```
$NOLIST
$MODLP51
$LIST

; Pin Declarations
SOUND_OUT     equ P1.1
First_Button  equ P0.1
Second_Button equ P0.3
Third_Button  equ P0.6
BOOT          equ P2.4
Fifth_Button  equ P2.2


CLK           EQU 22118400 ; Microcontroller system crystal frequency in Hz
TIMER0_RATE   EQU 4200     ; 2100Hz (High Freq)
TIMER0_RELOAD EQU ((65536-(CLK/TIMER0_RATE)))
TIMER1_RATE   EQU 4000     ; 2000Hz (Low Freq)
TIMER1_RELOAD EQU ((65536-(CLK/TIMER1_RATE)))
CReload EQU ((65536-(CLK/2093)))
EReload EQU ((65536-(CLK/2637)))
GReload EQU ((65536-(CLK/3135)))
GlReload EQU ((65536-(CLK/1568)))
TIMER1 EQU 4200
TIMER1R EQU ((65536-(CLK/TIMER1)))

; Timer 1,2,3 Declarations
org 0000H
   ljmp MyProgram

org 0x000B
      ljmp Timer0_ISR

org 0x0001B
      ljmp Timer1_ISR

; Timer/Counter 2 overflow interrupt vector
org 0x002B
      ljmp Timer2_ISR

; These register definitions needed by 'math32.inc'
      DSEG at 30H
      x:   ds 4
      y:   ds 4
      bcd: ds 5
;16-bit timer 2 overflow (to measure the period of very slow signals)
      sf: ds 3 ; status flag
      Seed: ds 4 ;random number seed
      counter: ds 1
      counterMajor: ds 1
      Period_A: ds 3
      Period_B: ds 3
      T2ov:     ds 1
      player1count: ds 1
      player2count: ds 1
      player3count: ds 1
      player4count: ds 1
      Count1ms:     ds 2
      speed: ds 1
```

```
; Flag Declarations
BSEG
      mf: dbit 1
      fr_flag: dbit 1 ;to determine frequency of speaker
      alarm_flag: dbit 1 ;to determine whether speaker is on/off
      start_flag: dbit 1 ;to determine whether to play starting sounds
      C_flag: dbit 1
      G_flag: dbit 1
      Gl_flag: dbit 1
      E_flag: dbit 1
      wait_flag: dbit 1
      win1_flag: dbit 1
      cheatcodeflag: dbit 1
      finalroundflag: dbit 1
      speedflag: dbit 1
      bigbugflag: dbit 1
      bigbugflag2: dbit 1
      bigbugflag3: dbit 1
      bigbugflag4: dbit 1

;Include math32.inc since without it were quite useless
$NOLIST
$include(math32.inc)
$LIST

; LCD pin declarations
cseg
; These 'equ' must match the hardware wiring
      LCD_RS equ P3.2
;LCD_RW equ PX.X ; Not used in this code, connect the pin to GND
      LCD_E  equ P3.3
      LCD_D4 equ P3.4
      LCD_D5 equ P3.5
      LCD_D6 equ P3.6
      LCD_D7 equ P3.7

;Including LCD_4BIT.inc
      $NOLIST
      $include(LCD_4bit.inc) ; A library of LCD related functions and utility
macros
      $LIST

;                     1234567890123456    <- This helps determine the location of
the counter (no one really knows how tho)

;String Messages for LCD
      Boot_Message1:     db 'Insert Coin', 0
      Boot_Message2:     db '  To Play  ', 0
      No_Signal_Str:    db 'No signal     %', 0
      C1:    db 'P1: ', 0
      C2:    db 'P2:', 0
      C3:    db 'P3: ', 0
      C4:    db 'P4:', 0
      clear:    db '  ', 0
      pressdetected: db 'Detected  ', 0
      win: db 'Wi', 0
      speeding: db '  Speeding up!  ' , 0
    speeding2:db  'LUDDDDAACCRRISSS' , 0
    finalround: db '  Final Round   ' , 0
```

```
    empty: db '                ' , 0
    roundd: db '   Round Two    ' , 0
