Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Generador de disparos configurable

Autor:

Nozal Fernández, Sara

Tutor:

Diez Muñoz, Pedro Luis
Tecnología Electrónica

Valladolid, Abril 2018

Resumen

EL objetivo del presente proyecto es la mejora de un sistema de generación de pulsos para metrología. El sistema actual está implementado en un ordenador personal que a través de la generación de pulsos con distintos intervalos de tiempo es capaz de comprobar la fiabilidad y precisión del instrumento de medida y verificar. Para mejorar la velocidad y versatilidad del sistema y reducir en tamaño, se ha optado por construir un sistema embebido basado en un microcontrolador del tipo PIC24FJ de 16 bit, capaz de generar distintos pulsos pudiendo el usuario variar la frecuencia a través de un oscilador externo.

Palabras clave

Interfaz de usuario

Microcontrolador

Oscilador

Puerto Serie

Señal de reloj

Agradecimientos

En primer lugar, quiero agradecérselo a mis padres y a mi hermana, sin su esfuerzo y apoyo esto no habría sido posible. A mi tía, a mi abuela, a mis primas y a mi familia en general, que han creído incondicionalmente en mí y me han animado a seguir adelante y no darme por vencida

En segundo lugar, quiero agradecérselo a mi tutor Pedro Luis Diez, que me ha cedido su tiempo y conocimiento para que la realización de este trabajo fin de grado haya sido posible.

En tercer lugar, imposible olvidar a todos los amigos que he conocido durante estos años de carrera, Ana, Sandra, Nuria, Marina, Rodri, Rubén… con los que he crecido como persona y con los comparto experiencias inolvidables.

Y, por último, a ellas, "Las ponchis". Esa segunda familia que siempre han sido incondicionales, siendo las primeras en alegrarse y en estar en los momentos importantes.

Índice

Índice de Figuras

1. Introducción y objetivos

EL objetivo del presente proyecto es la mejora de un sistema de generación de pulsos para metrología.

El sistema que actualmente están usando está implementado en un ordenador personal, que a través de la generación de pulsos con distintos intervalos de tiempo es capaz de comprobar la fiabilidad y precisión del instrumento de medida. Se quiere implementar mejoras desde el punto de vista de la rapidez, espacio y sencillez. Para ello, se ha construido un sistema embebido basado en un microcontrolador del tipo PIC24FJ de 16 bit, capaz de generar varios pulsos pudiendo el usuario variar la frecuencia. Esto ha conllevado una serie de estudios y actuaciones previos:

- Evaluación de la posibilidad de migrar a un nuevo entorno de desarrollo o IDE, basado en software libre, en lugar del actual sistema en uso.
- Estudio del modo de funcionamiento y lenguaje de los microcontroladores.
- Diseño y prueba de las funciones que permiten controlar los periféricos, así como de las tareas principales del sistema.
- Evaluación de las posibilidades de comunicación con la interfaz del operador.

Estos cuatro puntos engrosarían el objetivo global del proyecto.

Los microcontroladores de MICROCHIP utilizan el entorno de desarrollo MPLAB, el cual puede ser empleado para uso docente ya que es una herramienta de uso libre. Está basado en lenguaje C y ensamblador, lo que es una ventaja ya que proporciona cierta familiaridad a los estudiantes al haber cursado asignaturas de C y C++.

El uso de un PIC es muy versátil, ya que, aparte de tener numerosas librerías permite la adhesión de multitud de periféricos. Por otro lado, cuenta con Kits de aprendizaje y desarrollo como puede ser la Explorer 16 Development Board, con la que se inició este proyecto.

2. Recursos de Hardware y Software

2.1. MPLAB X IDE

2.1.1. CARACTERÍSTICAS

MPLAB X IDE es un entorno creado por Microchip que permite diseñar y desarrollar programas en C y ensamblador para los microcontroladores embebidos de esta marca. Se caracteriza por ser un entorno de desarrollo de software libre, que puede ser empleado en Windows, MAC OS y Linux. Realizar tareas de compilación, ensamblaje para convertir el código diseñado a código máquina, el cual será grabado posteriormente en la placa. Además, implementa una herramienta de depuración que permite ver el valor que adoptan las variables a lo largo de la ejecución del programa, para mejorarlo y/o eliminar errores. En este proyecto se ha empleado la versión 3.65.

A continuación, se muestra la interfaz del programa:

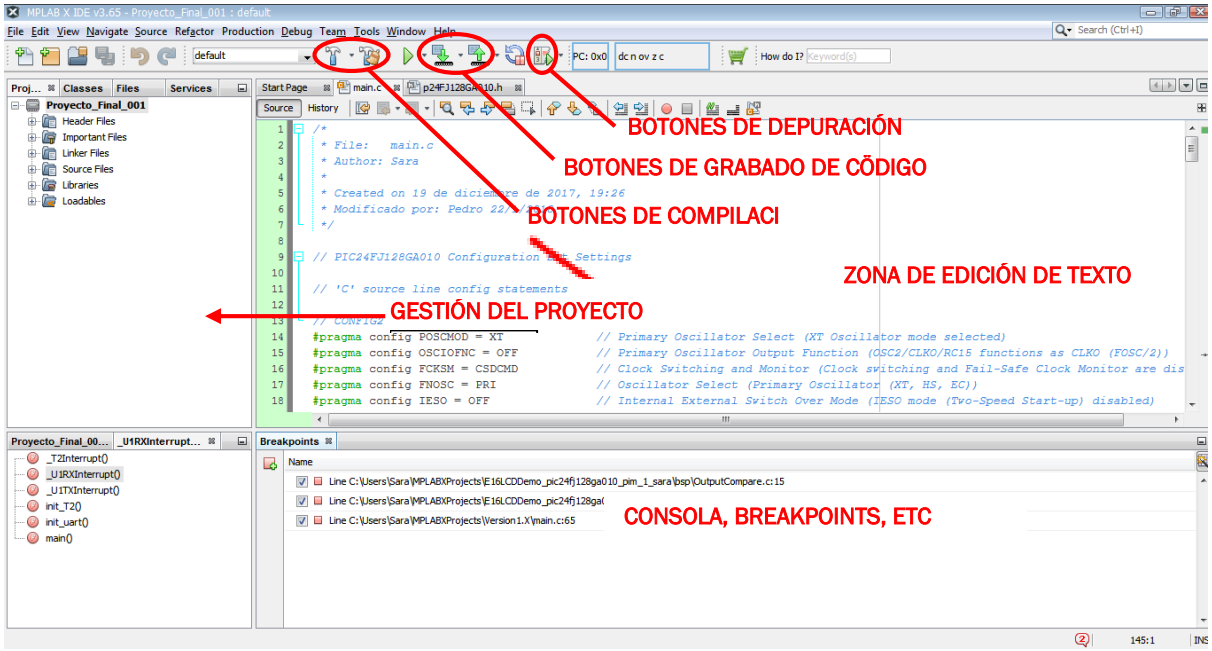


Figura 1. Interfaz MPLAB X IDE

Como se puede ver, se trata de una interfaz que resulta accesible y limpia para el usuario, con una zona donde dónde se muestra la gestión de contenidos del proyecto, otra para la edición del código y una zona inferior de la cual se obtiene información de los problemas, búsquedas o salidas de la consola, así como variables de configuración y "**breakpoints**". En la parte superior, se encuentran los botones de acceso directo para la compilación, depuración y grabado de código.

Todos los elementos de la pantalla son personalizables según el gusto y necesidades del usuario.

En cuanto al modo de depuración se pude comprobar que la interfaz también se puede configurar para un uso más cómodo, permitiendo controlar en cada momento lo que deseé ver el usuario: variables, "**breakpoints**", etc.

A continuación, se muestra la interfaz de depuración:

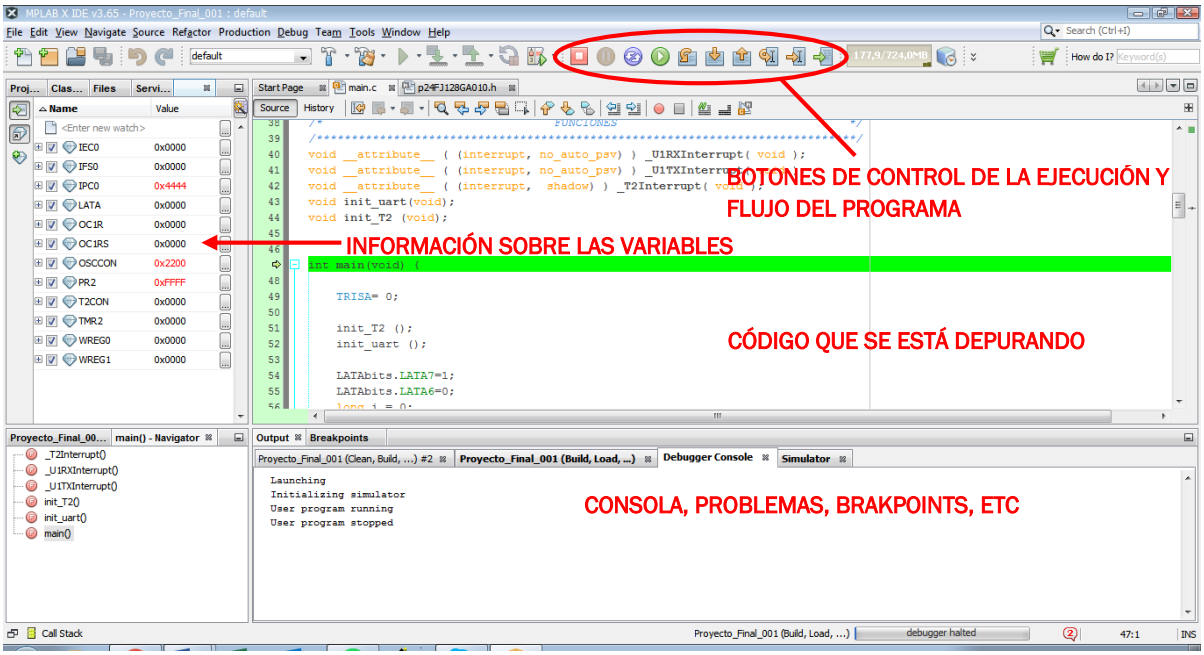


Figura 2. Interfaz de depuración de MPLAB X IDE

2.1.2. VENTAJAS E INCONVENIENTES

MPLAB X IDE tiene un gran número de puntos a favor:

- Interfaz amigable y accesible para el usuario.
- Opción de autocompletar el código.
- Sencillez para el manejo de varios proyectos a la vez.
- Acceder a librerías o definiciones con CTRL+Click.
- Gran versatilidad, soporta varías configuraciones para un proyecto.
- Puede formatear el código.
- Buena gestión de proyectos. Cuenta con la creación automatizada de plantillas.

Sin embargo, posee algunos inconvenientes:

- Ocupa bastante espacio lo que lo vuelve un poco pesado.
- Número de "**breakpoints**" limitado.

2.2. PLACA DE E/S

2.2.1. CARACTERÍSTICAS

La Explorer 16 Development Board es la placa de bajo costo sobre la que se desarrolla este proyecto. Se emplea usualmente para el aprendizaje con microcontroladores, ya que es muy versátil y se encuentran muchos ejemplos de código con los que puede iniciarse el estudiante.

Esta placa ha sido diseñada para adaptarse tanto a los montados de forma permanente (es decir, soldado sobre) y procesadores PIM separables. Con el control deslizante, S2, se puede seleccionar el modo de funcionamiento que se desea. Esto hace posible que la placa Explorer 16 sea compatible con la mayoría de PIM de 100 patillas. En la siguiente figura se muestra el esquema sobre el que se monta el PIC deseado:

En este caso se ha empleado el PIC24FJ128GA010, no soldado, y con S2 desplazado hacia el lado PIM.

A la hora de suministrar energía a la placa, se puede optar por una fuente de alimentación que proporcione de 9V a 15V a J12, o una fuente que proporcione 5V y 3.3V conectados a los terminales apropiados.

Para comprobar que se está suministrando 3.3V la placa dispone de un LED verde (D1) como indicador.

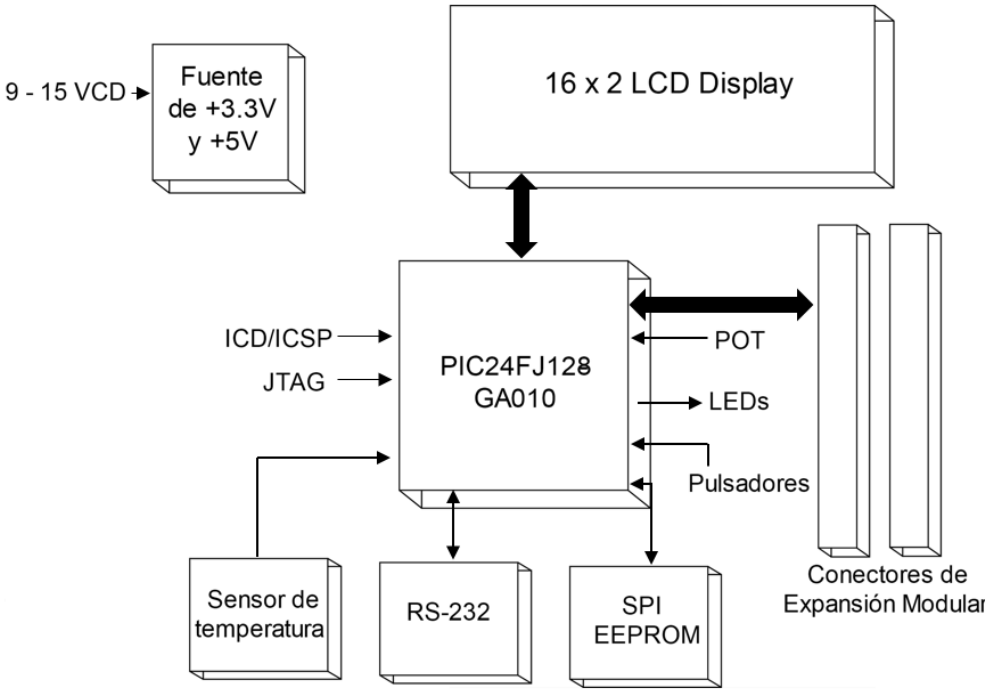


Figura 3. Esquema modular de la placa Explorer 16

2.2.2. PERIFÉRICOS

La placa dispone de controladores a 3V para comunicarse con dispositivos periféricos de 5V, entre los cuales cuenta con:

- DISPLAY LCD

Pantalla LCD alfanumérica que incluye dos líneas de 16 caracteres cada una. Se maneja con ocho líneas de datos y tres líneas de control, y se puede controlar tanto por el módulo PMP como por puerto de E/S. En caso de que se requiera hacer uso de diferentes terminales de E/S se deben cortar y crear nuevos puentes según como se muestra en el siguiente esquema:

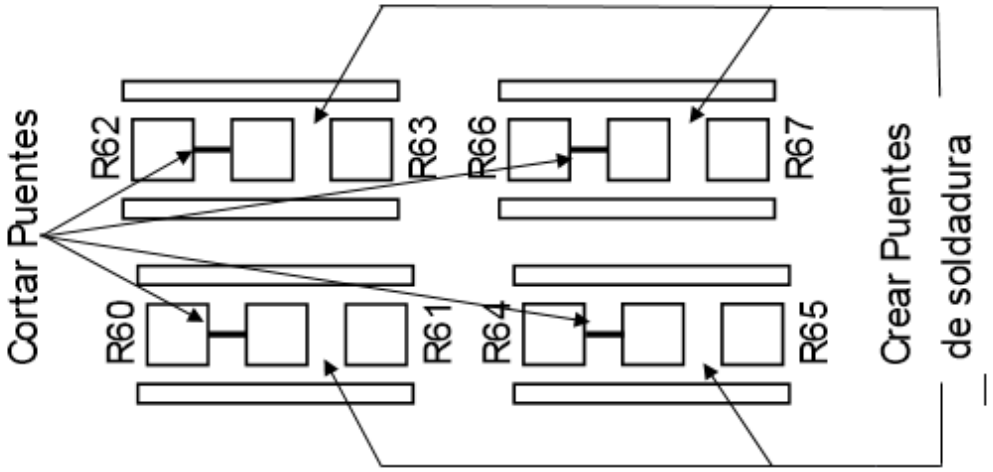


Figura 4. Esquema de conexión del Display LCD

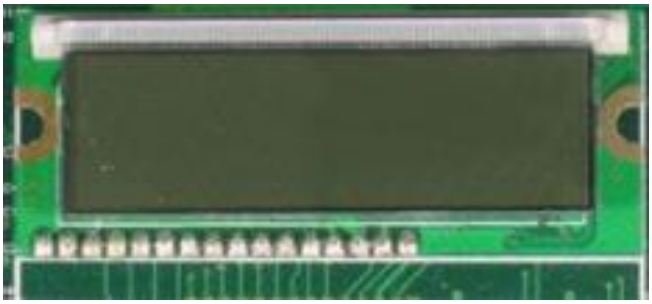


Figura 5. Interfaz física del Display LCD

- INDICADORES LED'S

La placa dispone de ocho LED's (D3 a D10). Para iluminarlos se debe configurar los correspondientes terminales de salida en alto nivel. Si se desea desactivar su funcionamiento se debe quitar el puente JP2.

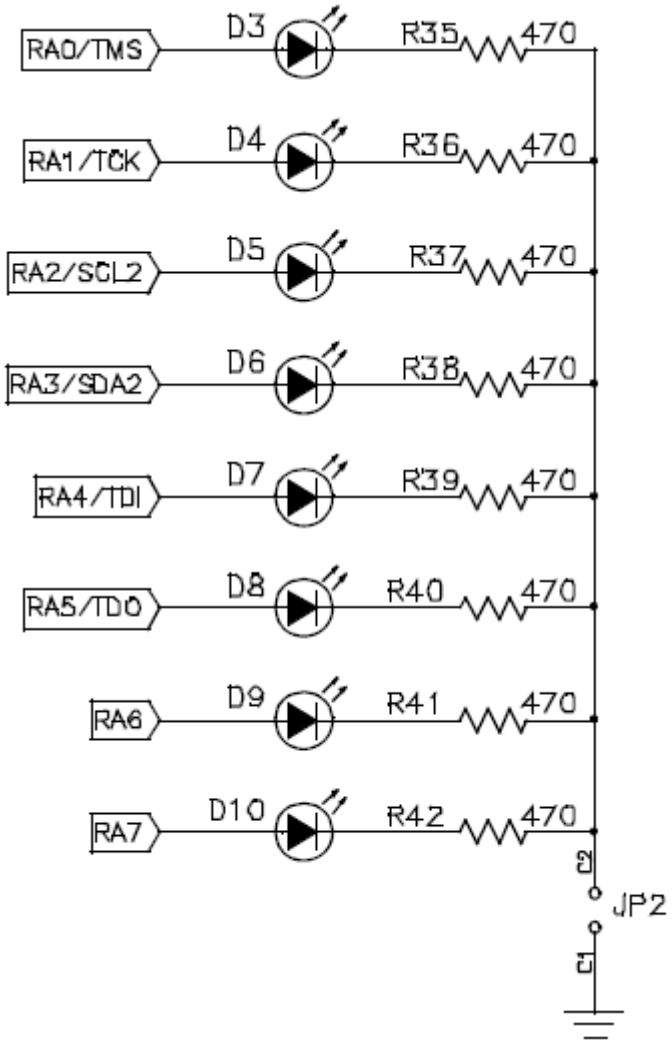


Figura 6. Esquema eléctrico de los indicadores LED's

Figura 7. Indicadoeres LED's de la placa Explorer 16

- **PULSADORES**

Se cuentan con cinco pulsadores de botón, de los cuales, el S1 está programado para restablecer el procesador, función MCLR. Todos ellos se activan a nivel bajo, es decir, que cuando se pulsan se ponen a tierra mientras que el resto de tiempo se encuentran a +3.3V.

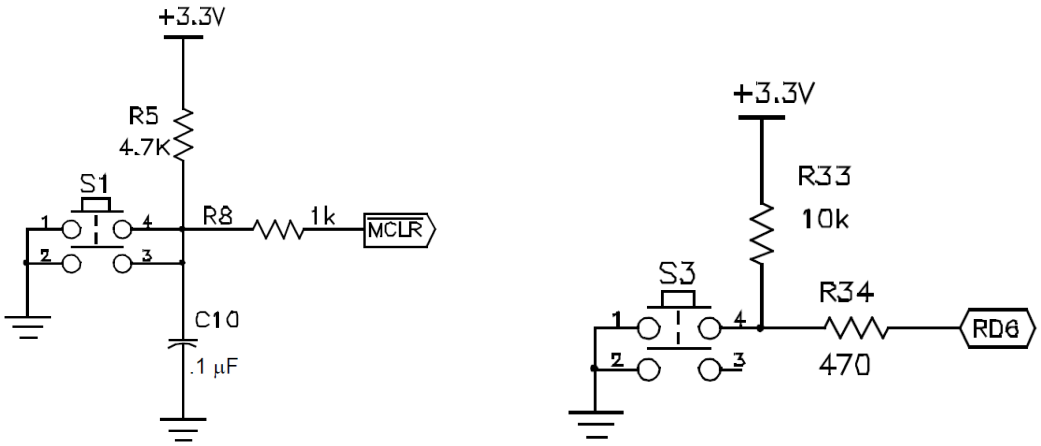


Figura 8. Esquema eléctrico de los pulsadores

Figura 9. Pulsadores de la placa Explorer 16

- PUERTO SERIE RS-232

Se puede hacer uso del puerto serie RS-232 con control de flujo de hardware a través del conector DB9. Está configurado como un dispositivo DCE, es decir, como un mecanismo que está conectado al controlador de la comunicación serie (dispositivo DTE). A su vez, se puede conectar con el PC a través de un cable directo. Sus patillas de lectura y escritura, RX y TX, están conectados a las líneas RX y TX de U3.

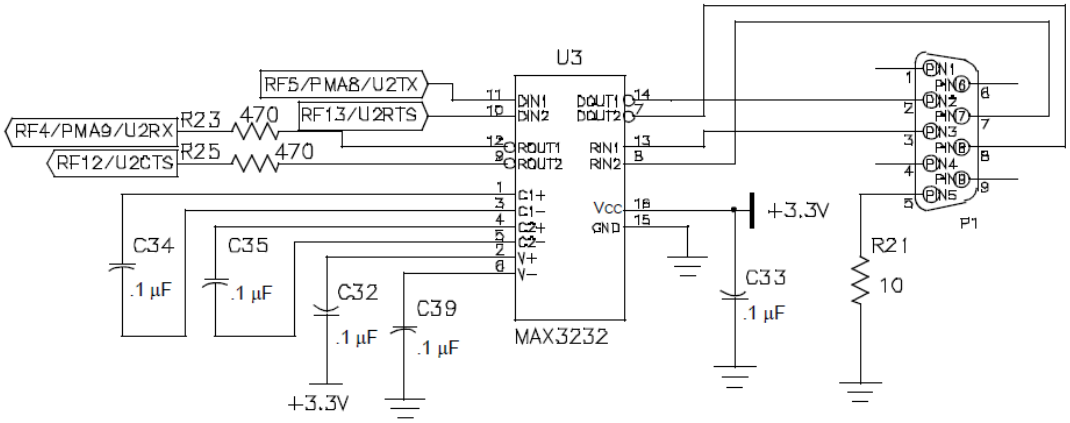


Figura 10. Esquema eléctrico del Puerto Serie RS-232

- PUERTO ICD

Este conector emplea los terminales de puerto RB6 y RB7 para la depuración de bajo coste a través del conector modular JP1.

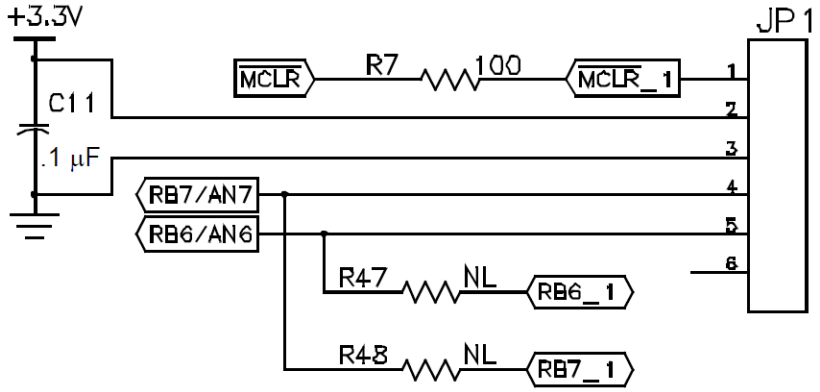


Figura 11. Esquema eléctrico del Puerto ICD

- INTERFAZ PARA PROGRAMADOR PICkit 3

El conector J14 proporciona una interfaz de 6 patillas para el programador de la casa PICkit. Esta herramienta permite depurar y programar el microcontrolador.

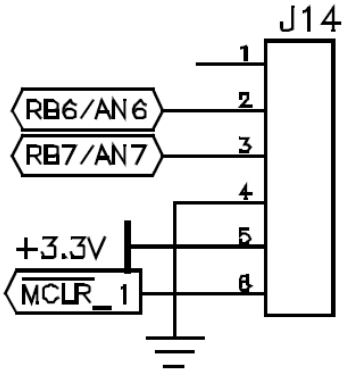


Figura 12. Esquema eléctrico de la Interfaz para PICkit 3

Figura 13. PICkit 3

2.3. PIC24FJ128GA010

2.3.1. CARACTERÍSTICAS

La familia PIC24FJ128GA engloba microcontroladores de 16 bits diseñados por Microchip. Debido a sus buenas prestaciones y bajo consumo están orientado a dispositivos portátiles. Además, sus periféricos especializados y software de conectividad como USB, CAN, hacen que sea fácil comunicarse con otros sistemas.

La comunidad de desarrolladores de aplicaciones con microcontroladores de la marca Microchip es muy grande, lo que posibilita que haya un enorme soporte a nivel de usuarios que facilita el inicio con estos sistemas.

A nivel de CPU, el set de instrucciones del PIC24FJ128GA010 que se emplea para desarrollar las aplicaciones es de 24 bits y de 16 bits para datos. Cuenta con 16 registros que pueden actuar como datos, direcciones o direcciones de desplazamiento. Su memoria flash es de 128Kbytes y posee 100 patillas de los cuales 87 que pueden ser del tipo I/O en 7 puertos. Cuenta con sistema de interrupciones y 7 niveles de prioridad configurables para controlarlas.

La placa de desarrollo utilizada para este proyecto, la Explorer 16, emplea esta versión del microcontrolador, que está especialmente diseñada para la emulación y depuración sobre la misma.

El PIC24FJ128GA010 cuenta con los siguientes periféricos:

- I2C: Dos interfaces de comunicación serie síncrona semi-duplex.
- SPI: Dos interfaces de comunicación serie síncrona full-duplex.
- UART: Dos interfaces de comunicación serie asíncrona.
- PMP: Puerto maestro paralelo.

- CRC: Redundancia cíclica programable.
- TEMPORIZADOR: Cinco temporizadores de propósito general.
- PWM: Generación de señales con modulación PWM.
- WDT: "**Watchdog Timer**", se encarga de reiniciar el microcontrolador en caso de que este se mantenga en un estado de error durante un cierto tiempo.
- RTCC: Reloj de tiempo real (aplicaciones de hora, fecha, etc).
- ADC: Conversor analógico-digital de alta velocidad.

2.3.2. FUNCIONES PRINCIPALES

A continuación, se va a explicar las principales funciones que se han implementado a la hora de configurar el microcontrolador del presente trabajo.

PALABRAS DE CONFIGURACIÓN (CONFIGURATION WORD)

En los dispositivos de la familia PIC24FJ128GA010, las dos palabras principales de la memoria del programa en el chip están reservadas para la información de configuración.

En la memoria del programa las palabras de configuración están integradas bajo un formato compacto. Pero en la configuración real, estos bits se asignan en varios registros diferentes en el espacio de memoria de configuración. Se pueden programar (nivel lógico 0) o dejar sin programar (nivel lógico 1) para seleccionar varias configuraciones de dispositivos. Su configuración, se implementan como memoria volátil, esto significa que los datos de configuración se deben programar cada vez el dispositivo se enciende. En cambio, para los reinicios ("**resets**"), se cargan automáticamente desde las Palabras de Configuración a los registros de Configuración adecuados.

