

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**BÁO CÁO**

**8051+BUZZER**

**TT Kiến Trúc Và Tổ Chức Máy Tính**

**GVHD: Huỳnh Hoàng Hà**

**Mã học phần: COOL325364\_23\_1\_10**

**Ngày, tiết học: Thứ sáu tiết 1-5**

**SVTH: Nguyễn Tài Như Tâm**

**MSSV: 21139053**

**TP. Thủ Đức - 11/2023**

Mục Lục

[**1.Buzzer thụ động:** 3](#_Toc151101887)

[**2.Schematic:** 4](#_Toc151101888)

[**3. Giải thích Code** 4](#_Toc151101889)

[**3.1. Alarm ( beep.c)** 4](#_Toc151101890)

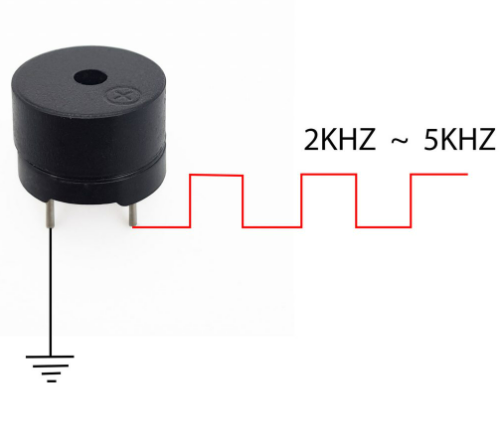
[**3.2 Happy (Buzz.asm)** 6](#_Toc151101891)

[**3.3 sound ( beep.c)** 7](#_Toc151101892)

[**3.4 Happy ( HAPPY.ASM)** 14](#_Toc151101893)

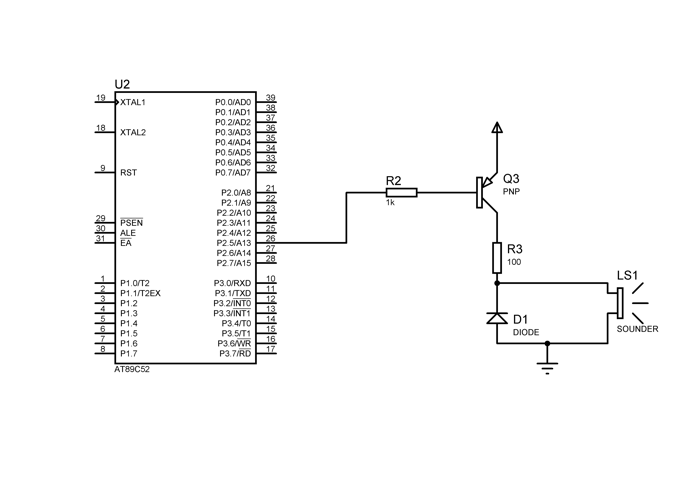
[**3.5 X ( znpa.asm)** 18](#_Toc151101894)

# **1.Buzzer thụ động:**

****

* Còi thụ động (passive buzzer) là bộ rung động tạo ra âm thanh có tần số khác nhau. Bên trong còi không có mạch dao động nên khi cấp nguồn 5V còi không tạo ra âm thanh giống như còi chủ động. Thay vào đó ta cần dùng sóng vuông có tần số từ 2 kHz đến 5 kHz để tạo tần số dao động phát ra âm thanh.
* Ta có thể dùng xung PWM tạo ra từ vi điều khiển để cấp cho còi thụ động. Việc thay đổi tần số xung PWM làm thay đổi tần số dao động và phát ra âm có tần số khác nhau. Ví dụ cấp xung 523 Hz sẽ tạo ra âm Do, 587 Hz tạo âm Re. Về cơ bản, ta có thể chơi một bản nhạc với còi thụ động

# **2.Schematic:**

****

**3. Giải thích Code**

**3.1. Alarm ( beep.c)**

|  |
| --- |
| #include <reg51.h>  sbit Beep = P2^5 ; // nối chân p2.5  void delay(unsigned int i) ;  void main()  {  Beep= 1; // Chân P2.5 được đặt lên mức logic high 🡪có tiếng beep  delay(5);  Beep= 0; //Chân P2.5 được đặt lên mức logic low 🡪 k có tiếng beep  delay(5);  }  // tạo hàm deley 1ms  void delay(unsigned int i)  {  char j;  for(i; i > 0; i--)  for(j = 200; j > 0; j--); |

Nội dung code : chương trình này sẽ kích hoạt loa (tạo ra tiếng beep) trong 5 miligiây, sau đó tắt loa trong 5 miligiây và lặp lại quy trình này.

