



Aguijón

Notas de aplicación

Nota de aplicación 31:

Input Capture

Descripción:

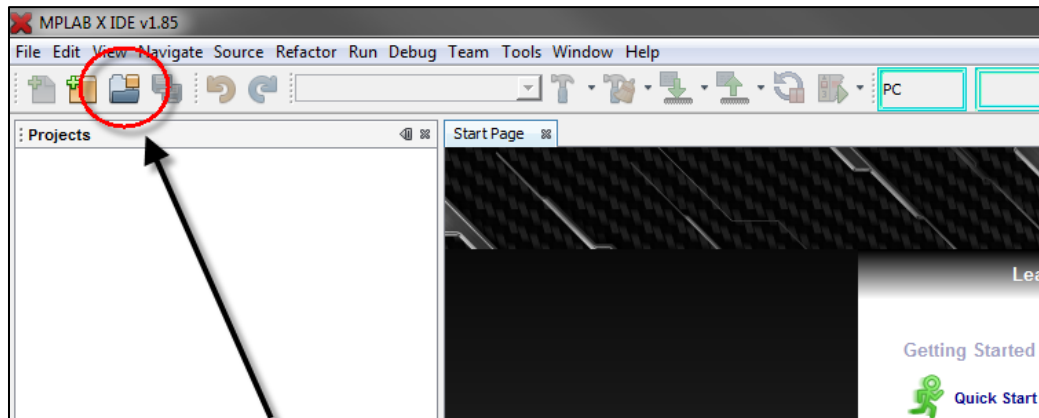
Medir tanto el periodo como la frecuencia de una onda PWM, esto utilizando el modulo Input Capture.

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 31.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      /*PWM*/
51      OC_PWMChannelConfig(1); //Assigns PWM to open collector output #1
52
53      /*Input Capture*/
54      InputCapture_Init();
55
56  #if defined(USE_LCD_EXTRA_FEATURES)
57      LCD_BacklightFadeIn();
58  #else
```

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 #54.

Utilizaremos la siguiente función:

```
47      /*LCD*/
48      LCD_Init(LCD_MODE_1);
49
50      /*PWM*/
51      OC_PWMChannelConfig(1); //Assigns PWM to open collector output #1
52
53      /*Input Capture*/
54      InputCapture_Init();
55
56      #if defined(USE_LCD_EXTRA_FEATURES)
57          LCD_BacklightFadeIn();
58      #else
59          LCD_BacklightSet(BLIGHT_LVL_4);
60      #endif
61
```

InputCapture_Init();

- Esta función inicializa el modo Input Capture;
Establece los registros necesarios para utilizar el modo Input Capture, Declara el pin Remapeable #3 del puerto de expansión 2, como entrada del modo Input Capture.
(EXP_PORT2_PIN3_RPIN)

5. Ir a la línea #81.

Utilizaremos la siguiente función:

```
74 HammerHead_Init(); //initialize [VD]HammerHead
75 LCD_IntroAnimation();
76 LCD_PutStr(1,0,"Vinagron Digital",TRUE);
77 LCD_PutStr(2,0,"Application Note 30",FALSE);
78
79 for(;;){
80
81     value=ADC_Range(5,1000); //Read ADC Range of 5 to 1000
82
83     OC_PWMSet(value,50); //Variable period and 50% duty
84
85     period=Input_Period(); //Read Period
86
87     frequency=Input_Frequency(); //Read Frequency
88 }
```

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).

6. Ir a la línea #83

Utilizaremos la siguiente función:

```
76 LCD_PutStr(1,0,"Vinagron Digital",TRUE);
77 LCD_PutStr(2,0,"Application Note 30",FALSE);
78
79 for(;;){
80
81     value=ADC_Range(5,1000);           //Read ADC Range of 5 to 1000
82
83     OC_PWMSet(value,50);               //Variable period and 50% duty
84
85     period=Input_Period();              //Read Period
86
87     frequency=Input_Frequency();        //Read Frequency
88
89     //Create a first string for the LCD//
90     sprintf(lcdMSG1,"Period= %i %lu",value,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)

