El MP3

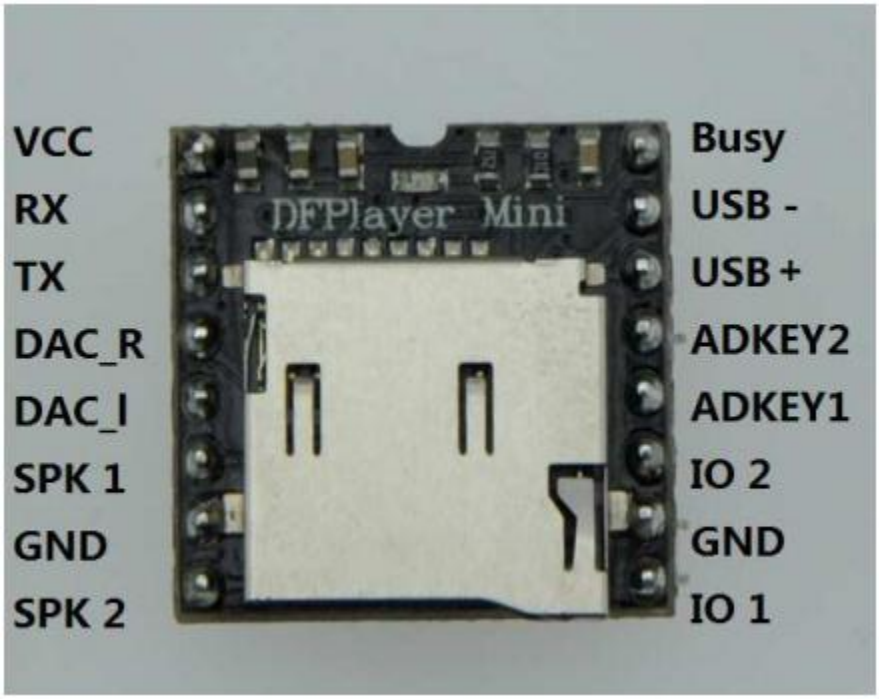
La tecnología Mp3 fue desarrollada por una empresa alemana Fraunhofer-Gesellshaft, la cual ahora tiene permiso a los derechos de patente a la tecnología de compresión de audio. Hay dos nombres que se asocian con el desarrollo del MP3. El instituto Fraunhofer contribuyo con el codificador de audio con la ayuda de Dieter Seitzer, un profesor de la Universidad de Erlangen. Dieter Seitzer trabajo en la transferencia de música a través de una línea telefónica estándar de calidad. Esta investigación fue dirigida por Karl Heinz Branden Burg, también conocido como el “padre del MP3”. Karl Heinz era un especialista en las matemáticas y en la electrónica y participó en la investigación de métodos para comprimir la música desde 1977.

Instrucción de aplicación de módulo

2.1. Descripción de la especificación

Articulo	Descripción
Formato MP3	1, admite decodificación de audio de capa 3 11172-3 e ISO13813-3
	2 rate Frecuencia de muestreo de soporte (KHZ): 8 / 11.025 / 12/16 / 22.05 / 24/32 / 44.1 / 48
	3, Soporte Normal, Jazz, Clásico, Pop, Rock, etc.
Puerto UART	serie estándar; Nivel TTL; Velocidad de transmisión ajustable (la velocidad de transmisión predeterminada es 9600)
Voltaje de trabajo	DC3.2~5.0V; Tipo :DC4.2V
Corriente en espera	20mA
Operando Temperatura	-40~+70
Humedad	5% ~95%

Descripción Del Pin



No.	Pin	Descripción	Nota
1	VCC	Voltaje de entrada	DC3.2 ~ 5.0V; Tipo: DC4.2V
2	RX	Entrada serial UART	
3	TX	Salida serial UART	
4	DAC_R	Salida de audio canal derecho	Conducir auriculares y amplificador
5	DAC_L	Salida de audio canal izquierdo	Conducir auriculares y amplificado
6	SPK2	Altavoz	Unidad de altavoz de menos de 3W

7	GND	Suelo	GND de potencia
8	SPKI	AltavoZ	Unidad de altavoz de menos de 3W
9	IOI	Activar puerto 1	Pulsación corta para reproducir la anterior（pulsación larga para bajar el volumen）
10	GND	Suelo	GND de potencia
11	IO2	Activar puerto 2	Presione brevemente para reproducir el siguiente（presione prolongadamente para aumenta el volumen)
12	ADKEY1	Puerto AD 1	Trigger play primer segmento
13	ADKEY2	Puerto AD 2	Trigger play quinto segmento
14	USB+	USB + DP	Puerto USB
15	USB-	USB-DM	Puerto USB