## **3.2 Happy (Buzz.asm)**

|  |
| --- |
| ORG 8000H  AJMP MAIN  ORG 8100H  MAIN:  MOV SP,#60  // vòng lặp gọi hàm MUSIC  LOOP: ACALL MUSIC  AJMP LOOP  // nhảy đến MUSIC và sau đó quay lại LOOP bằng AJMP LOOP.  MUSIC:  MOV R5,#60  MIC:  CPL P1.3 // đảo trạng thái của chân P1,3  ACALL DELAY5MS // gọi hàm DELAY5MS  DJNZ R5,MIC // Giảm R5 và nhảy đến MIC nếu R5 khác không. Vòng lặp này tạo ra một đợi dựa trên giá trị trong R5.  RET  // tạo hàm delay 5ms  DELAY5MS:  MOV R7,#3H  DELAY0: MOV R6,#40H  DELAY1: DJNZ R6,DELAY1  DJNZ R7,DELAY0  RET  ;END |

Nội dung Code : Chức năng chính của chương trình này là tạo ra một chu kỳ tiếng "beep" thông qua chân P1.3 bằng cách đảo ngược trạng thái logic của nó và tạo ra các khoảng trễ. Hàm MUSIC tạo ra tiếng "beep" và hàm DELAY5MS tạo ra một khoảng thời gian trễ 5 miligiây. Sau đó, chương trình lặp lại quá trình này vô hạn.

## **3.3 sound ( beep.c)**

|  |
| --- |
| #include <REG52.H>  #include <INTRINS.H>  sbit Beep = P2^5 ;    unsigned char n=0;  // dữ liệu bản nhạc  unsigned char code music\_tab[] ={  0x18, 0x30, 0x1C , 0x10,  0x20, 0x40, 0x1C , 0x10,  0x18, 0x10, 0x20 , 0x10,  0x1C, 0x10, 0x18 , 0x40,  0x1C, 0x20, 0x20 , 0x20,  0x1C, 0x20, 0x18 , 0x20,  0x20, 0x80, 0xFF , 0x20,  0x30, 0x1C, 0x10 , 0x18,  0x20, 0x15, 0x20 , 0x1C,  0x20, 0x20, 0x20 , 0x26,  0x40, 0x20, 0x20 , 0x2B,  0x20, 0x26, 0x20 , 0x20,  0x20, 0x30, 0x80 , 0xFF,  0x20, 0x20, 0x1C , 0x10,  0x18, 0x10, 0x20 , 0x20,  0x26, 0x20, 0x2B , 0x20,  0x30, 0x20, 0x2B , 0x40,  0x20, 0x20, 0x1C , 0x10,  0x18, 0x10, 0x20 , 0x20,  0x26, 0x20, 0x2B , 0x20,  0x30, 0x20, 0x2B , 0x40,  0x20, 0x30, 0x1C , 0x10,  0x18, 0x20, 0x15 , 0x20,  0x1C, 0x20, 0x20 , 0x20,  0x26, 0x40, 0x20 , 0x20,  0x2B, 0x20, 0x26 , 0x20,  0x20, 0x20, 0x30 , 0x80,  0x20, 0x30, 0x1C , 0x10,  0x20, 0x10, 0x1C , 0x10,  0x20, 0x20, 0x26 , 0x20,  0x2B, 0x20, 0x30 , 0x20,  0x2B, 0x40, 0x20 , 0x15,  0x1F, 0x05, 0x20 , 0x10,  0x1C, 0x10, 0x20 , 0x20,  0x26, 0x20, 0x2B , 0x20,  0x30, 0x20, 0x2B , 0x40,  0x20, 0x30, 0x1C , 0x10,  0x18, 0x20, 0x15 , 0x20,  0x1C, 0x20, 0x20 , 0x20,  0x26, 0x40, 0x20 , 0x20,  0x2B, 0x20, 0x26 , 0x20,  0x20, 0x20, 0x30 , 0x30,  0x20, 0x30, 0x1C , 0x10,  0x18, 0x40, 0x1C , 0x20,  0x20, 0x20, 0x26 , 0x40,  0x13, 0x60, 0x18 , 0x20,  0x15, 0x40, 0x13 , 0x40,  0x18, 0x80, 0x00  };    void int0() interrupt 1  { TH0=0xd8;  TL0=0xef;  n--;  // biến n đại diện cho nhịp của nốt nhạc. n sẽ đc giảm 1 (sau mỗi lần ngắt timer ) đến khi n = 0 tức là nốt nhạc đã chơi xong  }  // tạo hàm delay giữa các nốt nhạc  void delay (unsigned char m)  {  //m là giá trị tần số của nốt nhạc lấy từ bảng nhạc.  Biến i được khởi tạo bằng giá trị 3\*m. Đây chính là số lần lặp của vòng lặp while.  Khi i=0 thì kết thúc vòng lặp  unsigned i=3\*m;  //Vòng lặp while sẽ lặp giảm biến i đến 0  while(--i);  }    void delayms(unsigned char a)  {  while(--a);  }    void main()  { unsigned char p,m;  // p là tần số, m là nhịp  unsigned char i=0;  // cấu hình timer  TMOD&=0x0f;  TMOD|=0x01;  TH0=0xd8;TL0=0xef;  IE=0x82;  play:  while(1)  {  // chạy bản nhạc  a: p=music\_tab[i];  // nếu bài nhạc kết thúc thì đợi 1s và chơi lại  if(p==0x00) { i=0, delayms(1000); goto play;}  // nếu p== 0xff tức là tăng i để chuyển sang nốt khác ( chuyển bài ) , và delay trước khi lặp lại  else if(p==0xff) { i=i+1;delayms(100),TR0=0; goto a;}  // ngược p mà k phải là 2 trường hợp trên thì bản nhạc sẽ được chạy với m là thời gian, n là tần số  else {m=music\_tab[i++], n=music\_tab[i++];}  TR0=1; // timer đc bật để đếm n  // nếu n khác 0 thì tạo xung beep  while(n!=0) Beep=~Beep,delay(m);  // n = 0 thì kết thúc timer  TR0=0;  }  } |

