Guía N° 5: *Introducción al dsPIC*

Objetivo:

Familiarizarse con el controlador de señal [1] dsPIC33 y su entorno de programación

Fecha de entrega:

13/09/2019

Formato de entrega:

Esta guía se resuelve sobre un único proyecto base, en el cual cada ejercicio se corresponde con porciones de código, que serán comentadas/descomentadas para evaluar cada punto. Cada porción debe estar indicada con los comentarios necesarios.
Toda la carpeta del proyecto deberá comprimirse en un archivo .zip o .rar y enviarse por correo con título *Entrega Guía 5.*

Bibliografía recomendada:

Título: Programming dsPIC MCU in C
Authors: Zoran Milivojević, Djordje Šaponjić
Editorial: MikroElektronika
Disponible online en: http://www.mikroe.com/products/view/266/programming-dspic-mcu-in-c/

Manual de Referencia de la placa de desarrollo (Placa vieja - Se programa con Pickit 3)

Página web del dispositivo

Guía de uso para el bootloader de la placa nueva Pines placa nueva Esquemático

Requisitos

Tener instalado alguna versión del MPLABX de Microchip, disponible desde la sección Downloads en http://www.microchip.com/mplab/mplab-x-ide

Tener instalado el compilador de C XC16 de Microchip, disponible desde la sección Downloads en http://www.microchip.com/mplab/compilers

[1] https://en.wikipedia.org/wiki/Digital_signal_controller

Uso básico del entorno / Configuración del proyecto

Desde el menú hacer click en **File → New Project**, luego en la ventana Categories seleccionamos **Samples → Microchip Embedded** y en la ventana **Projects** seleccionamos **dsPIC33 C Template** y hacemos click en **Next**. En la siguiente ventana establecemos el nombre del proyecto, la ubicación de la carpeta del proyecto y lo marcamos como proyecto principal (tildando **Set Main Project**). Hacer click en **Finish** y tendremos creada toda la estructura de proyecto.
Luego, haciendo click derecho sobre el nombre del proyecto en la pestaña **Projects** seleccionamos **Properties**. Hacer click en **Manage Configuration**, seleccionar la primer configuración y renombrarla como XC16_dsPIC33FJ128GP804. El nombre aquí no tiene importancia, solo sirve como ayuda identificar la configuración. Eliminar todas las configuraciones restantes.
Elegir como dispositivo el dsPIC33FJ128GP804, de la familia dsPIC33. Dentro de **Hardware Tools** seleccionar Pickit 3 y como Compiler Chain seleccionar XC16 (Que deberá estar instalado previamente).

A continuación hacer click en el botón Clean and Build, que tiene como ícono un martillo y una escoba. Si todo está configurado correctamente, el proyecto se compilará sin errores. Llegado a este punto podemos comenzar a agregar código.

Pasos genéricos para la edición de la plantilla de proyecto y el agregado de código

P1 - El primer paso es la selección de los bits de configuración. Las instrucciones para esto están en el archivo *configuration_bits.c*

P2 - A menos que no se usen interrupciones en la aplicación, agregue todos los vectores de interrupción que se pretendan usar en el archivo *interrupts.c*. Se proveen segmentos de código que pueden usarse como ejemplo. Verifique con la hoja de datos que la interrupción usada esté presente en el dispositivo. Los mnemónicos de cada vector de interrupción se pueden encontrar en archivo *.h* del dispositivo.

P3 - Los parámetros de sistema como la frecuencia de operación se definen en el archivo *system.h*. Agregue funciones de sistema en *system.h* y *system.c*. Por ejemplo, una función para determinar el origen de un reset, la conmutación entre fuentes de reloj, el salto a un modo de bajo consumo, etc, son el tipo de funciones consideradas como funciones a nivel de sistema.

P4 - Las funciones a nivel de usuario irán en *user.h* y *user.c*. Estas son funciones que hacen cosas como inicializar I/O, periféricos como el ADC, cómputo de algoritmos del usuario y realizar cálculos sobre datos muestreados. Los prototipos de función irán en

user.h, mientras que la implementación de estas irá en el archivo user.c. Se recomienda que cada función tenga un nombre descriptivo como por ejemplo *ConfigurarADC()*.