16	BUSY	Estado de juego	Bajo significa Reproducir \ Alto significa no

Datos devueltos de error de módulo

El módulo está ocupado.	7E FF 06 40 00 00 00 xx xx EF
No se reciben todos los datos de trama	7E FF 06 40 00 00 01 xx xx EF
Error de verificación	7E FF 06 40 00 00 02 xx xx EF

1) Para fortalecer la estabilidad de la comunicación de datos, agregamos el manejo de errores de datos mecanismo. El módulo responderá la información después de recibir el formato de datos de error;

2) En el caso de un entorno relativamente hostil, se recomienda encarecidamente que los clientes procesen esto mando. Si el entorno de la aplicación en general, no necesita manejarlo;

3). El módulo vuelve a estar ocupado, básicamente cuando vuelve la inicialización de encendido del módulo, porque el

Los módulos necesitan inicializar el sistema de archivos.

Puertos clave

Utilizamos las teclas del módulo AD, en lugar del método tradicional de conexión de teclado matricial, es tomar ventaja de la funcionalidad AD de MCU cada vez más potente, nuestro módulo de configuración predeterminada 2 puertos AD,

Distribución de resistencia de 20 teclas, si se usa en interferencias electromagnéticas fuertes o inductivas fuertes,carga capacitiva de la ocasión, consulte nuestras "Notas".

1) Diagrama de referencia.

2) 20 tabla de asignación de teclas de función

Llave	Empuje corto	Empuje largo	Descripción
K1	Modo de juego		Cambiar a interrumpir / no interrumpido
K2	El dispositivo de reproducción cambia		U / TF / SPI / Sleep
K3	Modo operativo		Todo el ciclo
K4	Reproducir/pasusar		
K5	Anterior	Vol+	
K6	Siguiente	Vol-	
K7	4	Repita el reproducción seguimiento 4	Pulsación larga siempre para repetir la reproducción

K8	3	Repita el reproducción seguimiento 3	Pulsación larga siempre para repetir la reproducción
K9	2	Repita el reproducción seguimiento 2	Pulsación larga siempre para repetir la reproducción
K10	1	Repita el reproducción seguimiento 1	Pulsación larga siempre para repetir la reproducción
K11	5	Repita el reproducción seguimiento 5	Pulsación larga siempre para repetir la reproducción
K12	6	Repita el reproducción seguimiento 6	Pulsación larga siempre para repetir la reproducción
k13	7	Repita el reproducción seguimiento 7	Pulsación larga siempre para repetir la reproducción
K14	8	Repita el reproducción seguimiento 8	Pulsación larga siempre para repetir la reproducción
K15	9	Repita el reproducción seguimiento 9	Pulsación larga siempre para repetir la reproducción
K16	10	Repita el reproducción seguimiento 10	Pulsación larga siempre para repetir la reproducción

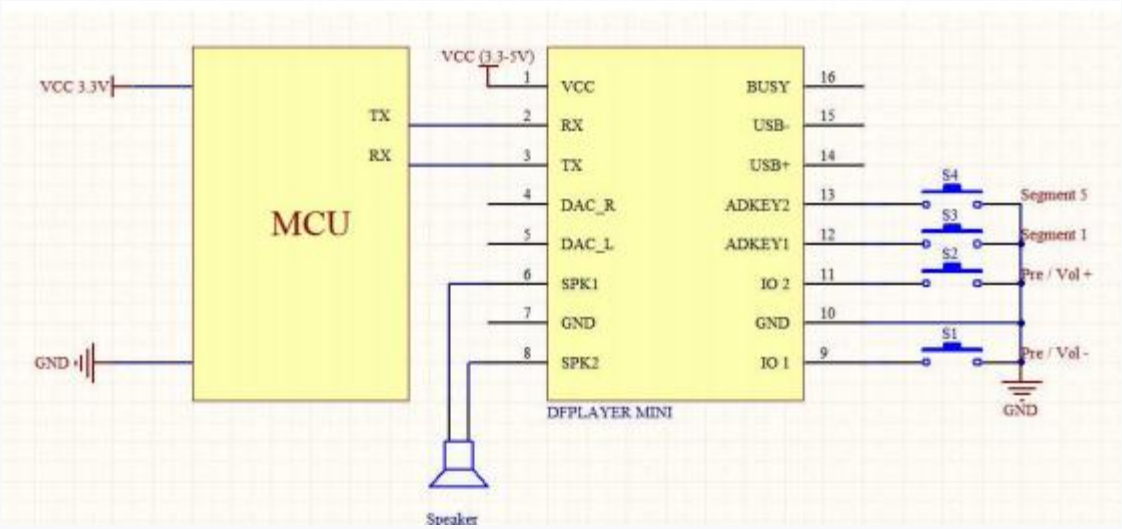
K17	11	Repita el reproducción seguimiento 11	Pulsación larga siempre para repetir la reproducción
K18	12	Repita el reproducción seguimiento 12	Pulsación larga siempre para repetir la reproducción
K19	13	Repita el reproducción seguimiento 13	Pulsación larga siempre para repetir la reproducción
K20	14	Repita el reproducción seguimiento 14	Pulsación larga siempre para repetir la reproducción