Nội dung Code: Đây là đoạn code phát nhạc có sử dụng timer, có các chức năng như chơi nhạc liên tục, chuyển nhạc. Tuy nhiên ta cần phải tìm hiểu sơ qua cách tạo ra mã code nhạc đó.

Tìm hiểu sơ qua nốt :

Một nốt nhạc hoàn chỉnh sẽ được cấu tạo từ cao độ ( đồ, rê, mi, pha, son...) và trường độ hay còn gọi là thời gian hoặc nhịp của nốt ( 1, ½, ¼ ,...)

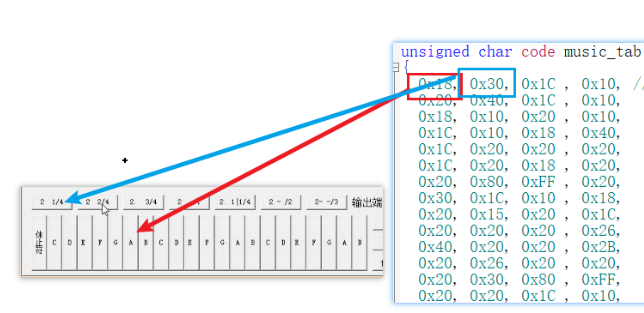


Bảng thể hiện nhịp của nốt



Bảng thể hiện Frequency (Hz) + Timer Count

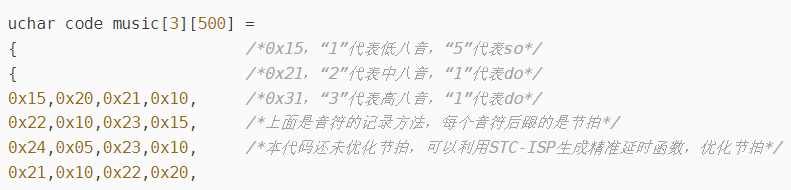
Có nhiều cách có thể tạo ra giai điệu cho bài hát ví dụ như code trên ta thấy :



Một nốt sẽ là 1 cặp mã hex đại diện cho cao độ và trường độ :Như ta có quan sát thì 0x18 sẽ đại diện cho nốt La , 0x30 đại diện cho ¼ tức là nhịp của nốt La này

Tương tự cho các nốt khác. Từ đó sẽ tạo được một bản nhạc nếu bạn có kiến thức về nhạc lý âm nhạc, tuy nhiên về việc tạo được mã này vẫn chưa thể tìm ra câu trả lời hợp lí nhất. Nhưng theo tìm hiểu thì có cách khác để tạo ra được nốt mong muốn.