P5 - Agregar código al archivo *main.c* en la línea que dice <INSERT USER APPLICATION CODE HERE>.
Por ejemplo, las llamadas a funciones en *user.c* o *system.c* o el bucle principal del programa.
Las variables globales pueden agregarse también en el archivo *main.c*. Como práctica general, las macros y prototipos de función debe ir en *archivos.h*, mientras que las declaraciones de variables deben ir en archivos .c.
Se recomienda que cada porción de código se ubique dentro de una función con nombre descriptivo y evitar la utilización de código suelto en el main.

P6 - Diseñar el resto de la aplicación. Agregar nuevos archivos y realizar las verificaciones necesarias.

P7 - Documente lo que hace el proyecto e incluya información adicional en el archivo *project_information.txt*.

1. Manejo básico de puertos en configuración básica

    1.1. Crear un programa que lea el estado del botón 0 (según la tabla en página 3 del manual de referencia de la placa de desarrollo) y lo asigne al LED 0 (también según la tabla de página 3). Ver referencia de la placa de desarrollo para identificar banco y bit para el botón y el LED. Para escribir un bit de un puerto de salida se debe primero configurar ese bit en modo salida, escribiendo 0 al bit correspondiente en el registro TRISX, donde X es la letra que designa al puerto. Una manera sencilla de hacerlo es mediante:

    ```
    TRISXbits.TRISXY = 0; // X Letra de Puerto, Y número de bit del puerto
    ```

    Luego se puede escribir el bit mediante

    ```
    LATXbits.LATXY = valor;
    ```

    Para leer un bit de un puerto se puede tomar el valor haciendo

    ```
    valor = PORTXbits.RXY;
    ```

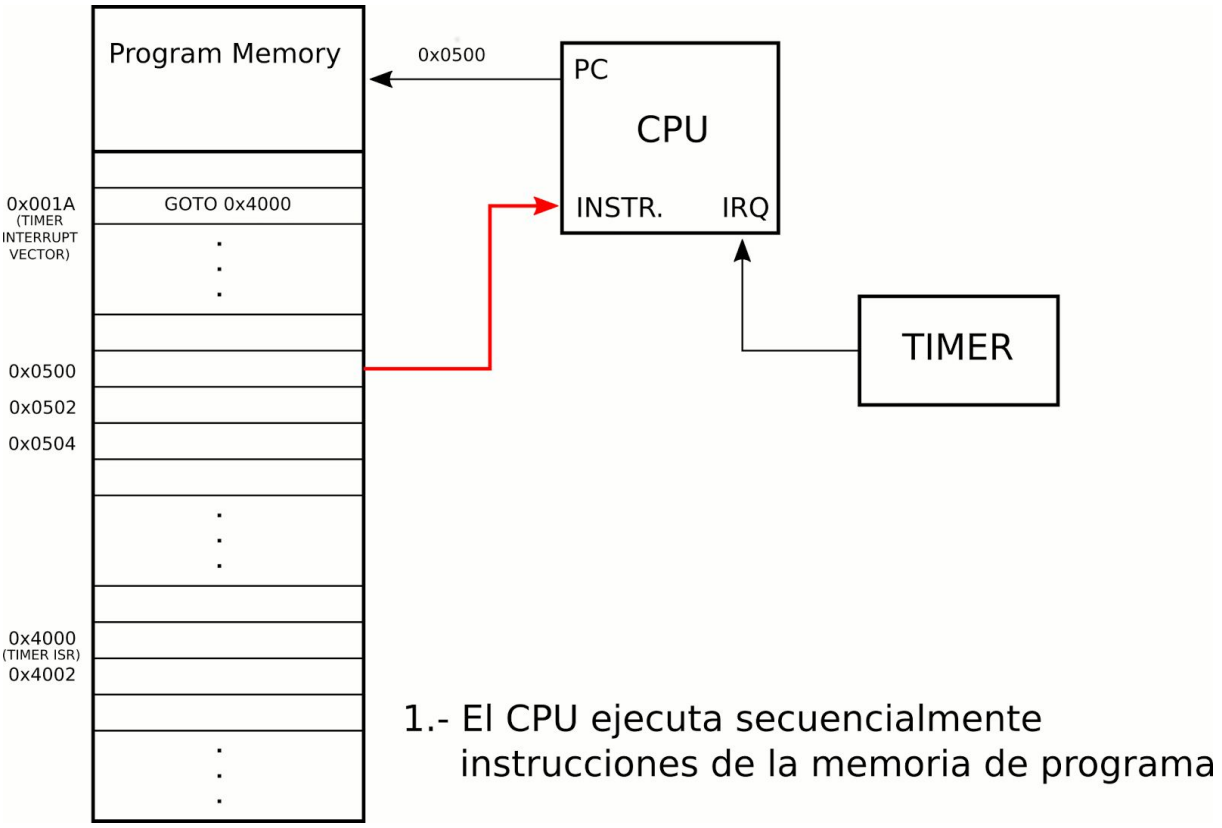
    Se puede encontrar información relacionada al acceso a los puertos en Section 10. I/O Ports - dsPIC33F/PIC24H FRM.
    Generar el código para los bits de configuración desde **Window → PIC Memory View → Configuration Bits**. Modificar el bit FWDTEN para apagar el