Aplicación Del Circuito

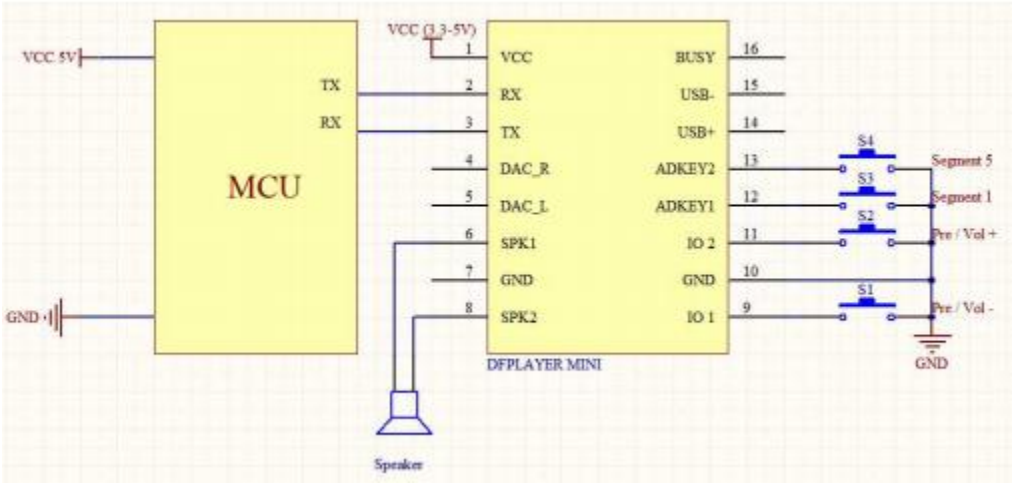
4.1 Conexión de comunicación en serie

El puerto serie del módulo tiene un nivel TTL de 3.3V, por lo que el nivel de interfaz predeterminado es 3.3V. Si el sistema MCU es de 5V. Eso

Se recomienda conectar una resistencia de 1K en serie.