Cách 1: Sử dụng trực tiếp các kí hiệu số của nốt nhạc



Giải thích vài mã hex : số hàng chục là sẽ tần số của nốt, số hàng chục là số thứ tự của nốt

Ví dụ :

0x15: 5 ( đơn vị) là nốt Son, 1( hàng chục) tần số 392( Hz) 🡪 nốt Son 392 (Hz)

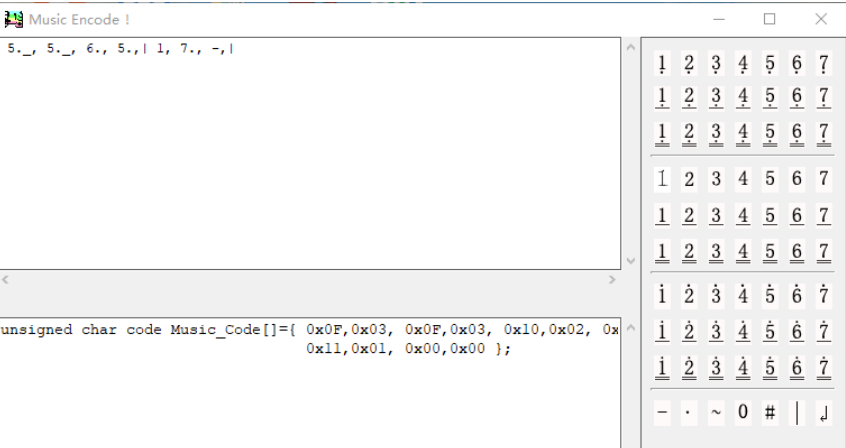
0x21: 1 (đơn vị) là nốt Đô, 2 (hàng chục) tần số 277(Hz) 🡪 nốt Đô 277 (Hz)

0x31: 1 (đơn vị) là nốt Đô, 3 (hàng chục) tần số 523(Hz) 🡪 nốt Đô 523 (Hz)

Tuy cách này có thể dễ hiểu hơn về nốt nhưng mà bị hạn chế về nhịp nên bài hát sẽ không được trơn tru.

Link tham khảo: [link tham khảo kí hiệu nhạc](https://blog.csdn.net/weixin_52470573/article/details/120071853?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522169984234916800188544179%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=169984234916800188544179&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-6-120071853-null-null.142%5ev96%5econtrol&utm_term=%E5%A6%82%E4%BD%95%E7%94%9F%E6%88%90%E8%9C%82%E9%B8%A3%E5%99%A8%E5%92%8C8051%E7%9A%84%E9%9F%B3%E4%B9%90%E4%BB%A3%E7%A0%81&spm=1018.2226.3001.4187)

Cách 2: Sử dụng ứng dụng Music Encode chuyển bản nhạc sang mã hex



Khi hiểu được nhạc lý ta có thể dùng ứng dụng này một cách chính xác !

Tuy nhiên ứng dụng này là nội địa bên Trung nên rất khó để có thể mua và tải về sử dụng.

Link tham khảo: [Link ứng dụng music encode](https://blog.csdn.net/Jason___xiao/article/details/125760751?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522169998818516800182742457%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=169998818516800182742457&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-6-125760751-null-null.142%5ev96%5econtrol&utm_term=music%20encode&spm=1018.2226.3001.4187)

Cách 3: Dùng timer để kiểm soát tần số âm thanh đầu ra cho buzzer

Ta cần phải chuyển tần số của nốt sang timer count nhằm tính toán THx, Tlx nhằm sử dụng timer hiệu quả

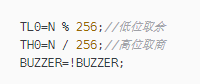
Ví dụ nốt Do có tần số 262 hz. Áp dụng công thức 2^16 -5x(10^5)/ tần số

Ta sẽ được timer count = 63628

THx = timer count/256 = 249

TLx = timer coun%256 = 140

Tiếp theo sử dụng interrupt 1 để xử lí ngắt cho timer



cài đặt bộ định thời (TLO và THO) dựa trên tần số đó, và chuyển đổi trạng thái của Buzzer để tạo ra âm thanh tương ứng.