7. Ir a la línea #85

Utilizaremos la siguiente función:

```
78
79     for(;;){
80
81         value=ADC_Range(5,1000);           //Read ADC Range of 5 to 1000
82
83         OC_PWMSet(value,50);               //Variable period and 50% duty
84
85         period=Input_Period();             //Read Period
86
87         frequency=Input_Frequency();       //Read Frequency
88
89         //Create a first string for the LCD//
90         sprintf(lcdMSG1,"Period= %i %lu",value,period);
91         LCD_PutStr(1,0,lcdMSG1,TRUE);
92
```

Input_Period ();

- Función que lee el periodo de la señal introducida en el pin de entrada Input Capture; donde:
- Regresa un valor de tipo entero sin signo de 32 bits (UINT32) proporcional a el periodo expresado en Micro Segundos

8. Ir a la línea #87

Utilizaremos la siguiente función:

```
80
81     value=ADC_Range(5,1000);           //Read ADC Range of 5 to 1000
82
83     OC_PWMSet(value,50);               //Variable period and 50% duty
84
85     period=Input_Period();             //Read Period
86
87     frequency=Input_Frequency();       //Read Frequency
88
89     //Create a first string for the LCD//
90     sprintf(lcdMSG1,"Period= %i %lu",value,period);
91     LCD_PutStr(1,0,lcdMSG1,TRUE);
92
93     //Create a second string for the LCD//
94     sprintf(lcdMSG2,"Frequency= %luHz",frequency);
```

Input_Frequency ();

- Función que lee la frecuencia de la señal introducida en el pin de entrada Input Capture; donde:
- Regresa un valor de tipo entero sin signo de 32 bits (UINT32) proporcional a la frecuencia expresado en Hz.

9. Ir a la línea #90

Utilizaremos la siguiente función:

```
83      OC_PWMSet(value,50);           //Variable period and 50% duty
84
85      period=Input_Period();          //Read Period
86
87      frequency=Input_Frequency();    //Read Frequency
88
89      //Create a first string for the LCD//
90      sprintf(lcdMSG1,"Period= %i %lu",value,period);
91      LCD_PutStr(1,0,lcdMSG1,TRUE);
92
93      //Create a second string for the LCD//
94      sprintf(lcdMSG2,"Frequency= %luHz",frequency);
95      LCD_PutStr(2,0,lcdMSG2,FALSE);
96
97      delayms(150);                   //Delay 150 milliseconds
```

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 #91

Utilizaremos la siguiente función:

```
84
85     period=Input_Period();           //Read Period
86
87     frequency=Input_Frequency();     //Read Frequency
88
89     //Create a first string for the LCD//
90     sprintf(lcdMSG1,"Period= %i %lu",value,period);
91     LCD_PutStr(1,0,lcdMSG1,TRUE);
92
93     //Create a second string for the LCD//
94     sprintf(lcdMSG2,"Frequency= %luHz",frequency);
95     LCD_PutStr(2,0,lcdMSG2,FALSE);
96
97     delayms(150);                     //Delay 150 milliseconds
98
```

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).

11. Ir a la línea #97

Utilizaremos la siguiente función:

```
86
87     frequency=Input_Frequency();    //Read Frequency
88
89     //Create a first string for the LCD//
90     sprintf(lcdMSG1,"Period= %i %lu",value,period);
91     LCD_PutStr(1,0,lcdMSG1,TRUE);
92
93     //Create a second string for the LCD//
94     sprintf(lcdMSG2,"Frequency= %luHz",frequency);
95     LCD_PutStr(2,0,lcdMSG2,FALSE);
96
97     delays(150);                    //Delay 150 milliseconds
98
99 }
100 return 0;
101 }
```

