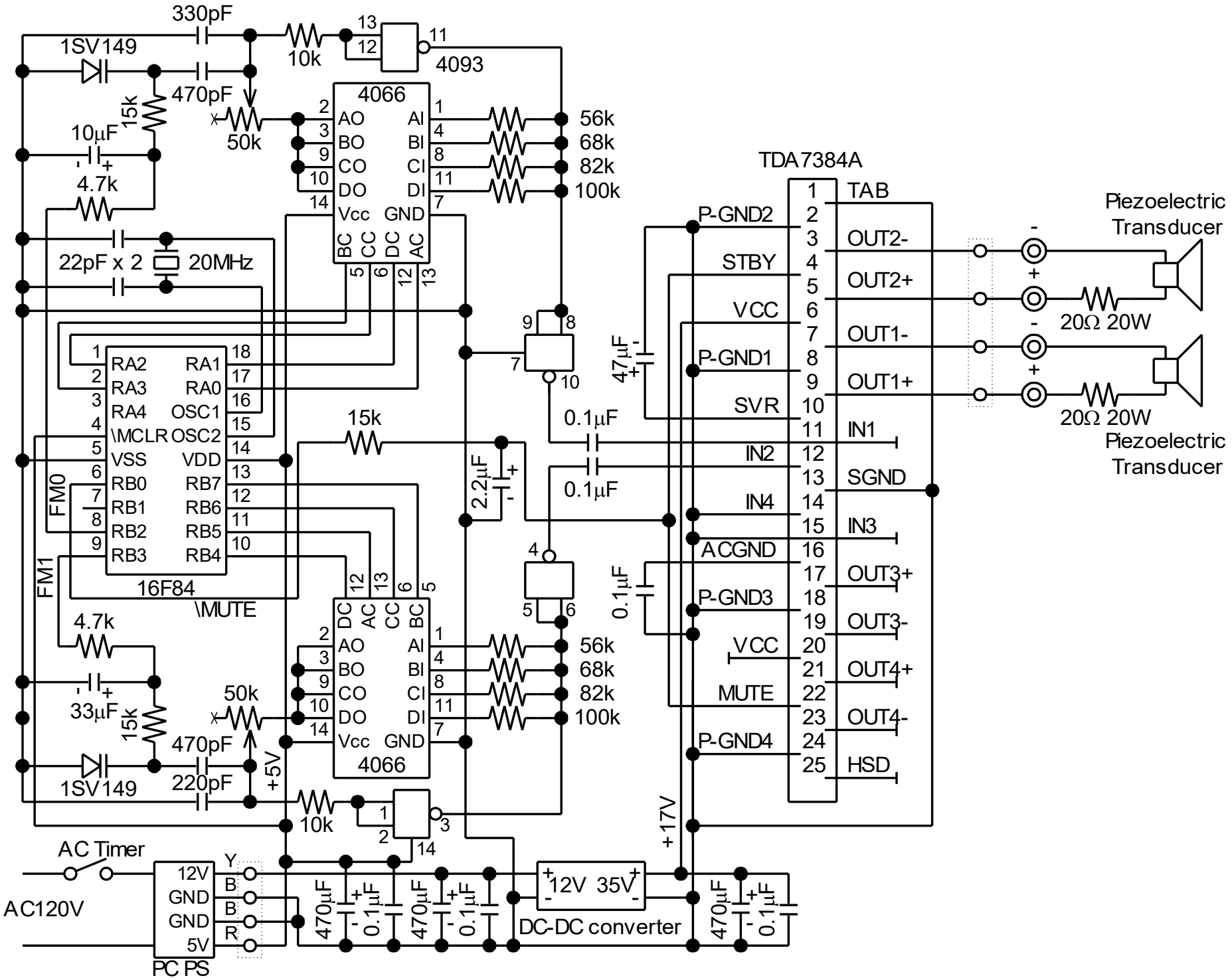


		(mSec)	/ 3.2	Counter_max	
		2400	750	2ee	9
		2250	703	2bf	2
		2100	656	28a	13
		1950	609	261	5
		1800	563	232	3
		1650	516	204	12
		1500	469	1d5	6
		1350	422	1a6	0
1	3.2 * (256 / 1) = 819.2 (mSec)	1200	375	177	7
2	3.2 * (256 / 2) = 409.6	1050	328	148	14
3	3.2 * (256 / 3) = 273.0	900	281	119	8
4	3.2 * (256 / 4) = 204.8	750	234	ea	10
5	3.2 * (256 / 5) = 163.8	600	188	ba	1
6	3.2 * (256 / 6) = 136.5	450	141	8d	4
7	3.2 * (256 / 7) = 117.0	300	94	5e	11
8	3.2 * (256 / 8) = 102.4	150	47	2f	15


1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D



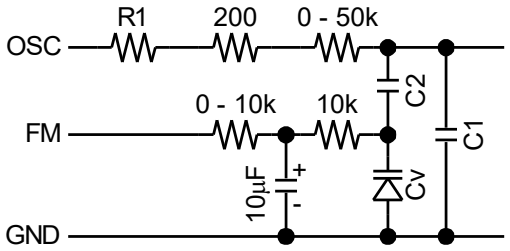
Rev.3 Design


330pF
1SV149
10k
4093
470pF
15k
50k
4066
56k
68k
82k
100k
10µF
4.7k
22pF x 2
20MHz
TDA7384A
TAB
P-GND2
OUT2-
STBY
OUT2+
VCC
OUT1-
P-GND1
OUT1+
SVR
IN1
IN2
SGND
IN4
IN3
ACGND
OUT3+
P-GND3
OUT3-
VCC
OUT4+
MUTE
OUT4-
P-GND4
HSD
Piezoelectric Transducer
20Ω 20W
Piezoelectric Transducer
20Ω 20W
47µF
0.1µF
0.1µF
15k
2.2µF
FM0
FM1
RA2
RA3
RA4
\MCLR
VSS
RB0
RB1
RB2
RB3
RA1
RA0
OSC1
OSC2
VDD
RB7
RB6
RB5
RB4
16F84
\MUTE
4.7k
33µF
50k
15k
470pF
220pF
1SV149
+5V
10k
AC Timer
AC120V
12V
GND
GND
5V
Y
B
B
R
PC PS
470µF
0.1µF
470µF
0.1µF
12V 35V
DC-DC converter
+17V
470µF
0.1µF

Frequency Modulation Calculation

C1 (nF)	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
C2 (nF)	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.47
C3 (nF)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cv (nF)	0.40	0.10	0.40	0.10	0.40	0.10	0.40	0.10
Ctotal (nF)	0.69	0.55	0.69	0.55	0.69	0.55	0.69	0.55
C%		24.19		24.19		24.19		24.19
R1 (ohm)	100,000	100,000	82,000	82,000	68,000	68,000	56,000	56,000
Rx_min (ohm)	200	200	200	200	200	200	200	200
f_max (Hz)	12121.84	15054.05	14776.26	18350.55	17809.51	22117.53	21612.26	26840.13
T_max (uSec)	82.50	66.43	67.68	54.49	56.15	45.21	46.27	37.26
Rx_max (ohm)	50200	50200	50200	50200	50200	50200	50200	50200
f_min (Hz)	8086.61	10042.71	9187.66	11410.10	10275.88	12761.55	11436.99	14203.53
T_min (uSec)	123.66	99.57	108.84	87.64	97.32	78.36	87.44	70.41

R (ohm)	1000	5000	10000
C (uF)	10	10	10
5T (mSec)	50	250	500



Pin Function of 16F84A

Port name	Pin name	Pin functions	I/O functions	Initial value
RA[4]		Not in use	O	0
RA[3:0]	FSEL_0[3:0]	Frequency select	O	0001
RB[7:4]	FSEL_1[3:0]	Frequency select	O	1000
RB[3]	FM_1	Frequency modulation	O	0
RB[2]	FM_0	Frequency modulation	O	0
RB[1]		Not in use	O	0
RB[0]	\MUTE	Mute	O	0

Interrupts (Global interrupt enable = INTCON[7], Peripheral interrupt enable = INTCON[6])

Timer 0 overflow	
Source	Fosc/4 ((20 / 4) MHz)
Prescaler	1/128
Initial value	TMR0[7:0] = 0
Period	(20,000,000 / 4) / 128 / 256 = 152.6 Hz (6.55 mSec)
Purpose	Basic timing for randomization
Interrupt enable	INTCON[5]; TMR0IE
Interrupt flag	INTCON[2]; TMR0IF