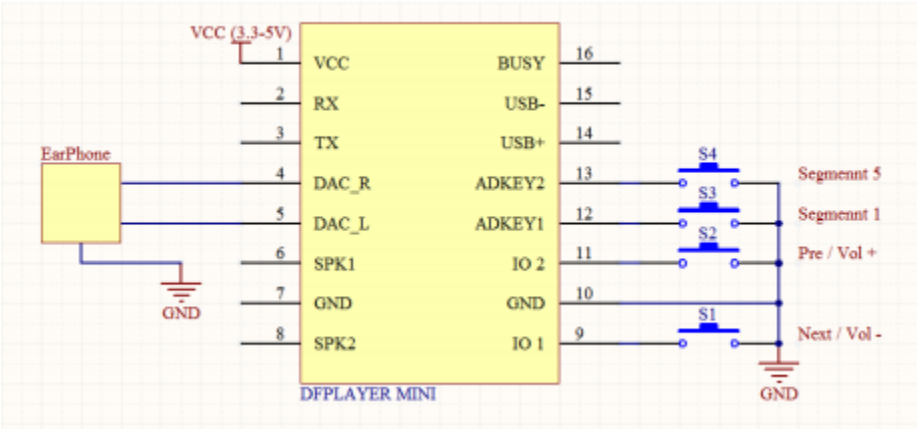


Conexión en serie



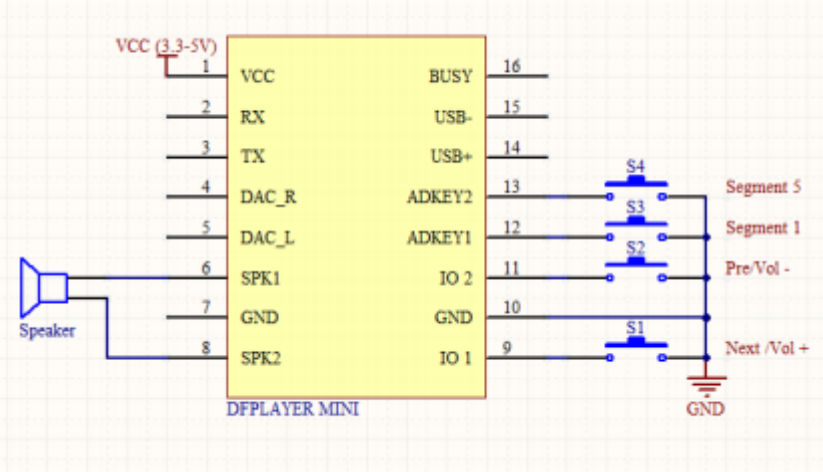
Conexión en serie

Otro diagrama de referencia

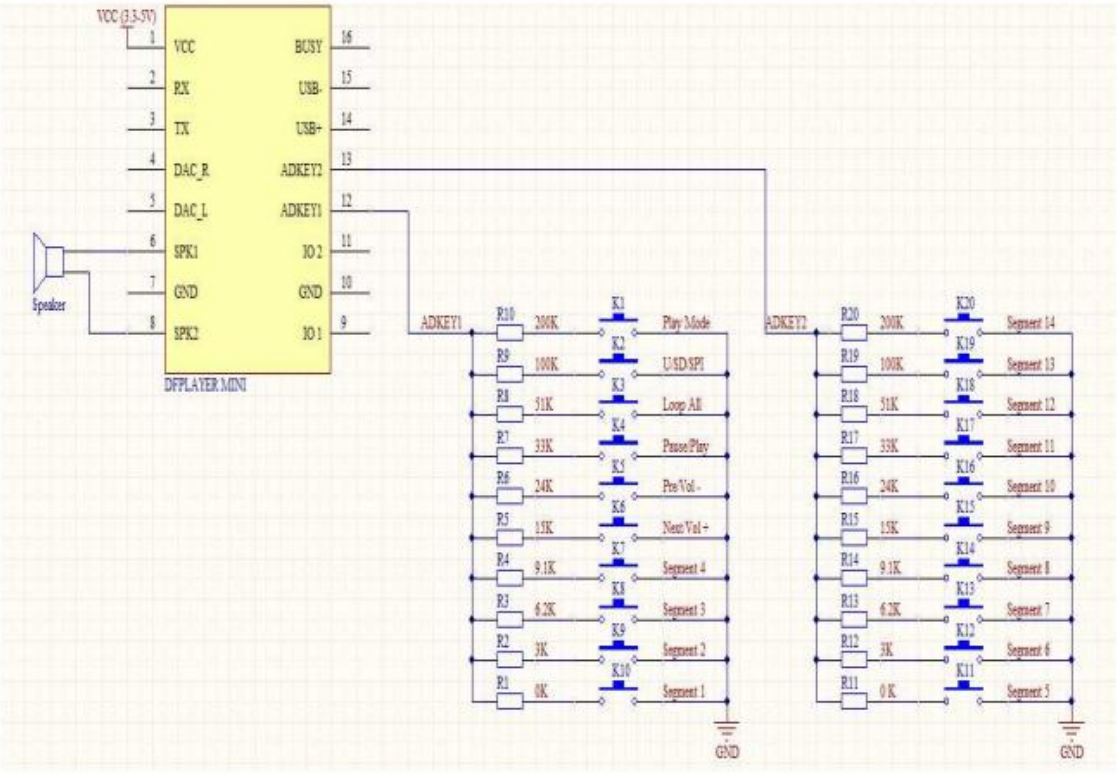


4.3 módulo de conexión de auriculares

Entre los auriculares y el módulo puede conectar una resistencia 100R, hacer una limitación