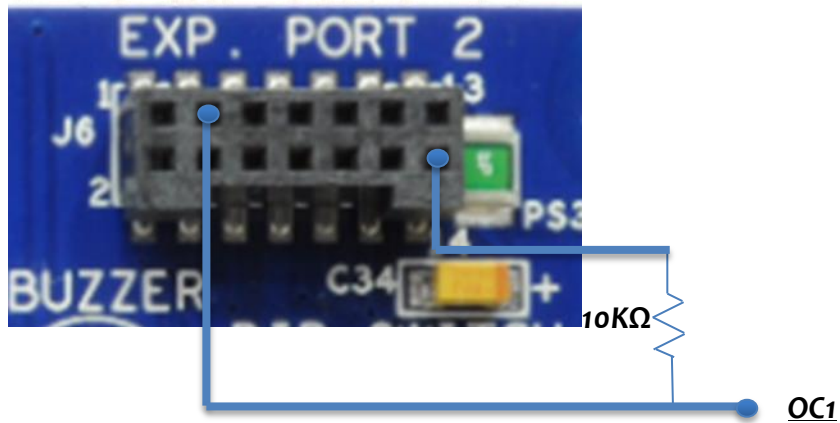
Delays (ms);

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

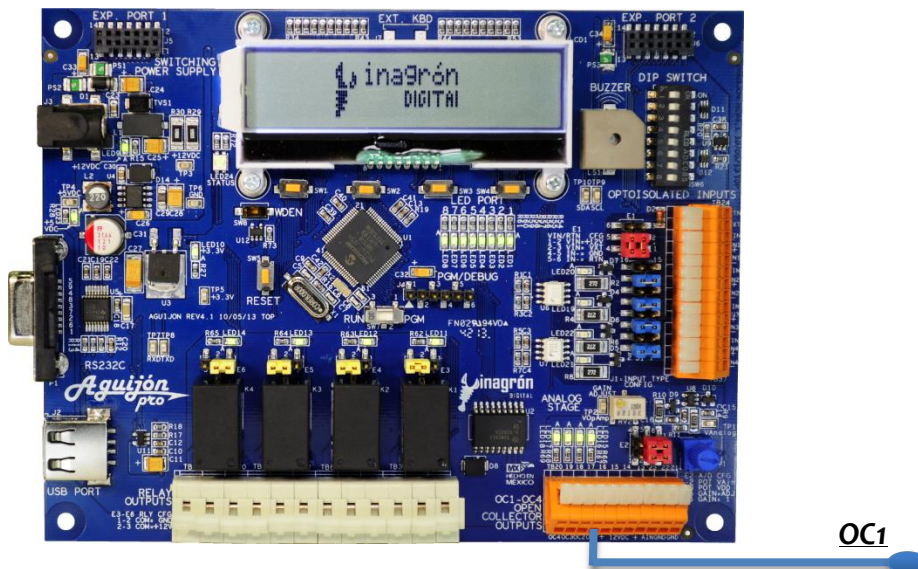
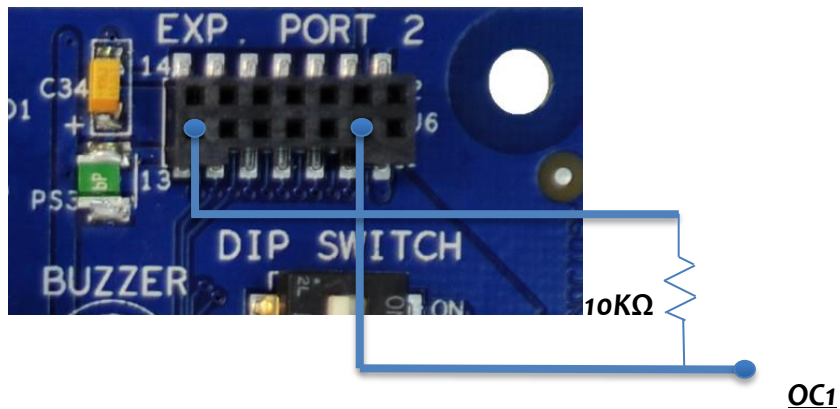
Diagrama de conexión

