



Aguijón

Notas de aplicación

Nota de aplicación 37:

Serial interface

Descripción:

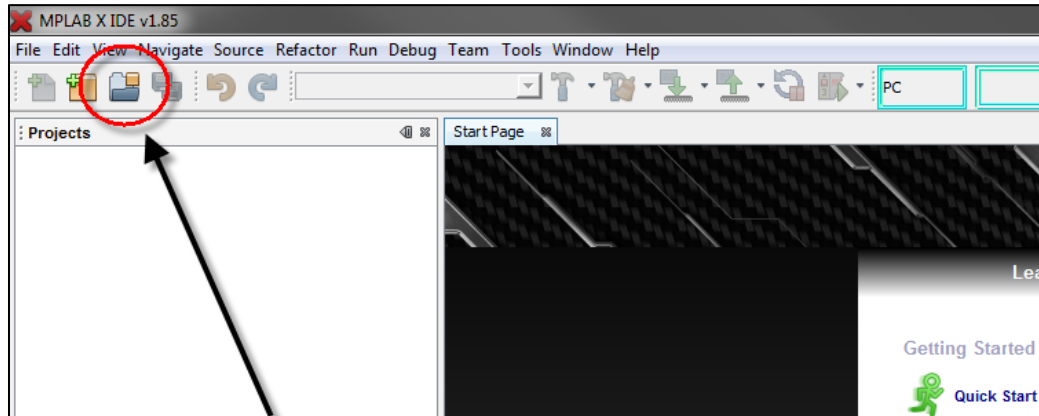
Controlar los principales periféricos de nuestra tarjeta, a través de una conexión serial RS232

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. Cable convertidor USB-SERIAL

Pasos:

1. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 37.X**



Haz 'clic' aquí y
abre el proyecto

2. Abrir el archivo **main.c**



3. Ir a la línea #51

Utilizaremos la siguiente función:

```
44      /*Inputs*/
45      ADC_Init();
46
47      /*LCD*/
48      LCD_Init(LCD_MODE_1);
49
50      /*RS232*/
51      RS232_Init(BAUD19200);
52
53      /*PWM*/
54      OC_PWMChannelConfig(4);
55
56  #if defined(USE_LCD_EXTRA_FEATURES)
57      LCD_BacklightFadeIn();
58  #else
```

RS232_Init (int baud);

- Función que inicializa los registros necesarios del puerto serial RS232
Esta función debe ser llamada antes que cualquier otra relacionada con el puerto serial;
donde:

Int baud = Velocidad de transferencia.

BAUD250K
BAUD115200
BAUD56000
BAUD38400
BAUD19200
BAUD9600

4. Ir a la línea #54

Utilizaremos la siguiente función:

```
47      /*LCD*/
48      LCD_Init(LCD_MODE_1);
49
50      /*RS232*/
51      RS232_Init(BAUD19200);
52
53      /*PWM*/
54      OC_PWMChannelConfig(4);
55
56  #if defined(USE_LCD_EXTRA_FEATURES)
57      LCD_BacklightFadeIn();
58  #else
59      LCD_BacklightSet(BLIGHT_LVL_4);
60  #endif
61
```

OC_PWMChannelConfig (int num);

- Función que configura un PWM a la salida Open Collector seleccionada; donde:
Int num = Número de OPEN COLLECTOR que queremos utilizar (Valor entero del 1 al 4.)

5. Ir a la línea #88

Utilizaremos la siguiente función:

```
81
82     for(;;){
83
84         switch (state) //Switch functions
85         {
86             /*HOME state*/
87             case HOME:
88                 RS232_PutStr((unsigned int *)"\n\r--SELECT FUNCTION--");
89                 RS232_PutStr((unsigned int *)"\r1.- RELAY SET");
90                 RS232_PutStr((unsigned int *)"\r2.- PWM SET");
91                 RS232_PutStr((unsigned int *)"\r3.- LCD PUT STRING");
92                 RS232_PutStr((unsigned int *)"\r4.- READ ADC PORT");
93                 RS232_PutStr((unsigned int *)"\r5.- READ DIP SWITCH\n");
94
95                 input = RS232_Read(); //Read RS232
```

RS232_PutStr (unsigned int *string);

- Función que manda una cadena de caracteres específica, por el puerto serial RS232; donde:
String = Mensaje a enviar (Cadena de caracteres 'String')