A continuación, se muestra una lista de los registros que se implementan en cada una de las palabras.

Palabra de configuración I

Esta palabra incluye los siguientes registros:

JTAGEN: bit de habilitación de puerto JTAG.

GCP: bit de protección del código de memoria del programa. Cuando se configura como un 0 implica que todo el código del programa se encuentra protegido en caso de querer ser modificado por una persona no autorizada.

GWRP: Bit de protección de escritura flash del código. Si se configura como un 0 no está permitido la escritura en la memoria volátil, lo que implica que no se podría cambiar la configuración de estas palabras.

DEBUG: bit de habilitación del depurador.

ICS: bit de selección del depurador. Indica que patillas emplea el depurador, EMUC1/EMUD1 o EMUC2/EMUD2, correspondientes a entradas/salidas de reloj o datos.

FWDTEN: bit de habilitación del temporizador de vigilancia o "**watchdog**".

WINDIS: bit de desactivación del temporizador de vigilancia o "**watchdog**" por ventana. Para que la opción de ver por ventana este temporizador esté activada, debe haberse activador FWDTEN.

FWPSA: bit de selección del ratio de pre-escala que usará WDT (temporizador de vigilancia).

WDTPS: bit de selección de la post-escala que empleará el temporizador de vigilancia WDT.

Palabra de configuración II

IESO: bit de conmutación externa interna.

FNOSC: bits de selección del oscilador inicial, ya sea primario, secundario, FRC…

FCKSM: Cambio de reloj y bits de configuración del monitor de reloj a prueba de errores.

Están disponibles todas las opciones: que no se pueda cambiar de reloj ni que esté activado el monitor de control de errores del reloj; que se pueda cambiar de reloj, pero que no esté activado el monitor de control de errores; y al contrario, que el monitor esté activado pero no el cambio de reloj.

OSCIOFCN: bit de configuración del terminal OSC2.

POSCMD: bits de configuración del oscilador primario. Se puede elegir entre XT, EC o HS.

OSCILADOR

A continuación, se describen las principales características y modos de funcionamiento para el oscilador:

- Dispone de cuatro opciones de funcionamiento como fuente de reloj para oscilador externo e interno, dando lugar a once modos de trabajo diferente.

- PLL para aumentar la frecuencia de la fuente de reloj externa o interna, o para proporcionar mayor precisión de reloj a los periféricos.
- Posibilidad de conmutar, vía software, de forma controlada varias fuentes de reloj.
- Posibilidad de seleccionar, vía software, una post-escala para relojes seleccionados.
- Un FSCM, monitor de reloj a prueba de fallos, que detecta fallos en el reloj y permite el apagado y recuperación seguros del sistema.
- Generador de reloj programable basado en una señal de referencia. Permite disponer de un mayor rango de frecuencias para sincronizar dispositivos periféricos.

A continuación, se observa un esquema del oscilador del PIC24:

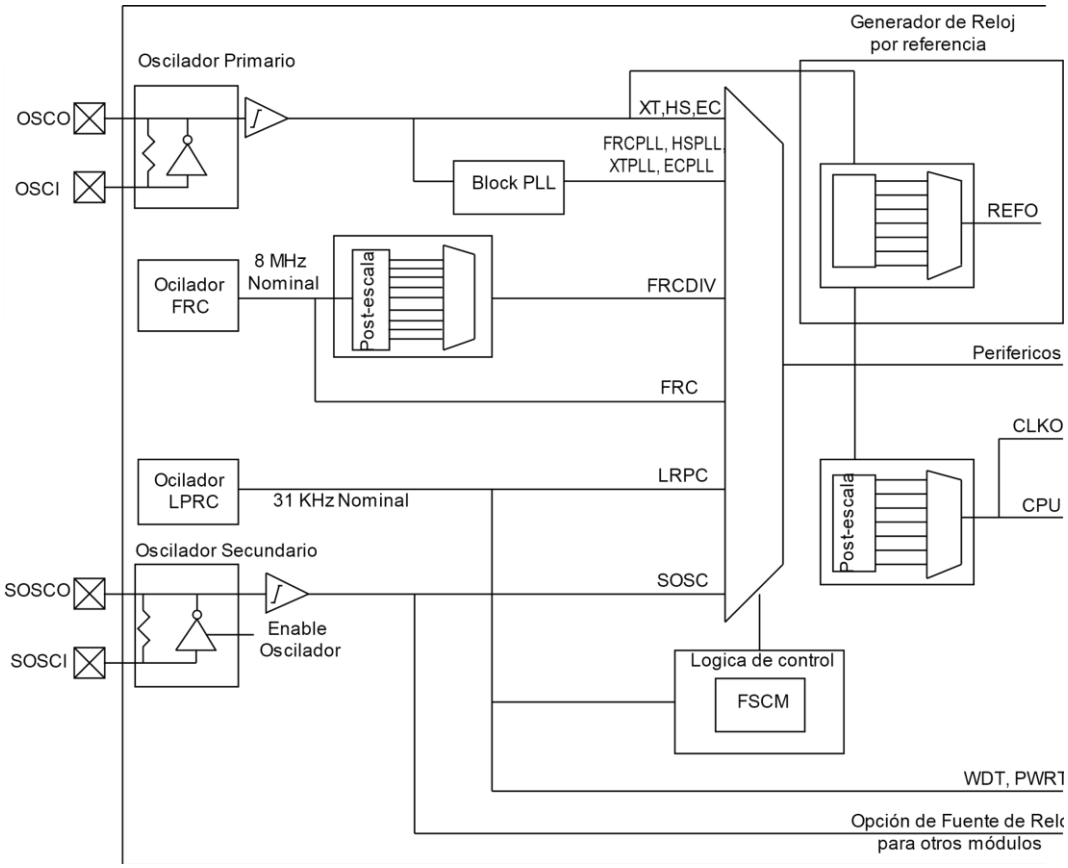


Figura 14. Esquema de Oscilador para PIC24

Registros principales del Oscilador

El funcionamiento del oscilador está controlado por tres registros de función (SFR):

- OSCCON
- CLKDIV
- OSCTUN

Registro de Control del Oscilador (OSCCON):

El registro OSCCON es el registro de control primordial del oscilador. Controla la conmutación y permite la monitorización de las fuentes de reloj. Sus principales bits son:

Los bits de estado COSC: son bits de solo lectura que indican que tipo de fuente de oscilador está actuando para que el dispositivo esté en funcionamiento. Si surge una operación de cambio de reloj, estos bits de estado cambiarán para indicar la nueva fuente que emplea el oscilador.

Los bits de estado de NOSC: se encargan de seleccionar la nueva fuente de reloj para la siguiente operación de cambio de reloj. Para las patillas de reinicio "**Power–On Reset**" (POR) y de "**Master Clear**" (MCLR) estos bits seleccionan automáticamente la fuente del oscilador definida por la Configuración FNOSC (Palabra de configuración).

El bit de estado LOCK: es de solo lectura e indica el estado del circuito PLL. Se activa cuando el PLL logra un bloqueo de frecuencia y se desactiva cuando se inicia una secuencia de cambio de reloj válida. Si la fuente de reloj que se encuentra en funcionamiento no emplea el PLL, su estado se programa a 0.

El bit de estado de CF: es un bit de estado de lectura / eliminable que indica cuando se produce un fallo del reloj, programándose a un nivel lógico 1. Se restablece cada vez que se produce un cambio de reloj válido.

El bit de control OSWEN: se utiliza para iniciar una operación de cambio de reloj. Este bit se borra automáticamente después de un cambio de reloj exitoso, si se ha completado un cambio de tipo de reloj a FRC o después de cualquier interrupción de reloj.

Registro de divisor de reloj (CLKDIV)

El registro del divisor de reloj controla las funciones asociadas con el "Modo Doze" (es un modo de ahorro de energía), así como la post-escala del oscilador FRC. Sus principales bits de configuración son:

Los bits CPDIV: permiten realizar una escala en la fuente de reloj en caso de que se requiera que vaya a menor velocidad.

El bit ROI: permite que una interrupción salga del modo Doze y selecciona automáticamente una Relación 1:1 para el procesador y los relojes

periféricos. Este bit se borra después de la salida del modo Doze, e implica que desde ese momento las interrupciones no afectan a ese modo.

Los bits DOZE: seleccionan la relación entre los relojes del procesador y los relojes periféricos. El rango es seleccionable por software entre 1:1 a 1:128. En los terminales de "**Master Clear**" (MCLR) y de reinicio (POR) por defecto la relación es de 1:1. Esta característica permite que la CPU consuma menos energía sin interrumpir las operaciones de los periféricos.

El bit DOZEN: Al programarlo como un 1, coloca el dispositivo en el "Modo Doze" y activa la post-escala del reloj del procesador. Este bit se borra cuando se activa el bit de ROI y se produce una interrupción.

Los bits RCDIV: seleccionan la opción de post-escala para la salida del oscilador FRC, lo que permite a los usuarios elegir una frecuencia de reloj inferior a los 8 MHz nominales.

Registro de sintonización del oscilador (OSCTUN)

El registro de sintonización del oscilador FRC permite al usuario ajustar con precisión el oscilador FRC.

La respuesta de sintonización del oscilador FRC puede que no sea monótona o lineal; La siguiente frecuencia más cercana, puede ser compensada por una cantidad de pasos. Se recomienda que se prueben múltiples valores de OSCTUN para encontrar el valor más cercano a la frecuencia deseada.

Tipos de Fuentes de Reloj

El PIC24 acepta diferentes formas de fuentes de reloj:

- Oscilador primario (POSC), asignado en los terminales OSC1 y OSC2. Pueden ser de cristal de cuarzo, resonador cerámico o circuito resistor-capacitor.
- Oscilador secundario (SOSC), asignado en los terminales SOSCI y SOSCO. Puede ser del mismo tipo que el anterior.
- Oscilador rápido interno RC (FRC).
- Oscilador RC de baja potencia interno (LPRC).

Las fuentes de oscilador primario y FRC tienen la opción de usar el 4x PLL interno. La frecuencia de la fuente de reloj FRC puede reducirse opcionalmente mediante el divisor de reloj programable.

La fuente de reloj seleccionada genera la fuente de reloj para el procesador y los periféricos. Esta fuente generada para el procesador está dividida por dos

para producir el reloj del ciclo de instrucción interno, FCY, también denominada FOSC / 2.

El siguiente diagrama de tiempo muestra la relación entre la fuente del reloj del procesador y la ejecución de la instrucción.

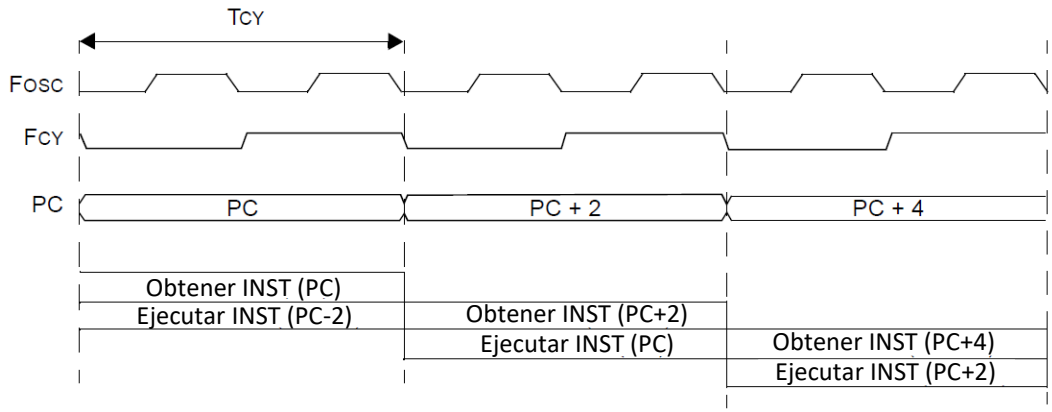


Figura 15. Diagrama de tiempos de instrucción

OSCILADOR PRIMARIO (POSC)

El oscilador primario se puede configurar para una entrada de reloj externo o para un cristal externo, por ello que sea un tipo de oscilador muy versátil. Para que pueda ser usado, debe conectarse a los terminales OSC1 y OSC2.

A continuación, se resumen los distintos modos de funcionamiento para este oscilador:

Modo EC: Fuente de reloj externa, con rango de frecuencia entre 0 - 32MHz. EL terminal OSC2 toma el valor del ciclo de instrucción, es decir, de la mitad de la frecuencia de oscilación (Fosc).

Modo ECPLL: Este modo se puede emplear si el PLL se encuentra en modo activo. Permite una entrada de reloj de 4 a 28 MHz. Y al igual que el anterior, el terminal OSC2 toma el valor del ciclo de instrucción, Fosc/2.

Modo HS: En este caso se hace uso de un cristal que actúa como oscilador, y se debe conectar a los terminales OSC1 y OSC2. Este modo proporciona unas frecuencias de entre 10 y 32 MHz.

Modo HSPLL: Se trata de un cristal que actúa como oscilador, con frecuencias de entre 10 y 32 MHz y se usa para el PLL.

Modo XT: Cristal con rango de frecuencia menor que los anteriores, entre 3.5 y 10 MHz, y el cual debe conectarse a los terminales OSC1 y OSC2.

Modo XTPLL: Similar al anterior con la diferencia de que se emplea cuando el bloque PLL está activado.

Los bits de configuración POSCMD y FNOSC seleccione el modo de funcionamiento del oscilador primario.

Los bits POSCMD seleccionan sub-modo particular que se utilizará (XT, HS o EC), mientras que los bits FNOSC determinan si el oscilador se usará solo o con el PLL interno.

Para que el PIC24F funcione con el modo oscilador primario los bits COSC deben estar configurados con '010' o '011'.

La principal diferencia entre los modos XT y HS es la ganancia del inversor interno del oscilador del circuito, que permite los diferentes rangos de frecuencia. El modo XT proporciona un rango de frecuencia media-baja, mientras que el modo HS proporciona las frecuencias más altas del oscilador con un cristal. La particularidad de estos dos modos es que poseen retroalimentación del cristal por parte de la patilla OSC2, con lo cual las variaciones en las frecuencias son menos acusadas. Por último, para estos casos se debe tener en cuenta que el diseño del oscilador PIC24F requiere el uso de un cristal de corte paralelo, ya que el uso de un cristal en serie puede dar lugar a una frecuencia fuera de las especificaciones del fabricante del cristal.