* Ver archivo Cambios REV4.1

- Aguijon 4.0

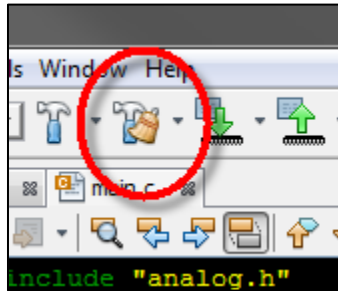


- Aguijon 4.1

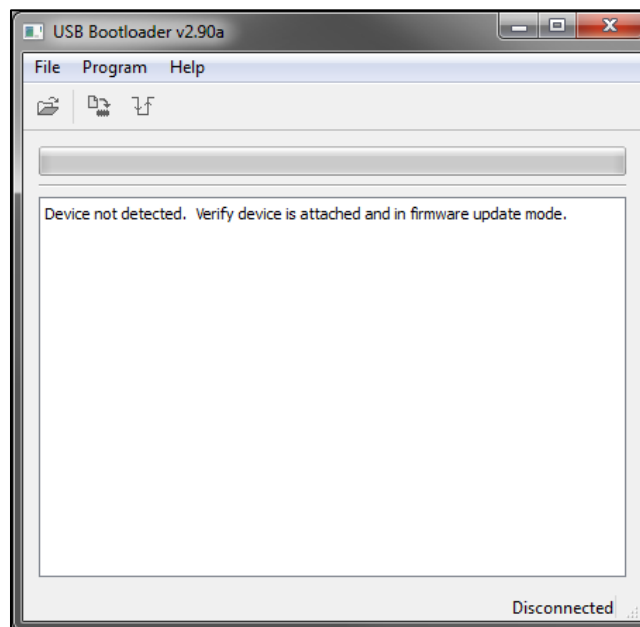


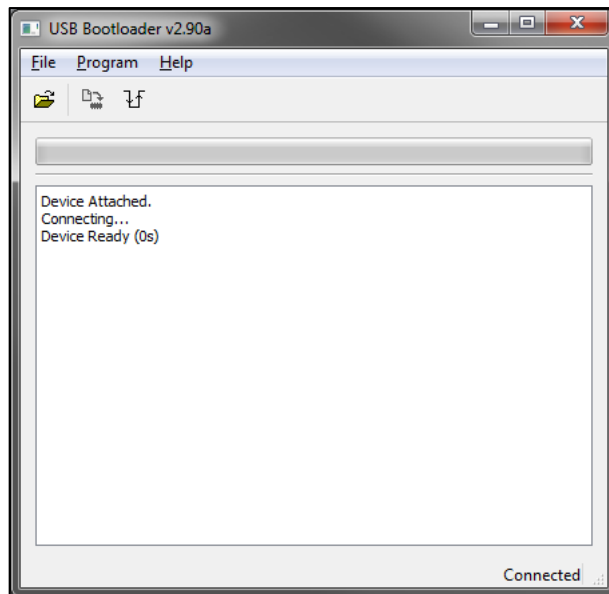
12. Compilar y programar

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



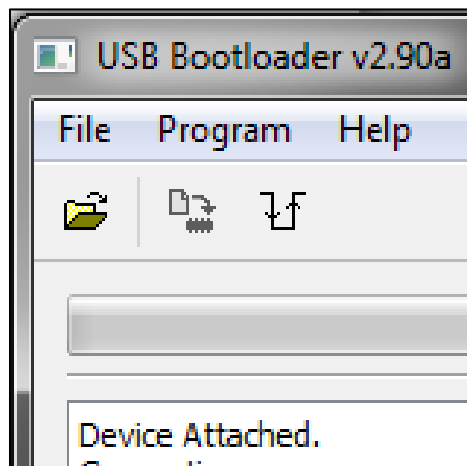
Cuando aparezca la ventana del bootloader, presione el Push-Button número 1 y conecte la fuente de voltaje o encienda el Aguijón y mantenga el PB1 presionado hasta que los LEDs empiecen a parpadear.