Từ đó ta có thể tạo ra các nốt do re mi pha son cực kì dễ dàng ! Để có thể chơi hoàn chỉnh bản nhạc ta cần tạo thêm khoảng delay giữa các nút để các nốt có độ dài khác nhau khi phát ra âm thanh

Link tham khảo: [Buzzer hoạt động với Timer](https://blog.csdn.net/m0_73803759/article/details/132063659?ops_request_misc=&request_id=&biz_id=102&utm_term=%E5%8D%95%E7%89%87%E6%9C%BA%E5%AD%A6%E4%B9%A0-%E8%9C%82%E9%B8%A3%E5%99%A8%E6%92%AD%E6%94%BE%E9%9F%B3%E4%B9%90&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-9-132063659.nonecase&spm=1018.2226.3001.4187)

## **3.4 Happy ( HAPPY.ASM)**

|  |
| --- |
| ORG 0000H  JMP MAIN  ORG 000BH  JMP INTT0  ORG 0100H  // thuc hien cau hinh thanh ghi va co de thiet lap bo dem timer  MAIN: MOV SP,#60H  MOV TMOD,#01H  SETB ET0  SETB EA  SETB TR0  START0: ;clr P1.5  MOV 30H,#00H  // Thuc hien viec truy xuat gia tri tu bang TABLE ,dieu khien timer 0  // dua tren cac gia tri lay tu bang  NEXT: MOV A,30H  MOV DPTR,#TABLE  MOVC A,@A+DPTR  MOV R2,A  JZ ENDD  ANL A,#0FH  MOV R5,A  MOV A,R2  SWAP A  ANL A,#0FH  JNZ SING  CLR TR0  JMP D1  // Thuc hien cac phep toan va cau hinh timer de dieu khien chu ky va tan so  // cua mot timer trong vi dieu khien  SING: DEC A  MOV 22H,A  RL A  MOV DPTR,#TABLE1  MOVC A,@A+DPTR  MOV TH0,A  MOV 21H,A  MOV A,22H  RL A  INC A  MOVC A,@A+DPTR  MOV TL0,A  MOV 20H,A  SETB TR0  D1: CALL DELAY  INC 30H  JMP NEXT  ENDD: CLR TR0  JMP START0  // cau hinh timer 0 va dao chan P2.5  INTT0:  PUSH PSW  PUSH ACC  MOV TL0,20H  MOV TH0,21H  CPL P2.5  POP ACC  POP PSW  RETI  // tao ham delay  DELAY: MOV R7,#02  DELAY0: MOV R4,#187  DELAY1: MOV R3,#248  DJNZ R3,$  DJNZ R4,DELAY1  DJNZ R7,DELAY0  DJNZ R5,DELAY  RET  // doan ma bang nhac HAPPY  TABLE: DB 82H,01H,81H,94H,84H,0B4H,0A4H,04H  DB 82H,01H,81H,94H,84H,0C4H,0B4H,04H  DB 82H,01H,81H,0F4H,0D4H,0B4H,0A4H,94H  DB 0E2H,01H,0E1H,0D4H,0B4H,0C4H,0B4H,04H  DB 82H,01H,81H,94H,84H,0B4H,0A4H,04H  DB 82H,01H,81H,94H,84H,0C4H,0B4H,04H  DB 82H,01H,81H,0F4H,0D4H,0B4H,0A4H,94H  DB 0E2H,01H,0E1H,0D4H,0B4H,0C4H,0B4H,04H  TABLE1: DW 64260,64400,64524,64580,64684,64777,64820,64898  DW 64968,65030,65058,65110,65157,65178,65217  END |

Nội dung chương trình : Chương trình này được viết để tạo ra một chuỗi âm thanh dựa trên dữ liệu được lưu trữ trong bảng TABLE và TABLE1. Nó sử dụng Timer 0 của vi điều khiển để kiểm soát thời gian phát ra từng âm thanh và có thể điều chỉnh tần số âm thanh thông qua việc thay đổi giá trị trong các bảng dữ liệu này. Ngoài ra, chương trình còn tích hợp xử lý ngắt, cho phép nó phản ứng với các sự kiện bên ngoài hoặc thay đổi trạng thái trong quá trình hoạt động.