6. Ir a la línea #95

Utilizaremos la siguiente función:

```
88 RS232_PutStr((unsigned int *)"\n\r--SELECT FUNCTION--");
89 RS232_PutStr((unsigned int *)"\r1.- RELAY SET");
90 RS232_PutStr((unsigned int *)"\r2.- PWM SET");
91 RS232_PutStr((unsigned int *)"\r3.- LCD PUT STRING");
92 RS232_PutStr((unsigned int *)"\r4.- READ ADC PORT");
93 RS232_PutStr((unsigned int *)"\r5.- READ DIP SWITCH\n");
94
95 input = RS232_Read(); //Read RS232
96 if (input == '1') {
97     RS232_PutStr((unsigned int *)"\n\r--RELAY SET--\n");
98     RS232_PutStr((unsigned int *)"\rSELECT RELAY (1-4)");
99     RS232_PutStr((unsigned int *)"\rOR PRESS '*' TO EXIT\n");
100     state = RELAY; /*Go to RELAY state*/
101 }
102 if (input == '2') {
```

RS232_Read ();

- Función que espera a que exista un dato disponible en el buffer de entrada serial para este ser leído; donde:
Devuelve el valor introducido en el puerto serial (byte representado en carácter)

7. Ir a la línea #142

Utilizaremos la siguiente función:

```
135         if(input == '3')
136             relay=3;
137         if(input == '4')
138             relay=4;
139         if(EXIT)
140             state = HOME; //Return to Home state
141         if(input=='1' || input=='2' || input=='3' || input=='4')
142             RLY_Toggle(relay);
143
144         if(RLY_Get(relay))
145             sprintf(rs232MSG, "\nRelay %i turn ON", relay);
146         else
147             sprintf(rs232MSG, "\nRelay %i turn OFF", relay);
148         RS232_PutStr((unsigned int*)rs232MSG);
149         break;
```

RLY_Toogle (int num);

- Función que cambia el estado lógico de un RELAY; donde:
Int num = Número de RELAY que queremos conmutar
(Valor entero del 1 al 4.)

8. Ir a la línea #144

Utilizaremos la siguiente función:

```
137         if(input == '4')
138             relay=4;
139         if(EXIT)
140             state = HOME; //Return to Home state
141         if(input=='1' || input=='2' || input=='3' || input=='4')
142             RLY_Toggle(relay);
143
144         if(RLY_Get(relay))
145             sprintf(rs232MSG, "\nRelay %i turn ON", relay);
146         else
147             sprintf(rs232MSG, "\nRelay %i turn OFF", relay);
148         RS232_PutStr((unsigned int*)rs232MSG);
149         break;
150
151         /*PWM state*/
```

RLY_Get (int num);

- Función que lee el puerto de RELAYs; donde:
Int num = Número de RELAY que queremos leer (Valor entero del 1 al 4.)
Devuelve un valor booleano equivalente al estado del relay
(ON = Encendido, OFF = apagado).

9. Ir a la línea #145

Utilizaremos la siguiente función:

```
138         relay=4;
139         if(EXIT)
140             state = HOME; //Return to Home state
141         if(input=='1' || input=='2' || input=='3' || input=='4')
142             RLY_Toggle(relay);
143
144         if(RLY_Get(relay))
145             sprintf(rs232MSG, "\nRelay %i turn ON", relay);
146         else
147             sprintf(rs232MSG, "\nRelay %i turn OFF", relay);
148         RS232_PutStr((unsigned int*)rs232MSG);
149         break;
150
151         /*PWM state*/
152         case PWM:
```

sprintf (char *, const char *, ...);

- Función que Genera una cadena de caracteres y lo asigna a una variable; Donde:
Char * = Variable a asignar cadena de caracteres (Variable de tipo char)
const char *,= Cadena de caracteres a asignar a la variable.

10. Ir a la línea #161

Utilizaremos la siguiente función:

```
154
155     if(input == '+')
156         duty += 5;
157     if(input == '-')
158         duty -= 5;
159     if(EXIT)
160         state = HOME; //Return to Home state
161     OC_PWMSet(200,duty);
162     sprintf(rs232MSG,"\rDUTY CYCLE=%i",duty);
163     RS232_PutStr((unsigned int *)rs232MSG);
164     break;
165
166     /*LCD state*/
167     case LCD:
168         input = RS232_Read(); //Read RS232
```

OC_PWMSet (int period, int duty_cycle);

- Función que establece el periodo y ciclo de trabajo de la señal PWM; donde:
Int period = Periodo de oscilación (Valor entero de 0 a 8000.)
Int duty_cycle = Ciclo de trabajo en %porcentaje (valor entero de 0 a 100)

11. Ir a la línea #184

Utilizaremos la siguiente función:

```
177         strsize=0;
178     }
179     else {
180         sprintf(preletter,"%c",input);
181         sprintf(letter2,"%1s",preletter);
182         if(strsize<20) {           //First line of LCD
183             strcat(lcdMSG1,letter2);
184             LCD_PutStr(1,0,lcdMSG1,TRUE);
185         }
186         if(strsize>=20 && strsize<40) { //Second line of LCD
187             strcat(lcdMSG2,letter2);
188             LCD_PutStr(2,0,lcdMSG2,FALSE);
189         }
190         strsize++;
191     }
```

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).