16F877A Source Code

```
;*********************************************************************
;Ultrasonic Bird Repeller
; Program name :  ubr
; Module name  :  ubr.asm
; Description  :
; Date         :  July 11, 2012
; Programmer   :  Tetsuji Oguchi
;(C) Oguchi R&D 2012
;*********************************************************************

      LIST  P = 16F877A
#include    "P16F877A.INC"
 __config   _HS_OSC & _WDT_OFF & _LVP_OFF
                        ; OSC = HS (20MHz), WDT off, LVP off

;*************************************
; Equates for general purpose registers *
;*************************************
Hour_0            equ   0x20  ; Hour[0] of clock (BCD)
Hour_1            equ   0x21  ; Hour[1] of clock (BCD)
Minute_0    equ   0x22  ; Minute[0] of clock (BCD)
Minute_1    equ   0x23  ; Minute[1] of clock (BCD)
Second            equ   0x24  ; Second for real time clock (00d --> 29d)
Hour_0_st   equ   0x25  ; Hour[0] for start timer (BCD)
Hour_1_st   equ   0x26  ; Hour[1] for start timer (BCD)
Minute_0_st equ   0x27  ; Minute[0] for start timer (BCD)
Minute_1_st equ   0x28  ; Minute[1] for start timer (BCD)
Hour_0_et   equ   0x29  ; Hour[0] for end timer (BCD)
Hour_1_et   equ   0x2a  ; Hour[1] for end timer (BCD)
Minute_0_et equ   0x2b  ; Minute[0] for end timer (BCD)
Minute_1_et equ   0x2c  ; Minute[1] for end timer (BCD)
            ;
Digit_tim   equ   0x2d  ; image of next {~DT[3:0], ~DT[3:0]}
Seg         equ   0x2e  ; image of next ~SEG[7:0]
Dt_val            equ   0x2f  ; sequencial counter of digit timing (0 to 0xff)
            ;
Key_code    equ   0x30  ; key code (0 to 0xf)
Key_image   equ   0x31  ; image of PORTB[7:4]
Dt_image    equ   0x32  ; image of digit timing when key depressed
Key_prefix  equ   0x33  ; key prefix (A:0, B:1, C:2, D:3, *:4, #:5)
Keyin_mode  equ   0x34  ; Mode of key input
                        ; 7:NA, 6:NA, 5:count_1, 4:count_0
                        ; 3:travel_cnt, 2:end timer, 1:start timer, 0:clock
            ;
Step_cnt    equ   0x35  ; step counter for stepper motor (0 --> 7)
Travel_cnt  equ   0x36  ; travel counter for stepper motor
Travel_max  equ   0x37  ; maximum travel counter for stepper motor
Travel_cnt_0      equ   0x38  ; travel counter for swing sequence 1
Travel_cnt_1      equ   0x39  ; travel counter for swing sequence 2
Travel_cnt_2      equ   0x3a  ; travel counter for swing sequence 3
Travel_cnt_3      equ   0x3b  ; travel counter for swing sequence 4
Travel_cnt_4      equ   0x3c  ; travel counter for swing sequence 5
Swing_num   equ   0x3d  ; swing sequence number
Step_mode   equ   0x3e  ; Mode of stepper motor
                        ; 7:NA, 6:NA, 5:elevation up/down, 4:elevation on/off
                        ; 3:NA, 2:NA, 1:azimuth right/left, 0:azimuth on/off
            ;
Disp_0            equ   0x3f  ; display left most digit
Disp_1            equ   0x40  ; display
Disp_2            equ   0x41  ; display
Disp_3            equ   0x42  ; display right most digit
Disp_mode   equ   0x43  ; Mode of display
                        ; 7:on/off, 6:NA, 5:NA, 4:travel counter
                        ; 3:key input, 2:end timer, 1:start timer, 0:clock
            ;
```

```
Numeric_0   equ   0x44  ; numerics keyin left most digit
Numeric_1   equ   0x45  ; numerics keyin
Numeric_2   equ   0x46  ; numerics keyin
Numeric_3   equ   0x47  ; numerics keyin right most digit
            ;
Misc_flag   equ   0x4f  ; Miscellaneous flags
                        ; {0000000, randomize change}
            ;
Rand_cnt    equ   0x50  ; randomize counter
Rand_var_h  equ   0x51  ; randomize variable high byte
Rand_var_l  equ   0x52  ; randomize variable low  byte
            ;
Temp_tmr0   equ   0x7b  ; temporary register
Temp_travel equ   0x7c  ; temporary register
Temp_key    equ   0x7d  ; temporary register
Temp_wait   equ   0x7e  ; temporary register
            ;
;************************
; Equates for constants *
;************************
second_max  equ   0d30  ; maximum value of Second
            ;

;***************
; Main program *
;***************
 org  0x0000
      ;
      nop               ; reserved for ICD (In-Circuit Debug)
      goto  main        ; start processing
      ;
;************************
; Interrupt processing *
;************************
 org  0x0004
      ;
      btfsc INTCON, TMR0IF
      goto  tmr0_int    ; timer 0 overflow interrupt
      btfsc PIR1, TMR1IF
      goto  tmr1_int    ; timer 1 overflow interrupt
      retfie
      ;
;----------------------
; I/O port definition
;----------------------
 org  0x0100
      ;
main
      bcf   STATUS, RP1 ;
      bcf   STATUS, RP0 ; <page = 0>
      ;
      movlw 0xee
      movwf Digit_tim   ; Digit_tim = 1110b
      movwf PORTA       ; PORTA[3:0] = ~DT[3:0] = 1110b
      ;
      clrf  PORTB       ; PORTB[3:0] = V_DRV[3:0] = 0000b
      ;
      clrf  PORTC       ; PORTC[7:2] = {H_DRV[3:0], !STBY, !MUTE} = 000000b
      ;
      movlw 0x03
      movwf Seg         ; Seg[7:0] = 00000011b
      movwf PORTD       ; PORTD[7:0] = ~SEG[7:0] = 00000011b
      ;
      clrf  PORTE       ; PORTE[2:0] = {FM, RAND[1:0]} = 000b
      ;
      bsf   STATUS, RP0 ; <page = 1>
      ;
```

```
      movlw 0x06
      movwf ADCON1            ; all digital I/Os, not analog
      movlw 0x30
      movwf TRISA       ; PORTA[5:4] = input, PORTA[3:0] = ~DT[3;0] = output
      ;
      movlw 0xf0
      movwf TRISB       ; PORTB[7:4] = ~KEY[3:0] = input
                        ; PORTB[3:0] = V_DRV[3:0] = output
      bcf   OPTION_REG, NOT_RBPU    ; enable pullups for PORTB[7:4] = ~KEY[3:0]
      ;
      movlw 0x03
      movwf TRISC       ; PORTC[7:2] = {H_DRV[3:0], !STBY, !MUTE} = output
                        ; PORTC[1:0] = {T1OSI, T1OSO} = input
      ;
      clrf  TRISD       ; PORTD[7:0] = ~SEG[7:0] = output
      ;
      clrf  TRISE       ; PORTE[2:0] = {FM, RAND[1:0]} = output
      ;
      bcf   STATUS, RP0 ; <page = 0>
      ;
      movlw 0x07        ; <special treatment for _TB>
      movwf PCLATH            ; PCLATH = 0x07 <special treatment for _TB>
      ;
;--------------------------------------
; Initialize general purpose registers
;--------------------------------------
      clrf  Hour_1            ; Hour_1 = 0
      clrf  Hour_0            ; Hour_0 = 0
      clrf  Minute_1    ; Minute_1 = 0
      clrf  Minute_0    ; Minute_0 = 0
      clrf  Disp_0            ; Disp_0 = 0
      clrf  Disp_1            ; Disp_1 = 0
      clrf  Disp_2            ; Disp_2 = 0
      clrf  Disp_3            ; Disp_3 = 0
      clrf  Second            ; Second = 0
      ;
; Clock = 03:59:00
      movlw 0x00
      movwf Hour_1            ; Hour_1 = 0
      movwf Disp_0            ; Disp_0 = 0
      movlw 0x03
      movwf Hour_0            ; Hour_0 = 3
      movwf Disp_1            ; Disp_1 = 3
      movlw 0x05
      movwf Minute_1    ; Minute_1 = 5
      movwf Disp_2            ; Disp_2 = 5
      movlw 0x09
      movwf Minute_0    ; Minute_0 = 9
      movwf Disp_3            ; Disp_3 = 9
      clrf  Second            ; Second = 0
      ;
; Timer start time = 04:00
      movlw 0x00
      movwf Hour_1_st   ; Hour_1_st = 0
      movlw 0x04
      movwf Hour_0_st   ; Hour_0_st = 4
      movlw 0x00
      movwf Minute_1_st ; Minute_1_st = 0
      movlw 0x00
      movwf Minute_0_st ; Minute_0_st = 0
      ;
; Timer end time = 22:00
      movlw 0x02
      movwf Hour_1_et   ; Hour_1_et = 2
      movlw 0x02
      movwf Hour_0_et   ; Hour_0_et = 2
      movlw 0x00
```

```
      movwf Minute_1_et ; Minute_1_et = 0
      movlw 0x00
      movwf Minute_0_et ; Minute_0_et = 0
      ;
; !STBY = 1, !MUTE = 1
      movlw 0x0c
      movwf PORTC       ; PORTC[7:2] = {H_DRV[3:0], !STBY, !MUTE} = 000011b
      ;
      clrf  Dt_val            ; Dt_val = 0
      clrf  Step_cnt    ; Step_cnt = 0
      clrf  Travel_cnt  ; Travel_cnt = 0
      ;
      movlw 0xf0
      movwf Travel_cnt_0      ; Travel_cnt_0 = 0xf0 (240d)
      movlw 0x20
      movwf Travel_cnt_1      ; Travel_cnt_1 = 0x20 (32d)
      movlw 0xe0
      movwf Travel_cnt_2      ; Travel_cnt_2 = 0xe0 (224d)
      movlw 0x10
      movwf Travel_cnt_3      ; Travel_cnt_3 = 0x10 (16d)
      movlw 0xa0
      movwf Travel_cnt_4      ; Travel_cnt_4 = 0xa0 (160d)
      ;
      clrf  Rand_cnt    ; Rand_cnt = 0
      clrf  Rand_var_h  ; Rand_var_h = 0
      movlw 0x2f
      movwf Rand_var_l  ; Rand_var_l = 0x2f
      ;
      clrf  Key_prefix  ; Key_prefix = 0 (no key prefix)
      clrf  Keyin_mode  ; Keyin_mode = 0
      movlw 0x81
      movwf Disp_mode   ; Disp_mode = 0x81 (On, clock)
      clrf  Step_mode   ; Step_mode = 0
      call  get_next_dt
      call  get_next_seg_TB
      ;
;---------------------------------
; Interrupt related initialization
;---------------------------------
      bsf   STATUS, RP0 ; <page = 1>
; Timer 0 overflow interrupt
      bcf   OPTION_REG, PSA   ; prescaler assigned to TMR0
      bcf   OPTION_REG,T0CS   ; TMR0 clock = CLKO
      bsf   OPTION_REG, PS2
      bcf   OPTION_REG, PS1
      bsf   OPTION_REG, PS0   ; prescaler rate = 1:256
      bcf   STATUS, RP0 ; <page = 0>
      clrf  TMR0        ; TMR0[7:0] = 0
; Timer 1 overflow interrupt
      movlw 0x0b
      movwf T1CON       ; enable {oscillation, sync, external clock, timer 1 on}
      clrf  TMR1H       ; TMR1H[7:0] = 0
      clrf  TMR1L       ; TMR1L[7:0] = 0
; clear interrupt flags