Los modos EC y HS que usan el circuito PLL proporcionan las frecuencias operativas más altas del dispositivo.

Si el oscilador primario está configurado para una entrada de reloj externa, EC o ECPLL, no se necesita del terminal OSC2 para cumplir la función del oscilador. Por el contrario, y como ventaja, en estos modos la patilla OSC2 se puede usar como un terminal de tipo entrada/salida del dispositivo o un terminal de salida del reloj. El uso de una u otra forma del terminal se determina en la palabra de configuración OSCIOFCN. En el caso de que se opte por usarlo como un terminal de salida de reloj, toma un valor de frecuencia de FOSC / 2.

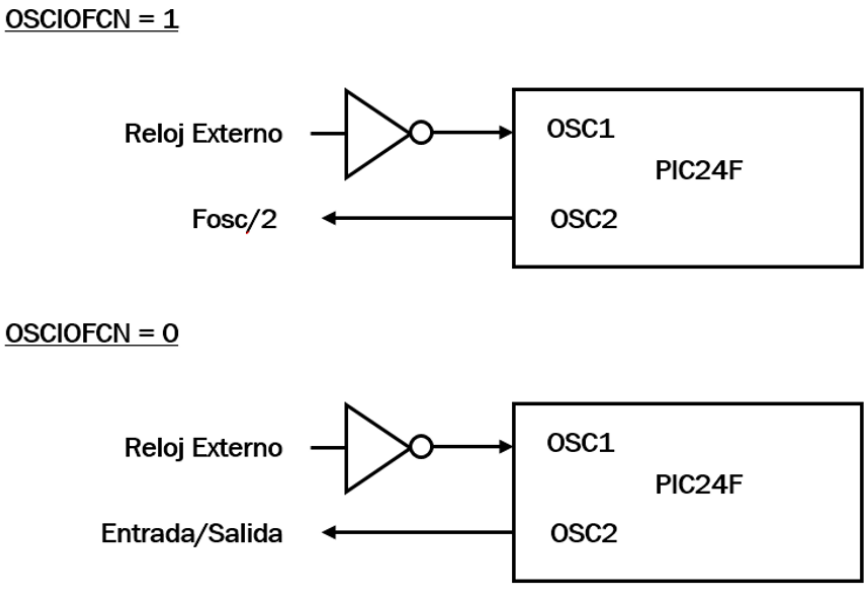


Figura 16. Esquema de Oscilador primario en función del valor del registo OSCIOFCN

Aspectos a tener en cuenta a la hora de usar este tipo de osciladores:

Una vez que el voltaje del dispositivo empiece a incrementarse desde VSS, el oscilador comenzará a realizar las oscilaciones. Este tiempo requerido para que el oscilador comience a oscilar depende de muchos factores, entre los que se encuentran:

- Frecuencia de cristal.
- Valores capacitivos empleados.
- Resistencia de serie, si se usa, y su valor y tipo.
- Tiempo de subida del dispositivo VDD.
- Temperatura del sistema.
- Ganancia del oscilador inversor interno.
- Calidad de cristal.
- Disposición del circuito del oscilador.
- Ruido en el sistema.

Hay que tener en cuenta, que el tiempo para que comience a oscilar de forma estable no es instantáneo, sino que requiere de cierto tiempo. A continuación, se muestra la gráfica que comúnmente siguen los osciladores:

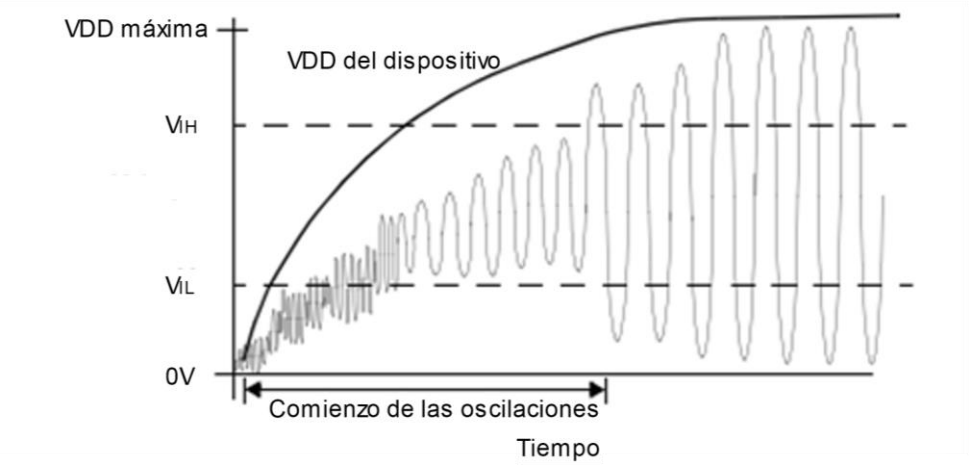


Figura 17. Gráfica de funcionamiento de un oscilador

OSCILADOR SECUNDARIO

El oscilador secundario de baja potencia (SOSC) se implementa, en la mayoría de los casos, para funcionar con un cristal de 32.768 kHz. El oscilador está ubicado en los terminales del dispositivo SOSCO y SOSCI y sirve como una fuente secundaria de reloj de cristal para realizar operaciones de baja potencia. Se utiliza sobre todo para controlar el "Timer1", reloj en tiempo real y calendario (RTCC), pero también para otros módulos que requieren una señal de reloj de baja potencia.

RELOJ INTERNO RÁPIDO (FRC)

El oscilador FRC es un oscilador RC interno rápido (8 MHz nominales). Este oscilador está pensado para proporcionar velocidades razonables de operación sin necesidad de emplear un cristal externo o un resonador cerámico. El PIC24F hace uso del oscilador FRC siempre que los bits COSC sean '111', '001' o '000'.

Dado que sirve como el reloj de referencia durante la inicialización del dispositivo, el oscilador FRC está siempre habilitado cuando se produce un reset. Después de configurar el dispositivo y expirar PWRT (temporizador de encendido), FRC permanece activo solo si se selecciona como fuente de reloj del dispositivo.

Uso de la post-escala en osciladores FRC.

Si se desea emplear un oscilador FRC como fuente de reloj del sistema no implica que deba quedarse anclado en los 8MHz nominales, sino que puede

hacerse uso del modo adicional, FRCDIV. Esta opción implementa una post-escala seleccionable que permite la elección de una frecuencia de reloj inferior, pudiendo elegir entre 7 opciones diferentes, más la salida directa de 8 MHz. La post-escala se configura utilizando los bits RCDIV. Suponiendo una salida nominal de 8 MHz, las opciones de frecuencia más bajas disponibles van desde 4 MHz (se aplica una división por 2) hasta 31 kHz (se aplica una división por 256). Esta versatilidad en el rango de frecuencias otorga la capacidad de ahorrar energía en cualquier momento en una aplicación simplemente cambiando los bits RCDIV.

El modo FRCDIV se selecciona siempre que los bits COSC sean '111'.

FRC y el bloque PLL

El modo FRCPLL se selecciona siempre que los bits COSC sean '001'. Además, este modo solo funciona cuando se seleccionan las opciones de post-escala de FRC directo (8 MHz) o dividido por 2 (4 MHz).

Para dispositivos con el bloque 4x PLL básico, la salida del bloque de post-escala del FRC también puede combinarse con el PLL para producir un reloj de sistema de 16 MHz o 32 MHz nominales. Tiene la desventaja de que es un poco menos preciso en frecuencia que el uso del oscilador primario con un cristal o resonador, pero como contrapunto, permite el funcionamiento a alta velocidad del dispositivo sin el uso de componentes externos del oscilador.

OSCILADOR INTERNO DE BAJA POTENCIA RC (LPRC)

Este oscilador es empleado para funciones específicas del sistema, como pueden ser, la fuente de reloj para el temporizador de encendido (PWRT), el temporizador de vigilancia (WDT) y los circuitos FSCM. Se encuentra separado del FRC y oscila a una frecuencia nominal de 31 kHz. También se puede utilizar para proporcionar una opción de fuente de reloj de baja frecuencia para el dispositivo en aquellas aplicaciones en las que el consumo de energía es crítico y no se requiere de tanta precisión en la frecuencia.

Lazo de seguimiento de fase (PLL)

Para todos los dispositivos PIC24FJ, el reloj del sistema incluye una rama multiplicadora de frecuencia construida alrededor de un "**Phase Lock Loop**" o Lazo de seguimiento de fase (PLL). Esta rama permite obtener una velocidad de reloj más alta usando un oscilador primario de baja velocidad o una fuente de reloj externo, eliminando la necesidad de un cristal o resonador de alta velocidad que son más costosos económicamente. También permite el uso de

un oscilador FRC, como reloj del sistema, trabajando a su máxima velocidad de operación sin el uso de un oscilador externo.

En el mercado se encuentran distintos tipos de bloques PLL, que generan mayores o menores velocidades, pero el más común es el 4x. Este es el que se emplea en el PIC24FJ128GA010.

Bloque Básico 4x

Este bloque proporciona un multiplicador 4x fijo, que se puede usar con los osciladores primarios XT y EC y el oscilador FRC. El PLL acepta cualquier entrada de frecuencia de aproximadamente 3.5 MHz a 8 MHz, que puede ser la frecuencia directa de la fuente de reloj o el resultado de un proceso de post-escala.

Siempre que se cambie la fuente de reloj del PLL, el temporizador de PLL se debe reiniciar. Esta acción va a permitir al PLL sincronizar con la nueva fuente de reloj. Una vez que el temporizador haya contado el tiempo requerido, la salida de PLL estará lista para ser usada, ya que la frecuencia y fase se mantendrán en una situación estable.

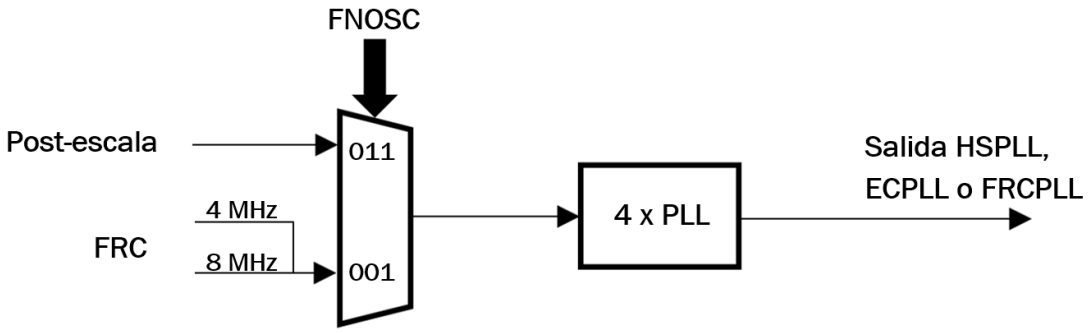


Figura 18. Esquema de funcionamiento de un Bloque Básico 4x

Consideraciones para usar el bloque PLL

El uso conjunto de un bloque PLL con un oscilador de tipo FRC proporciona al microcontrolador un reloj estable. Pero hay que tener en cuenta, que para las operaciones que necesitan de comunicaciones síncronas, como puede ser una comunicación en serie empleando un UART, es requisito que la precisión de la frecuencia sea de un ± 2%.

Todos los bloques PLL usan el bit LOCK como un bit de estado de solo lectura para indicar el estado de bloqueo del PLL. Se programa a 1 automáticamente después del típico retraso de tiempo para que el PLL logre el bloqueo, designado como TLOCK. Se borra cuando se produce un reinicio en el sistema

o cuando se producen interruptores de reloj como resultado de haber seleccionado el PLL como una fuente de reloj. Permanece a nivel lógico 0 cuando se selecciona cualquier fuente de reloj que no use el PLL.

Si el PLL no se estabiliza adecuadamente durante el arranque, es posible que el bit LOCK no sea capaz de leer su estado real, ni que detecte cuándo el PLL pierde el seguimiento de la fase durante el funcionamiento normal.

TEMPORIZADOR

El PIC24FJ128GA010 cuenta con cinco temporizadores de 16 bits, de los cuales, cuatro de ellos se pueden combinar para formar dos temporizadores de 32 bits. A su vez, estos temporizadores se pueden clasificar en tres tipos dependiendo de sus diferencias funcionales:

- Base de tiempo tipo A
- Base de tiempo tipo B
- Base de tiempo tipo C

A continuación, se va a mostrar de forma resumida que funciones tiene cada uno de estos tipos.

TIPO A

Nuestro microcontrolador cuenta con un temporizador de tipo A, conocido como “Timer1”. Tiene las siguientes características únicas sobre otros tipos:

- Se puede operar desde el dispositivo oscilador de baja potencia de 32 kHz.
- Se puede operar en un modo asincrónico desde una fuente de reloj externa.

En particular, las características únicas de un temporizador tipo A permiten su uso para funciones de cronometraje o como una fuente de reloj del sistema secundario.

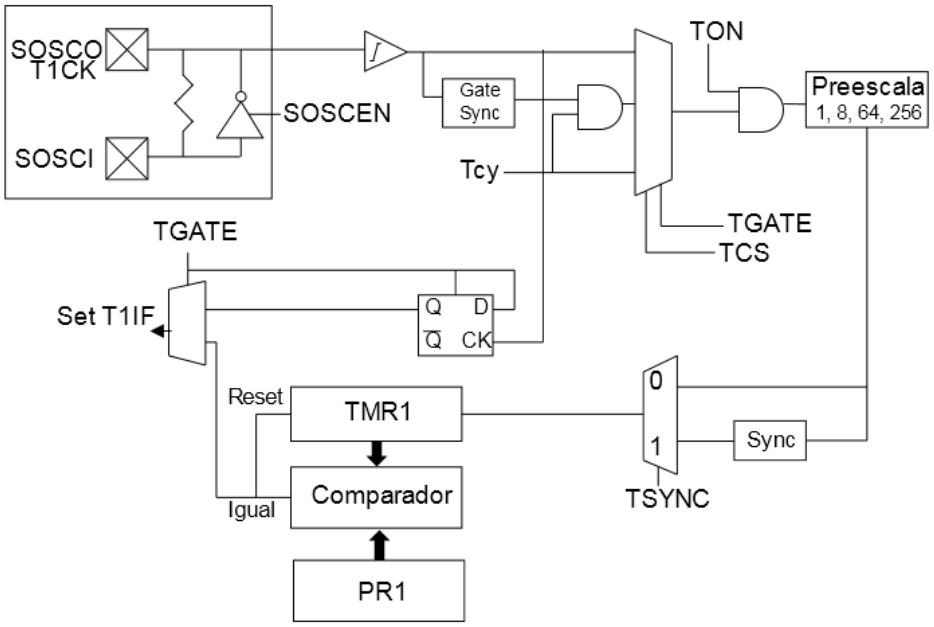


Figura 19. Esquema de un Temporizadorr tipo A para un PIC24

TIPO B

Para nuestro microcontrolador los temporizadores de este tipo son el "Timer2" y el "Timer4". Estos cuentan con las siguientes características únicas sobre los otros tipos de temporizadores:

• Un temporizador de tipo B se puede concatenar con un temporizador de tipo C para formar un temporizador de 32 bits. Para habilitar está opción, se debe configurar el registro TxCON del correspondiente timer de tipo B.

