



Aguijón

Notas de aplicación

Nota de aplicación 12:

Open Collector PWM

Descripción:

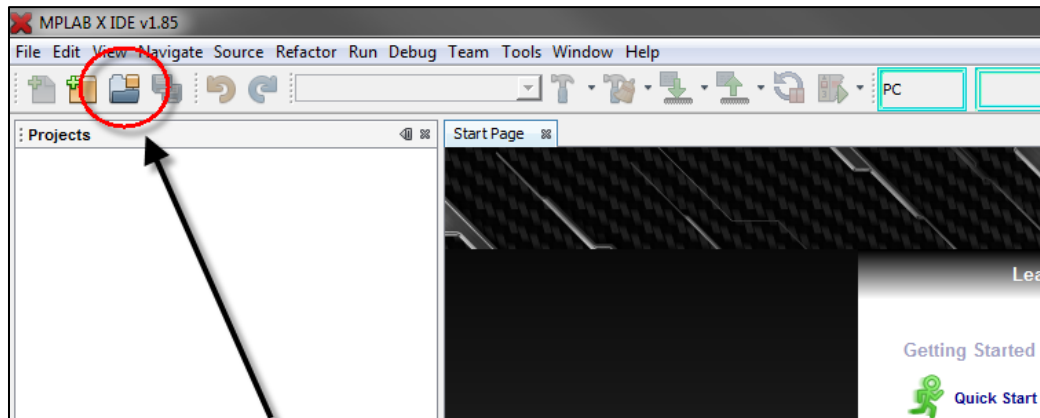
Generar una señal cuadrada variando el ancho de pulso (PWM)

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.

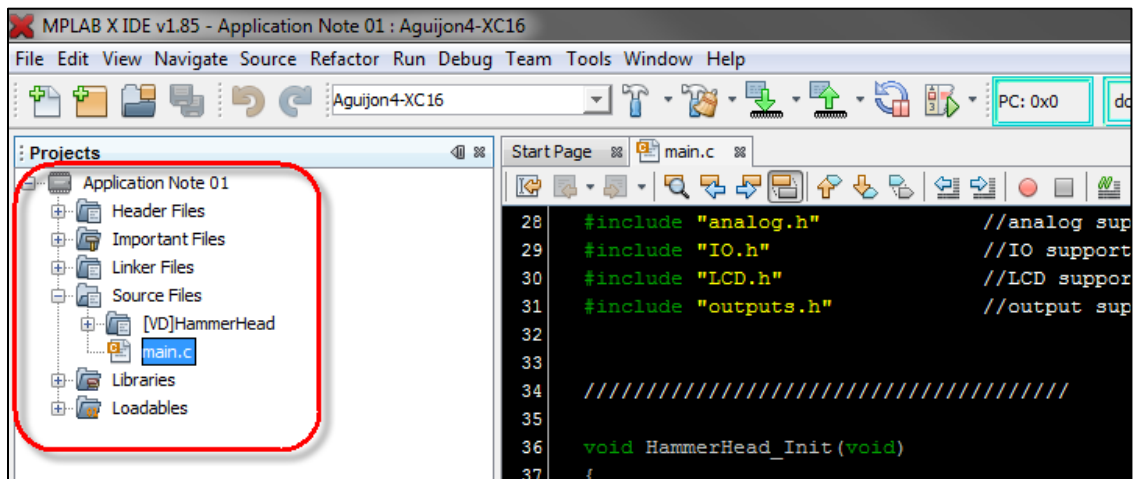
Pasos:

1. Abrir MPLAB X® y cargar el archivo del proyecto: **Application Note 12.X**



Haz 'clic' aquí y
abre el proyecto

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



3. Ir a la línea #58

Utilizaremos la siguiente función:

```
51     LCD_BacklightFadeIn();
52 #else
53     LCD_BacklightSet(BLIGHT_LVL_4);
54 #endif
55
56     LEDport_Set(0,ON);
57
58     OC_PWMChannelConfig(1);           //assigns PWM to open collector output #
59 }
60
61 //////////////////////////////////////
62
63
64 int main(void)
65 {
```

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.)

