



Aguijón

Notas de aplicación

Nota de aplicación 35:

Digital Recorder

Descripción:

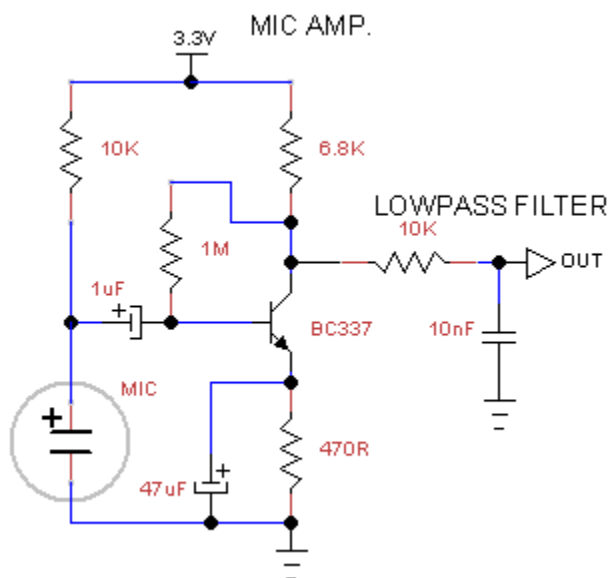
Generar y guardar en una memoria USB un archivo de audio con formato '.WAV', Grabado desde un micrófono conectado a un puerto de expansión.

Herramientas:

1. Aguijón 3.0, Aguijón 4.0 ó Aguijón 4.1
2. MPLAB X®
3. Aguijón HID bootloader
4. Cable USB 'A' to 'A'
5. Librerías HammerHead.
6. Memoria usb 2.0
7. Micrófono Electret
8. Transistor NPN MPSA06
9. Resistencias (varias)
10. Capacitores (varios)

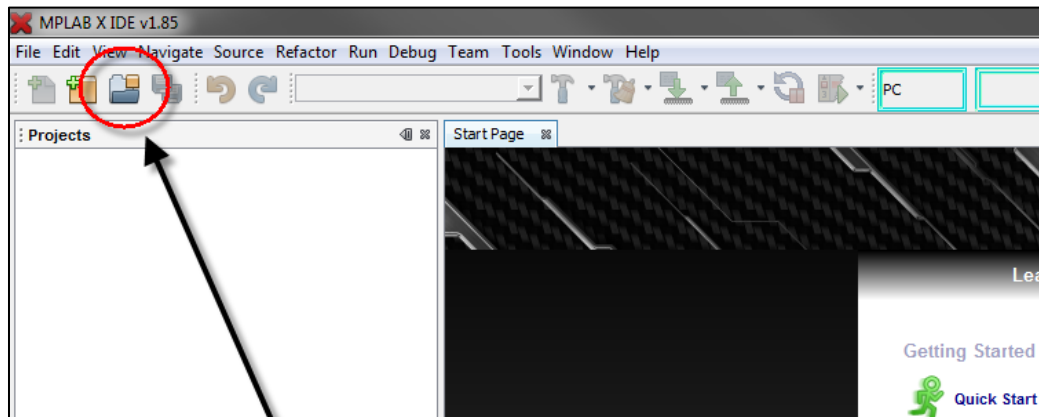
Pasos:

1. Diagrama de conexión



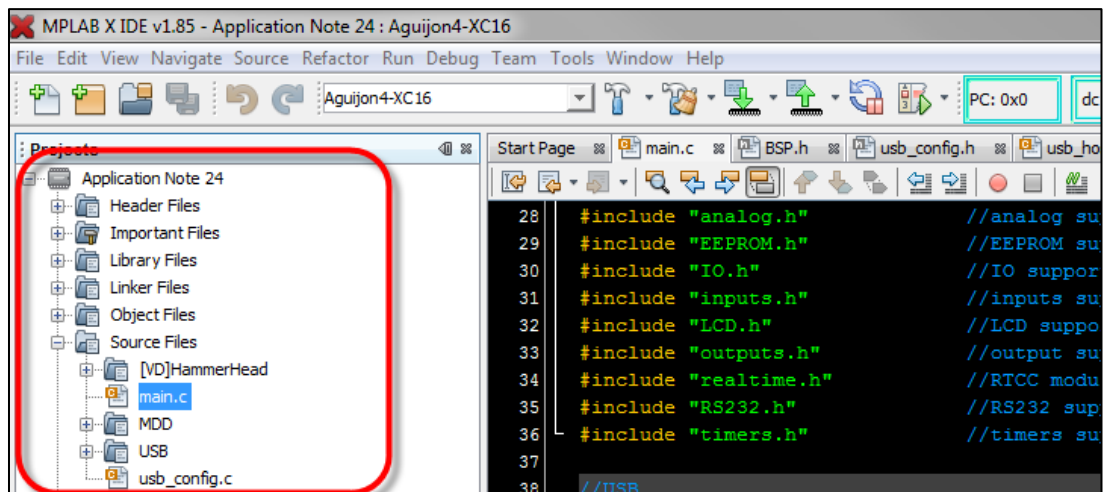
	Signal In	+3.3V	GND
Aguijon 4.0	Puerto de Expansión 2, Pin 10	Puerto de Expansión 2, Pin 14	Puerto de Expansión 2, Pin 1
Aguijon 4.1	Puerto de Expansión 2, Pin 6	Puerto de Expansión 2, Pin 13	Puerto de Expansión 2, Pin 14

2. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 35.X**



Haz 'clic' aquí y
abre el proyecto

3. Abrir el archivo **main.c**



4. Ir a la línea #155.

Utilizaremos la siguiente función:

```
148
149     LCD_IntroAnimation();
150     LCD_PutStr(1,0,"Aguijon USB MSD",TRUE);
151     LCD_PutStr(2,0,"Insert flash drive",FALSE);
152
153     for(;;){
154         //USB stack process function
155         USBTasks();
156
157         //if thumbdrive is plugged in
158         if(USBHostMSDSCSIMediaDetect()){
159             InsertChime();
160             //now a device is attached
161
162             //See if the device is attached and in the right format
```

USBTasks ();

- Esta función es la herramienta necesaria para configurar e inicializar nuestro dispositivo USB, esta debe ser llamada antes que cualquier otra función relacionada con el puerto USB

5. Ir a la línea #158

Utilizaremos la siguiente función:

```
151 LCD_PutStr(2,0,"Insert flash drive",FALSE);
152
153 for(;;){
154     //USB stack process function
155     USBTasks();
156
157     //if thumbdrive is plugged in
158     if(USBHostMSDSCSIMediaDetect()){
159         InsertChime();
160         //now a device is attached
161
162         //See if the device is attached and in the right format
163         if(FSInit()){
164
165             /*Create a string for the LCD*/
```

USBHostMSDSCSIMediaDetect ();

- Función que determina si un dispositivo de almacenamiento masivo está conectado y listo para utilizar; donde:
Devuelve un valor booleano dependiendo del estado del dispositivo
(TRUE= MSD presente y listo, FALSE = MSD no está presente o no está listo).

6. Ir a la línea #163.

Utilizaremos la siguiente función:

```
156
157     //if thumbdrive is plugged in
158     if(USBHostMSDSCSIMediaDetect()){
159         InsertChime();
160     //now a device is attached
161
162     //See if the device is attached and in the right format
163     if(FSInit()){
164
165         /*Create a string for the LCD*/
166         LCD_PutStr(1,0,"Digital Recorder ",FALSE);
167         LCD_PutStr(2,0,"REC ",FALSE);
168
169         while(SW_Read()!=1);
170         /*Create a string for the LCD*/
```

FSInit ();

- Esta función inicializa el dispositivo físico que debe ser conectado al microcontrolador ;
donde:
Devuelve un valor booleano dependiendo del estado de la inicialización
(TRUE= Inicialización exitosa, FALSE = Inicialización no exitosa).

7. Ir a la línea #166

Utilizaremos la siguiente función:

```
159     InsertChime();
160     //now a device is attached
161
162     //See if the device is attached and in the right format
163     if(FSInit()){
164
165         /*Create a string for the LCD*/
166         LCD_PutStr(1,0,"Digital Recorder ",FALSE);
167         LCD_PutStr(2,0,"REC ",FALSE);
168
169         while(SW_Read()!=1);
170         /*Create a string for the LCD*/
171         LCD_PutStr(1,0,"Recording.. ",FALSE);
172         LCD_PutStr(2,0,"STOP ",FALSE);
173     }
```

LCD_PutStr (int y, int x, char *msg, BOOL clear);

- Función que muestra una cadena de caracteres en la pantalla LCD; donde:
Int y = Coordenada en y (Valor entero del 1 al 2.)
Int X = Coordenada en x (Valor entero del 0 al 20.)
Char *msg = Cadena de caracteres (De 0 a 20 caracteres)
BOOL clear = Determina si se borra la pantalla antes de escribir
(TRUE =Borrar, FALSE =No borrar).