• La sincronización del reloj para un temporizador Tipo B se realiza después de la lógica de pre-escala.

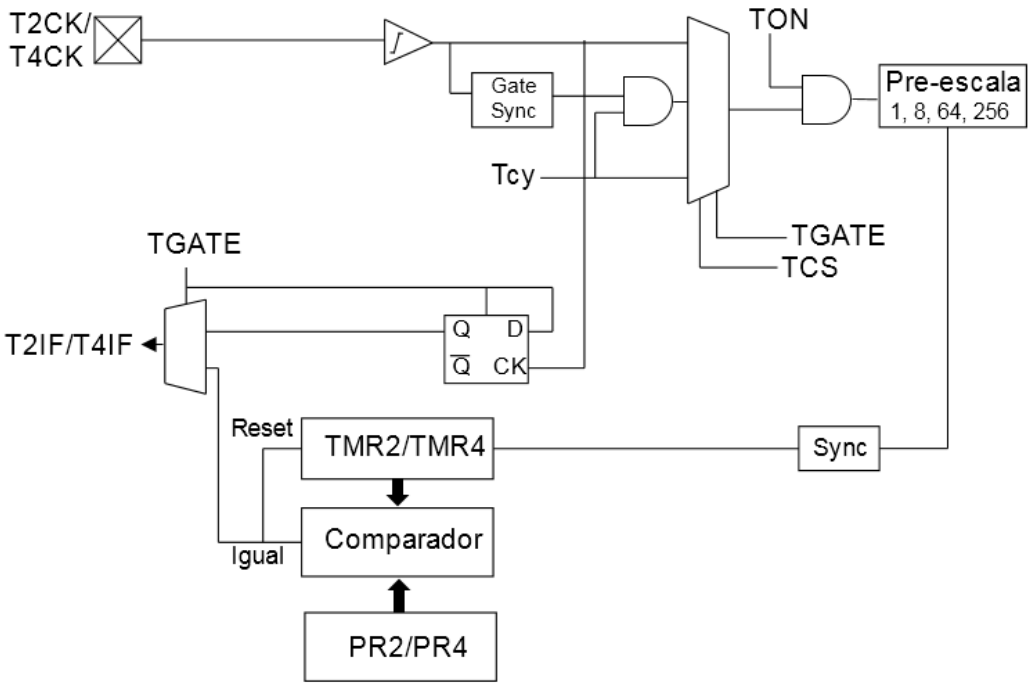


Figura 20. Esquema de un Temporizador tipo B para un PIC24

TIPO C

Para nuestro microcontrolador los temporizadores de este tipo son el "Timer3" y el "Timer5". Estos cuentan con las siguientes características únicas sobre los otros tipos de temporizadores:

• Un temporizador de tipo C se puede concatenar con un temporizador de tipo B para formar un temporizador de 32 bits.

• En un dispositivo dado, al menos un temporizador tipo C tiene la capacidad de activar una conversión A / D.

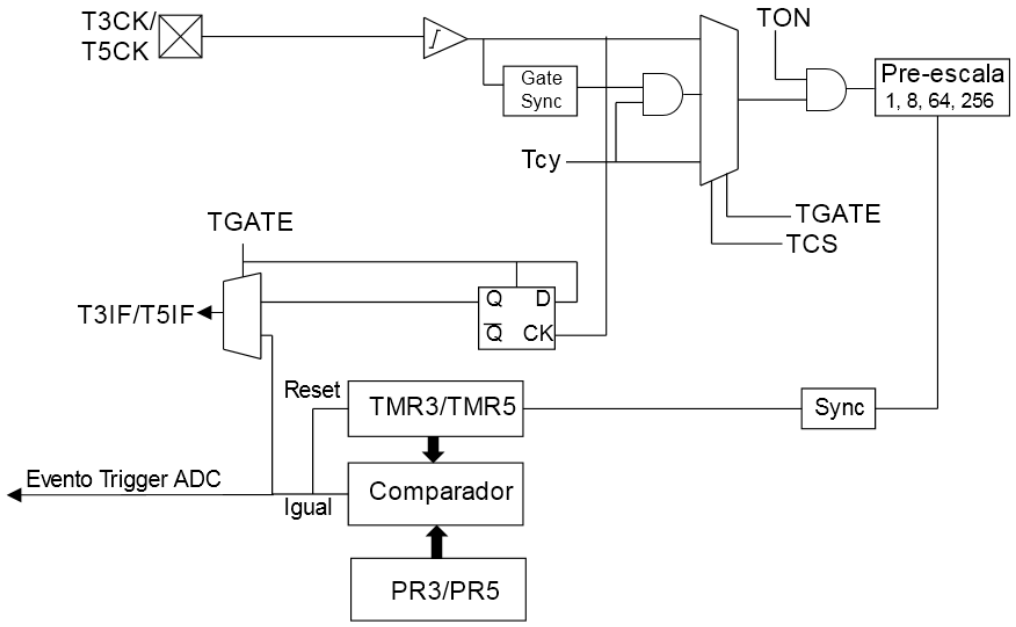


Figura 21. Esquema de un Temporizador tipo C para un PIC24

Registros principales del Temporizador

El principal registro que emplean los temporizadores sean del tipo A, B o C es:

- TxCON: Registro de control de los temporizadores.

A su vez, este registro dependiendo del tipo de temporizador que se emplee sufre ligeras variaciones con respecto a los otros tipos.

En general, con este registro se puede controlar cuando se activa y desactivan los temporizadores (TON), indicar si la fuente de reloj es externa o interna (Fosc/2), en caso de ser externa tener la opción de sincronizarla o no (TSYNC), posibilidad de aplicar una pre-escala, controlar el bloqueo del temporizador (TGATE), así como de interrumpir o no el funcionamiento del temporizador cuando el dispositivo ingresa al modo inactivo.

A mayores, en el caso de los temporizadores de tipo B, existe otro byte en este registro con el que se puede distinguir si se quieren concatenar o no un temporizador tipo B con un temporizador tipo C para conseguir un temporizador de 32 bits en lugar de uno simple de 16 bits.

Modos operación de los temporizadores

Cada módulo de temporizador puede operar en uno de los siguientes modos:

- Temporizador.

- Contador sincrónico.
- Modo de acumulación de tiempo de bloqueo.
- Contador asincrónico (solo con temporizadores de tipo A y C).

A continuación, se describe brevemente la forma de funcionamiento de cada uno de estos modos de operación.

A tener en cuenta que cuando el temporizador es de 32 bits los modos de operaciones son similares a los temporizadores de 16 bits.

Temporizador

Todos los tipos de temporizadores tienen la capacidad de operar en modo temporizador tomando como referencia el reloj del sistema. En este modo el reloj de entrada al temporizador se proporciona desde el reloj interno del sistema (FOSC / 2). Cuando está habilitado, el temporizador se incrementa una vez por ciclo de instrucción si el valor de pre-escala es de 1:1. El modo del temporizador se selecciona borrando el bit de control del TCS. El bit de control del modo síncrono, TSYNC, no tiene ningún efecto, ya que la fuente del reloj del sistema se utiliza para generar el reloj del temporizador.

Modo de contador síncrono usando entrada de reloj externo.

Cuando se programa a 1 el bit de control TCS, la fuente de reloj del temporizador se proporciona de forma externa y el temporizador seleccionado aumenta en cada flanco ascendente de la entrada del reloj en la patilla TxCK.

Para una base de tiempo Tipo A, la sincronización del reloj externo debe estar habilitada para ejecutarlo en el modo contador síncrono. Esto se logra al configurar el bit de control. En cambio, para las bases de tiempo Tipo B y Tipo C, la entrada del reloj externo siempre está sincronizada con el reloj del ciclo de instrucciones del sistema, TCY.

Cuando el temporizador opera en este modo, existen requisitos mínimos para el reloj externo, el tiempo de subida y el tiempo bajada. La sincronización de la fuente de reloj externa con el reloj de instrucción (TCY) del dispositivo se realiza muestreando la señal del reloj externo en dos momentos diferentes dentro de un mismo ciclo de instrucción.

La desventaja de trabajar en este modo es que no funcionará cuando el sistema entre en suspensión, ya que el circuito de sincronización se apaga cuando se suspende el dispositivo.

Temporizador asíncrono de tipo A usando una fuente externa

Para este modo de funcionamiento es necesario usar un temporizador de Tipo A, usando una fuente de reloj externa conectada a la patilla TxCK. Este modo de operación se basa en el borrado del bit de control TSYNC, debido a que la entrada del reloj externo no se sincroniza con la fuente de reloj del sistema del dispositivo, lo que provoca que el temporizador funcione de forma asíncrona.

La base de tiempo asincrónica se emplea para las siguientes aplicaciones:

- El temporizador asíncrono puede operar durante el modo de suspensión y tiene la capacidad de generar una interrupción cuando coincide el valor de la base de tiempo con el valor del registro de período, lo que reactivaría el procesador.
- La base de tiempo se puede sincronizar desde el oscilador de baja potencia de 32 kHz para proporcionar una fuente de reloj secundaria del sistema.

Operación del temporizador con fuente de reloj externa rápida

En algunas aplicaciones, se necesita de un segundo temporizador que cuente el momento en que se producen los flancos de subida o bajada de una fuente externa que emplee una frecuencia relativamente alta. Para estos casos, se opta por los temporizadores de Tipo A y Tipo B, ya que son las opciones más adecuadas debido a que la lógica de sincronización del reloj para estos temporizadores está ubicada después de la pre-escala. Esto permite que se use una frecuencia de reloj externo más alta que no viole los tiempos mínimos y máximos requeridos para la pre-escala. Se debe tener en cuenta que una base de tiempo de tipo A si opera en modo asíncrono elimina cualquier requisito de temporización de la pre-escala.

Modo de acumulación de tiempo de bloqueo

Este modo permite que el registro del temporizador interno se incremente en función de la duración del tiempo en estado alto aplicado al terminal TxCK. La fuente de reloj que emplea el temporizador se deriva del reloj interno del sistema (Fosc/2). Cuando el estado de la patilla TxCK es alto, el registro del temporizador irá aumentando hasta que haya ocurrido una coincidencia de período o se cambie a un estado bajo la patilla TxCK. Cuando se produce una transición de estado alto a bajo se activa el indicador de interrupción TxIF. Dependiendo de cuándo se produzca esta activación, la interrupción se realizará 1 o 2 ciclos de instrucciones después del cambio a estado bajo de la señal en la patilla TxCK.

El bit de control TGATE debe configurarse para habilitar el modo de acumulación de tiempo de bloqueo. Además de tener el temporizador en modo activo, es decir, TON programado a 1 y el bit de fuente de reloj del temporizador, TxCON, establecido como reloj interno.

TERMINAL ENTRADA SALIDA

Son el periférico más simple y a su vez de los más importantes, ya que permiten conectar la unidad de control del microcontrolador, MCU, con el resto de los periféricos. Además, algunos de ellos tienen la característica de poder de combinar otras funciones, lo que aporta mayor flexibilidad a la hora de programar el PIC. Se debe tener en cuenta, que, si una patilla de este tipo toma la función de otro periférico, normalmente, este ya no puede actuar como entra/salida.

Registros principales de los terminales entrada/salida

Todos los puertos de E/S tienen cuatro registros asociados directamente con la operación del puerto, siendo x una letra que determina que puerto de E/S es:

- TRISx: registro de control de dirección de datos PORTx. Es decir, determina si el terminal asociado al puerto es de entrada, programado a 1, o de salida, programado a 0.
- PORTx: registro de puerto de E/S. Este registro permite tanto la lectura como la escritura y el cambio de modo de las patillas del puerto.
- LATx: registro PORTx de almacenamiento de datos. Al igual que el anterior, permite leer, escribir y cambiar de modo, pero la diferencia es que este no lee o escribe el valor en las patillas, sino que lo realiza directamente en el biestable, ahorrando los problemas que pueden surgir de la lectura/escritura directa de los terminales.
- ODCx: Registro de control del modo de salida del PORTx. Con él se estable el modo de funcionamiento de los terminales, si se programa a 0 seguirá un funcionamiento normal, pero si se programa a 1 funcionaran como patillas de defensa frente a sobrecargas de tensión (Open-Drain). Es decir, que podrán funcionar a voltajes superiores o inferiores a VDD. Aunque nunca podrá sobrepasar las condiciones impuestas por el fabricante, V_{IH}, ni aplicar un voltaje inferior a VSS.

Cada patilla de E/S en el dispositivo tiene un bit asociado en los registros TRIS, PORT, LAT y ODC.

Funcionamiento

Las patillas pueden configurarse como entradas o salidas digitales, y entradas o salidas analógicas. Cuando configurados como entradas digitales, son búfers TTL o disparadores Schmitt. Cuando se configura como salidas digitales, son controladores CMOS o salidas de drenaje abierto "**Open-Drain**". Muchos terminales también admiten uno o más módulos periféricos.

Cuando está configurado para operar con un periférico, un terminal no se puede como entrada o salida. Pero, hay algunos dispositivos PIC24F, como es nuestro caso, que para múltiples funciones periféricas pueden ser multiplexados en cada patilla de E/S. La prioridad de la función periférica depende del orden que viene estipulado en la Hoja de características del dispositivo.