4. Ir a la línea #76.

Utilizaremos la siguiente función:

```
69     HammerHead_Init(); //initialize [VD]HammerHead
70     LCD_IntroAnimation();
71     LCD_PutStr(1,0,"Vinagron Digital",TRUE);
72     LCD_PutStr(2,0,"Application Note 12",FALSE);
73
74     for(;;){
75
76         newDuty=ADC_Range(5,95); //Read ADC range to (5-
77
78         OC_PWMSet(400,newDuty); //400uS( 2.5 Khz) perio
79
80         sprintf(lcdMSG,"Duty Cycle: %i",newDuty); //Create the string
81         LCD_PutStr(1,0,lcdMSG,TRUE); //Write lcdMSG
82
83         delayms(100);
```

ADC_Range (int min_val, int max_val);

- Esta función lee el puerto ADC y devuelve un valor proporcional al Rango establecido;
Donde:
Int min_val = valor mínimo del rango (Entero de 0 a 1023 y menor que **int max_val**)
Int max_val = valor máximo del rango (Entero de 0 a 1023 y mayor que **int min_val**)
Devuelve un valor proporcional al rango (Entero de min_val a max_val).

5. Ir a la línea #78

Utilizaremos la siguiente función:

```
71 LCD_PutStr(1,0,"Vinagron Digital",TRUE);
72 LCD_PutStr(2,0,"Application Note 12",FALSE);
73
74 for(;;){
75
76     newDuty=ADC_Range(5,95);           //Read ADC range to (5-
77
78     OC_PWMSet(400,newDuty);             //400uS( 2.5 Khz) perio
79
80     sprintf(lcdMSG,"Duty Cycle: %i",newDuty); //Create the string
81     LCD_PutStr(1,0,lcdMSG,TRUE);         //Write lcdMSG
82
83     delayms(100);
84
85 }
```

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)

Duty_cycle está definido por el valor del de entrada del puerto ADC en un rango de 5 a 95. Es decir la salida Open collector 1 generara una onda pulsante con una frecuencia de 2.5 KHz, y un ciclo de trabajo de 5% a 95%

6. Ir a la línea #80

Utilizaremos la siguiente función:

```
73
74     for(;;){
75
76         newDuty=ADC_Range(5,95);           //Read ADC range to (5-
77
78         OC_PWMSet(400,newDuty);           //400uS( 2.5 Khz) perio
79
80         sprintf(lcdMSG,"Duty Cycle: %i",newDuty); //Create the string
81         LCD_PutStr(1,0,lcdMSG,TRUE);       //Write lcdMSG
82
83         delayms(100);
84
85     }
86     return 0;
87 }
```

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.

7. Ir a la línea #81

Utilizaremos la siguiente función:

```
74     for(;;){
75
76         newDuty=ADC_Range(5,95);           //Read ADC range to (5-
77
78         OC_PWMSet(400,newDuty);           //400uS( 2.5 Khz) perio
79
80         sprintf(lcdMSG,"Duty Cycle: %i",newDuty); //Create the string
81         LCD_PutStr(1,0,lcdMSG,TRUE);       //Write lcdMSG
82
83         delayms(100);
84
85     }
86     return 0;
87 }
```

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 #81.