      bcf   INTCON, TMR0IF    ; clear TMR0 overflow interrupt flag
      bcf   PIR1, TMR1IF      ; clear TMR1 overflow interrupt flag
; set interrupt enables
      bsf   INTCON, TMR0IE    ; set timer 0 overflow interrupt enable
      bsf   STATUS, RP0 ; <page = 1>
      bsf   PIE1, TMR1IE      ; set timer 1 overflow interrupt enable
      bcf   STATUS, RP0 ; <page = 0>
      bsf   INTCON, PEIE      ; set peripheral interrupt enable
      bsf   INTCON, GIE ; set global interrupt enable
      ;
;-------------
; Key polling
;-------------
```

```
key_polling
      call  check_key
      btfsc STATUS, Z
      goto  key_polling ; no key depressed
      call  wait_32msec ; eliminate key chattering
      ;
key_polling_1
      btfsc PORTA, 0
      goto  key_polling_1     ; wait until PORTA[0] = ~DT[0] = 0
      call  check_key
      btfsc STATUS, Z
      goto  key_polling_2     ; no key when ~DT[0] = 0
      ;
key_polling_3
      decf  Dt_val, W   ; W = Dt_val -1
      andlw 0x3
      movwf Dt_image    ; Dt_image = Dt_val[1:0] for release check
      call  key_proc    ; start key code processing
      goto  key_rel_chk ; start key releasing check
      ;
key_polling_2
      btfsc PORTA, 1
      goto  key_polling_2     ; wait until PORTA[1] = ~DT[1] = 0
      call  check_key
      btfss STATUS, Z
      goto  key_polling_3     ; detected key when ~DT[1] = 0
      ;
key_polling_4
      btfsc PORTA, 2
      goto  key_polling_4     ; wait until PORTA[2] = ~DT[2] = 0
      call  check_key
      btfss STATUS, Z
      goto  key_polling_3     ; detected key when ~DT[2] = 0
      ;
key_polling_5
      btfsc PORTA, 3
      goto  key_polling_5     ; wait until PORTA[3] = ~DT[3] = 0
      call  check_key
      btfss STATUS, Z
      goto  key_polling_3     ; detected key when ~DT[3] = 0
      goto  key_polling ; key scan from beginning
      ;
key_rel_chk
      call  wait_32msec
key_rel_chk_1
      decf  Dt_val, W   ; W = Dt_val - 1
      andlw 0x03
      subwf Dt_image, W
      btfss STATUS, Z
      goto  key_rel_chk_1     ; wait till digit timing matches
      call  check_key
      btfss STATUS, Z
      goto  key_rel_chk ; previous key depressed still
      call  wait_32msec ; previous key released
      goto  key_polling ; previous key not depressed this time
      ;
;----------------
; Key processing
;----------------
key_proc
      call  key_enc_TB
      movwf Key_code    ; Key_code = 0 --> 0xf
      addlw 0xf6
      btfss STATUS, C
      goto  key_proc_2  ; W = 0xf6 --> 0xff
      call  key_prefix_enc_TB
      movwf Key_prefix  ; set Key_prefix[7:0]
```

```
      ;
disp_mode_chk
      btfsc Disp_mode, 0
      goto  disp_clock  ; Clock
      btfsc Disp_mode, 1
      goto  disp_stimer ; Start timer
      btfsc Disp_mode, 2
      goto  disp_etimer ; End timer
      btfsc Disp_mode, 3
      goto  disp_numeric      ; Numerics (numerical key input)
      btfsc Disp_mode, 4
      goto  disp_travel_c     ; Travel counter (binary to BCD)
      ;
disp_clock
      movf  Hour_1, W   ; Clock display mode
      movwf Disp_0            ; Disp_0 = Hour_1
      movf  Hour_0, W
      movwf Disp_1            ; Disp_1 = Hour_0
      movf  Minute_1, W
      movwf Disp_2            ; Disp_2 = Minute_1
      movf  Minute_0, W
      movwf Disp_3            ; Disp_3 = Minute_0
      return
      ;
disp_stimer
      movf  Hour_1_st, W      ; Clock display mode
      movwf Disp_0            ; Disp_0 = Hour_1_st
      movf  Hour_0_st, W
      movwf Disp_1            ; Disp_1 = Hour_0_st
      movf  Minute_1_st, W
      movwf Disp_2            ; Disp_2 = Minute_1_st
      movf  Minute_0_st, W
      movwf Disp_3            ; Disp_3 = Minute_0_st
      return
      ;
disp_etimer
      movf  Hour_1_et, W      ; Clock display mode
      movwf Disp_0            ; Disp_0 = Hour_1_et
      movf  Hour_0_et, W
      movwf Disp_1            ; Disp_1 = Hour_0_et
      movf  Minute_1_et, W
      movwf Disp_2            ; Disp_2 = Minute_1_et
      movf  Minute_0_et, W
      movwf Disp_3            ; Disp_3 = Minute_0_et
      return
      ;
disp_numeric
      movf  Numeric_0, W      ; Numerics display mode
      movwf Disp_0            ; Disp_0 = Numeric_0
      movf  Numeric_1, W      ; Numerics display mode
      movwf Disp_1            ; Disp_1 = Numeric_1
      movf  Numeric_2, W      ; Numerics display mode
      movwf Disp_2            ; Disp_2 = Numeric_2
      movf  Numeric_3, W      ; Numerics display mode
      movwf Disp_3            ; Disp_3 = Numeric_3
      return
      ;
disp_travel_c
      movf  Travel_cnt, W     ; W = Travel_cnt
      movwf Temp_travel ; Temp_travel = Travel_cnt
      clrf  Disp_0            ; Disp_0 = 0
      clrf  Disp_1            ; Disp_1 = 0
      clrf  Disp_2            ; Disp_2 = 0
      movlw 0x64        ; W = 100
disp_travel_2
      subwf Temp_travel, F    ; Temp_travel = Temp_travel - 100
      btfss STATUS, C
```

```
      goto  disp_travel_1
      incf  Disp_1, F   ; Disp_1 ++
      goto  disp_travel_2
      ;
disp_travel_1
      addwf Temp_travel, F
      movlw 0x0a        ; W = 10
disp_travel_4
      subwf Temp_travel, F    ; Temp_travel = Temp_travel - 10
      btfss STATUS, C
      goto  disp_travel_3
      incf  Disp_2, F   ; Disp_2 ++
      goto  disp_travel_4
      ;
disp_travel_3
      addwf Temp_travel, F
      movf  Temp_travel, W    ; W = Temp_travel
      movwf Disp_3
      return
      ;
key_proc_2
      btfsc Disp_mode, 3
      goto  nkey_in1_TB ; key input = 1
      addlw 0x0a        ; W = 0 to 9 ("0" to "9")
      btfsc Key_prefix, 0
      goto  key_pre_a_TB      ; key prefix was "A"
      btfsc Key_prefix, 1
      goto  key_pre_b_TB      ; key prefix was "B"
      btfsc Key_prefix, 2
      goto  key_pre_c_TB      ; key prefix was "C"
      btfsc Key_prefix, 3
      goto  key_pre_d   ; key prefix was "D"
      btfsc Key_prefix, 4
      goto  key_pre_s   ; key prefix was "*"
      goto  key_pre_p   ; key prefix was "#"
      ;
;-----------------------------
; "A-0", "A-1",,,, "A-9"
;  (return from key_pre_a_TB)
;-----------------------------
key_pre_a0
      goto  main        ; restart program
      ;
key_pre_a1
      movlw 0x81        ; display on, clock
      goto  key_pre_a9_1
      ;
key_pre_a2
      movlw 0x82        ; display on, start timer
      goto  key_pre_a9_1
      ;
key_pre_a3
      movlw 0x84        ; display on, end timer
      goto  key_pre_a9_1
      ;
key_pre_a4
      movlw 0x31        ; keyin_cnt = 3, clock
key_pre_a41
      movwf Keyin_mode
      call  clr_numerics
      movlw 0x88        ; display on, key input
      goto  key_pre_a9_1
      ;
key_pre_a5
      movlw 0x32        ; keyin_cnt = 3, start timer
      goto  key_pre_a41
      ;
```

```
key_pre_a6
      movlw 0x34        ; keyin_cnt = 3, end timer
      goto  key_pre_a41
      ;
key_pre_a7
      bcf   PORTC, 3    ; !STBY = 0
      bcf   PORTC, 2    ; !MUTE = 0
      goto  key_pre_a1
      ;
key_pre_a8
      bsf   PORTC, 3    ; !STBY = 1
      bsf   PORTC, 2    ; !MUTE = 1
      goto  key_pre_a1
      ;
key_pre_a9
      movlw 0x01        ; "A-9" Dispay off, clock
key_pre_a9_1
      movwf Disp_mode
      clrf  Key_prefix  ; Key_prefix = 0
      goto  disp_mode_chk
      ;
;-----------------------------
; "B-0", "B-1",,,, "B-9"
;  (return from key_pre_b_TB)
;-----------------------------
key_pre_b0
      clrf  Step_mode   ; Step_mode = 0
set_disp_tc
      movlw 0x90
      movwf Disp_mode   ; display on, travel counter
      goto  disp_travel_c     ;
      ;
key_pre_b1
      movlw 0x01
      movwf Step_mode   ; Step_mode = 0x01, azimuth left, on
clr_step_cnt
      movlw 0xff
      movwf Travel_cnt  ; Travel_cnt = 0xff
      clrf  Step_cnt    ; Step_cnt = 0
      goto  set_disp_tc
      ;
key_pre_b2
      movlw 0x03
      movwf Step_mode   ; Step_mode = 0x03, azimuth right, on
      goto  clr_step_cnt
      ;
key_pre_b3
      movlw 0x10
      movwf Step_mode   ; Step_mode = 0x10, elevation left, on
      goto  clr_step_cnt
      ;
key_pre_b4
      movlw 0x30
      movwf Step_mode   ; Step_mode = 0x30, elevation right, on
      goto  clr_step_cnt
      ;
key_pre_b9              ; "B-5, B-6, B-7, B-8, B-9"
      movwf Swing_num   ; Swing_num = W (5, 6, 7, 8, 9)
      movlw 0x05
      subwf Swing_num, F      ; Swing_num -= 0x05
      movlw 0x38        ; keyin_cnt = 3, travel count
      goto  key_pre_a41
      ;
;-----------------------------
; "C-0", "C-1",,,, "C-9"
;  (return from key_pre_c_TB)
;-----------------------------
```

```
key_pre_c0
      movlw 0x01
      movwf Step_mode   ; Step_mode = 0x01, azimuth left, on
key_pre_c11
      movf  Travel_cnt_0, W   ; W = Travel_cnt_0
set_tcsc
      movwf Travel_cnt  ; Travel_cnt = Travel_cnt_0
      clrf  Step_cnt    ; Step_cnt = 0
      goto  set_disp_tc
      ;
key_pre_c1
      movlw 0x03
      movwf Step_mode   ; Step_mode = 0x03, azimuth right, on
      goto  key_pre_c11
      ;
key_pre_c2
      movlw 0x30
      movwf Step_mode   ; Step_mode = 0x30, elevation up, on
key_pre_c31
      movf  Travel_cnt_1, W   ; W = Travel_cnt_1
      goto  set_tcsc
      ;
key_pre_c3
      movlw 0x10
      movwf Step_mode   ; Step_mode = 0x10, elevation down, on
      goto  key_pre_c31
      ;
key_pre_c4
      movlw 0x01
      movwf Step_mode   ; Step_mode = 0x01, azimuth left, on
key_pre_c51
      movf  Travel_cnt_2, W   ; W = Travel_cnt_2
      goto  set_tcsc
      ;
key_pre_c5
      movlw 0x03