Módulo de conexión de altavoz



Clave de anuncio conectar referir

Para Programarlo en C

```
#include<stdio.h>

#include<stdlib.h>
```

```
#include<string.h>
#include<conio.h>
#include<ctype.h>
#define MAX 128
#define BIT 30
#pragma warning(disable:4996)

#define CREAR 1
#define AGREGAR 2
#define BORRAR 3
#define MODIFICAR 4
#define GUARDAR 5
#define CARGAR 6
#define EJECUTAR 7
#define SALIR 8


typedef struct
{
	char Titulo[BIT],Artista[BIT],Album[BIT],Coment[29],Genero[1],Anio[4];


}ID3;
void crear(void);
```

```
int borrardato(ID3 mp[MAX], int cant);
int cargardatos(ID3 mp[MAX], int cant);
int guardardatos(ID3 mp[MAX], int cant);
int ingresardatos(ID3 mp[MAX], int cant);
int ingresarCadena (char[],int,char*);
int verificarfecha (ID3 mp[MAX], int cant);
void ejecutardatos(ID3 mp[MAX], int cant);
void modificardatos(ID3 mp[MAX], int cant);
int menu(void);


int main()
{
	ID3 mp[MAX];
	int opc;
	int cant=0;
	printf("	Data de MP3\n");

	while((opc=menu())!=SALIR ){
		switch(opc){
			case CREAR:
				crear();
				break;
			case AGREGAR:
```

```
                cant=ingresardatos(mp,cant);
                break;
            case BORRAR:
                cant = borrardato(mp,cant);
                break;
            case MODIFICAR:
                modificardatos(mp,cant);
                break;
            case GUARDAR:
                //cant=guardardatos(mp,cant);
                printf("guardar");
                break;
            case CARGAR:
                //cant=cargardatos(mp,cant);
                printf("cargar");
                break;
            case EJECUTAR:
                //ejecutardatos(mp,cant);
                printf("ejecutar");
                break;
        }
    }
    return 0;
}
```

```
int menu(void)
{
	int opcion;
	printf("\n");
	printf("1 - Crear Archivo MP3\n");
	printf("2 - Ingresar nueva pista de MP3\n");
	printf("3 - Eliminar pista\n");
	printf("4 - Modificar pista deseada\n");
	printf("5 - Guardar Contenido\n");
	printf("6 - Cargar historial\n");
	printf("7 - Ejecutar MP3\n");
	printf("8 - Salir\n");
	printf("\n");
	do{
		scanf("%d",&opcion);
	}while(opcion<CREAR || opcion>SALIR)
		;
	fflush(stdin);
	return opcion;
}
```

```
void crear(void){
   FILE *arch;
   arch = fopen("MP3.dat","wb");
   printf("El creado ha sido existoso\n");
   if(arch==NULL)
   {
                printf("Error de apuertura del archivo MP3\n");
       fclose(arch);
   }
}

int ingresardatos(ID3 mp[MAX], int cant){
          FILE *arch;
          int auxCant = 0, cont=0,auxAnio=0
          arch=fopen("MP3.dat","wb");
          if(arch==NULL){
                puts("No se peude crear el archivo");
          }else{
                printf("Ingrese el Titulo de la pista\n");
                scanf("%s",mp[cant].Titulo);
                fflush(stdin);
```

```
printf("Ingrese el nombre del Artista o Grupo:\n");
scanf("%s",mp[cant].Artista);
fflush(stdin);

printf("Ingrese el Album al que pertenece la pista anteriormente ingresada:\n");
scanf("%s",mp[cant].Album);
fflush(stdin);

printf("Ingrese el Anio de dicha pista:\n");
scanf("%c",mp[cant].Anio);
fflush(stdin);

printf("Ingrese un comentario: \n");
scanf("%s",mp[cant].Coment);
fflush(stdin);


printf("ingrese el genero blab labla \n");
scanf("%s",mp[cant].Genero);
fflush(stdin);
fwrite (&mp,sizeof(ID3),1,arch);
fclose(arch);
```

```
            }
            cant++;
            return cant;
}



/*void borrardato(void){
   FILE *arch,*archAux;
   ID3 mp;
   int encontrado=0;
            char Aux[BIT];

   printf("Introduce el nombre de la pista que deseas borrar: ");
   gets(Aux);
   printf(" BORRANDO REGISTROS\n");
   arch = fopen("MP3.dat","rb");
   if(arch== NULL)
            printf("Error de apuertura del archivo MP3\n");
   else{
                        archAux = fopen("MP3aux.dat","wb");
      if(archAux==NULL)
         printf("Error Imposible crear archivo auxiliar\n");
      else{
         while ((fread(&mp, sizeof(ID3), 1, arch))!=0)
```

```
                                if(stricmp(mp.Titulo,Aux)!=0)
                    fwrite(&mp, sizeof(ID3), 1, archAux);
                  else
                    encontrado=1;
                fclose(archAux);