8. Ir a la línea #177

Utilizaremos la siguiente función:

```
170      /*Create a string for the LCD*/
171      LCD_PutStr(1,0,"Recording..      ",FALSE);
172      LCD_PutStr(2,0,"STOP ",FALSE);
173
174      //Opening a wav file in mode "w" will create the file if it
175      // exist. If the file does exist it will delete the old f
176      // and create a new one that is blank.
177      WAV_Init("test.wav", S_RATE);
178
179      Timer1_Enable();          //Timer 1 eneable
180      while(SW_Read() !=1)
181      {
182          if(state_a && Write)
183          {
184              WAV_Add(buffer_a,sizeof (buffer_a));          //Add s
```

WAV_Init (char * filename, int s_rate);

- Función que inicializa un archivo de audio tipo WAV; donde:
Char *FileName = El nombre del archivo a crear. (Cadena de caracteres, incluyendo extensión del archivo.)
Int s_rate = Velocidad de muestreo para el audio incluido en el archivo a crear.

1. Ir a la línea #179 y #93

Utilizaremos las siguientes funciones:

```
172 LCD_PutStr(2,0,"STOP ",FALSE);
173
174 //Opening a wav file in mode "w" will create the file if it
175 // exist. If the file does exist it will delete the old fi
176 // and create a new one that is blank.
177 WAV_Init("test.wav", S_RATE);
178
179 Timer1_Enable(); //Timer 1 eneable
180 while(SW_Read() !=1)
181 {
182     if(state_a && Write)
183     {
184         WAV_Add(buffer_a,sizeof (buffer_a)); //Add sa
185         Write=0;
186     }
```

Timer1_Eneable ();

- Función que habilita el módulo Timer 1;

Timer1_Disable ();

- Función que deshabilita el módulo Timer 1;

9. Ir a la línea #180.

Vamos a utilizar la siguiente función:

```
173
174 //Opening a wav file in mode "w" will create the file if it
175 // exist. If the file does exist it will delete the old f
176 // and create a new one that is blank.
177 WAV_Init("test.wav", S_RATE);
178
179 Timer1 Enable(); //Timer 1 enable
180 while(SW_Read() !=1)
181 {
182     if(state_a && Write)
183     {
184         WAV_Add(buffer_a,sizeof (buffer_a)); //Add s
185         Write=0;
186     }
187     if(state_b && Write)
```

SW_Read ();

- Función que lee el puerto de PUSH BUTTONS;
Devuelve un valor equivalente al Push-Button presionado (carácter del 1 al 4)

10. Ir a la línea #184

Utilizaremos la siguiente función:

```
177     WAV_Init("test.wav", S_RATE);
178
179     Timer1_Enable();           //Timer 1 enable
180     while(SW_Read() !=1)
181     {
182         if(state_a && Write)
183         {
184             WAV_Add(buffer_a,sizeof (buffer_a)); //Add s
185             Write=0;
186         }
187         if(state_b && Write)
188         {
189             WAV_Add(buffer_b,sizeof (buffer_b)); //Add s
190             Write=0;
191         }
192     }
```

WAV_Add (int * data, int size);

- Función que agrega muestras al archivo de audio tipo WAV, previamente inicializado; donde:
int *data= Apuntador a la ubicación de los datos a agregar al archivo (Datos de tipo entero sin signo)
Int size = Cantidad de datos a agregar (Entero sin signo).