      movwf Step_mode   ; Step_mode = 0x03, azimuth right, on
      goto  key_pre_c51
      ;
key_pre_c6
      movlw 0x30
      movwf Step_mode   ; Step_mode = 0x30, elevation up, on
key_pre_c71
      movf  Travel_cnt_3, W   ; W = Travel_cnt_3
      goto  set_tcsc
      ;
key_pre_c7
      movlw 0x10
      movwf Step_mode   ; Step_mode = 0x10, elevation down, on
      goto  key_pre_c71
      ;
key_pre_c8
      movlw 0x01
      movwf Step_mode   ; Step_mode = 0x01, azimuth left, on
key_pre_c91
      movf  Travel_cnt_4, W   ; W = Travel_cnt_4
      goto  set_tcsc
      ;
key_pre_c9
      movlw 0x03
      movwf Step_mode   ; Step_mode = 0x03, azimuth right, on
      goto  key_pre_c91
      ;
key_pre_d
      goto  key_pre_a_TB      ; treated as "A" prefix
      ;
key_pre_s
```

```
      goto  key_pre_a_TB      ; treated as "A" prefix
      ;
key_pre_p
      goto  key_pre_a_TB      ; treated as "A" prefix
      ;
;-------------------------------------------
; Input 4 digits numerics for travel counter
;  (return from nkey_in1_TB)
;-------------------------------------------
nkey_tc_3               ; keyin counter = 3, travel counter
      movf  Key_code, W ; W = Key_code
      btfsc STATUS, Z
      goto  set_numeric_0     ; Key_code = 0
      goto  recall_tc_TB      ; Key_code ~= 0
;-----------------------------
; Recall travel count
;  (return from recall_tc_TB)
;-----------------------------
recall_tc_00
      movf  Travel_cnt_0, W   ; W = Travel_cnt_0
      movwf Travel_cnt  ; Travel_cnt = Travel_cnt_0
      goto  set_disp_tc
      ;
recall_tc_01
      movf  Travel_cnt_1, W   ; W = Travel_cnt_1
      movwf Travel_cnt  ; Travel_cnt = Travel_cnt_1
      goto  set_disp_tc
      ;
recall_tc_02
      movf  Travel_cnt_2, W   ; W = Travel_cnt_2
      movwf Travel_cnt  ; Travel_cnt = Travel_cnt_2
      goto  set_disp_tc
      ;
recall_tc_03
      movf  Travel_cnt_3, W   ; W = Travel_cnt_3
      movwf Travel_cnt  ; Travel_cnt = Travel_cnt_3
      goto  set_disp_tc
      ;
recall_tc_04
      movf  Travel_cnt_4, W   ; W = Travel_cnt_4
      movwf Travel_cnt  ; Travel_cnt = Travel_cnt_4
      goto  set_disp_tc
      ;
nkey_tc_2               ; keyin counter = 2, travel counter
      movf  Key_code, W ; W = Key_code
      goto  set_numeric_1
      ;
nkey_tc_1               ; keyin counter = 1, travel counter
      movf  Key_code, W ; W = Key_code
      goto  set_numeric_2
      ;
nkey_tc_0               ; keyin counter = 0, travel counter
      movf  Key_code, W ; W = Key_code
      movwf Numeric_3   ; Numeric_3 = Key_code
      clrf  Keyin_mode  ; Keyin_mode = 0
      movf  Numeric_3, W      ; W = Numeric_3
      movwf Travel_cnt  ; Travel_cnt = Numeric_3
      movf  Numeric_1, W      ; W = Numeric_1
      movwf Temp_key    ; Temp_key = Numeric_1
      movlw 0x64        ; W = 0x64 (100d)
      call  bcd_to_bin
      movf  Numeric_2, W      ; W = Numeric_2
      movwf Temp_key    ; Temp_key = Numeric_2
      movlw 0x0a        ; W = 0x0a (10d)
      call  bcd_to_bin
      ;
      goto  save_tc_TB
```

```
;--------------------------
; Save travel count
;  (return from save_tc_TB)
;--------------------------
nkey_tc_00              ; Swing_num = 0
      movf  Travel_cnt, W     ; W = Travel_cnt
      movwf Travel_cnt_0      ; Travel_cnt_0 = Travel_cnt
      goto  set_disp_tc
      ;
nkey_tc_01              ; Swing_num = 1
      movf  Travel_cnt, W     ; W = Travel_cnt
      movwf Travel_cnt_1      ; Travel_cnt_1 = Travel_cnt
      goto  set_disp_tc
      ;
nkey_tc_02              ; Swing_num = 2
      movf  Travel_cnt, W     ; W = Travel_cnt
      movwf Travel_cnt_2      ; Travel_cnt_2 = Travel_cnt
      goto  set_disp_tc
      ;
nkey_tc_03              ; Swing_num = 3
      movf  Travel_cnt, W     ; W = Travel_cnt
      movwf Travel_cnt_3      ; Travel_cnt_3 = Travel_cnt
      goto  set_disp_tc
      ;
nkey_tc_04              ; Swing_num = 4
      movf  Travel_cnt, W     ; W = Travel_cnt
      movwf Travel_cnt_4      ; Travel_cnt_4 = Travel_cnt
      goto  set_disp_tc
      ;
;-----------------------------------
; Input 4 digits numerics for clock
;  (return from nkey_in1_TB)
;-----------------------------------
nkey_cl_3               ; keyin counter = 3, clock
      movf  Key_code, W ; W = Key_code
      movwf Hour_1            ; Hour_1 = Key_code
set_numeric_0
      movwf Numeric_0   ; Numeric_0 = Key_code
dec_keyin_cnt
      movlw 0x10
      subwf Keyin_mode, F     ; Keyin_mode -= 0x10
      goto  disp_numeric
      ;
nkey_cl_2               ; keyin counter = 2, clock
      movf  Key_code, W ; W = Key_code
      movwf Hour_0            ; Hour_0 = Key_code
set_numeric_1
      movwf Numeric_1   ; Numeric_1 = Key_code
      goto  dec_keyin_cnt
      ;
nkey_cl_1               ; keyin counter = 1, clock
      movf  Key_code, W ; W = Key_code
      movwf Minute_1    ; Minute_1 = Key_code
set_numeric_2
      movwf Numeric_2   ; Numeric_2 = Key_code
      goto  dec_keyin_cnt
      ;
nkey_cl_0               ; keyin counter = 0, clock
      movf  Key_code, W ; W = Key_code
      movwf Minute_0    ; Minute_0 = Key_code
      movwf Numeric_3   ; Numeric_3 = Key_code
      clrf  Keyin_mode  ; Keyin_mode = 0
      movlw 0x81
      movwf Disp_mode   ; display on, clock
      goto  disp_mode_chk
      ;
;----------------------------------------
```

```
; Input 4 digits numerics for start timer
;  (return from nkey_in1_TB)
;---------------------------------------
nkey_st_0               ; keyin counter = 0, start timer
      movf  Key_code, W ; W = Key_code
      movwf Minute_0_st ; Minute_0_st = Key_code
      movwf Numeric_3   ; Numeric_3 = Key_code
      clrf  Keyin_mode  ; Keyin_mode = 0
      movlw 0x82
      movwf Disp_mode   ; display on, start timer
      goto  disp_mode_chk
      ;
nkey_st_1               ; keyin counter = 1, start timer
      movf  Key_code, W ; W = Key_code
      movwf Minute_1_st ; Minute_1_st = Key_code
      goto  set_numeric_2
      ;
nkey_st_2               ; keyin counter = 2, start timer
      movf  Key_code, W ; W = Key_code
      movwf Hour_0_st   ; Hour_0_st = Key_code
      goto  set_numeric_1
      ;
nkey_st_3               ; keyin counter = 3, start timer
      movf  Key_code, W ; W = Key_code
      movwf Hour_1_st   ; Hour_1_st = Key_code
      goto  set_numeric_0
      ;
;---------------------------------------
; Input 4 digits numerics for end timer
;  (return from nkey_in1_TB)
;---------------------------------------
nkey_et_3               ; keyin counter = 3, end timer
      movf  Key_code, W ; W = Key_code
      movwf Hour_1_et   ; Hour_1_et = Key_code
      goto  set_numeric_0
      ;
nkey_et_2               ; keyin counter = 2, end timer
      movf  Key_code, W ; W = Key_code
      movwf Hour_0_et   ; Hour_0_et = Key_code
      goto  set_numeric_1
      ;
nkey_et_1               ; keyin counter = 1, end timer
      movf  Key_code, W ; W = Key_code
      movwf Minute_1_et ; Minute_1_et = Key_code
      goto  set_numeric_2
      ;
nkey_et_0               ; keyin counter = 0, end timer
      movf  Key_code, W ; W = Key_code
      movwf Minute_0_et ; Minute_0_et = Key_code
      movwf Numeric_3   ; Numeric_3 = Key_code
      clrf  Keyin_mode  ; Keyin_mode = 0
      movlw 0x84
      movwf Disp_mode   ; display on, end timer
      goto  disp_mode_chk
      ;
;***********************************
; Timer 0 overflow interrupt process *
;  - Digit timing                    *
;  - Stepper motor increment         *
;  Timing generation (3.2 mSec)      *
;***********************************
tmr0_int
      bcf   INTCON, TMR0IF    ; clear interrupt flag bit
      ;
;-----------
; Randomize
;-----------
```

```
      btfsc PORTC, 2    ;
      goto  randomize_1 ; if (!MUTE = 1) goto
      btfsc Misc_flag, 0      ;
      goto  randomize_2 ; if (change = 1) goto
; detecting frequency randomization timing during (!MUTE = 0) & (change = 0)
      decf  Rand_var_l, F     ; Rand_var_l --
      btfss STATUS, Z
      goto  tmr0_int_1  ; if (Z = 0) goto
      call  randm_8_TB  ; _TB on page 6
      incf  PCLATH, F   ; PCLATH = 0x07 <special treatment for _TB>
      movwf PORTE       ; PORTE = W
      addwf Rand_cnt, W ; W += Rand_cnt
      movwf Temp_tmr0   ; Temp_tmr0 = W
      call  randm_varh_TB
      movwf Rand_var_h  ; Rand_var_h = W
      movf  Temp_tmr0, W      ; W = Temp_tmr0
      call  randm_varl_TB
      movwf Rand_var_l  ; Rand_var_l = W
      incf  Rand_cnt, F ; Rand_cnt ++
      bsf   Misc_flag, 0      ; change = 1
      goto  tmr0_int_1
      ;
; detecting !MUTE low to high transition during (!MUTE = 0) & (change = 1)
randomize_2
      decf  Rand_var_l, F     ; Rand_var_l --
      btfss STATUS, Z
      goto  tmr0_int_1  ; if (Z = 0) goto
      movf  Rand_cnt, W ; W = Rand_cnt
      call  randm_varh_TB
      movwf Rand_var_h  ; Rand_var_h = W
      movf  Rand_cnt, W ; W = Rand_cnt
      call  randm_varl_TB
      movwf Rand_var_l  ; Rand_var_l = W
      bsf   PORTC, 2    ; !MUTE = 1
      goto  randomize_3
      ;
; detecting !MUTE high to low transition during (!MUTE = 1)
randomize_1
      decf  Rand_var_l, F     ; Rand_var_l --
      btfss STATUS, Z
      goto  tmr0_int_1  ; if (Z = 0) goto
      clrw              ; W = 0
      iorwf Rand_var_h, F     ; Rand_var_h |= W
      btfss STATUS, Z
      goto  randomize_4 ; if (Z = 0) goto
      movlw 0x8d
      movwf Rand_var_l  ; Rand_var_l = 0x8d (fixed value)