Watchdog y hacer click en **Generate Source Code**. Pegar el código generado en el archivo *configuration_bits.c*.

1.2. Crear un programa que encienda y apague el led 1 aproximadamente cada 1 segundo. Utilizar para el retardo un bucle FOR con una variable de 32 bits como índice (tipo de dato `uint32_t`). La duración total del retardo, en iteraciones del bucle FOR será de aproximadamente 155 mil ciclos.

2. Temporizador con interrupción [2] y uso del PLL

Las interrupciones son señales que indican al procesador que un evento necesita atención inmediata. Estas causan que la CPU detenga momentáneamente el flujo de ejecución normal para ejecutar el código asociado a la interrupción en particular. Algunos registros importantes como el contador de programa y el registro de estado se almacenan automáticamente en la memoria de stack, de manera de poder continuar con la aplicación en curso luego de atender la interrupción. Puede consultarse información adicional sobre el funcionamiento de las interrupciones en Section 32. Interrupts (Part III) - dsPIC33F/PIC24H FRM.

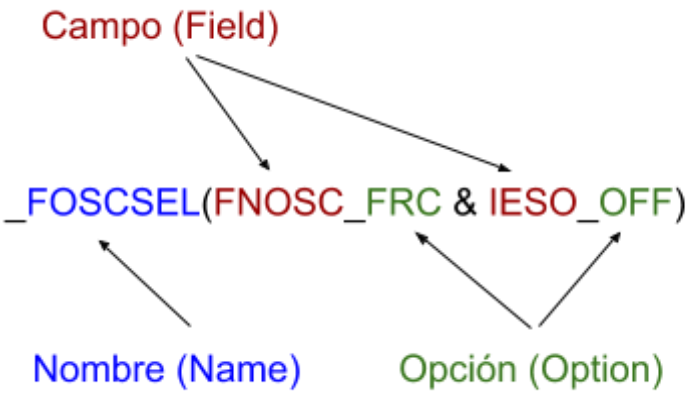


2.1. Crear un programa para encender y apagar un led mediante una interrupción generada cada un segundo. Se deberá configurar un temporizador en modo 32bits, dentro de una función en el archivo user.c. El código de configuración para el timer se encuentra en el manual de referencia para temporizadores,

[2] https://en.wikipedia.org/wiki/Interrupt

Section 11. Timers - dsPIC33F/PIC24H FRM , página 18, ejemplo 11-6. Traducir todos los comentarios del ejemplo al castellano. Copiar la rutina de interrupción al archivo *interrupts.c* y agregar el código necesario para cambiar el estado de led 2. Configurar el registro PR3 con 0x40 y PR2 con 0x00 dentro de la rutina de configuración del timer.

2.2. Configurar la frecuencia del oscilador del sistema F_{OSC} a 80MHz, de manera que la frecuencia de ejecución de instrucciones F_{CY} sea de 40MIPS, utilizando el ejemplo de código disponible en el manual de oscilador Section 39. Oscillator (Part III) - dsPIC33F FRM. Página 23, ejemplo 39-1. La primer parte del ejemplo corresponde a la definición de los bits de configuración que hay que deben modificarse desde configuration_bits.c. Las macros que figuran en el ejemplo permiten configurar los bits directamente, pero se desaconseja su utilización. Para utilizar el método moderno de configuración, se deben traducir las macros a la configuración correcta en la herramienta de generación de la siguiente manera:



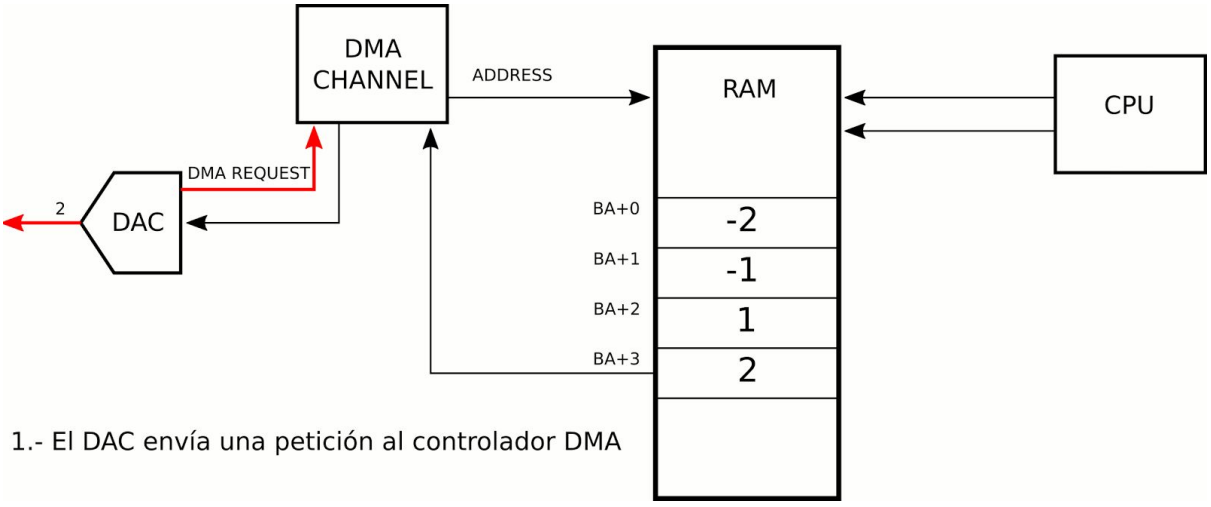
Luego tomar la rutina de configuración de PLL del ejemplo y colocarla en una función dentro de system.c, que puede ser la misma función que ya existe para la configuración del oscilador.
Por último calcular el valor a colocar en los registros PR2 y PR3 del Timer para obtener una interrupción exactamente cada 1 segundo, teniendo en cuenta que con el nuevo reloj de sistema el Timer se incrementa en uno cada 1/40e6 seg y que contará hasta que la cuenta llegue al valor de 32 bits formado por la concatenación de los registros PR3 y PR2.

3. Uso del DAC con interrupciones y DMA [3]

El controlador de acceso directo a memoria es un bloque que permite el intercambio de datos entre memoria y periféricos, sin asistencia del CPU. Una vez configurado permite disminuir considerablemente la carga de trabajo sobre la unidad de procesamiento automatizando la transferencia de datos. Los detalles de operación del

[3] https://en.wikipedia.org/wiki/Direct_memory_access

controlador de acceso directo a la memoria se pueden ver en Section 38. Direct Memory Access (DMA) (Part III) - dsPIC33F/PIC24H FRM.



3.1. Crear un programa que permita escribir dos valores alternados a través del DAC de audio. Para ello crear una tarea en user.c que configure el DAC de acuerdo al ejemplo 33-1 de la página 11 en el manual del DAC: Section 33. Audio Digital-to-Analog Converter (DAC) - dsPIC33F FRM. Este ejercicio y los que siguen necesitan tener el PLL configurado para generar una frecuencia de oscilador F_{OSC} de 80MHz (F_{CY} 40MIPS).
Revisar cada una de las siguientes configuraciones y modificar el ejemplo de ser necesario:
- Interrupción si la fifo está vacía. (ver campos del registro DAC1STAT)
- Salida del canal izquierdo y salida del canal derecho habilitadas. (ver campos del registro DAC1STAT)
- Frecuencia de trabajo del DAC de 39062.5. Tener en cuenta que la frecuencia de entrada para el DAC, como figura en el manual del oscilador, es la frecuencia denominada F_{VCO} que tiene un valor de 160MHz en nuestra configuración de PLL y es dividida por 256 por el hardware del DAC. Por lo tanto el valor resultante debe dividirse por el contenido del campo APSTSCLR del registro ACLKCON (ver manual del oscilador) y luego por el contenido del campo DACFDIV del registro DAC1CON más 1 (DACFDIV+1), de manera que las divisiones resulten en la frecuencia esperada.
- Formato de datos signado. Ver campo FORM en el registro DAC1CON.
- Valor por defecto en 0. Ver registro DAC1DFLT
Luego, en la rutina de interrupción se deberá escribir alternadamente a los registros DAC1RDAT y DAC1LDAT los valores 0x7FFF (máximo positivo) y 0x8000 (máximo negativo), de manera que las dos salidas del DAC alternen entre el voltaje máximo y mínimo. Se recomienda el uso de una variable global para llevar el estado de la salida, que deberá ser definida en el archivo *main.c* e inicializada en la función main de dicho archivo. Esta variable global podrá ser

