



Aguijón

Notas de aplicación

Nota de aplicación 30:

[Generate a Square Wave](#)

Descripción:

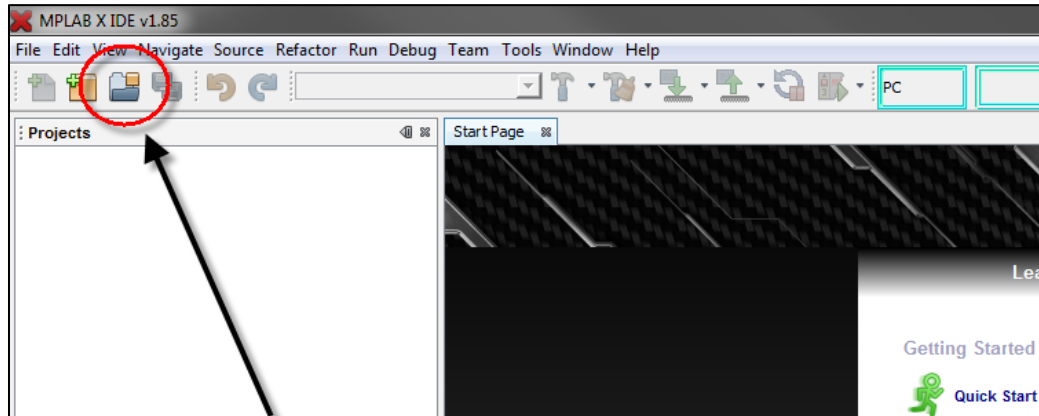
Generar una onda cuadrada programable la cual se pueda manipular, el periodo de oscilación.

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 30.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      /*ADC*/
45      ADC_Init();
46
47      /*LCD*/
48      LCD_Init(OFF);
49
50      /*PWM*/
51      OC_PWMChannelConfig(1); // Set the PWM to Use
52
53
54  #if defined(USE_LCD_EXTRA_FEATURES)
55      LCD_BacklightFadeIn();
56  #else
57      LCD_BacklightSet(BLIGHT_LVL_4);
58  #endif
```

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. En a la línea # 87 utilizaremos la siguiente función:

```
80     delayms(1000);
81     LCD_Clear();
82
83     for(;;){
84
85
86
87         period = ADC_Range(100,1000);           // Read ADC with
88
89         OC_PWMSet(period,duty_cycle);           // Variable peri
90
91         finalPeriod = (double) (period);
92
93         frequency = (1 / (finalPeriod/(1000000))); // Formula for F
94
```

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)
Int max_val = valor máximo del rango (Entero de 0 a 1023)
Devuelve un valor proporcional al rango (Entero de min_val a max_val).

5. En a la línea # 89 utilizaremos la siguiente función:

```
82
83     for(;;){
84
85
86
87         period = ADC_Range(100,1000);           // Read ADC wi
88
89         OC_PWMSet(period,duty_cycle);           // Variable pe
90
91         finalPeriod = (double) (period);
92
93         frequency = (1 / (finalPeriod/(1000000))); // Formula for
94
95         // Create the string for lcdMSG1
96         sprintf(lcdMSG1,"Period: %d", period);
```

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).

6. En a la línea # 96 utilizaremos la siguiente función:

```
89         OC_PWMSet(period,duty_cycle);           // Variable peri
90
91         finalPeriod = (double) (period);
92
93         frequency = (1000000 / finalPeriod);      // Formula for F
94
95         // Create the string for lcdMSG1
96         sprintf(lcdMSG1,"Period: %d", period);
97         LCD_PutStr(1,0,lcdMSG1,ON);
98
99         // Create the string for lcdMSG2
100        sprintf(lcdMSG2,"Frequency: %.2f", frequency);
101        LCD_PutStr(2,0,lcdMSG2,OFF);
102
103        delayms(150);                             // Delay 150 mill
```

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. En a la línea # 97 utilizaremos la siguiente función:

```
90
91     finalPeriod = (double) (period);
92
93     frequency = (1000000 / finalPeriod);           // Formula for F
94
95     // Create the string for lcdMSG1
96     sprintf(lcdMSG1,"Period: %d", period);
97     LCD_PutStr(1,0,lcdMSG1,ON);
98
99     // Create the string for lcdMSG2
100    sprintf(lcdMSG2,"Frequency: %.2f", frequency);
101    LCD_PutStr(2,0,lcdMSG2,OFF);
102
103    delayms(150);                                   // Delay 150 mill
104
```

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. En a la línea # 97 utilizaremos la siguiente función:

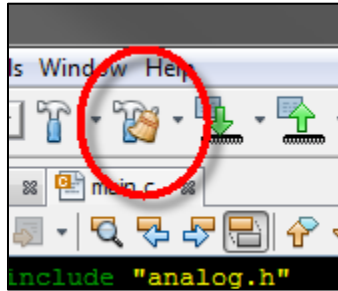
```
96     sprintf(lcdMSG1,"Period: %d", period);
97     LCD_PutStr(1,0,lcdMSG1,ON);
98
99     // Create the string for lcdMSG2
100    sprintf(lcdMSG2,"Frequency: %.2f", frequency);
101    LCD_PutStr(2,0,lcdMSG2,OFF);
102
103    delayms(150); // Delay 150 mill
104
105
106    }
107
108    return 0;
109 }
```

Delayms (ms);