      bcf   PORTC, 2    ; !MUTE = 0
randomize_3
      bcf   Misc_flag, 0      ; change = 0
      goto  tmr0_int_1
      ;
randomize_4
      decf  Rand_var_h, F     ; Rand_var_h --
      ;
;---------
; Display
;---------
tmr0_int_1
      movf  Digit_tim, W      ; W = Digit_tim
      movwf PORTA       ; PORTA = new ~DT[3:0]
      movf  Seg, W            ; W = Seg
      movwf PORTD       ; PORTD = new ~SEG[7:0]
      call  get_next_dt
      call  get_next_seg_TB
      ;
; Stepper motor function has been removed
```

```
      retfie
      ;
;---------------
; Stepper motor
;---------------
      btfsc Step_mode, 0
      goto  az_motor_on ; if azimuth motor on
      btfss Step_mode, 4
      retfie
el_motor_on             ; if elevation motor on
      call  motor_drv_el_TB   ; get motor drive data
      movwf PORTB       ; PORTB[3:0] = E_DRV[3:0]
      btfsc Step_mode, 5
      goto  inc_step_cnt      ; if Up/Down = 1
dec_step_cnt                  ; if Up/Down = 0 or Right/Left = 0
      movf  Step_cnt, F
      btfsc STATUS, Z
      goto  dec_step_cnt_1    ; if Step_cnt = 0
      decf  Step_cnt, F ; Step_cnt --
      movf  Step_cnt, F
      btfss STATUS, Z
      goto  tmr0_int_end
      goto  travel_cnt_chk
      ;
dec_step_cnt_1
      movlw 0x07
      movwf Step_cnt    ; Step_cnt = 7
      goto  tmr0_int_end
      ;
az_motor_on
      call  get_stby_mute
      call  motor_drv_az_TB   ; get motor drive data
      iorwf Temp_tmr0, W      ; W = {A_DRV[3:0], !STBY, !MUTE, 00b}
      movwf PORTC       ; PORTC[7:2] = {A_DRV[3:0], !STBY, !MUTE}
      btfss Step_mode, 1
      goto  dec_step_cnt      ; if Right/Left = 0
inc_step_cnt                  ; if Right/Left = 1
      incf  Step_cnt, F ; Step_cnt ++
      btfsc Step_cnt, 3
      goto  inc_step_cnt_1
tmr0_int_end
      call  disp_travel_c     ; display travel counter
      retfie
      ;
inc_step_cnt_1
      clrf  Step_cnt    ; Step_cnt = 0
travel_cnt_chk
      movf  Travel_cnt, F
      btfsc STATUS, Z
      goto  swing_end
      decf  Travel_cnt, F     ; Travel_cnt --
      goto  tmr0_int_end
      ;
swing_end
      clrf  Step_mode   ; Step_mode = 0
      clrf  PORTB       ; PORTB[3:0] = E_DRV[3:0] = 0
      call  get_stby_mute
      clrw              ; W = 0
      iorwf Temp_tmr0, W      ; W = {A_DRV[3:0], !STBY, !MUTE, 00b}
      movwf PORTC       ; PORTC[7:2] = {A_DRV[3:0], !STBY, !MUTE}
      goto  tmr0_int_end
      ;
;*****************************************************
; Timer 1 overflow interrupt (every 2 seconds) process *
;*****************************************************
tmr1_int
      bcf   PIR1, TMR1IF
```

```
      ;
      incf  Second, F   ; Second ++
      movlw 0x1e
      subwf Second, W
      btfss STATUS, Z
      goto  clock_cnt_3
      clrf  Second
      incf  Minute_0, F ; Increment minute
      movlw 0x0a
      subwf Minute_0, W
      btfss STATUS, Z
      goto  clock_cnt_4
      clrf  Minute_0
      incf  Minute_1, F
      movlw 0x06
      subwf Minute_1, W
      btfss STATUS, Z
      goto  clock_cnt_4
      clrf  Minute_1
      movlw 0x02
      subwf Hour_1, W
      btfss STATUS, Z
      goto  clock_cnt_1
      movlw 0x03
      subwf Hour_0, W
      btfss STATUS, Z
      goto  clock_cnt_1
      clrf  Hour_0
      clrf  Hour_1
      goto  clock_cnt_4
      ;
clock_cnt_1
      incf  Hour_0, F
      movlw 0x0a
      subwf Hour_0, W
      btfss STATUS, Z
      goto  clock_cnt_4
      clrf  Hour_0
      incf  Hour_1, F
      ;
clock_cnt_4
      btfss PORTC, 3    ; if (!STBY = 1) skip
      goto  clock_cnt_5 ; if (!STBY = 0) goto
      movf  Minute_0, W ; W = Minute_0
      subwf Minute_0_et, W    ; W -= Minute_0_et
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
      movf  Minute_1, W ; W = Minute_1
      subwf Minute_1_et, W    ; W -= Minute_1_et
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
      movf  Hour_0, W   ; W = Hour_0
      subwf Hour_0_et, W      ; W -= Hour_0_et
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
      movf  Hour_1, W   ; W = Hour_1
      subwf Hour_1_et, W      ; W -= Hour_1_et
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
      bcf   PORTC, 3    ; !STBY = 0
      goto  clock_cnt_3
      ;
clock_cnt_5
      movf  Minute_0, W ; W = Minute_0
      subwf Minute_0_st, W    ; W -= Minute_0_st
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
```

```
      movf  Minute_1, W ; W = Minute_1
      subwf Minute_1_st, W    ; W -= Minute_1_st
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
      movf  Hour_0, W   ; W = Hour_0
      subwf Hour_0_st, W      ; W -= Hour_0_st
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
      movf  Hour_1, W   ; W = Hour_1
      subwf Hour_1_st, W      ; W -= Hour_1_st
      btfss STATUS, Z   ; if (Z = 1) skip
      goto  clock_cnt_3 ; if (Z = 0) goto
      bsf   PORTC, 3    ; !STBY = 0
      ;
clock_cnt_3
      call  disp_mode_chk
      retfie
      ;
;**************
; Subroutines *
;**************
 org  0x0500
      ;
;-----------------------------------------
; Get next Digit_tim = {~DT[3:0], ~DT[3:0]}
;-----------------------------------------
get_next_dt
      incf  Dt_val, F
      rlf   Digit_tim, F      ; W = Digit_tim rotate left
      btfsc STATUS, C   ; if (C = 0) skip
      goto  get_next_dt1      ; C = 0
      bcf   Digit_tim, 0      ; reset Digit_tim[0]
      return
get_next_dt1
      bsf   Digit_tim, 0      ; set Digit_tim[0]
      return
      ;
;-----------------------------------------
; Get next segment data (return from TB_)
;-----------------------------------------
get_next_seg1
      movf  Disp_0, W   ; W = Disp_0
      goto  get_next_seg5
get_next_seg2
      movf  Disp_1, W   ; W = Disp_1
      goto  get_next_seg5
get_next_seg3
      movf  Disp_2, W   ; W = Disp_2
      goto  get_next_seg5
get_next_seg4
      movf  Disp_3, W   ; W = Disp_3
get_next_seg5
      call  seg_dec_TB
      movwf Seg
      btfsc Disp_mode, 7
      return                  ; display on
get_next_seg6
      movlw 0xff        ; display off
      movwf Seg
      return
      ;
;-------------------
; Wait for 32 mSec
;-------------------
wait_32msec
      movf  Dt_val, W   ; W = Dt_val
      addlw 0x0a        ; W += 10
```

```
      movwf Temp_wait   ; Temp_wait = Dt_val + 10
wait_32msec_1
      movf  Temp_wait, W
      subwf Dt_val, W
      btfss STATUS, Z
      goto  wait_32msec_1
      return
      ;
;------------------
; Check key status
;------------------
check_key
      swapf PORTB, W    ; W[7:0] = {PORTB[3:0], PORTB[7:4]}
      andlw 0x0f
      movwf Key_image   ; Key_image = PORTB[7:4]
      sublw 0x0f
      return
      ;
;------------------------
; Clear numerics buffers
;------------------------
clr_numerics
      movlw 0x10
      movwf Numeric_0   ; Numeric_0 = 0x10
      movwf Numeric_1   ; Numeric_1 = 0x10
      movwf Numeric_2   ; Numeric_2 = 0x10
      movwf Numeric_3   ; Numeric_3 = 0x10
      return
      ;
;-------------------------
; BCD to binary conversion
;-------------------------
bcd_to_bin
      movf  Temp_key, F
      btfsc STATUS, Z
      return
      decf  Temp_key, F ; Temp_key --
      addwf Travel_cnt, F     ; Travel_cnt += W
      goto  bcd_to_bin
      ;
;---------------------------------
; Store !STBY & !MUTE to Temp_tmr0
;---------------------------------
get_stby_mute
      movlw 0x0c
      andwf PORTC, W    ; W = {0000b, !STBY, !MUTE, 00b}
      movwf Temp_tmr0   ; Temp_tmr0 = W
      return
      ;
;*****************************
; Table Branches (_TB) page 06 *
;*****************************
 org  0x0600
      ;
;-------------------------
; Randomize {FM, RAND[1:0]}
;-------------------------
randm_8_TB
      decf  PCLATH, F   ; PCLATH = 0x06 <special treatment for _TB>
      movf  Rand_cnt, W ; W = Rand_cnt
      andlw 0x3f
      addwf PCL, F            ; PCL = PCL + W
      retlw 0           ; W = 0
      retlw 7
      retlw 2
      retlw 3
      retlw 6
```

```
      retlw 1
      retlw 4
      retlw 5
      retlw 2           ; W = 8
      retlw 6
      retlw 1
      retlw 5
      retlw 0
      retlw 4
      retlw 3
      retlw 7
      retlw 3           ; W = 0x10
      retlw 1
      retlw 6
      retlw 5
      retlw 7
      retlw 0
      retlw 4
      retlw 2
      retlw 4           ; W = 0x18
      retlw 1
      retlw 2
      retlw 0
      retlw 3
      retlw 7
      retlw 6
      retlw 5
      retlw 3           ; W = 0x20
      retlw 6
      retlw 7
      retlw 4
      retlw 0
      retlw 2
      retlw 5
      retlw 1
      retlw 0           ; W = 0x28
      retlw 1
      retlw 3
      retlw 5
      retlw 6
      retlw 7
      retlw 2
      retlw 4
      retlw 5           ; W = 0x30
      retlw 0
      retlw 2
      retlw 6
      retlw 4
      retlw 7
      retlw 3
      retlw 1
      retlw 6           ; W = 0x38
      retlw 1
      retlw 0
      retlw 3
      retlw 2
      retlw 5
      retlw 4
      retlw 7           ; W = 0x3f
      ;
;*******************************
; Table Branches (_TB) page 07 *
;*******************************
 org  0x0700
      ;
;--------------------------------------------------------------------------
; 7 segment decoder (input : W = 0 to 10, output : W = segment drive data)
```

```
;-------------------------------------------------------------------------
seg_dec_TB
      addwf PCL, F            ; PCL = PCL + W
      retlw b'00000011' ; if (W = 0x00) 0