            }
            fclose(arch);
            if(encontrado){
                remove("MP3.dat");
                rename("MP3aux.dat","MP3.dat");
                printf("Registro BORRADO con Exito!!!\n");
            }
            else{
                remove("MP3aux.dat");
                printf(" El Registro NO SE ENCONTRO, no fue borrado!!!\n");
            }
        }
}
 void borrardato(ID3 mp[MAX], int cant){
            FILE*arch;
            arch = fopen ("MP3.dat","rb");
            char tituloaux[BIT] ;
            int posicion=0,j,bandera=0;
```

```
printf("Ingrese el titulo de la pista que desea eliminar: ");
printf("\n");
fgets(tituloaux,BIT,stdin);
fflush(stdin);
fread(&mp,sizeof(ID3),1,arch);
while (!feof(arch)){
	for(j=0;j<cant+1;j++){
		if(strcmp(mp[j].Titulo,tituloaux)==0){
				posicion=j;
				bandera=1;
			}
		}
	j=0;
	if(bandera!=0){
		if(posicion!=cant-1){
			while(j<cant-1){
				mp[posicion]=mp[posicion+1];
				j++;
			}
			cant--;
			printf("El borrado ha finalizado con exito\n");
		}else{
			cant--;
			printf("El borrado ha finalizado con exito\n");
```

```
                        }
                }else{
                        printf("El titulo de la pista no esta cargada en la base de datos \n");
                }
                fread(&mp,sizeof(ID3),1,arch);
        }
        fclose(arch);
}
*/



int borrardato(ID3 mp[MAX],int cant){
        FILE *arch;
        int j;
        int k=0;
        int posicion=0;
        int bandera=0;
        char titulo[BIT];
        arch = fopen("MP3.dat","wb+");
        printf("La apertura ha sido existosa\n\n\n");
        if(arch==NULL)
```

```
{
                printf("Error de apuertura del archivo MP3\n");
        fclose(arch);
 }
printf("Ingrese la pista que desea eliminar:\n ");
gets(titulo);
fflush(stdin);
fread(&mp,sizeof(mp),1,arch);
//while (!feof(arch)){

        for(j=0;j<cant;j++){
                if(strcmpi(mp[j].Titulo,titulo)==0){
                                posicion=j;
                                bandera=1;
                        }
        }
        k=posicion+1;
        if(bandera!=0){
                if(posicion!=cant-1){
                        while(posicion<cant-1 && k!=cant){
                                mp[posicion]=mp[k];
                                posicion++;
                                k++;
                        }
```

```
                cant++;
                printf("El borrado ha finalizado con exito\n");
            }else{
                cant--;
                printf("El borrado ha finalizado con exito\n");
            }
            cant--;
        }else{
            printf("la pista ingresada no se encuentra en la base de datos\n");
        }
    //}
    //printf(" la cagamos");
    //system("pause");
    fwrite(&mp,sizeof(mp),1,arch);
    fclose (arch);
    return cant;
}



void modificardatos(ID3 mp[MAX], int cant){
    FILE*arch;
    char aux[BIT];
    int i,contador=0,opcAux=0;
    arch=fopen("MP3.dat","wb+");
```

```
        printf("Ingrese el nombre de la pista que desea modificar: ");
        fgets(aux,BIT,stdin);
        fflush(stdin);
        fread(&mp,sizeof(ID3),1,arch);
        while(!feof(arch)){
                for(i=0;i<cant;i++){

                                fseek(arch,0,SEEK_SET);
                                fwrite(&mp,sizeof(ID3),1,arch);
                                printf("Ingrese un nuevo titulo de lo contrario escriba el
mismo:\n");
                                scanf("%s",mp[i].Titulo);
                                fflush(stdin);
                                printf("Ingrese nuevo nombre de  artista o grupo:\n");
                                scanf("%s",mp[i].Artista);
                                fflush(stdin);
                                printf("Ingrese el Anio de dicha pista:\n");
                                scanf("%s",mp[i].Anio);
                                fflush(stdin);
                                printf("Ingrese un nuevo comentario: \n");
                                scanf("%s",mp[i].Coment);
                                fflush(stdin);
                        }
        }
```

```
        fclose(arch);
}
```