      seven: db '   ', 0
; Sends 10-digit BCD number in bcd to the LCD
Display_10_digit_BCD:
      Display_BCD(bcd+4)
      Display_BCD(bcd+3)
      Display_BCD(bcd+2)
      Display_BCD(bcd+1)
      Display_BCD(bcd+0)
      ret

Timer0_Init:
      mov a, TMOD
      anl a, #0xf0 ; Clear the bits for timer 0
      orl a, #0x01 ; Configure timer 0 as 16-timer
      mov TMOD, a
      mov TH0, #high(TIMER0_RELOAD)
      mov TL0, #low(TIMER0_RELOAD)
      ; Set autoreload value
      mov RH0, #high(TIMER0_RELOAD)
      mov RL0, #low(TIMER0_RELOAD)
      ; Enable the timer and interrupts
    setb ET0  ; Enable timer 0 interrupt
    setb TR0  ; Start timer 0
      ret

CNote:
      mov TH0, #high(CReload)
      mov TL0, #low(CReload)
      ; Set autoreload value
      mov RH0, #high(CReload)
      mov RL0, #low(CReload)
      ljmp HighFr

GNote:
      mov TH0, #high(GReload)
      mov TL0, #low(GReload)
      ; Set autoreload value
      mov RH0, #high(GReload)
      mov RL0, #low(GReload)
      ljmp HighFr

GlNote:
      mov TH0, #high(GlReload)
      mov TL0, #low(GlReload)
      ; Set autoreload value
      mov RH0, #high(GlReload)
      mov RL0, #low(GlReload)
      ljmp HighFr

ENote:
      mov TH0, #high(EReload)
      mov TL0, #low(EReload)
      ; Set autoreload value
      mov RH0, #high(EReload)
      mov RL0, #low(EReload)
      ljmp HighFr
```

```
Timer0_ISR:
      ;clr TF0  ; According to the data sheet this is done for us already.
      jnb alarm_flag, Done
      jb C_flag, CNote
      jb G_flag, GNote
      jb Gl_flag, GlNote
      jb E_flag, ENote
      jnb fr_flag, LowFr
      mov TH0, #high(TIMER0_RELOAD)
      mov TL0, #low(TIMER0_RELOAD)
      ; Set autoreload value
      mov RH0, #high(TIMER0_RELOAD)
      mov RL0, #low(TIMER0_RELOAD)
      ljmp HighFr

LowFr:
      mov TH0, #high(TIMER1_RELOAD)
      mov TL0, #low(TIMER1_RELOAD)
      ; Set autoreload value
      mov RH0, #high(TIMER1_RELOAD)
      mov RL0, #low(TIMER1_RELOAD)

HighFr:
      cpl SOUND_OUT ; Connect speaker to P1.1!

Done:
      reti



Timer1_Init:
      mov a, TMOD
      anl a, #0xf0 ; Clear the bits for timer 1
      orl a, #0x01 ; Configure timer 1 as 16-timer   mov TMOD, a
      mov TH1, #high((TIMER1R))
      mov TL1, #low((TIMER1R))
 ;Set autoreload value
      mov RH1, #high((TIMER1R))
      mov RL1, #low((TIMER1R))
 ;Enable the timer and interrupts
      setb ET1  ; Enable timer 0 interrupt
      setb TR1  ; Start timer 0
      ret

Timer1_ISR:
      ; The two registers used in the ISR must be saved in the stack
      ;push acc
      ;push psw
      jnb wait_flag, Timer1_ISR_done ;maybe delete this
      ; Increment the 16-bit one mili second counter
      mov a, counter
    add a, #0x01
    da a
    mov counter, a
      cjne a, #0x99, Timer1_ISR_done

Inc_Done:
;try regular flag here?
      ; Check if half second has passed
```

```
	; 500 milliseconds have passed.  Set a flag so the main program knows
	clr wait_flag ; Let the main program know half second had passed
	;cpl TR0 ; Enable/disable timer/counter 0. This line creates a beep-silence-
beep-silence sound.