      retlw b'10011111' ; if (W = 0x01) 1
      retlw b'00100101' ; if (W = 0x02) 2
      retlw b'00001101' ; if (W = 0x03) 3
      retlw b'10011001' ; if (W = 0x04) 4
      retlw b'01001001' ; if (W = 0x05) 5
      retlw b'01000001' ; if (W = 0x06) 6
      retlw b'00011011' ; if (W = 0x07) 7
      retlw b'00000001' ; if (W = 0x08) 8
      retlw b'00001001' ; if (W = 0x09) 9
      retlw b'00010001' ; if (W = 0x0a) A
      retlw b'11000001' ; if (W = 0x0b) b
      retlw b'11100101' ; if (W = 0x0c) c
      retlw b'10000101' ; if (W = 0x0d) d
      retlw b'01100001' ; if (W = 0x0e) E
      retlw b'01110001' ; if (W = 0x0f) F
      retlw b'11111111' ; if (W = 0x10) Blank
      ;
;-----------------------------
; Azimuth motor drive decoder
;-----------------------------
motor_drv_az_TB
      movf  Step_cnt, W ; W = Step_cnt
      andlw 0x07
      addwf PCL, F            ; PCL = PCL + W
      retlw b'11000000' ; if (W = 0)
      retlw b'01000000' ; if (W = 1)
      retlw b'01100000' ; if (W = 2)
      retlw b'00100000' ; if (W = 3)
      retlw b'00110000' ; if (W = 4)
      retlw b'00010000' ; if (W = 5)
      retlw b'10010000' ; if (W = 6)
      retlw b'10000000' ; if (W = 7)
      ;
;-------------------------------
; Elevation motor drive decoder
;-------------------------------
motor_drv_el_TB
      movf  Step_cnt, W ; W = Step_cnt
      andlw 0x07
      addwf PCL, F            ; PCL = PCL + W
      retlw b'00001100' ; if (W = 0)
      retlw b'00000100' ; if (W = 1)
      retlw b'00000110' ; if (W = 2)
      retlw b'00000010' ; if (W = 3)
      retlw b'00000011' ; if (W = 4)
      retlw b'00000001' ; if (W = 5)
      retlw b'00001001' ; if (W = 6)
      retlw b'00001000' ; if (W = 7)
      ;
;-------------
; Key encoder
;-------------
key_enc_TB
      movf  Dt_image, W ; W = Dt_image
      btfss Key_image, 0
      goto  key_enc_1_TB      ; Key_image[0] = 0
      btfss Key_image, 1
      goto  key_enc_2_TB      ; Key_image[1] = 0
      btfss Key_image, 2
      goto  key_enc_3_TB      ; Key_image[2] = 0
      addwf PCL, F            ; PCL = PCL + W
      retlw b'00000001' ; if (W = 0)
      retlw b'00000100' ; if (W = 1)
```

```
      retlw b'00000111' ; if (W = 2)
      retlw b'00001110' ; if (W = 3)
      ;
key_enc_1_TB
      addwf PCL, F            ; PCL = PCL + W
      retlw b'00001010' ; if (W = 0)
      retlw b'00001011' ; if (W = 1)
      retlw b'00001100' ; if (W = 2)
      retlw b'00001101' ; if (W = 3)
      ;
key_enc_2_TB
      addwf PCL, F            ; PCL = PCL + W
      retlw b'00000011' ; if (W = 0)
      retlw b'00000110' ; if (W = 1)
      retlw b'00001001' ; if (W = 2)
      retlw b'00001111' ; if (W = 3)
      ;
key_enc_3_TB
      addwf PCL, F            ; PCL = PCL + W
      retlw b'00000010' ; if (W = 0)
      retlw b'00000101' ; if (W = 1)
      retlw b'00001000' ; if (W = 2)
      retlw b'00000000' ; if (W = 3)
      ;
;-------------------
; Encode key prefix
;-------------------
key_prefix_enc_TB
      addwf PCL, F            ; PCL = PCL + W
      retlw b'00000001' ; "A"
      retlw b'00000010' ; "B"
      retlw b'00000100' ; "C"
      retlw b'00001000' ; "D"
      retlw b'00010000' ; "*"
      retlw b'00100000' ; "#"
      ;
;--------------------------
; Randomize for Rand_var_h
;--------------------------
randm_varh_TB
      andlw 0x0f
      addwf PCL, F            ; PCL = PCL + W
      retlw 1           ; 1500 mSec ( 9)  W = 0
      retlw 0           ;  450 mSec ( 2)  W = 1
      retlw 2           ; 2100 mSec (13)  W = 2
      retlw 1           ;  900 mSec ( 5)  W = 3
      retlw 0           ;  600 mSec ( 3)  W = 4
      retlw 2           ; 1950 mSec (12)  W = 5
      retlw 1           ; 1050 mSec ( 6)  W = 6
      retlw 0           ;  150 mSec ( 0)  W = 7
      retlw 1           ; 1200 mSec ( 7)  W = 8
      retlw 2           ; 2250 mSec (14)  W = 9
      retlw 1           ; 1350 mSec ( 8)  W = 10
      retlw 2           ; 1650 mSec (10)  W = 11
      retlw 0           ;  300 mSec ( 1)  W = 12
      retlw 0           ;  750 mSec ( 4)  W = 13
      retlw 2           ; 1800 mSec (11)  W = 14
      retlw 2           ; 2400 mSec (15)  W = 15
      ;
;--------------------------
; Randomize for Rand_var_l
;--------------------------
randm_varl_TB
      andlw 0x0f
      addwf PCL, F            ; PCL = PCL + W
      retlw 0xd5        ; 1500 mSec ( 9)  W = 0
      retlw 0x8d        ;  450 mSec ( 2)  W = 1
```

```
      retlw 0x8a        ; 2100 mSec (13)  W = 2
      retlw 0x19        ;  900 mSec ( 5)  W = 3
      retlw 0xba        ;  600 mSec ( 3)  W = 4
      retlw 0x61        ; 1950 mSec (12)  W = 5
      retlw 0x48        ; 1050 mSec ( 6)  W = 6
      retlw 0x2f        ;  150 mSec ( 0)  W = 7
      retlw 0x77        ; 1200 mSec ( 7)  W = 8
      retlw 0xbf        ; 2250 mSec (14)  W = 9
      retlw 0xa6        ; 1350 mSec ( 8)  W = 10
      retlw 0x04        ; 1650 mSec (10)  W = 11
      retlw 0x5e        ;  300 mSec ( 1)  W = 12
      retlw 0xea        ;  750 mSec ( 4)  W = 13
      retlw 0x32        ; 1800 mSec (11)  W = 14
      retlw 0xee        ; 2400 mSec (15)  W = 15
      ;
;-----------------------
; Get next segment data
;-----------------------
get_next_seg_TB
      movf  Dt_val, W   ; W = dt_val
      andlw 0x03        ; W = (dt_val & 0x03)
      addwf PCL, F            ; PCL = PCL + W
      goto  get_next_seg1     ; if (dt_val = 0)
      goto  get_next_seg2     ; if (dt_val = 1)
      goto  get_next_seg3     ; if (dt_val = 2)
      goto  get_next_seg4     ; if (dt_val = 3)
      ;
;-----------------------
; "A-0", "A-1",,,, "A-9"
;-----------------------
key_pre_a_TB
      addwf PCL, F            ; PCL = PCL + W
      goto  key_pre_a0  ; "A-0"
      goto  key_pre_a1  ; "A-1"
      goto  key_pre_a2  ; "A-2"
      goto  key_pre_a3  ; "A-3"
      goto  key_pre_a4  ; "A-4"
      goto  key_pre_a5  ; "A-5"
      goto  key_pre_a6  ; "A-6"
      goto  key_pre_a7  ; "A-7"
      goto  key_pre_a8  ; "A-8"
      goto  key_pre_a9  ; "A-9"
      ;
;-----------------------
; "B-0", "B-1",,,, "B-9"
;-----------------------
key_pre_b_TB
      addwf PCL, F            ; PCL = PCL + W
      goto  key_pre_b0  ; "B-0"
      goto  key_pre_b1  ; "B-1"
      goto  key_pre_b2  ; "B-2"
      goto  key_pre_b3  ; "B-3"
      goto  key_pre_b4  ; "B-4"
      goto  key_pre_b9  ; "B-5"
      goto  key_pre_b9  ; "B-6"
      goto  key_pre_b9  ; "B-7"
      goto  key_pre_b9  ; "B-8"
      goto  key_pre_b9  ; "B-9"
      ;
;-----------------------
; "C-0", "C-1",,,, "C-9"
;-----------------------
key_pre_c_TB
      addwf PCL, F            ; PCL = PCL + W
      goto  key_pre_c0  ; "C-0"
      goto  key_pre_c1  ; "C-1"
      goto  key_pre_c2  ; "C-2"
```

```
      goto  key_pre_c3  ; "C-3"
      goto  key_pre_c4  ; "C-4"
      goto  key_pre_c5  ; "C-5"
      goto  key_pre_c6  ; "C-6"
      goto  key_pre_c7  ; "C-7"
      goto  key_pre_c8  ; "C-8"
      goto  key_pre_c9  ; "C-9"
      ;
;-------------------------------------------
; Input 4 digits numerics
;  for clock, start/end timer, travel counter
;-------------------------------------------
nkey_in1_TB
      swapf Keyin_mode, W     ; W = Keyin_mode{[3:0], [7:4]}
      andlw 0x03        ; W = keyin counter (3, 2, 1, 0)
      btfsc Keyin_mode, 0
      goto  nkey_in_clock     ; clock = 1
      btfsc Keyin_mode, 1
      goto  nkey_in_stimer    ; start timer = 1
      btfsc Keyin_mode, 2
      goto  nkey_in_etimer    ; end timer = 1
nkey_in_tc              ; travel counter = 1
      addwf PCL, F            ; PCL = PCL + W
      goto  nkey_tc_0   ; keyin counter = 0, travel counter
      goto  nkey_tc_1   ; keyin counter = 1, travel counter
      goto  nkey_tc_2   ; keyin counter = 2, travel counter
      goto  nkey_tc_3   ; keyin counter = 3, travel counter
      ;
nkey_in_clock
      addwf PCL, F            ; PCL = PCL + W
      goto  nkey_cl_0   ; keyin counter = 0, clock
      goto  nkey_cl_1   ; keyin counter = 1, clock
      goto  nkey_cl_2   ; keyin counter = 2, clock
      goto  nkey_cl_3   ; keyin counter = 3, clock
      ;
nkey_in_stimer
      addwf PCL, F            ; PCL = PCL + W
      goto  nkey_st_0   ; keyin counter = 0, start timer
      goto  nkey_st_1   ; keyin counter = 1, start timer
      goto  nkey_st_2   ; keyin counter = 2, start timer
      goto  nkey_st_3   ; keyin counter = 3, start timer
      ;
nkey_in_etimer
      addwf PCL, F            ; PCL = PCL + W
      goto  nkey_et_0   ; keyin counter = 0, end timer
      goto  nkey_et_1   ; keyin counter = 1, end timer
      goto  nkey_et_2   ; keyin counter = 2, end timer
      goto  nkey_et_3   ; keyin counter = 3, end timer
      ;
;-------------------
; Save travel count
;-------------------
save_tc_TB
      movf  Swing_num, W      ; W = Swing_num
      addwf PCL, F            ; PCL = PCL + W
      goto  nkey_tc_00  ; Swing_num = 0
      goto  nkey_tc_01  ; Swing_num = 1
      goto  nkey_tc_02  ; Swing_num = 2
      goto  nkey_tc_03  ; Swing_num = 3
      goto  nkey_tc_04  ; Swing_num = 4
      ;
;---------------------
; Recall travel count
;---------------------
recall_tc_TB
      movf  Swing_num, W      ; W = Swing_num
      addwf PCL, F            ; PCL = PCL + W
```

```
goto  recall_tc_00      ; Swing_num = 0
goto  recall_tc_01      ; Swing_num = 1
goto  recall_tc_02      ; Swing_num = 2
goto  recall_tc_03      ; Swing_num = 3
goto  recall_tc_04      ; Swing_num = 4
;
```