accedida desde el archivo interrupts.c utilizando una declaración de variable externa, del siguiente modo:

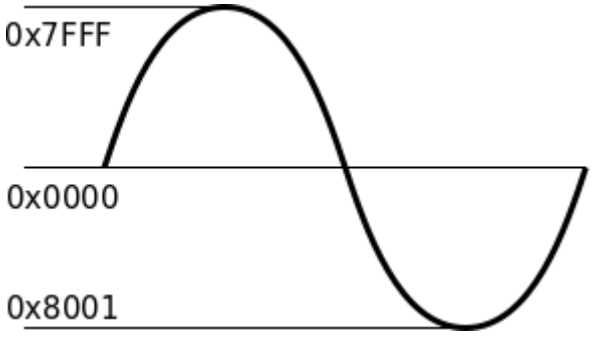
```
extern uint16_t mi_variable; // Colocar esta línea en el archivo
                             // interrupts.c
```

Chequear finalmente la salida del DAC con un osciloscopio. Identificar los pines de salida mediante el manual de referencia de la placa dsPIC.

3.2. Crear un programa que genere una salida senoidal de 1220Hz mediante el DAC, utilizando acceso directo a memoria (DMA). Ver para ello el ejemplo 33-2 (página 14) del manual de referencia del DAC y modificar los mismos registros que fueron modificados en el punto anterior.
Las primeras líneas del ejemplo declaran la memoria DMA que se utilizará para almacenar la señal senoidal a transmitir, es conveniente que estas líneas se declaren en el archivo *main.c* y luego se utilicen desde user.c mediante la palabra clave `extern`.
El uso del atributo `__attribute__((space(dma)))` le indica al linker que ubique la memoria en la zona de memoria DMA válida. Esta memoria puede cargarse con valores de la manera en que se inicializa un arreglo en c, mediante el uso corchetes en la declaración del arreglo, por ejemplo:

```
int mi_arreglo[4]={0x0000,0x18F8,0x30FB,0x471C};
```

Para obtener los elementos del arreglo, se deberá crear una senoidal en Octave/MATLAB de frecuencia digital 1/16, con un ciclo de duración, y cuyos valores deberán mapearse a enteros de la siguiente manera:



Una manera de realizar lo anterior es escalando la señal entre +-32767 (+-2^15-1), luego aplicar la operación **floor** para convertir a entero y por último sumar 65536 (2^16) sólo a los números que sean negativos. Los pasos para cargar el seno los podemos listar así:
1- Crear un vector con las muestras necesarias para nuestro seno de 1 ciclo
2- Convertir a número entero con el procedimiento descrito anteriormente
3- Crear una cadena de caracteres que comience en llave e incorpore todos los

elementos enteros separados por coma. Cada elemento se deberá convertir a hexadecimal con la función **dec2hex** y se le deberá anteponer la cadena '0x'. Se pueden concatenar caracteres/cadenas con el operador de corchete o la función **strcat**.
4- Copiar el código y pegarlo como inicialización del vector en el MPLAB