COMPARADOR DE SALIDA

El módulo de comparación de salida tiene la capacidad de comparar el valor de una base de tiempo seleccionada con el valor de uno o dos registros de comparación (dependiendo del modo de operación seleccionado). Además, tiene la capacidad de generar un solo pulso de salida, o un tren de pulsos de salida. Al igual que la mayoría de los periféricos, también tiene es capaz de generar interrupciones cuando se produce una coincidencia entre el valor del registro y el valor que toma la base de tiempo. En el caso de nuestro PIC24FJ128GA010, dispone de cinco canales de comparación de salida, que son funcionalmente idénticos. Cada canal de comparación de salida puede usar una de las dos bases de tiempo seleccionables. Esto se determina mediante el uso del bit OCTSEL. Las bases de tiempo disponibles para esta funcionalidad son el "Timer2" y "Timer3", pero se debe tener en cuenta que no son compatibles con el modo asincrónico. Por lo tanto, este modo de funcionamiento solo se puede aplicar para sistemas síncronos.

A continuación, se muestra el esquema general electrónico de funcionamiento que sigue un módulo de comparación de salida.

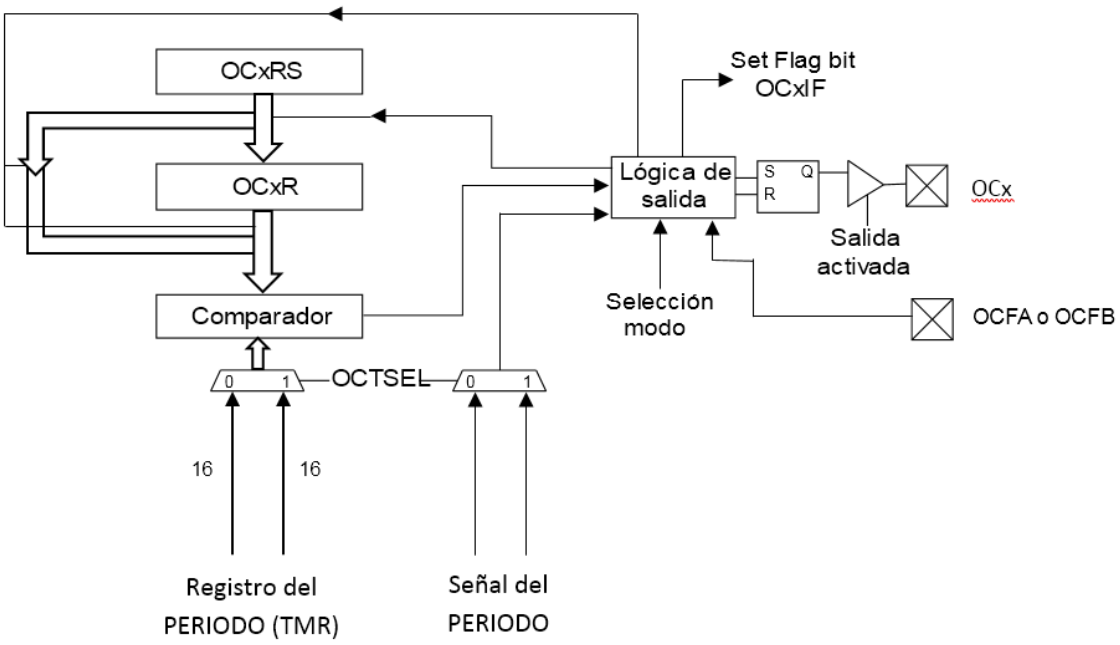


Figura 22. Esquema de un Comparador de Salida para un PIC24

Registros principales del Comparador de salida

Cada canal de comparación de salida tiene los siguientes registros:

- OCxCON: el registro de control para el módulo de comparación de salida.
- OCxR: primer registro de datos para el canal de comparación de salida.
- OCxRS: segundo registro de datos para el canal de comparación de salida.

Los registros de control para los 5 canales de comparación de salida se denominan de OC1CON a OC5CON. Todos ellos, tienen definiciones de bit idéntica y están representados por un registro común.

Modos de operación del Comparador de salida

Los módulos de comparación de salida son versátiles a la hora de regular su funcionamiento, ya que disponen de distintos modos de actuar:

- Modo de generación de pulso único por comparación con un único registro.
- Modo de generación de pulso dual por comparación con los dos registros disponibles.

- Modo de pulso de salida única.
- Modo de pulso de salida continúa.

- Modo con modulación de ancho de pulso simple:
  - Con entrada protectora ante fallos.
  - Sin entrada protectora de fallos.

A continuación, se va a esclarecer cada uno de estos modos de funcionamiento.

Modo de generación de pulso único por comparación con un único registro.

Para configurar este modo de funcionamiento del módulo de comparación de salida los bits de control, OCM, deben estar programados con el valor "001". El TMRy, registro del temporizador, también debe estar habilitado. Una vez que se haya habilitado este modo de comparación, la patilla de salida, OCx, inicialmente se reducirá y permanecerá bajo hasta que se produzca una coincidencia entre los registros TMRy y OCxR. Algunas consideraciones que se deben tener presentes son:

- Una vez que se produzca la coincidencia de los registros TMRy y OCxR, deberá pasar un ciclo de instrucción para que el terminal OCx se active como flanco de subida. Permanecerá a nivel alto hasta que haya un cambio de modo o se produzca la inhabilitación del módulo.
- El registro del temporizador, TMRy, irá aumentando hasta alcanzar el valor contenido en el registro del período asociado, PRy. Una vez lo alcance, se restablece a 0000h en el siguiente reloj de instrucciones.

El indicador de interrupción del canal respectivo, OCxIF, se activará dos relojes de instrucción después de que en el terminal OCx se genere el flanco de subida.

A continuación, se puede observar un ejemplo de este modo de funcionamiento:

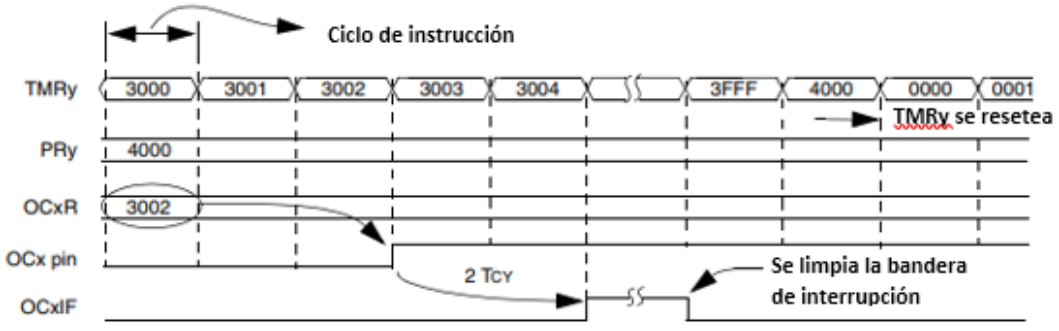


Figura 23. Diagrama de funcionamiento de generación de pulso único por comparación con un único registro

En algunas ocasiones, es preferible trabajar a nivel bajo. Para ello, se debe configurar los bits de control, OCM, con un valor de "010". Al final, la forma de funcionar es muy similar al anterior caso, con la diferencia de que se comienza trabajando a nivel alto, y cuando se produzca la coincidencia entre los registros TMRy y OCxR se genera el flanco de bajada que genera el pulso deseado. Al igual que antes, los tiempos necesarios para generar el flanco de bajada o activar la interrupción son los mismos que se indican anteriormente.

Otra forma comúnmente empleada, es aquella en la que el OCx va alternando su flanco cada vez que se produce una coincidencia. Inicialmente, pasará a flanco de bajada y en la siguiente coincidencia a flanco de subida, así continuamente hasta que se produzca una inhabilitación o un cambio de modo. En caso de desear este modo de funcionamiento, del cual se muestra un ejemplo más abajo, se necesita configurar los bits de control OCM como "011".

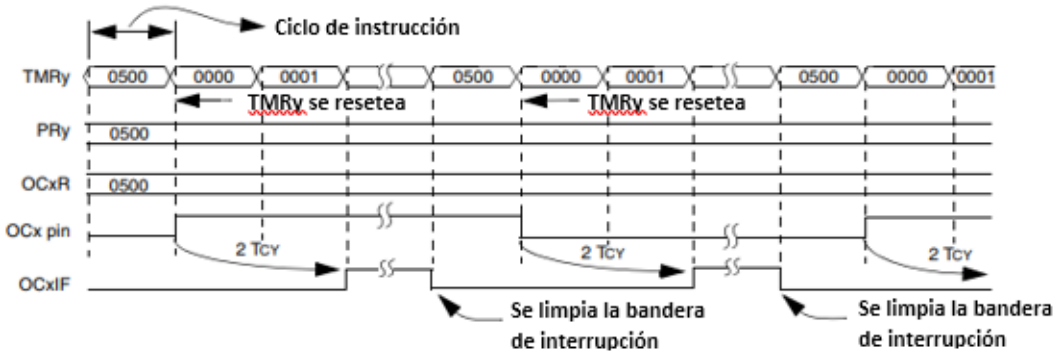


Figura 24. Diagrama de funcionamiento de un Comparador de Salida por cambio de flanco

Modo de generación de pulso dual por comparación con los dos registros disponibles

En este caso, los bits de control OCM deben tener asignado los valores 100 o 101, dependiendo de qué modo de pulso se desea obtener:

- Modo de pulso de salida única.
- Modo de pulso de salida continúa.

En este modo de funcionamiento, el módulo de generación de pulsos usa los registros OCxR y OCxRS para comparar eventos. Para empezar, el registro OCxR se compara con el registro del temporizador, TMRy, y cuando se produce la coincidencia de valores, se genera el flanco de subida correspondiente al pulso que se genera en el terminal OCx. Ahora bien, este evento se mantendrá a nivel alto hasta que el registro OCxRS coincida con el

valor de TMRy, momento en el que se genera el flanco de bajada del pulso en el terminal OCx.

PULSO DE SALIDA SIMPLE

Para configurar el módulo de comparación de salida para el modo Pulso de salida simple, los bits de control deben estar programados para un valor "100". Además, el TMRy debe estar seleccionado y habilitado. Una vez que este modo ha sido habilitado, el terminal de salida, OCx, se reducirá y permanecerá bajo hasta que ocurra una coincidencia entre la base de tiempo y el registro OCxR. Hay ciertos eventos de sincronización que es necesario tener en cuenta:

• El terminal OCx se activa generando un flanco de subida un ciclo de instrucción después de que se produce la coincidencia de comparación entre el registro TMRy y OCxR. La patilla OCx permanecerá a nivel alto hasta que la base de tiempo y el registro OCxRS coincidan. En este momento, en el terminal se genera el flanco de bajada. Permanecerá a este nivel hasta que se haya realizado un cambio de modo, o el módulo sea deshabilitado.

• TMRy contará hasta alcanzar el valor del registro del periodo, PRy, y luego reiniciará a 0000h en el próximo ciclo de instrucciones.

• Si el contenido del registro TMRy es menor que el contenido del registro OCxRS, entonces no se podrá producir el flanco de bajada en la patilla. Con lo cual, OCx permanecerá a nivel alto hasta que se cumpla que OCxRS ≤ PRy, o se produzca un cambio de modo o se deshabilite el mismo.

• La bandera de la interrupción, OCxIF, se activa dos ciclos de reloj después de que se haya producido el flanco de bajada del terminal OCx.

A continuación, se muestra un ejemplo del funcionamiento de este modo:

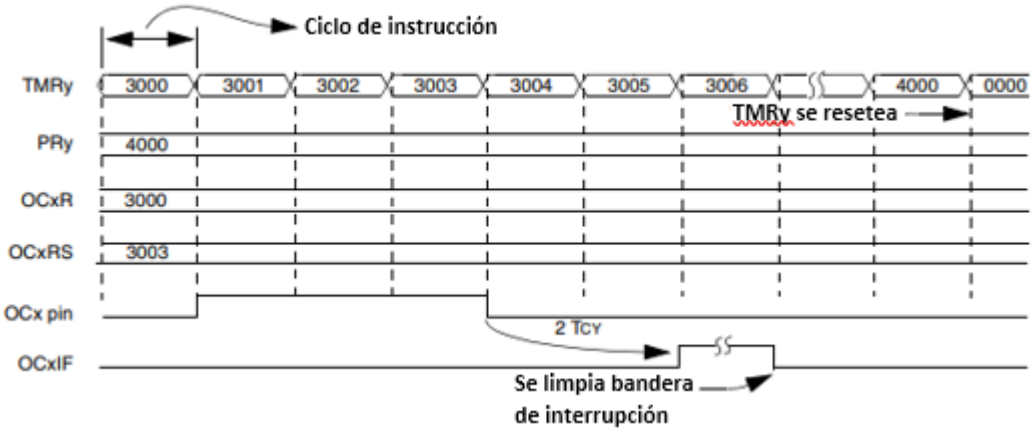


Figura 25. Diagrama de funcionamiento de generación de pulso simple

PULSO DE SALIDA CONTINUA

Este formato de modo de funcionamiento sigue las mimas características que el anterior, con la única diferencia de que se generan pulsos de forma continúa sin necesidad de que intervenga el usuario. Los tiempos de ejecución de la interrupción y generación del pulso siguen los mismos patrones que para la salida de pulso simple. Para configurar este modo, los bits de control, OCM, deben estar programados a "101".

Modo con modulación de ancho de pulso simple

Cuando los bits de control, OCM, se establecen en '110' o '111', el módulo de comparación está configurado para operar como PWM (Modulación del ancho de pulso). Y puede trabajar con entrada con protección ante fallos o sin protección ante fallos.

Para estos modos de funcionamiento, se establece una peculiaridad en cuanto a los registros. OCxR se convierte en un registro de ciclo de trabajo y de solo lectura, mientras que OCxRS es un registro de escritura en el que se actualiza el valor del ciclo de trabajo.

Cuando se produce la coincidencia de valores entre los registros TMRy y PRy se generan los siguientes acontecimientos:

- TMRy se restablece a cero y reanuda el conteo.
- OCx se activa, por tanto, se genera el flanco de subida a menos que OCxRS = 0.
- Se transfiere el valor del ciclo de trabajo de OCxRS a OxCR.
- La bandera de la interrupción, TyIF, se activa cuando TMRy y OCxR coinciden, y como consecuencia OCx genera el flanco de bajada.