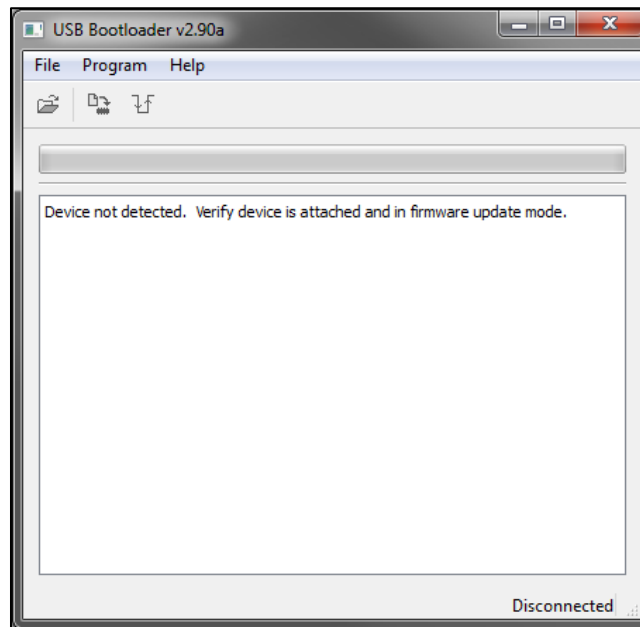
- Función que Pausa el programa por un tiempo determinado (en milisegundos); donde: **ms** = el número de milisegundos que se desea pausar el programa.

9. Compilar y programar

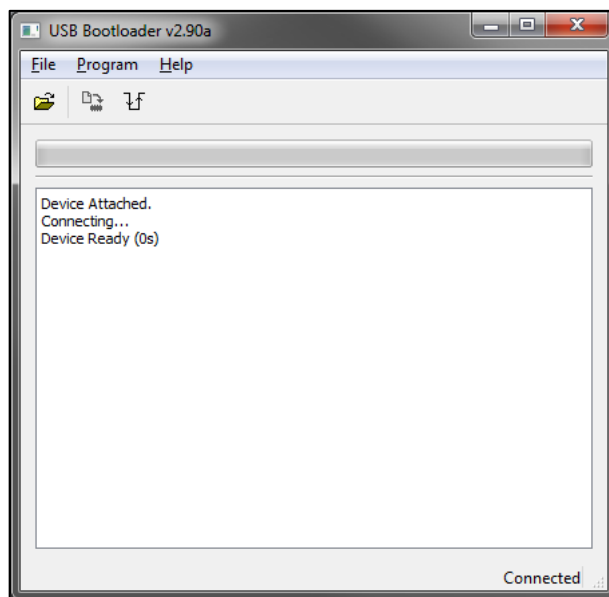
Al hacer clic en el ícono de compilar, y si no hay errores de compilación, el bootloader será cargado automáticamente.



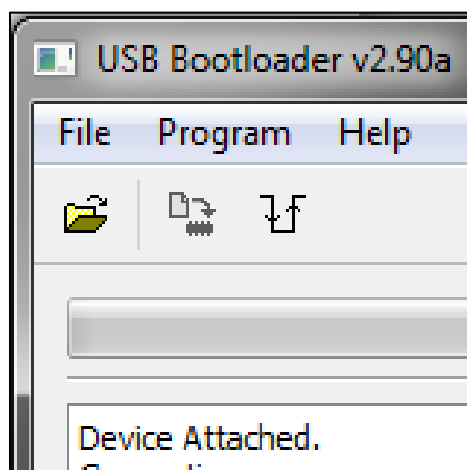
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear



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

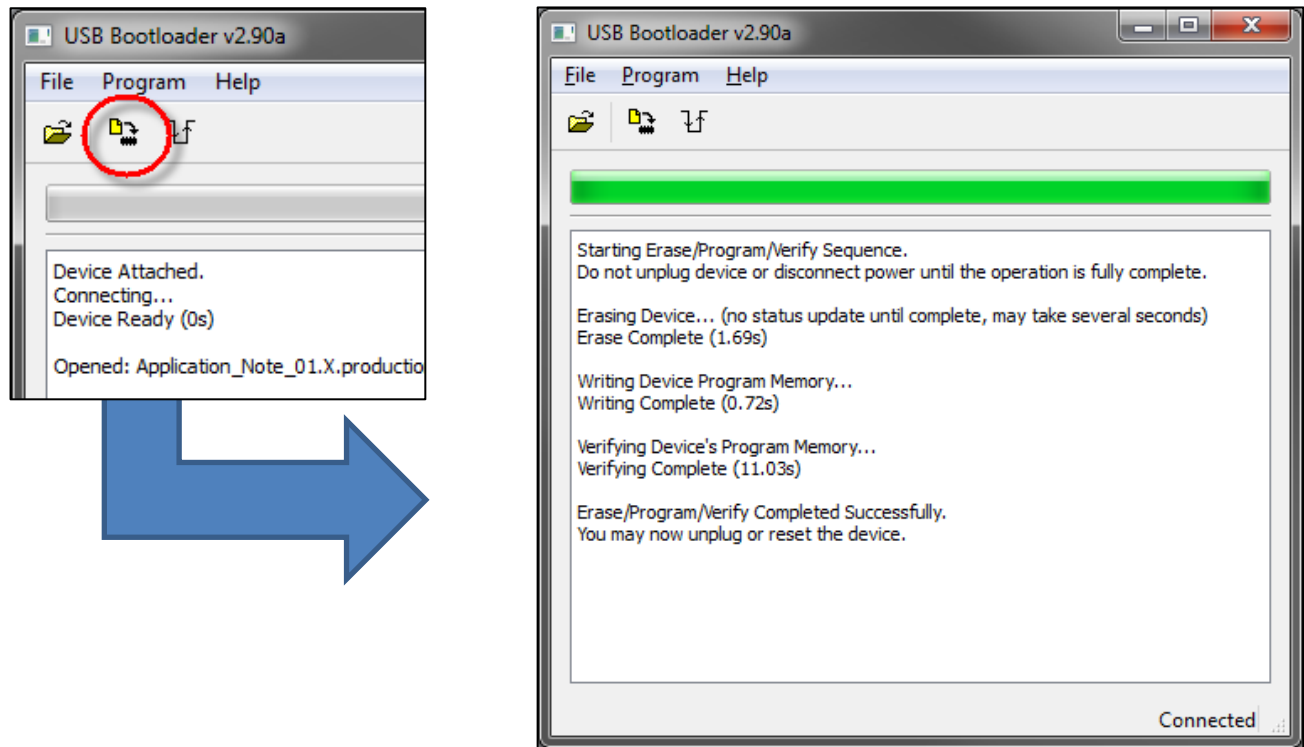


10. Hacer Clic en Abrir y Cargar el archivo **Application Note 31.X.production.hex**

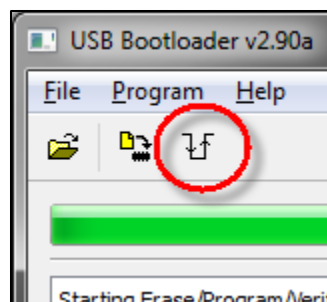


El archivo, depende de la plataforma de hardware.

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 verificar que al variar el puerto ADC, varié a su vez la frecuencia y el periodo. La frecuencia es calculada por el periodo establecido por el puerto ADC.