## **3.5 X ( znpa.asm)**

|  |
| --- |
| ORG 0000H  LJMP START  ORG 000BH  INC 20H  MOV TH0,#0D8H  MOV TL0,#0EFH  RETI  // cau hinh timer0  START:  MOV SP,#50H  MOV TH0,#0D8H  MOV TL0,#0EFH  MOV TMOD,#01H  MOV IE,#82H  // khoi tao cac thanh ghi va bien truoc khi thuc hien cac hoat dong  // khac trong MUSIC0  MUSIC0:  NOP  MOV DPTR,#DAT  MOV 20H,#00H  MOV B,#00H  // kiem tra va dieu khien luon chuong trinh trong dua tren gia tri trong thanh ghi A  MUSIC1:  NOP  CLR A  MOVC A,@A+DPTR  JZ END0  CJNE A,#0FFH,MUSIC5  LJMP MUSIC3  // di chuyen gia tri giua cac thanh ghi  // tang DPTR  // lay du lieu de dieu khien timer0  // de xu ly am thanh  MUSIC5:  NOP  MOV R6,A  INC DPTR  MOV A,B  MOVC A,@A+DPTR  MOV R7,A  SETB TR0  MUSIC2:  NOP  CPL P2.5  MOV A,R6  MOV R3,A  LCALL DEL  MOV A,R7  CJNE A,20H,MUSIC2    MOV 20H,#00H  INC DPTR  ; INC B  LJMP MUSIC1  MUSIC3:  NOP  CLR TR0  MOV R2,#0DH  MUSIC4:  NOP  MOV R3,#0FFH  LCALL DEL  DJNZ R2,MUSIC4  INC DPTR  LJMP MUSIC1  END0:  NOP  MOV R2,#10H  MUSIC6:  MOV R3,#00H  LCALL DEL  DJNZ R2,MUSIC6  LJMP MUSIC0  // tao ham delay  DEL:  NOP  DEL3:  MOV R4,#02H  DEL4:  NOP  DJNZ R4,DEL4  NOP  DJNZ R3,DEL3  RET  NOP  // ma hex cua ban nhac  DAT:  db 26h,20h,20h,20h,20h,20h,26h,10h,20h,10h,20h,80h,26h,20h,30h,20h  db 30h,20h,39h,10h,30h,10h,30h,80h,26h,20h,20h,20h,20h,20h,1ch,20h  db 20h,80h,2bh,20h,26h,20h,20h,20h,2bh,10h,26h,10h,2bh,80h,26h,20h  db 30h,20h,30h,20h,39h,10h,26h,10h,26h,60h,40h,10h,39h,10h,26h,20h  db 30h,20h,30h,20h,39h,10h,26h,10h,26h,80h,26h,20h,2bh,10h,2bh,10h  db 2bh,20h,30h,10h,39h,10h,26h,10h,2bh,10h,2bh,20h,2bh,40h,40h,20h  db 20h,10h,20h,10h,2bh,10h,26h,30h,30h,80h,18h,20h,18h,20h,26h,20h  db 20h,20h,20h,40h,26h,20h,2bh,20h,30h,20h,30h,20h,1ch,20h,20h,20h  db 20h,80h,1ch,20h,1ch,20h,1ch,20h,30h,20h,30h,60h,39h,10h,30h,10h  db 20h,20h,2bh,10h,26h,10h,2bh,10h,26h,10h,26h,10h,2bh,10h,2bh,80h  db 18h,20h,18h,20h,26h,20h,20h,20h,20h,60h,26h,10h,2bh,20h,30h,20h  db 30h,20h,1ch,20h,20h,20h,20h,80h,26h,20h,30h,10h,30h,10h,30h,20h  db 39h,20h,26h,10h,2bh,10h,2bh,20h,2bh,40h,40h,10h,40h,10h,20h,10h  db 20h,10h,2bh,10h,26h,30h,30h,80h,00H  END |

Nội dung chương trình : điều khiển module âm thanh thông qua timer 0