	; Reset to zero the milli-seconds counter, it is a 16-bit variable
	mov a, #0x00
	mov counter, a

Timer1_ISR_done:

	;pop psw
	;pop acc
	reti

;Initializes timer/counter 2 as a 16-bit timer
Timer2_ISR:
	clr TF2  ; Timer 2 doesn't clear TF2 automatically. Do it in ISR.
	inc T2ov

	jnb wait_flag, Timer2_ISR_done ;maybe delete this
	; Increment the 16-bit one mili second counter
	mov a, counter
    add a, #0x01
    da a
    mov counter, a
    jb finalroundflag, HighSpeed
    jb speedflag, MedSpeed
LowSpeed:
	cjne a, #0x99, Timer2_ISR_done ;add variable in place of 99 so i can
decrement it
	ljmp Inc_Major
MedSpeed:
	cjne a, #0x60, Timer2_ISR_done ;add variable in place of 99 so i can
decrement it
	ljmp Inc_Major
HighSpeed:
	cjne a, #0x45, Timer2_ISR_done ;add variable in place of 99 so i can
decrement it
Inc_Major:
	mov a, #0x00
	mov counter, a

	mov a, counterMajor
	add a, #0x01
	da a
	mov counterMajor, a
	cjne a, #0x03 , Timer2_ISR_done


Inc_Done2:
;try regular flag here?
	; Check if half second has passed


	; 500 milliseconds have passed.  Set a flag so the main program knows
	clr wait_flag ; Let the main program know half second had passed
	;cpl TR0 ; Enable/disable timer/counter 0. This line creates a beep-silence-
```

```
beep-silence sound.
      ; Reset to zero the milli-seconds counter, it is a 16-bit variable
      mov a, #0x00
      mov counterMajor, a
Timer2_ISR_done:
      reti

; When using a 22.1184MHz crystal in fast mode
; one cycle takes 1.0/22.1184MHz = 45.21123 ns
; (tuned manually to get as close to 1s as possible)
Wait1s:
    mov R2, #176
X3: mov R1, #250
X2: mov R0, #166
X1: djnz R0, X1 ; 3 cycles->3*45.21123ns*166=22.51519us
    djnz R1, X2 ; 22.51519us*250=5.629ms
    djnz R2, X3 ; 5.629ms*176=1.0s (approximately)
    ret

;Initializes timer/counter 2 as a 16-bit timer
InitTimer2:
      mov T2CON, #0b_0000_0000 ; Stop timer/counter.  Set as timer (clock input is
pin 22.1184MHz).
      ; Set the reload value on overflow to zero (just in case is not zero)
      mov RCAP2H, #0
      mov RCAP2L, #0
      setb ET2  ; Enable timer 2 interrupt to count overflow
    ret

;Converts the hex number in T2ov-TH2 to BCD in R2-R1-R0
hex2bcd5:
      clr a
    mov R0, #0  ;Set BCD result to 00000000
    mov R1, #0
    mov R2, #0
    mov R3, #16 ;Loop counter.

hex2bcd_loop:
    mov a, TH2 ;Shift T2ov-TH2 left through carry
    rlc a
    mov TH2, a

    mov a, T2ov
    rlc a
    mov T2ov, a

      ; Perform bcd + bcd + carry
      ; using BCD numbers
      mov a, R0
      addc a, R0
      da a
      mov R0, a

      mov a, R1
      addc a, R1
      da a
      mov R1, a

      mov a, R2
```

```
      addc a, R2
      da a
      mov R2, a

      djnz R3, hex2bcd_loop
      ret

; Dumps the 5-digit packed BCD number in R2-R1-R0 into the LCD
DisplayBCD_LCD:
      ; 5th digit:
    mov a, R2
    anl a, #0FH
    orl a, #'0' ; convert to ASCII
      lcall ?WriteData
      ; 4th digit:
    mov a, R1
    swap a
    anl a, #0FH
    orl a, #'0' ; convert to ASCII
      lcall ?WriteData
      ; 3rd digit:
    mov a, R1
    anl a, #0FH
    orl a, #'0' ; convert to ASCII
      lcall ?WriteData
      ; 2nd digit:
    mov a, R0
    swap a
    anl a, #0FH
    orl a, #'0' ; convert to ASCII
      lcall ?WriteData
      ; 1st digit:
    mov a, R0
    anl a, #0FH
    orl a, #'0' ; convert to ASCII
      lcall ?WriteData

    ret

SeedGen:
      setb TR2
      jb BOOT, $
      mov Seed+0, TH2
      mov Seed+1, #0x01