12. Ir a la línea #192.

Utilizaremos la siguiente función:

```
191         }
192         break;
193
194     /*ADC state*/
195     case ADC:
196         input = RS232_Read(); //Read RS232
197         if(input == 32) {
198             sprintf(rs232MSG, "\rADC VALUE=%i", ADC_Read()); //Read
199             RS232_PutStr((unsigned int *)rs232MSG);
200         }
201         if(EXIT)
202             state=HOME; //Return to Home state
203         break;
204
205     /*DIP satte*/
```

ADC_Read ();

- Esta función lee el puerto ADC;
Regresa un valor equivalente al Valor análogo introducido (entero del 0 al 1024)

13. Ir a la línea #209.

Utilizaremos la siguiente función:

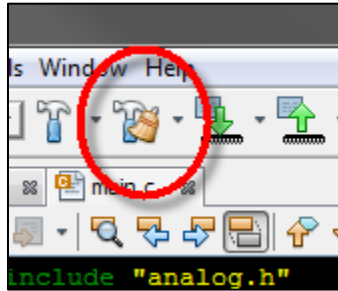
```
202         state=HOME; //Return to Home state
203         break;
204
205     /*DIP satte*/
206     case DIP:
207         input = RS232_Read(); //Read RS232
208         if(input == 32) {
209             sprintf(rs232MSG, "\rDIP SWITCH VALUE=%i", DIPSW_Read()); //R
210             RS232_PutStr((unsigned int *)rs232MSG);
211         }
212         if(EXIT)
213             state=HOME; //Return to Home state
214         break;
215
216     default:
```

DIPSW_Read ();

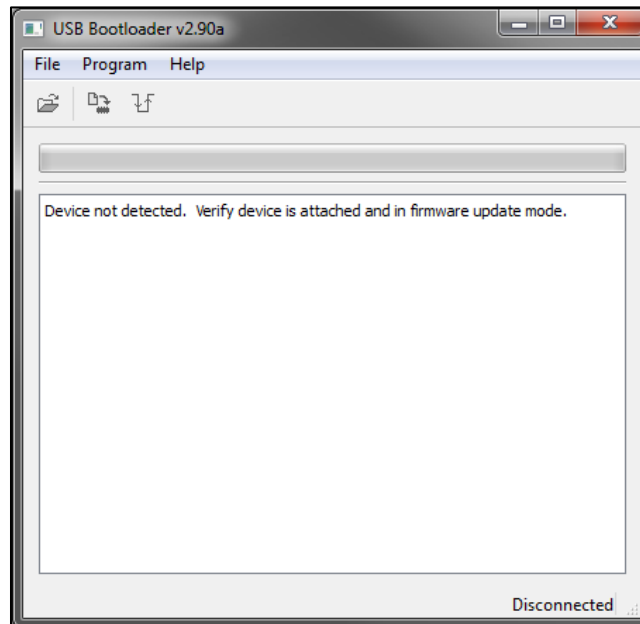
- Esta función lee el puerto de DIP-SWITCH;
Regresa un valor equivalente al valor del DIP-SWITCH (carácter del 1 al 255)