end

16F84A Source Code

```
;*********************************************************************
;Ultrasonic Bird Repeller
; Program name :  ubr2
; Module name  :  ubr2.asm
; Description  :
; Date         :  July 4, 2013
; Programmer   :  Tetsuji Oguchi
;(C) Oguchi R&D 2012-2013
;*********************************************************************

      LIST  P = 16F84A
#include    "P16F84A.INC"
 __config   _HS_OSC & _WDT_OFF & _PWRTE_ON
                        ; OSC = HS (20MHz), WDT off,
                        ; Power-up timer enabled

;*************************************
; Equates for general purpose registers *
;*************************************
Int_cnt           equ   0x10  ; Interrupt counter
Mute_Tcnt   equ   0x11  ; Mute period counter
                        ;  for Mute period definition
                        ;  randomized by "Mute_Tindex"
Mute_Tcnt_buf     equ   0x12  ; Mute period counter buffer
Mute_Tindex equ   0x13  ; Mute index counter
                        ;  for Mute period randomization index
Mute_mul_cnt      equ   0x14  ; Mute multiplication factor counter
                        ;  for Mute period definition
                        ;  randomized by "Int_cnt"
FM_Tcnt_0   equ   0x15  ; FM period counter 0
                        ;  for FM period at same frequency
                        ;  randomized by "Int_cnt"
FM_Fcnt_0   equ   0x16  ; FM frequency counter 0
                        ;  for FM pulse generation
                        ;  randomized by "FM_Findex_0"
FM_Findex_0 equ   0x17  ; FM frequency index 0
                        ;  for FM frequency randomization index
FM_Tcnt_1   equ   0x18  ; FM period counter 1
                        ;  for FM period at same frequency
                        ;  randomized by "Int_cnt"
FM_Fcnt_1   equ   0x19  ; FM frequency counter 1
                        ;  for FM pulse generation
                        ;  randomized by "FM_Findex_1"
FM_Findex_1 equ   0x1a  ; FM frequency index 1
                        ;  for FM frequency randomization index
OSC_Tcnt_0  equ   0x1b  ; OSC period counter 0
                        ;  for OSC period at same frequency
                        ;  randomized by "Int_cnt"
OSC_index_0 equ   0x1c  ; OSC index 0
                        ;  for OSC randomization (+= Int_cnt)
OSC_Tcnt_1  equ   0x1d  ; OSC period counter 1
                        ;  for OSC period at same frequency
                        ;  randomized by "Int_cnt"
OSC_index_1 equ   0x1e  ; OSC index 1
                        ;  for OSC randomization (+= Int_cnt)
Temp        equ   0x1f  ; Temporary register
            ;
;************************
; Equates for constants *
;************************
            ;
;mdelay_max equ   0x1f  ; maximum mute delay (not in use)

;***************
; Main program *
```

```
;***************
 org  0x0000
      ;
      nop               ; reserved for ICD (In-Circuit Debug)
      goto  main        ; start processing
      ;
;***********************
; Interrupt processing *
;***********************
 org  0x0004
      ;
      btfsc INTCON, TMR0IF
      goto  tmr0_int    ; timer 0 overflow interrupt
      retfie
      ;
;----------------------
; I/O port definition
;----------------------
 org  0x0100
      ;
main
      bcf   STATUS, RP1 ;
      bcf   STATUS, RP0 ; <page = 0>
      ;
      movlw 0x01
      movwf PORTA       ; PORTA[4] = FM = 0b
                        ; PORTA[3:0] = SEL_0[3:0] = 0001b
      ;
      movlw 0x80
      movwf PORTB       ; PORTB[7:4] = SEL_1[3:0] = 1000b
                        ; PORTB[3:2] = FM[1:0] = 00b
                        ; PORTB[1] (not in use) = 0b
                        ; PORTB[0] = ~MUTE = 0b
      ;
      bsf   STATUS, RP0 ; <page = 1>
      ;
      clrf  TRISA       ; PORTA[4] = FM = output (needs to pull-up)
                        ; PORTA[3:0] = SEL_0[3:0] = output
      ;
      clrf  TRISB       ; PORTB[7:4] = SEL_1[3:0] = output
                        ; PORTB[3:2] = FM[1:0] = output
                        ; PORTB[1] (not in use) = output
                        ; PORTB[0] = ~MUTE = output
      ;
      bcf   STATUS, RP0 ; <page = 0>
      ;
      movlw 0x03        ; <special treatment for _TB>
      movwf PCLATH            ; PCLATH = 0x03 <special treatment for _TB>
      ;
;--------------------------------------
; Initialize general purpose registers
;--------------------------------------
      clrf  Mute_Tindex ; Mute_Tindex = 0
      movlw 0xb1
      movwf Mute_Tcnt   ; Mute_Tcnt = 0xb1
      movwf Mute_Tcnt_buf     ; Mute_Tcnt_buf = 0xb1
      clrf  Mute_mul_cnt      ; Mute_mul_cnt = 0
      clrf  Int_cnt           ; Int_cnt = 0
      movlw 0x30
      movwf FM_Tcnt_0   ; FM_Tcnt_0 = 0x30
      movlw 0x18
      movwf FM_Fcnt_0   ; FM_Fcnt_0 = 0x18
      clrf  FM_Findex_0 ; FM_Findex_0 = 0
      movlw 0x57
      movwf OSC_Tcnt_0  ; OSC_Tcnt_0 = 0x57
      clrf  OSC_index_0 ; OSC_index_0 = 0
      movlw 0x75
```

```
      movwf OSC_Tcnt_1  ; OSC_Tcnt_1 = 0x75
      clrf  OSC_index_1 ; OSC_index_1 = 0
      ;
;---------------------------------
; Interrupt related initialization
;---------------------------------
      bsf   STATUS, RP0 ; <page = 1>
      ;
; Timer 0 overflow interrupt
      bcf   OPTION_REG,T0CS   ; TMR0 clock = CLKO
      bcf   OPTION_REG, PSA   ; prescaler assigned to TMR0
      bsf   OPTION_REG, PS2
      bsf   OPTION_REG, PS1
      bcf   OPTION_REG, PS0   ; prescaler rate PS[2:0] = 110b = 1/128
      ;
      bcf   STATUS, RP0 ; <page = 0>
      ;
      clrf  TMR0        ; TMR0[7:0] = 0
; clear interrupt flags
      bcf   INTCON, TMR0IF    ; clear TMR0 overflow interrupt flag
; set interrupt enables
      bsf   INTCON, TMR0IE    ; set TMR0 overflow interrupt enable
      bsf   INTCON, GIE ; set global interrupt enable
;-------------------------------------------------------
; Event driven by only TMR0 interrupt (every 6.55 mSec)
;-------------------------------------------------------
loop_main
      nop
      goto  loop_main
      ;
;*********************************************
; Timer 0 overflow interrupt (every 6.55 mSec) *
;*********************************************
tmr0_int
      bcf   INTCON, TMR0IF    ; clear TMR0 overflow interrupt flag
      ;
      incf  Int_cnt, F  ; Int_cnt++
      ;
;-------------------------------------------------
; Mute period randomization for both channel 0 & 1
;-------------------------------------------------
mute_proc
      decf  Mute_Tcnt, F      ; Mute_Tcnt--
      btfss STATUS, Z
      goto  fm0_proc    ; if (Z = 0) goto
      btfsc PORTB, 0
      goto  mute_offchk ; if (~Mute = 1) goto
;
; Mute On now (~Mute = 0)
;
mute_onchk
      movf  Mute_mul_cnt, W   ; W = Mute_mul_cnt
      btfss STATUS, Z
      goto  mute_offchk_0     ; if (Z = 0) goto
; Mute_mul_cnt = 0
      bsf   PORTB, 0    ; ~Mute = 1
      goto  mute_offchk_2
      ;
;
; Mute Off now (~Mute = 1)
;
mute_offchk
      movf  Mute_mul_cnt, W   ; W = Mute_mul_cnt
      btfsc STATUS, Z
      goto  mute_offchk_1     ; if (Z = 1) goto
; Mute_mul_cnt ~= 0
mute_offchk_0
```

```
      decf  Mute_mul_cnt, F   ; Mute_mul_cnt--
      movf  Mute_Tcnt_buf, W; W = Mute_Tcnt_buf
      movwf Mute_Tcnt   ; Mute_Tcnt = W
      goto  fm0_proc
      ;
; Mute_mul_cnt = 0
mute_offchk_1
      bcf   PORTB, 0    ; ~Mute = 0
      incf  Mute_Tindex, F    ; Mute_Tindex++ (make async Ton&Toff)
mute_offchk_2
      incf  Mute_Tindex, F    ; Mute_Tindex++
      movf  Mute_Tindex, W    ; W = Mute_Tindex
      call  monoff_cnt_TB
      iorlw 0x80        ; W |= 0x80
      movwf Mute_Tcnt   ; Mute_Tcnt = W
      movwf Mute_Tcnt_buf     ; Mute_Tcnt_buf = W
      movf  Int_cnt, W  ; W = Int_cnt
      call  t_mul_TB
      movwf Mute_mul_cnt      ; Mute_mul_cnt = W
      ;
;----------------------------------------
; Channel 0 processing
;  (1) Frequency moduration randomization
;  (2) Frequency oscillation randomization
;----------------------------------------
;-------------------------------------
; Frequency modulation randomization
;-------------------------------------
fm0_proc
      decf  FM_Fcnt_0, F      ; FM_Fcnt_0--
      btfsc STATUS, Z
      goto  fm0fcnt_zero      ; if (Z = 1) goto
      decf  FM_Tcnt_0, F      ; FM_Tcnt_0--
      btfss STATUS, Z
      goto  fsel0_proc  ; if (Z = 0) goto
      ;
fm0tcnt_zero
      movf  Int_cnt, W  ; W = Int_cnt
      call  monoff_cnt_TB
      movwf FM_Tcnt_0   ; FM_Tcnt_0 = W
      incf  FM_Findex_0, F    ; FM_Findex_0++
      movf  FM_Findex_0, W    ; W = FM_Findex_0
      call  monoff_cnt_TB
      movwf FM_Fcnt_0   ; FM_Fcnt_0 = W
      bcf   STATUS, C   ; C = 0
      rrf   FM_Fcnt_0, F      ; FM_Fcnt_0 >> 1
      goto  fsel0_proc
      ;
fm0fcnt_zero
      movf  FM_Findex_0, W    ; W = FM_Findex_0
      call  monoff_cnt_TB
      movwf FM_Fcnt_0   ; FM_Fcnt_0 = W
      btfsc PORTB, 2
      goto  fm0fcnt_zero_1    ; if (FM0 = 1) goto
      bsf   PORTB, 2    ; FM0 = 1
      goto  fsel0_proc
      ;
fm0fcnt_zero_1
      bcf   PORTB, 2    ; FM0 = 0
      ;
;-------------------------------------
; Frequency oscillation randomization
;-------------------------------------
fsel0_proc
      decf  OSC_Tcnt_0, F     ; OSC_Tcnt_0--
      btfss STATUS, Z
      goto  fm1_proc    ; if (Z = 0) goto
```

```
      movf  Int_cnt, W  ; W = Int_cnt
      call  monoff_cnt_TB
      movwf OSC_Tcnt_0  ; OSC_Tcnt_0 = W
      movf  Int_cnt, W  ; W = Int_cnt
      addwf OSC_index_0, F    ; OSC_index_0 += Int_cnt
      movf  OSC_index_0, W    ; W = OSC_index_0
      call  f_sel_TB
      btfss PORTA, 4
      goto  fsel0_1           ; if (FM = 0) goto
      addlw 0x10        ; W += 0x10
fsel0_1
      movwf PORTA       ; PORTA = W
      ;
;----------------------------------------
; Channel 1 processing
;  (1) Frequency moduration randomization
;  (2) Frequency oscillation randomization
;----------------------------------------
;-----------------------------------
; Frequency modulation randomization
;-----------------------------------
fm1_proc
      decf  FM_Fcnt_1, F      ; FM_Fcnt_1--
      btfsc STATUS, Z
      goto  fm1fcnt_zero      ; if (Z = 1) goto
      decf  FM_Tcnt_1, F      ; FM_Tcnt_1--
      btfss STATUS, Z
      goto  fsel1_proc  ; if (Z = 0) goto
      ;
fm1tcnt_zero
      movf  Int_cnt, W  ; W = Int_cnt
      call  monoff_cnt_TB
      movwf FM_Tcnt_1   ; FM_Tcnt_1 = W
      incf  FM_Findex_1, F    ; FM_Findex_1++
      movf  FM_Findex_1, W    ; W = FM_Findex_1
      call  monoff_cnt_TB
      movwf FM_Fcnt_1   ; FM_Fcnt_1 = W
      bcf   STATUS, C   ; C = 0
      rrf   FM_Fcnt_1, F      ; FM_Fcnt_1 >> 1
      goto  fsel1_proc
      ;
fm1fcnt_zero
      movf  FM_Findex_1, W    ; W = FM_Findex_1
      call  monoff_cnt_TB
      movwf FM_Fcnt_1   ; FM_Fcnt_1 = W
      btfsc PORTB, 3
      goto  fm1fcnt_zero_1    ; if (FM1 = 1) goto
      bsf   PORTB, 3    ; FM1 = 1
      goto  fsel1_proc
      ;
fm1fcnt_zero_1
      bcf   PORTB, 3    ; FM1 = 0
      ;
;-----------------------------------
; Frequency oscillation randomization
;-----------------------------------
fsel1_proc
      decf  OSC_Tcnt_1, F     ; OSC_Tcnt_1--
      btfss STATUS, Z
      retfie                  ; if (Z = 0)
      movf  Int_cnt, W  ; W = Int_cnt
      call  monoff_cnt_TB
      movwf OSC_Tcnt_1  ; OSC_Tcnt_1 = W
      movf  Int_cnt, W  ; W = Int_cnt
      addwf OSC_index_1, F    ; OSC_index_1 += Int_cnt
      movf  OSC_index_1, W    ; W = OSC_index_1
      call  f_sel_TB
```