      mov Seed+2, #0x87
      mov Seed+3, TL2
      clr TR2
      ret

Random:
      mov x+0, Seed+0
      mov x+1, Seed+1
      mov x+2, Seed+2
      mov x+3, Seed+3
      Load_y(214013)
      lcall mul32
      Load_y(2531011)
      lcall add32
      mov Seed+0, x+0
```

```
      mov Seed+1, x+1
      mov Seed+2, x+2
      mov Seed+3, x+3
      ret

Wait_Random:
      Wait_Milli_Seconds(Seed+0)
      Wait_Milli_Seconds(Seed+1)
      Wait_Milli_Seconds(Seed+2)
      Wait_Milli_Seconds(Seed+3)
      ret

Wait_Random_Compounded:
      lcall Wait_Random
      lcall Wait_Random
      lcall Wait_Random
      lcall Wait_Random
      ret

Wait:
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#50)
      ret

Wait_Full:
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#100)
      ret

Wait_Half:
      Wait_Milli_Seconds(#100)
      Wait_Milli_Seconds(#50)
      ret

;---------------------------------;
; Hardware initialization         ;
;---------------------------------;

Initialize_All:
      lcall LCD_4BIT ; Initialize LCD

      Set_Cursor(1, 1)
   Send_Constant_String(#Boot_Message1)
   Set_Cursor(2, 1)
   Send_Constant_String(#Boot_Message2)

   setb start_flag
   lcall Timer0_Init
      lcall SeedGen
   lcall InitTimer2
   setb EA
   clr win1_flag
   clr fr_flag
```

```
   mov a, #0x00
      mov counter, a
      mov a, #0x00
      mov counterMajor, a
      clr speedflag
      clr finalroundflag

      clr bigbugflag
      clr bigbugflag2
      clr bigbugflag3
      clr bigbugflag4

      ret

;---------------------------------;
; Main program loop               ;
;---------------------------------;

MyProgram:
    ; Initialize the hardware:
    mov SP, #7FH
    lcall Initialize_All
    setb P2.0 ; Pin is used as input for player 1
       setb P2.1 ; Pin is used as input for player 2

       Set_Cursor(1, 1)
    Send_Constant_String(#C1)
    Set_Cursor(1,10)
       Send_Constant_String(#C2)
       Set_Cursor(2, 1)
    Send_Constant_String(#C3)
    Set_Cursor(2,10)
       Send_Constant_String(#C4)
       Set_Cursor(1,7)
       Send_Constant_String(#seven)
       Set_Cursor(2,7)
       Send_Constant_String(#seven)



       mov a, #0x00
      mov player1count, a

            mov a, #0x00
      mov player2count, a

             mov a, #0x00
      mov player3count, a

            mov a, #0x00
      mov player4count, a

forever:                                                              ;player 1
  ; Measure the period applied to pin P2.0
    clr TR2 ; Stop counter 2
    mov TL2, #0
    mov TH2, #0
    mov T2ov, #0
    jb P2.0, $
```

```
    jnb P2.0, $
    mov R0, #0 ; 0 means repeat 256 times
    setb TR2 ; Start counter 0
meas_loop1:
    jb P2.0, $
    jnb P2.0, $
    djnz R0, meas_loop1 ; Measure the time of 100 periods
    clr TR2 ; Stop counter 2, TH2-TL2 has the period
    ; save the period of P2.0 for later use

      mov x+0, TL2
      mov x+1, TH2
      mov x+2, #0
      mov x+3, #0

      Load_y(2400)
    lcall sub32

    ;load_y(1500)
;     lcall x_gt_y
;     jnb mf, okayyylessgoo
;     Load_x(1500)

    okayyylessgoo:
  ;  Set_Cursor(1,1)

  ;  lcall hex2bcd
;     lcall Display_10_digit_BCD
;
      jb win1_flag, contt
      Set_Cursor(1, 14)
      Display_BCD(player1count)

;     bigbugdetection3:
;     mov a, player1count
;     cjne a, #0x01, bigbugdetection4
;     setb bigbugflag3
;;
;     bigbugdetection4:
;;    mov a, player1count
;     cjne a, #0x09, contt
;     setb bigbugflag4