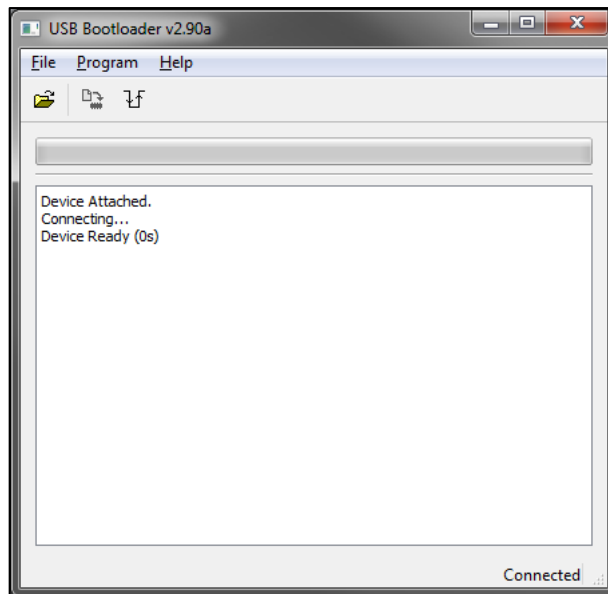
14. Compilar y programar

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



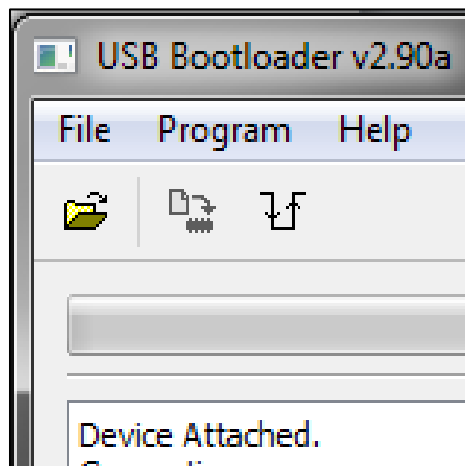
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





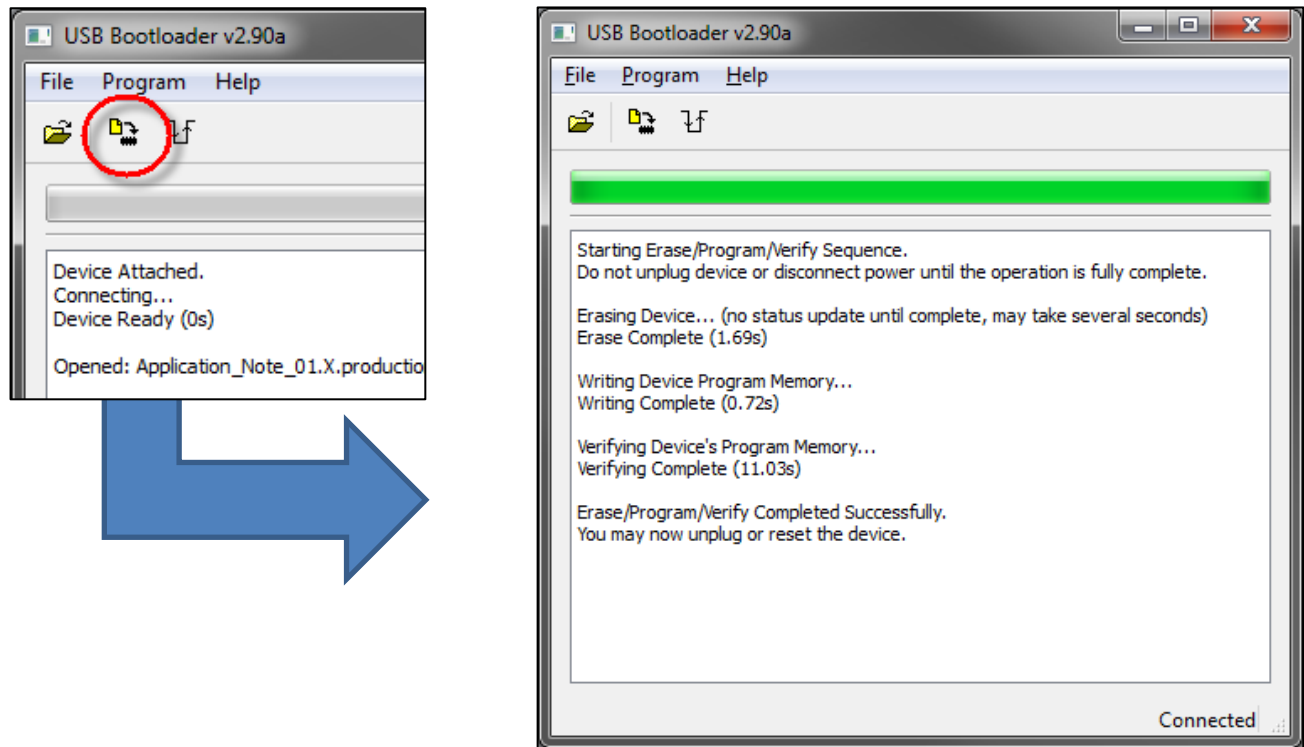
La ventana del Bootloader indicará la conexión establecida con el aguijón:

15. Hacer Clic en Abrir y Cargar el archivo **Application Note 37.X.production.hex**

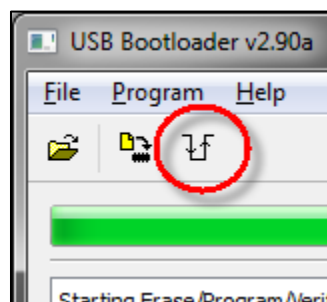


El archivo, depende de la plataforma de hardware.

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.

16. Para verificar el funcionamiento del programa verifique que al conectar nuestra tarjeta a una terminal serial, en esta se visualice un menú, desde el cual se puedan controlar los periféricos de nuestra tarjeta.