11. Ir a la línea #195.

Vamos a utilizar la siguiente función:

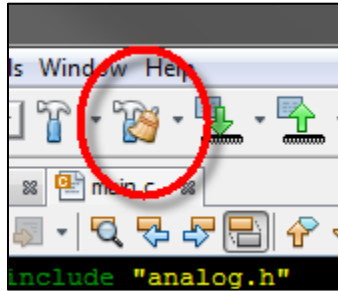
```
188         {
189             WAV_Add(buffer_b,sizeof (buffer_b));           //Add s
190             Write=0;
191         }
192     }
193     Timer1_Disable();           //Timer 1 disable
194
195     WAV_Close();           //Closing a wav file
196
197     /*Create a string for the LCD*/
198     LCD_PutStr(1,0,"DONE!",TRUE);
199     LCD_PutStr(2,0,"Safe to remove USB",FALSE);
200
201     //Just sit here until the device is removed.
202     while(USBHostMSDSCSIMediaDetect()){
```

WAV_Close ();

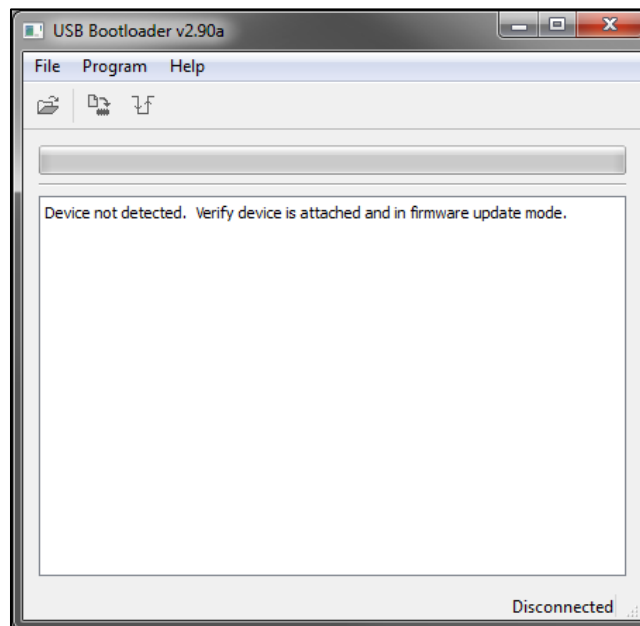
- Función que finaliza el archivo de audio en formato WAV. Este le agrega finalmente la longitud del archivo, con esto se proporciona la posibilidad de agregar cuantos datos sean necesarios;

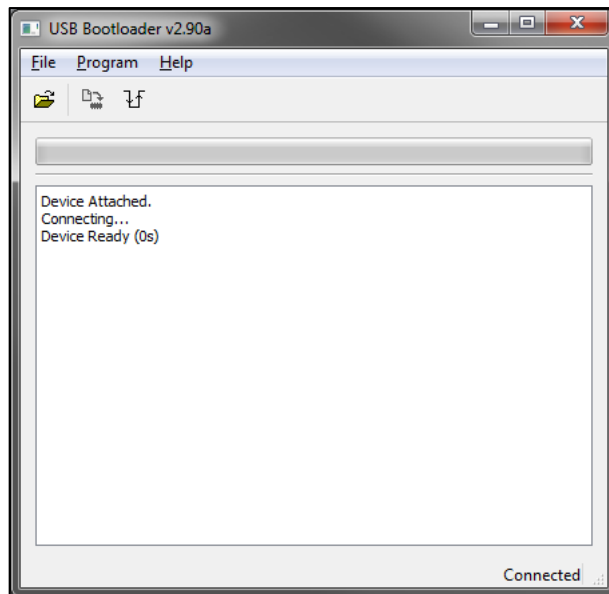
12. Compilar y programar

Al hacer clic en el ícono de compilar, y si no hay errores de compilación, el bootloader será cargado automáticamente.



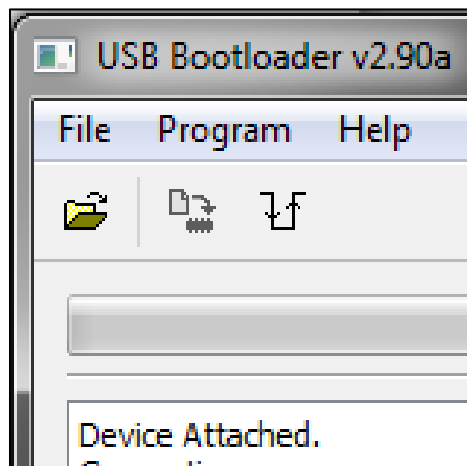
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