Utilizaremos la siguiente función:

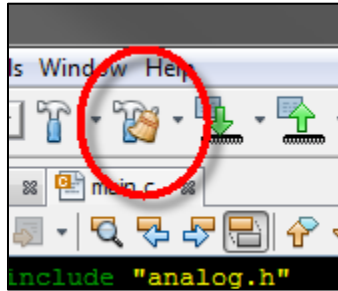
```
76     newDuty=ADC_Range(5,95);           //Read ADC range to (5-
77
78     OC_PWMSet(400,newDuty);             //400uS( 2.5 Khz) perio
79
80     sprintf(lcdMSG,"Duty Cycle: %i",newDuty); //Create the string
81     LCD_PutStr(1,0,lcdMSG,TRUE);         //Write lcdMSG
82
83     delayms(100);
84
85 }
86 return 0;
87 }
```

Delayms (ms);

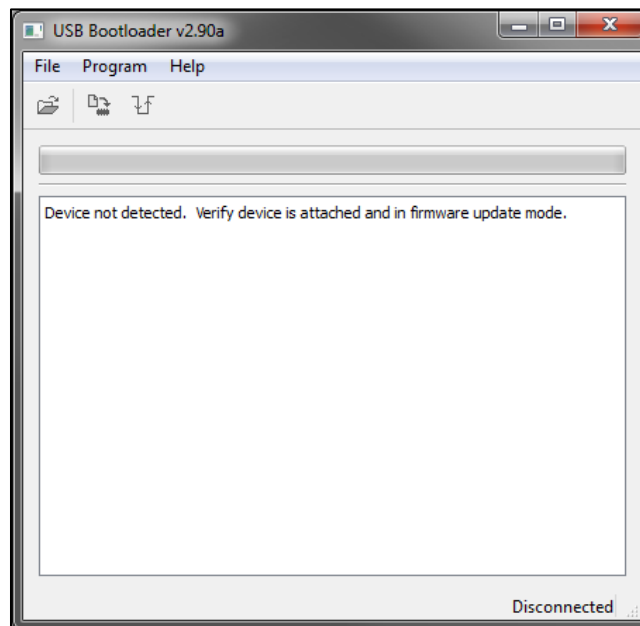
- Función que Pausa el programa por un tiempo determinado (en milisegundos); donde:
ms = el número de milisegundos que se desea pausar el programa.

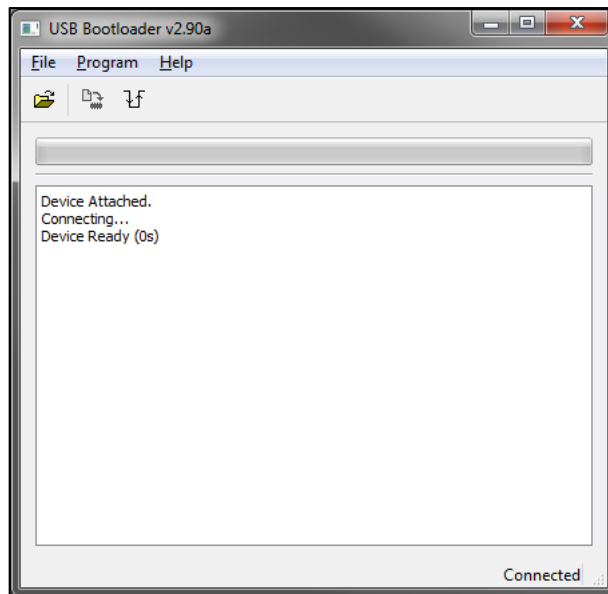
9. Compilar y programar

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



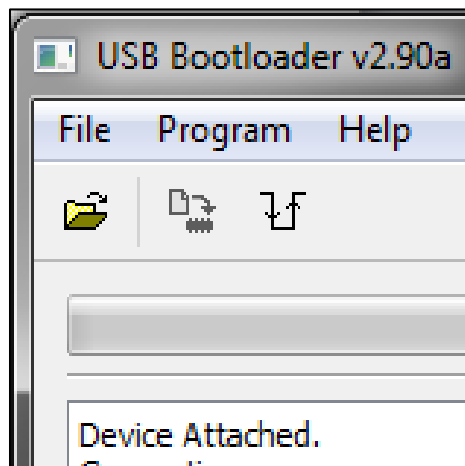
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