4. Uso del ADC mediante DMA

4.1. Crear un programa que lea en conversor AD y sobreescriba uno de los buffers DMA del DAC (Por ejemplo el buffer asociado al canal izquierdo) . En primer lugar se debe configurar el canal DMA para el ADC, siguiendo estos pasos:
Tomar la configuración DMA del ejercicio 3.2, aplicarla al canal DMA 2 y modificar los siguientes registros:
- DMA2PAD → Establecer el valor definido para el ADC en la tabla 38-1 del manual DMA: Section 38. Direct Memory Access (DMA) (Part III) - dsPIC33F/PIC24H FRM . También se puede tomar la dirección de ADC1BUF0.
- DMA2REQ → De la misma tabla tomar el valor para ADC1 Convert Done
- DMA2STA → Apuntar al buffer derecho B
- DMA2STB → Apuntar al buffer derecho A
También se deberá cambiar la dirección de la transacción DMA para que sea de periférico a RAM. Ver campo DIR en registro DMA2CON
Luego se debe configurar el ADC, siguiendo el manual Section 16. Analog-to-Digital Converter (ADC) - dsPIC33F/PIC24H FRM de la siguiente manera:
- ADC en modo 12 bits, ver campo AD12B en el registro AD1CON1
- Vdd y Vss como referencia de voltaje. Ver campo VCFG en registro AD1CON2
- Tcy como clock de conversión. Ver campo ADRC en registro AD1CON3
- Divisor de clock en 10 (Tad: 10xTcy). Ver campo ADCS en registro AD1CON3
- Poner todos los bits del registro AD1PCFGL como entradas digitales excepto el bit correspondiente a AN0, que deberá configurarse como entrada analógica
- Muestreo únicamente en canal cero (CH0). Ver campo CHPS en el registro AD1CON2bits
- Entrada positiva en AN0. Ver campo CH0SA en registro AD1CHS0
- Habilitar muestreo automático. Ver campo ASAM en registro AD1CON1
- Tipo de datos fraccional con signo. Ver campo FORM en registro AD1CON1
- Un incremento por muestra. Ver campo SMPI en registro AD1CON2
- Timer 5 como temporizador de muestro. Ver campo SSRC en registro AD1CON1
- Período de muestreo: 5xTad. Ver campo SAMC en registro AD1CON3
Como el muestreo se generará a partir del Timer 5, se deberá configurar este para generar una interrupción cada 39062,5Hz. No es necesario habilitar interrupciones pues solo se usará para comandar al ADC. Se puede usar el

código de configuración de timer de 32 bits del punto 2.1 y modificarlo para trabajar en 16 bits.
- Encender el ADC con el campo ADON del registro AD1CON1
Para comprobar el funcionamiento del ADC, conectar la salida del DAC cuyo buffer NO haya sido modificado (de manera que continúe generando la señal senoidal) y observar la salida en el pin del canal DAC cuyo buffer está siendo sobrescrito por el ADC. Se debería ver la misma senoidal, con una amplitud menor. Al cortar la conexión entre el ADC y el DAC, la señal debería interrumpirse (conectar la entrada del ADC a masa).

5. Uso de la UART mediante interrupciones

5.1. Crear un programa que permita recibir un carácter del puerto serie y reenviarlo como eco, a través del MCP2200. Comenzar tomando el ejemplo 7-1 (página 25) del manual de referencia de la UART: dsPIC33/PI4 FRM, UART. Modificar este ejemplo de manera que las interrupciones se generan a partir de caracteres recibidos (RX). Antes de habilitar la UART, activar la interrupción poniendo en 1 el bit U1RXIE del registro IEC0. Luego habilitar la transmisión UART poniendo en 1 el bit UTXEN del registro U1STA. Por último habilitar la UARTX (como en el ejemplo).
En la rutina de interrupción por recepción se debe leer el caracter recibido y escribirlo en el buffer UART de TX. También se deberá conmutar el estado del Led 3 de manera que cada vez que llegue un carácter cambie de estado. Para poder utilizar la conexión serie disponible en la placa se deberá configurar el multiplexor de selección de pines para que la salida TX sea mapeada al puerto RP19, configurable desde el registro RPOR9 y la entrada RX alpuerto RP20, configurable desde el registro RPINR18. Ver los detalles desde el manual de configuración de pines remapeables: Section 30. I/O Ports with Peripheral Pin Select (PPS) - dsPIC33F/PIC24H FRM.
Modificación para placa nueva:
Salida TX al pin RP6
Entrada RX al pin RP5
La comunicación se podrá probar con cualquier programa de acceso al puerto serie disponible.
Ver en la página http://www.microchip.com/wwwproducts/en/en546923 información relacionada al driver para Linux/Windows en la sección Documentation.