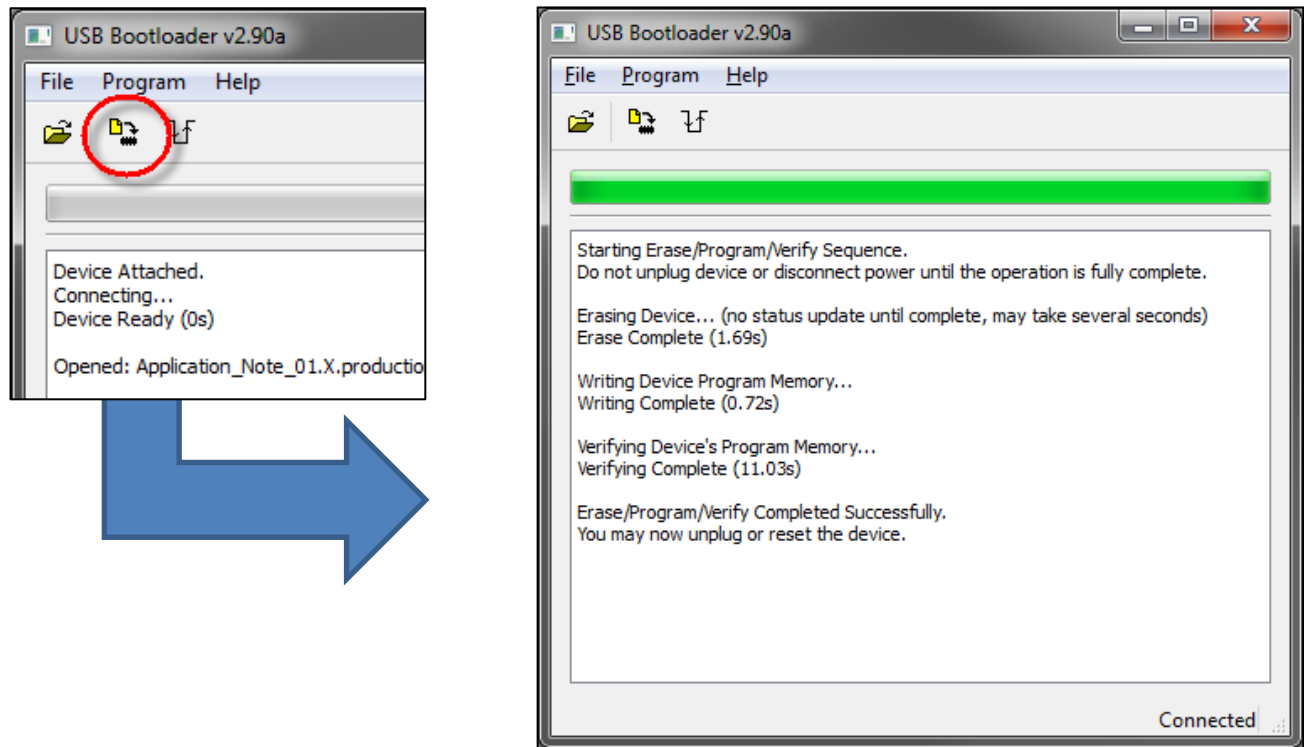
La ventana del Bootloader indicará la conexión establecida con el aguijón:

13. Hacer Clic en Abrir y Cargar el archivo **Application Note 35.X.production.hex**

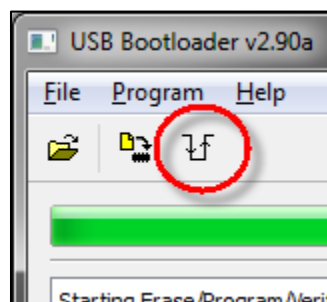


El archivo, depende de la plataforma de hardware.

Una vez cargado el archivo, hacer clic en el ícono de programar.



Hacer clic en el ícono de Reset cuando en la ventana del Bootloader se indique que se terminó de programar con éxito.



Una vez programado podemos verificar el programa corriendo en la tarjeta.

14. Para verificar el funcionamiento del programa verifique que al tener la conexión necesaria y conectar un dispositivo de almacenamiento USB, en este se genere un archivo con el nombre 'TEST.wav', y que este a su vez contenga el sonido generado a la hora de la grabación.