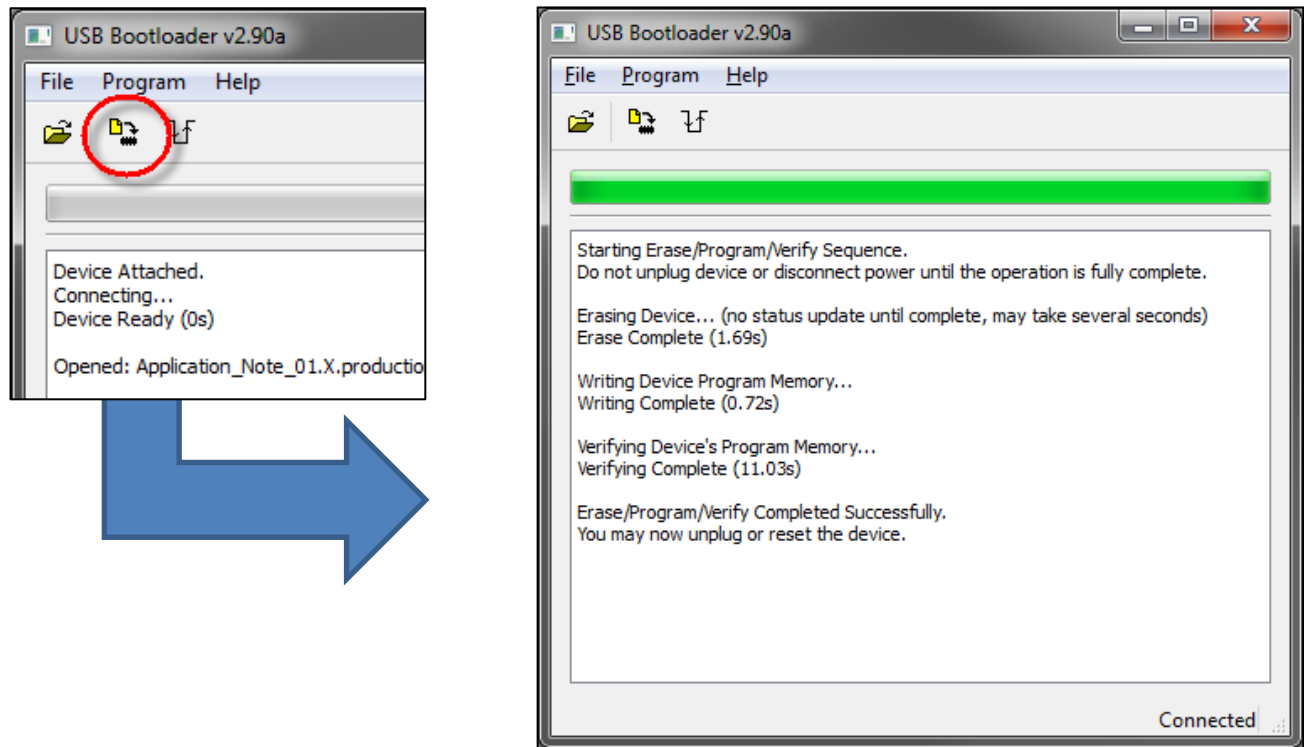
La ventana del Bootloader indicará la conexión establecida con el aguijón:

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

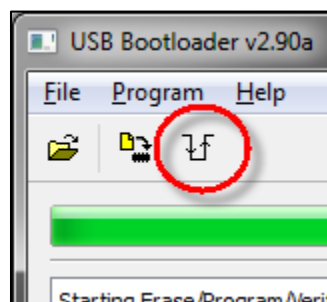


El archivo, depende de la plataforma de hardware.

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.

14. Para verificar el funcionamiento del programa verifique que al variar la entrada del puerto ADC, varié tanto la frecuencia como el periodo monitoreado constantemente en la pantalla LCD. De igual manera verifique que los dos valores mostrados en la primera línea de la pantalla coincidan, es decir que el periodo de salida sea igual al periodo leído.