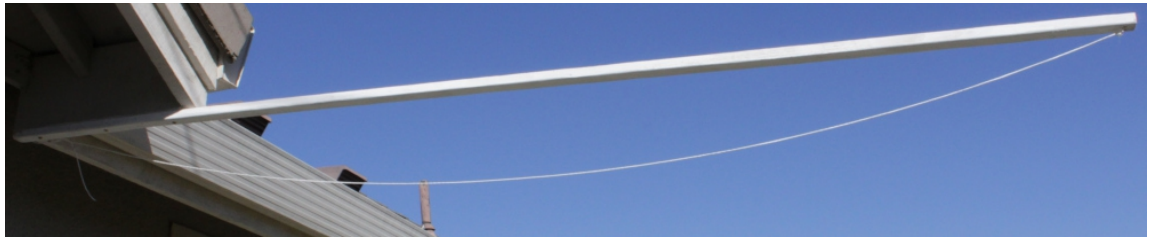
```
      movwf Temp        ; Temp = W
      swapf Temp, F           ; Temp[7:0] = Temp[3:0, 7:4]
      movf  PORTB, W    ; W = PORTB
      andlw 0x0f
      iorwf Temp, W           ; W |= Temp
      movwf PORTB       ; PORTB = W
      retfie
      ;
;**************
; Subroutines *
;**************
 org  0x0200
      ;
;******************************
; Table Branches (_TB) page 03 *
;******************************
 org  0x0300
      ;
;-------------------------
; Frequency randomization
;-------------------------
f_sel_TB
      andlw 0x3f
      addwf PCL, F            ; PCL = PCL + W
      retlw 1           ; W = 0
      retlw 2
      retlw 4
      retlw 8
      retlw 4
      retlw 2
      retlw 1
      retlw 2
      retlw 1           ; W = 8
      retlw 4
      retlw 1
      retlw 8
      retlw 2
      retlw 1
      retlw 2
      retlw 4
      retlw 2           ; W = 0x10
      retlw 8
      retlw 4
      retlw 1
      retlw 4
      retlw 2
      retlw 4
      retlw 8
      retlw 1           ; W = 0x18
      retlw 8
      retlw 2
      retlw 8
      retlw 4
      retlw 8
      retlw 4
      retlw 2
      retlw 1           ; W = 0x20
      retlw 2
      retlw 4
      retlw 8
      retlw 4
      retlw 8
      retlw 2
      retlw 8
      retlw 1           ; W = 0x28
      retlw 4
      retlw 8
```

```
      retlw 4
      retlw 2
      retlw 4
      retlw 1
      retlw 2
      retlw 8           ; W = 0x30
      retlw 2
      retlw 4
      retlw 2
      retlw 1
      retlw 4
      retlw 2
      retlw 8
      retlw 2           ; W = 0x38
      retlw 4
      retlw 1
      retlw 8
      retlw 2
      retlw 1
      retlw 4
      retlw 8           ; W = 0x3f
      ;
;-------------------------------------
; Mute on & off period randomization
;-------------------------------------
monoff_cnt_TB
      andlw 0x1f
      addwf PCL, F            ; PCL = PCL + W
      retlw 0xb1        ; 1200 mSec ()  W =  0
      retlw 0x72        ;  750 mSec ()  W =  1
      retlw 0x3a        ;  400 mSec ()  W =  2
      retlw 0xf0        ; 1650 mSec ()  W =  3
      retlw 0xa3        ; 1100 mSec ()  W =  4
      retlw 0x80        ;  850 mSec ()  W =  5
      retlw 0x4f        ;  550 mSec ()  W =  6
      retlw 0x79        ;  800 mSec ()  W =  7
      retlw 0xd4        ; 1450 mSec ()  W =  8
      retlw 0x41        ;  450 mSec ()  W =  9
      retlw 0x9c        ; 1050 mSec ()  W = 10
      retlw 0xc6        ; 1350 mSec ()  W = 11
      retlw 0x25        ;  250 mSec ()  W = 12
      retlw 0x87        ;  900 mSec ()  W = 13
      retlw 0xaa        ; 1150 mSec ()  W = 14
      retlw 0x1e        ;  200 mSec ()  W = 15
      retlw 0xdb        ; 1500 mSec ()  W = 16
      retlw 0x5d        ;  650 mSec ()  W = 17
      retlw 0xb8        ; 1250 mSec ()  W = 18
      retlw 0x33        ;  350 mSec ()  W = 19
      retlw 0xe9        ; 1600 mSec ()  W = 20
      retlw 0x95        ; 1000 mSec ()  W = 21
      retlw 0x17        ;  150 mSec ()  W = 22
      retlw 0xbf        ; 1300 mSec ()  W = 23
      retlw 0x48        ;  500 mSec ()  W = 24
      retlw 0x8e        ;  950 mSec ()  W = 25
      retlw 0xf7        ; 1700 mSec ()  W = 26
      retlw 0x56        ;  600 mSec ()  W = 27
      retlw 0xcd        ; 1400 mSec ()  W = 28
      retlw 0x64        ;  700 mSec ()  W = 29
      retlw 0xe2        ; 1550 mSec ()  W = 30
      retlw 0x2c        ;  300 mSec ()  W = 31
      ;
;-------------------------------------
; Multiplication factor randomization
;-------------------------------------
t_mul_TB
      andlw 0x07
      addwf PCL, F            ; PCL = PCL + W
```

```
retlw 0x04        ; 4 * Tcnt mSec ()  W =  0
retlw 0x00        ; 0 * Tcnt mSec ()  W =  1
retlw 0x07        ; 7 * Tcnt mSec ()  W =  2
retlw 0x03        ; 3 * Tcnt mSec ()  W =  3
retlw 0x06        ; 6 * Tcnt mSec ()  W =  4
retlw 0x01        ; 1 * Tcnt mSec ()  W =  5
retlw 0x05        ; 5 * Tcnt mSec ()  W =  6
retlw 0x02        ; 2 * Tcnt mSec ()  W =  7
;
```

end

Bird Repeller by Eagle Kite



Installing

Eagle kite violently flying (Birds scared and cleared out...)

Two eagle kites flying vividly