      contt:
 ;   Set_Cursor(1, 1)
;     lcall hex2bcd5
 ;   lcall DisplayBCD_LCD

    jb cheatcodeflag, display
      jnb alarm_flag, display
      jb start_flag, display
      jnb fr_flag, decre
    load_y(1400)
      lcall x_gt_y
      jnb mf, wejump
    mov a, player1count
    add a, #0x01
    da a
    mov player1count, a
```

```
    setb cheatcodeflag
    sjmp display

    decre:
    mov a, player1count
    cjne a, #0x00,decc
    sjmp display

    decc:
    load_y(1400)
      lcall x_gt_y
      jnb mf, display
    dec player1count
    setb cheatcodeflag
    sjmp display

wejump3:
ljmp round2
wejump:
ljmp cont

    display:
    mov a, player1count
    cjne a, #0x05, wejump3
;    jnb bigbugflag3, wejump3
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#roundd)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
    setb speedflag
 ;   clr bigbugflag3
    ljmp round2

wejump2:
ljmp wincondition

    round2:
    mov a, player1count
    cjne a, #0x010, wejump2
  ;  jnb bigbugflag4, wejump2

 ;   Set_Cursor(1, 1)
```

```
;     Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;    Set_Cursor(1, 1)
;     Send_Constant_String(#finalround)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
      ;clr bigbugflag4
    setb finalroundflag

    wincondition:
    cjne a, #0x15, cont
    Set_Cursor(1, 14)
      Send_Constant_String(#win)
      setb win1_flag


cont:                                                                   ;play
er 2
    ; Measure the period applied to pin P2.1
    clr TR2 ; Stop counter 2
    mov TL2, #0
    mov TH2, #0
    mov T2ov, #0
    jb P2.1, $
    jnb P2.1, $
    mov R0, #0 ; 0 means repeat 256 times
    setb TR2 ; Start counter 0
meas_loop2:
    jb P2.1, $
    jnb P2.1, $
    djnz R0, meas_loop2 ; Measure the time of 100 periods
    clr TR2 ; Stop counter 2, TH2-TL2 has the period
    ; save the period of P2.1 for later use

      mov x+0, TL2
      mov x+1, TH2
      mov x+2, #0
      mov x+3, #0

      Load_y(5200)
    lcall sub32

;    load_y(1300)
;     lcall x_gt_y
;     jnb mf, okayyylessgo
```

```
;     Load_x(1300)

      okayyylessgo:

  ; Set_Cursor(2,1)
 ;   lcall hex2bcd
;     lcall Display_10_digit_BCD

      jb win1_flag, conttt
      Set_Cursor(1, 5)
      Display_BCD(player2count)

;     bigbugdetection:
;     mov a, player2count
;     cjne a, #0x01, bigbugdetection2
;     setb bigbugflag
;
;     bigbugdetection2:
;     mov a, player2count
;     cjne a, #0x09, bigbugdetection2
;     setb bigbugflag2

      conttt:
 ;   Set_Cursor(2, 1)
;     lcall hex2bcd5
 ;   lcall DisplayBCD_LCD

      jb cheatcodeflag, display2
      jnb alarm_flag, display2
      jb start_flag, display2
      jnb fr_flag, decre2
    load_y(1600)
      lcall x_gt_y
      jnb mf, display2
    mov a, player2count
    add a, #0x01
    da a
    mov player2count, a
    setb cheatcodeflag
    sjmp display2

    decre2:
    mov a, player2count
    cjne a, #0x00,deccc
    sjmp display2

    deccc:
    Load_y(1600)
      lcall x_gt_y
      jnb mf, display2
    dec player2count
    setb cheatcodeflag
    sjmp display2

wejump4:
ljmp rround2

    display2:
    mov a, player2count
```

```
    cjne a, #0x05, wejump4
 ;   jnb bigbugflag, wejump4
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#roundd)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
    setb speedflag
  ;  clr bigbugflag
    sjmp rround2

    wejump5:
    ljmp wincondition2

    rround2:
    mov a, player2count
    cjne a, #0x10, wejump5
   ; jnb bigbugflag2, wejump5
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#finalround)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
      ;clr bigbugflag2
    setb finalroundflag
```

```
wincondition2:

    mov a, player2count
    cjne a, #0x15, player3
    Set_Cursor(1, 5)
      Send_Constant_String(#win)
      setb win1_flag

      ;                                                                    player 3
player3:
    ; Measure the period applied to pin P2.1
    clr TR2 ; Stop counter 2
    mov TL2, #0
    mov TH2, #0
    mov T2ov, #0
    jb P0.0, $
    jnb P0.0, $
    mov R0, #0 ; 0 means repeat 256 times
    setb TR2 ; Start counter 0

meas_loop3:
    jb P0.0, $
    jnb P0.0, $
    djnz R0, meas_loop3 ; Measure the time of 100 periods
    clr TR2 ; Stop counter 2, TH2-TL2 has the period
    ; save the period of P2.1 for later use

      mov x+0, TL2
      mov x+1, TH2
      mov x+2, #0
      mov x+3, #0

      Load_y(8000)
    lcall sub32

      okayyylessgo2:

 ;  Set_Cursor(1,1)
  ;  lcall hex2bcd
      ;lcall Display_10_digit_BCD

      jb win1_flag, contttt
      Set_Cursor(2, 5)
      Display_BCD(player3count)

;     bigbugdetection:
;     mov a, player2count
;     cjne a, #0x01, bigbugdetection2
;     setb bigbugflag
;
;     bigbugdetection2:
;     mov a, player2count
;     cjne a, #0x09, bigbugdetection2
;     setb bigbugflag2

      contttt:
 ;   Set_Cursor(2, 1)
;     lcall hex2bcd5
 ;   lcall DisplayBCD_LCD
```

```
      jb cheatcodeflag, display3
      jnb alarm_flag, display3
      jb start_flag, display3
      jnb fr_flag, decre3
    load_y(1400)
      lcall x_gt_y
      jnb mf, display3
    mov a, player3count
    add a, #0x01
    da a
    mov player3count, a
    setb cheatcodeflag
    sjmp display3

    decre3:
    mov a, player3count
    cjne a, #0x00,deccc3
    sjmp display3

    deccc3:
    Load_y(1400)
      lcall x_gt_y
      jnb mf, display3
    dec player3count
    setb cheatcodeflag
    sjmp display3

wejump7:
ljmp rround3

    display3:
    mov a, player2count
    cjne a, #0x05, wejump7
 ;   jnb bigbugflag, wejump4
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#roundd)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
    setb speedflag
  ;;  clr bigbugflag
    sjmp rround3
```

```
    wejump8:
    ljmp wincondition3

    rround3:
    mov a, player3count
    cjne a, #0x10, wejump8
   ; jnb bigbugflag2, wejump5
  ;  Set_Cursor(1, 1)
      ;Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#finalround)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
      ;clr bigbugflag2
    setb finalroundflag


wincondition3:

    mov a, player3count
    cjne a, #0x15, player4
    Set_Cursor(2,5)
      Send_Constant_String(#win)
      setb win1_flag
;                                                                       player 4

player4:
    ; Measure the period applied to pin P2.1
    clr TR2 ; Stop counter 2
    mov TL2, #0
    mov TH2, #0
    mov T2ov, #0
    jb P0.1, $
    jnb P0.1, $
    mov R0, #0 ; 0 means repeat 256 times
    setb TR2 ; Start counter 0

meas_loop4:
    jb P0.1, $
    jnb P0.1, $
    djnz R0, meas_loop4 ; Measure the time of 100 periods
    clr TR2 ; Stop counter 2, TH2-TL2 has the period
    ; save the period of P2.1 for later use
```

```
      mov x+0, TL2
      mov x+1, TH2
      mov x+2, #0
      mov x+3, #0

      Load_y(12000)
    lcall sub32

      okayyylessgo3:

  ;  Set_Cursor(2,14)
   ; lcall hex2bcd
      ;lcall Display_10_digit_BCD

      jb win1_flag, contttttt
      Set_Cursor(2, 14)
      Display_BCD(player4count)

;     bigbugdetection:
;     mov a, player2count
;     cjne a, #0x01, bigbugdetection2
;     setb bigbugflag
;
;     bigbugdetection2:
;     mov a, player2count
;     cjne a, #0x09, bigbugdetection2
;     setb bigbugflag2

      contttttt:
 ;   Set_Cursor(2, 1)
;     lcall hex2bcd5
 ;   lcall DisplayBCD_LCD

      jb cheatcodeflag, display4
      jnb alarm_flag, display4
      jb start_flag, display4
      jnb fr_flag, decre4
    load_y(1000)
      lcall x_gt_y
      jnb mf, display4
    mov a, player4count
    add a, #0x01
    da a
    mov player4count, a
    setb cheatcodeflag
    sjmp display4

    decre4:
    mov a, player4count
    cjne a, #0x00,deccc4
    sjmp display4

    deccc4:
    Load_y(1000)
      lcall x_gt_y
      jnb mf, display4
    dec player4count
    setb cheatcodeflag
```

```
    sjmp display4

wejump12:
ljmp rround4

    display4:
    mov a, player4count
    cjne a, #0x05, wejump12
 ;   jnb bigbugflag, wejump4
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#roundd)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding)
    setb speedflag
  ;  clr bigbugflag
    sjmp rround4

    wejump15:
    ljmp wincondition4

    rround4:
    mov a, player4count
    cjne a, #0x10, wejump15
   ; jnb bigbugflag2, wejump5
;    Set_Cursor(1, 1)
;     Send_Constant_String(#empty)
;     Set_Cursor(2, 1)
;     Send_Constant_String(#empty)
 ;   Set_Cursor(1, 1)
;     Send_Constant_String(#finalround)
;     lcall Wait_Full
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#speeding2)
;     lcall Wait_Half
;     Set_Cursor(2,1)
;     Send_Constant_String(#empty)
```

```
;      lcall Wait_Half
;      Set_Cursor(2,1)
;      Send_Constant_String(#speeding2)
;      ;clr bigbugflag2
    setb finalroundflag


wincondition4:
    mov a, player4count
    cjne a, #0x15, soundstuff
    Set_Cursor(2, 14 )
      Send_Constant_String(#win)
      setb win1_flag
      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
soundstuff:
      jb win1_flag, connect
    jnb start_flag, NoSoundBridge

      ljmp StartingSounds

NoSoundBridge:
      ljmp NoSound

StartingSounds:
      setb fr_flag
      setb E_flag
      setb alarm_flag
      lcall Wait_Half

      clr alarm_flag
      Wait_Milli_Seconds(#20)

      setb alarm_flag
      lcall Wait_Half

      clr alarm_flag
      lcall Wait_Half

      setb alarm_flag
      lcall Wait_Half

      clr alarm_flag
      lcall Wait_Half

      setb alarm_flag
      setb C_flag
      lcall Wait_Half

      clr alarm_flag
      clr C_flag
      Wait_Milli_Seconds(#20)

      setb alarm_flag
      lcall Wait_Full

      clr alarm_flag
      Wait_Milli_Seconds(#20)

      setb alarm_flag
```

```
      setb G_flag
      lcall Wait_Full

      clr alarm_flag
      clr G_flag
      lcall Wait_Full

      setb alarm_flag
      setb Gl_flag
      lcall Wait_Full
      lcall Wait_Full
      clr Gl_flag
      clr E_flag
      clr start_flag
      sjmp NoSound

connect:
ljmp theend

NoSound:

      jb wait_flag, Waiting
      clr alarm_flag
      lcall Random
      lcall Wait_Random_Compounded
      lcall Random
      mov a, Seed+1
      mov c, acc.3
      mov fr_flag, c
      setb alarm_flag
      setb wait_flag
      clr cheatcodeflag

Waiting:
sjmp repeat
theend:
clr alarm_flag
Repeat:
    ljmp forever ; Repeat!


end
```