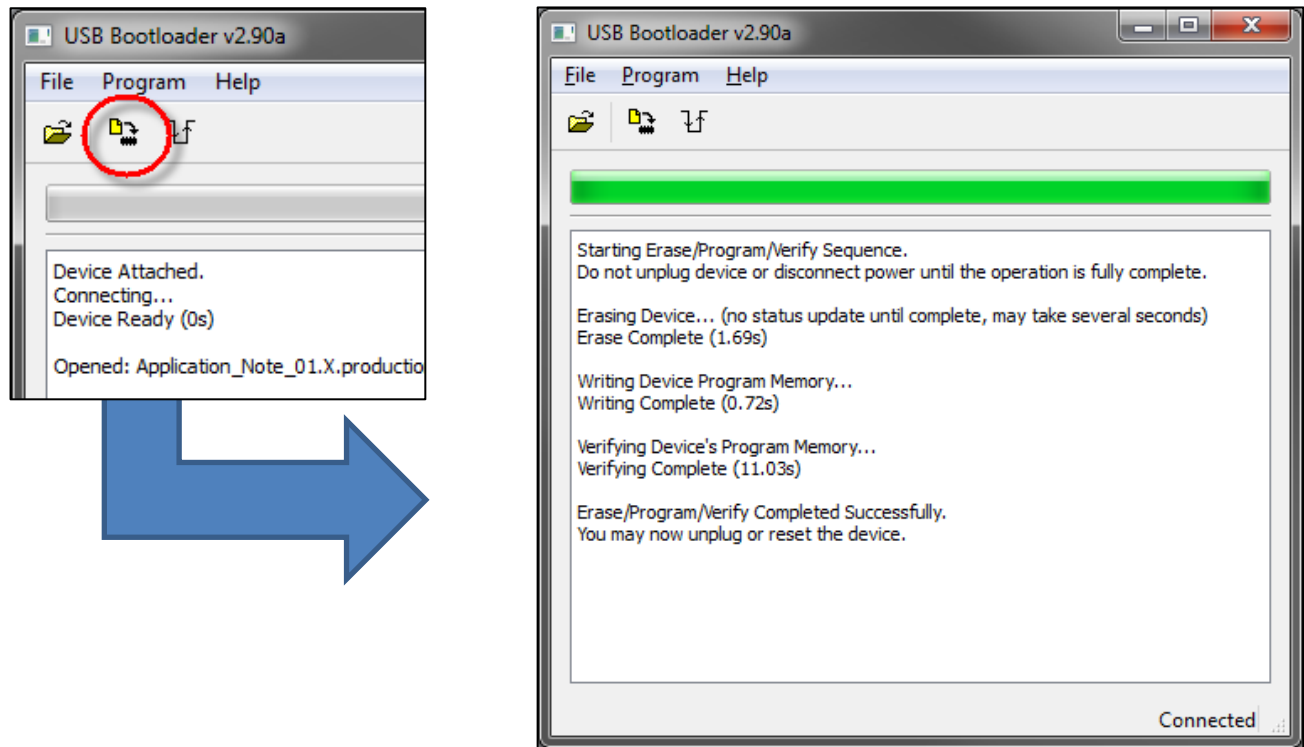
La ventana del Bootloader indicará la conexión establecida con el aguijón:

10. Hacer Clic en Abrir y Cargar el archivo **Application Note 12.X.production.hex**

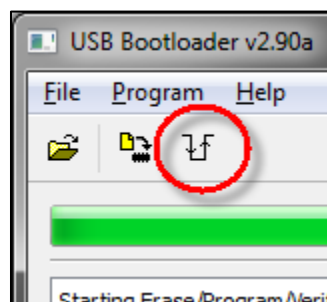


El archivo, depende de la plataforma de hardware.

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.

11. Para verificar el funcionamiento del programa verifique que la intensidad de luz del led indicador de la salida OPEN COLLECTOR 1 varia proporcionalmente al valor introducido en el puerto ADC.