INTERRUPCIONES

Una de las ventajas del empleo de un PIC24F es el trato de interrupciones. Este microcontrolador, posee un módulo de control de las mismas, para en lugar de permitir la generación de numerosas solicitudes de interrupción por parte de cada uno de los periféricos, se produzca una única señal. Este módulo, puede procesar hasta 8 excepciones de procesador y trampas de software. Contiene 7 niveles de prioridad entre los que puede elegir el usuario. Y para cada interrupción, se asigna un vector único, que a su vez pertenece a una tabla base, IVT, que dispone de 118 posiciones. Además, aunque el usuario asigne cierto nivel de prioridad a distintas interrupciones, dentro del mismo, existe una prioridad fija entre ellas.

Dentro de la CPU, los primeros 7 niveles de prioridad son los disponibles para las interrupciones de periféricos, mientras que los niveles de 8 a 15 están reservados para los "fuentes trampa". Es decir, son los encargados de encontrar problemas de hardware y software, por ello que dispongan de los niveles más altos.

UART

El módulo del Transmisor/Receptor asíncrono universal (UART) es una de las E/S en serie disponibles en los dispositivos PIC24F. El UART es un dúplex completo, que emplea un canal de comunicación asíncrono para comunicarse con dispositivos periféricos y computadoras personales, utilizando protocolos RS-232. Una de sus características principales es que puede transmitir o recibir datos de 8 o 9 bits a través de los terminales UxTX y UxRx. Además, se puede elegir si la transmisión de datos es par, impar o sin paridad, así como el número de bits de parada. Como el resto de los módulos, también tiene la opción de contar con interrupciones.

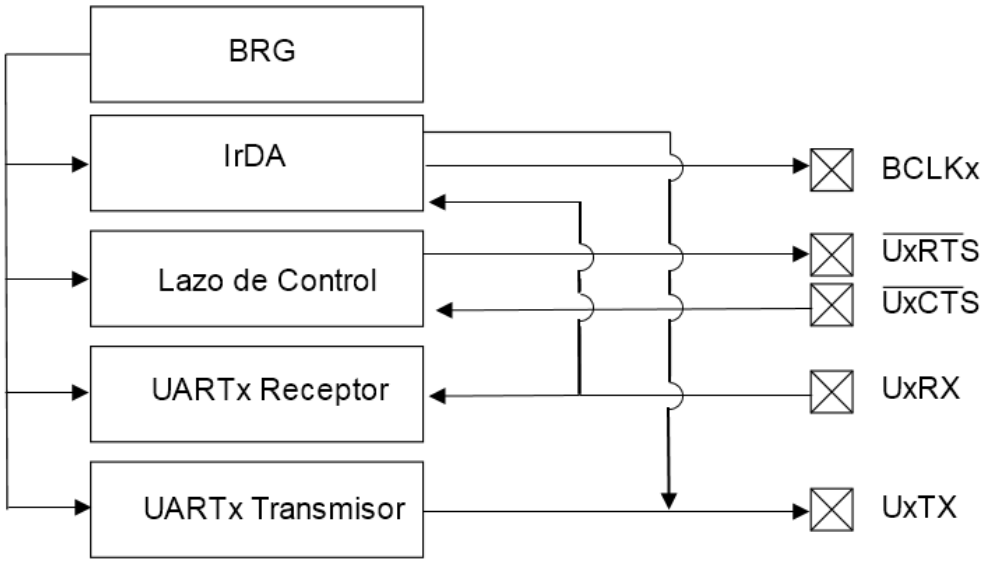


Figura 26. Esquema de un UART para PIC24

Una parte importante para el buen funcionamiento de este módulo es el cálculo de la variable "**Baud Rate**". El registro UxBRG controla el periodo de un temporizador de 16 bits de ejecución libre. Dicho de otra forma, establece la velocidad con la que se transmiten y reciben los bits. Para su cálculo se puede hacer uso de la siguiente formula, aunque el microcontrolador posee una opción de cálculo automático del mismo:

$$Baud\ Rate\ = \frac{Fcy}{4 \cdot (UxBRG + 1)}$$

$$\text{UxBRG}\ = \frac{\text{Fcy}}{4 \cdot \text{Baud Rate}} - 1$$

Figura 27. Fórmulas para el cálculo de la variable "Baud Rate"

La máxima velocidad posibles es $\frac{Fcy}{4}$ y el mínimo $\frac{Fcy}{4 \cdot 65536}$.

El módulo UART se habilita configurando el bit de activación de la recepción UARTEN y el bit de activación de la transmisión UTXEN. Una vez habilitado, las patillas UxTX y UxRX se configuran como salida y entrada, respectivamente, anulando los ajustes de bit de registro TRIS y PORT para los puertos de E/S correspondientes. El terminal UxTX toma el valor lógico 1 cuando no está teniendo lugar ninguna transmisión.

Transmisión de datos

La parte fundamental en la que se basa el transmisor es el Registro de desplazamiento de transmisión UxTSR. Este registro obtiene sus datos del búfer tipo FIFO de transmisión UxTXREG, que carga sus datos en el software. Para que el registro UxTSR pueda ser cargado, primero tiene que haber sido transmitido el bit de parada de la carga anterior, y segundo, UxTXREG debe tener datos disponibles.

La transmisión se habilita configurando el bit de habilitación UTXEN. La transmisión real no ocurrirá hasta que el registro UxTXREG haya sido cargado con los datos y la velocidad en baudios del generador "**Baud Rate**", UxBRG, haya producido un cambio de reloj. La transmisión también puede comenzar cargando primero el registro UxTXREG y luego configurando el bit de habilitación UTXEN. Normalmente, cuando se inicia la transmisión por primera vez, el registro UxTSR está vacío, por lo que una transferencia al registro UxTXREG dará como resultado una transferencia inmediata a UxTSR. Este último registro, dispone de 9 bits de ancho y 4 niveles de profundidad, que junto con el nivel en el que se produce el cambio de un registro a otro, hace un total de 5 niveles de profundidad. Se caracteriza por ser de tipo FIFO, es decir que el primer dato que entra será el primero en salir.

Una vez que el contenido de UxTXREG se transfiere al registro UxTSR, la ubicación actual en la que se encontraba el dato estará disponible para que se escriban nuevos datos y la siguiente ubicación del búfer será el que origina el siguiente registro de UxTSR. Otro bit importante, es el de estado UTXBF, que

adquiere el valor lógico 1 siempre que el búfer está lleno. Si el búfer se encuentra lleno, y se intenta escribir nuevos datos, esto no será viable debido a que los nuevos datos no serán aceptados en el FIFO.

Para este modo de funcionamiento se puede elegir cuando se generan las interrupciones:

- Se activa la bandera de interrupción cuando se transfiere un carácter del UxTXREG al UxTSR.
- Se activa la bandera de interrupción cuando se transfiere el último carácter del dato desde UxTXREG a UxTSR.
- Se activa la bandera de interrupción cuando se transfiere un carácter del UxTXREG al UxTSR y este último registro está vacío.

Recepción de datos.

La función de recepción se basa en el registro de desplazamiento UxRSR. En este caso los datos se reciben en el terminal UxRX y se envían a un bloque de recuperación que opera 16 veces más rápido que la velocidad de Baud Rate. Una vez que es leído el bit de parada, estos datos se transfieres al FIFO de la recepción si tiene hueco disponible.

Al igual que en el caso de la transmisión, el búfer FIFO dispone de 9 bits de ancho y cuatro niveles de recepción, además del quinto nivel para el que se produce el cambio entre registros.

Para el caso de las interrupciones sigue el mismo modus operandi que en las transmisiones.

3. CONFIGURACIÓN DEL ENTORNO Y EL PIC24FJ128GA010

En este apartado se va a explicar paso a paso desde cómo se crea un proyecto en el entorno MPLAB hasta cómo se ha configurado cada uno de los registros de los distintos módulos, y el porqué de dicha elección. Además, se explica cómo se ha realizado las comunicaciones entre placa y computador personal, que tipos de interfaces pueden ser empleadas y con cual se ha empezado a trabajar.

Si se desea más información en relación al código este se presenta en el anejo "Disparador de pulsos. Anejo I".

3.1. CREACIÓN DEL PROYECTO EN EL ENTORNO MPLAB X IDE

Para comenzar, se abre el entorno de trabajo MPLAB X IDE y se realizan los siguientes pasos:

1. Seleccionar "File" → "New Project" o pulsar sobre el icono de la carpeta amarilla con el más:

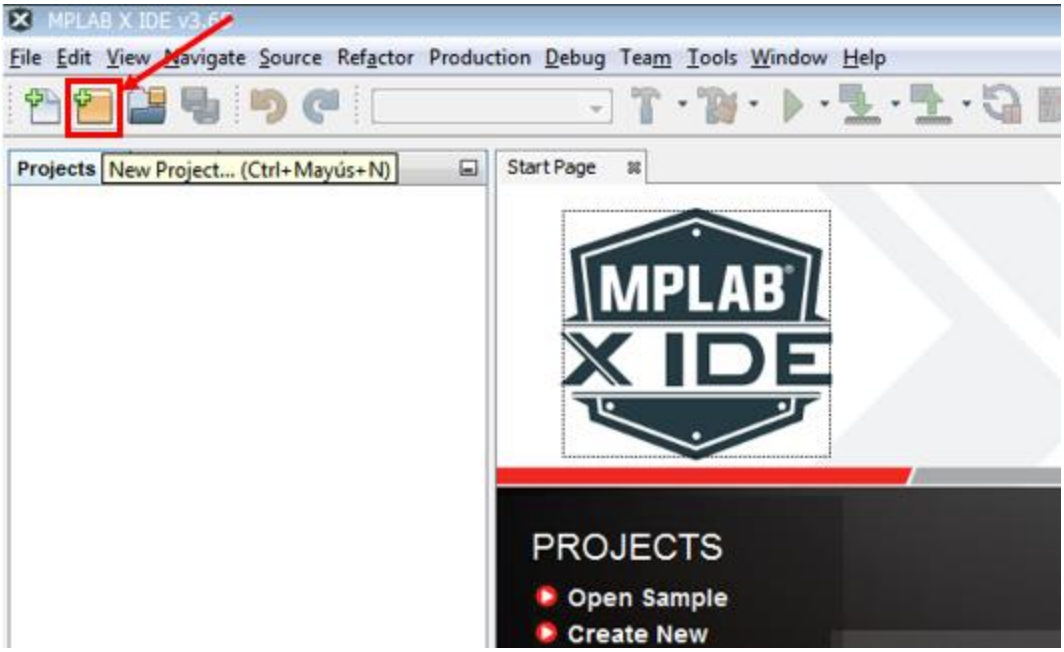


Figura 28. Icono de MPLAB X IDE para la creación de un nuevo proyecto

2. Una vez se ha seleccionado la opción de crear un nuevo proyecto, aparece una pantalla en la que se comienza por seleccionar el tipo de dispositivo que se quiere emplear y que tipo de proyecto se desea crear. En nuestro caso, se seleccionar un dispositivo de Microchip embebido, y una plantilla de proyecto estándar:

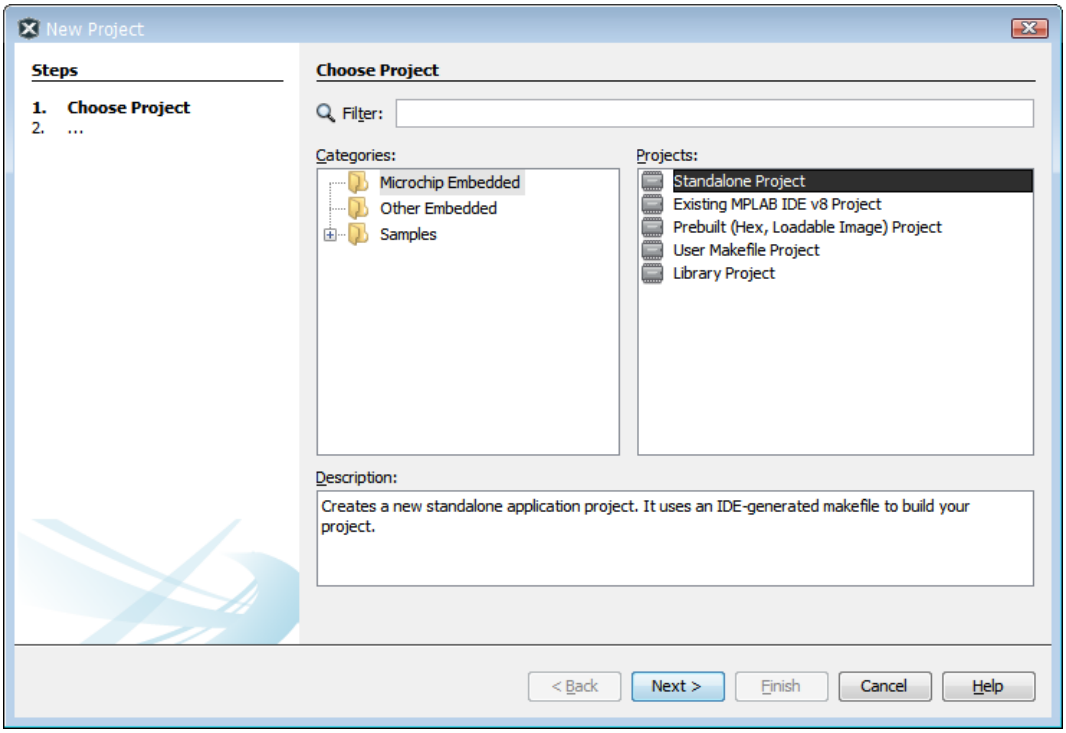


Figura 29. Pantalla MPLAB X IDE para selección de tipo de proyecto

3. A continuación, se debe seleccionar a que familia pertenece el dispositivo que se va a emplear, y el modelo del mismo. Para nuestro microcontrolador, la familia es la de microcontroladores de 16 bits, más concretamente la rama de los PIC24. Y el dispositivo empleado, como ya se ha mencionado en varias ocasiones, es el PIC24FJ128GA010:

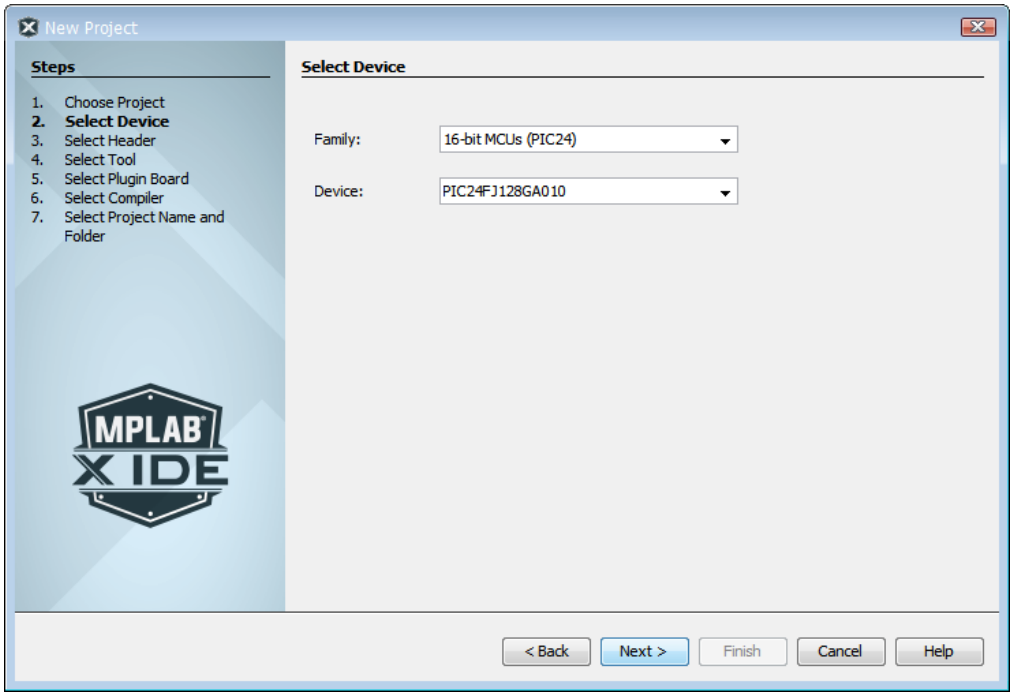


Figura 30. Pantalla de MPLAB X IDE para selección de dispositivo

4. También te da la opción de seleccionar, si se quiere, de qué tipo de adaptador para conectar un dispositivo de depuración se quiere emplear. Esto se conoce como "**Debug Header**". En este caso, se ha seleccionado la opción que viene por defecto que es ninguna:

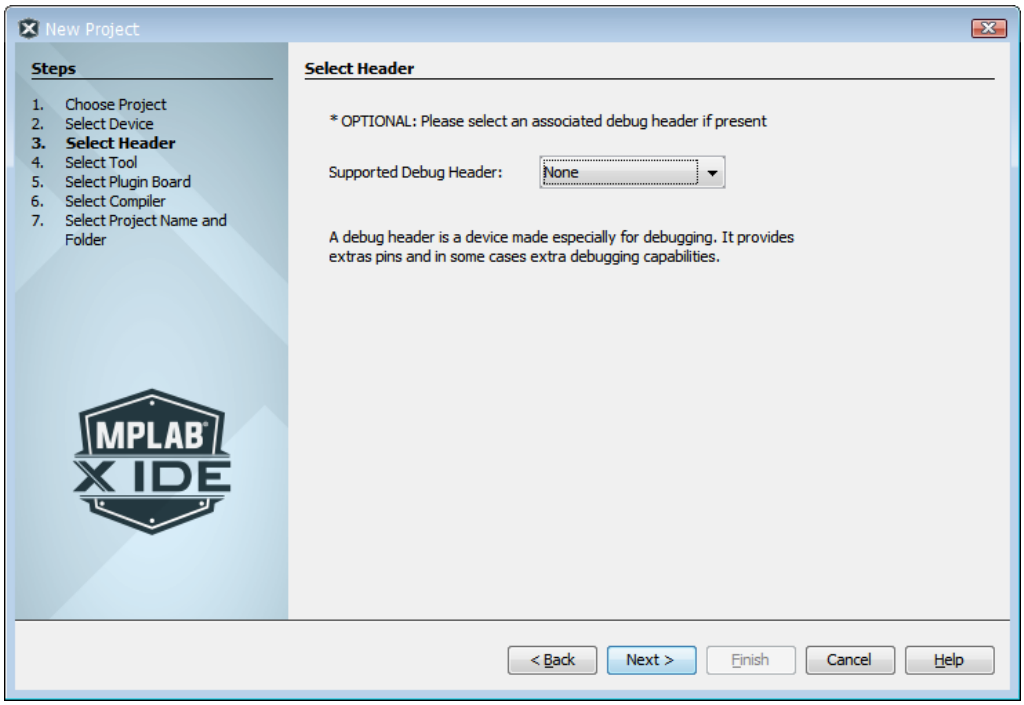


Figura 31. Pantalla de MPLAB X IDE para selección de un conector para el dispositivo de depuración

5. En la siguiente pantalla se debe elegir qué tipo de herramienta hardware se va a emplear. Seleccionamos el PICkit3:

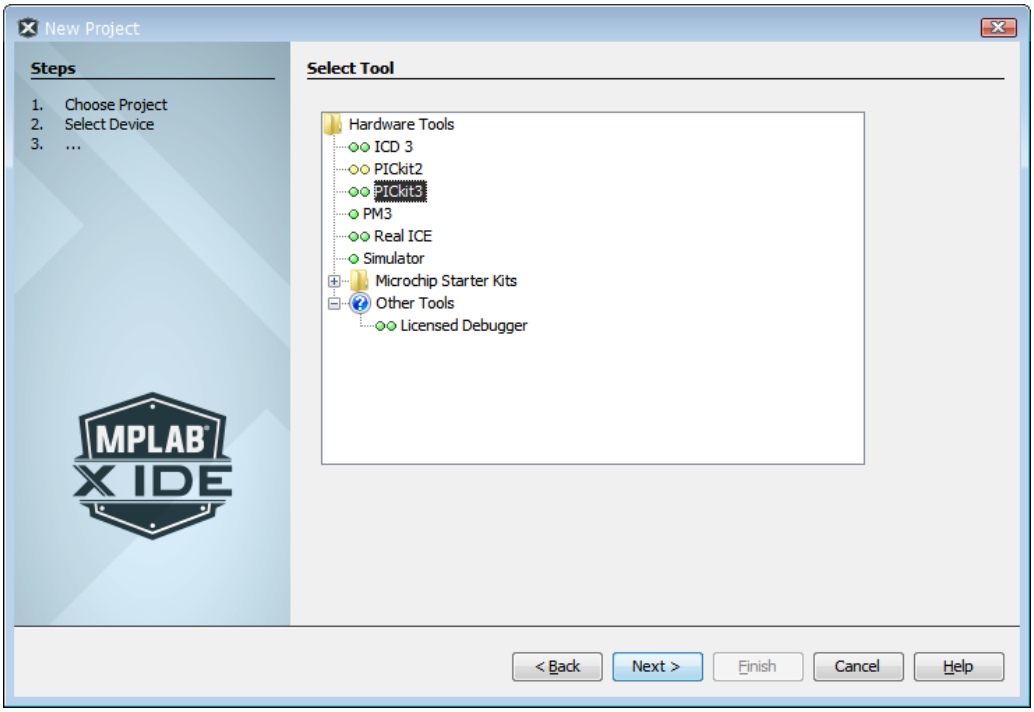


Figura 32. Pantalla de MPLAB X IDE para selección de herramienta hardware

6. El penúltimo paso, será seleccionar el tipo de compilador que se desea emplear. Es conveniente investigar qué tipo de compiladores tolera el microcontrolador con el que se va a trabajar. Por ello, en este caso, se ha optado por el compilador XC16:

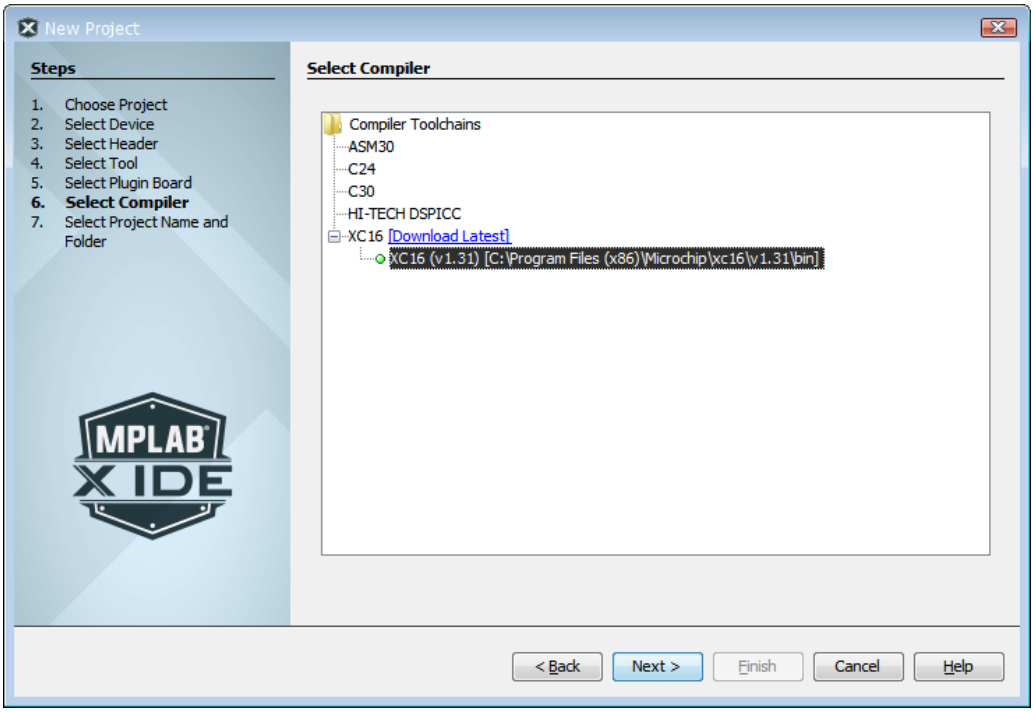


Figura 33. Pantalla de MPLAB X IDE para selección de compilador

7. Para terminar, se debe el elegir el nombre del proyecto, y el tipo de reglas que se desea emplear para la codificación. Se ha elegido la ISO-8859-1 que es la establecida por defecto y la que se emplea en Europa occidental. Además, se acepta la opción de elegir este proyecto como el principal del entorno. Esto quiere decir, que las reglas que se ajusten para depurar se aplicaran a este proyecto.

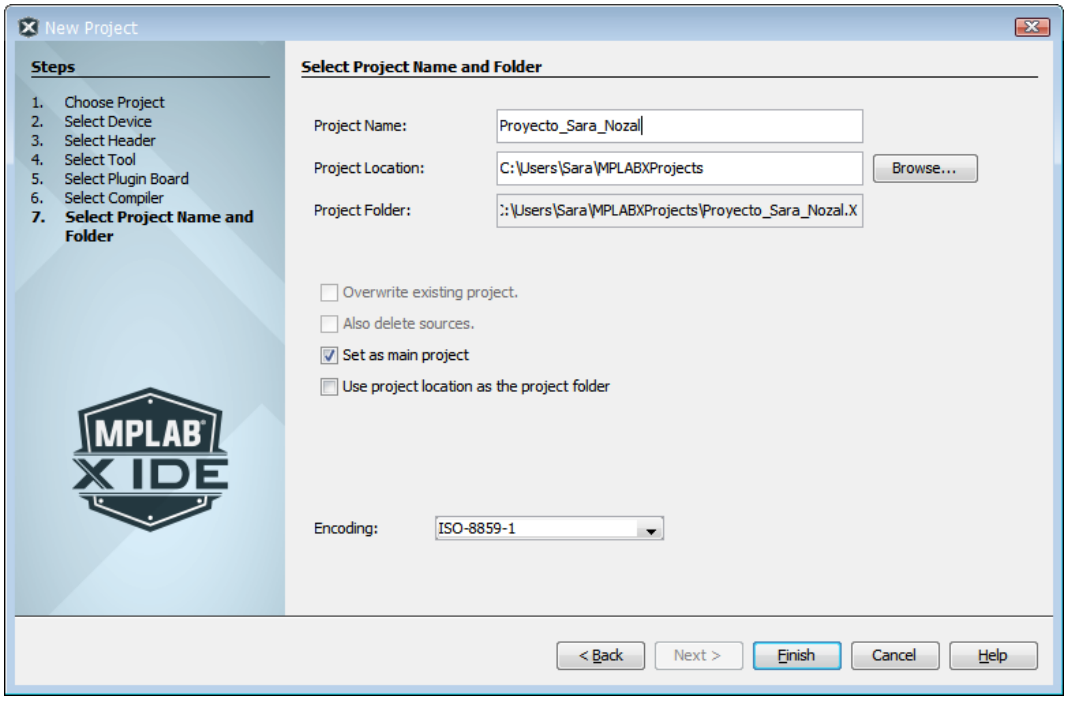


Figura 34. Pantalla MPLAB X IDE de selección de nombre y ruta del nuevo proyecto

Una vez que se ha creado el proyecto, se plantea como se va a configurar. En este caso, se ha optado por trabajar con un fichero de cabeceras, "**header**", así como de distintos ficheros que tendrán contenido las distintas funciones de cada módulo de operación, ficheros .c, como pueden ser la declaración de los Temporizadores, de los Comparadores de Salida, del módulo de comunicación UART… para seguir cierta organización:

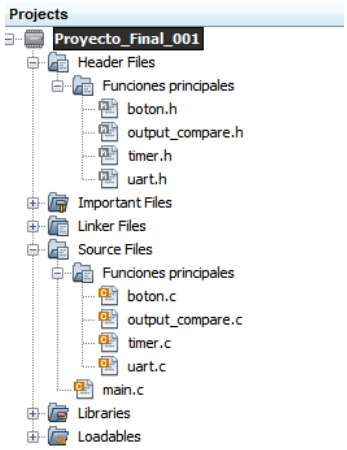


Figura 35. Estructura de carpetas del presente proyecto

Para realizar esta estructura de organización, hacemos click derecho sobre la carpeta "Header Files" → "New Logical Folder", para crear la subcarpeta de

Funciones principales. Una vez se ha creado esta, se hace de nuevo click derecho sobre la última carpeta creada Funciones principales → "New" → xc16_header.h:

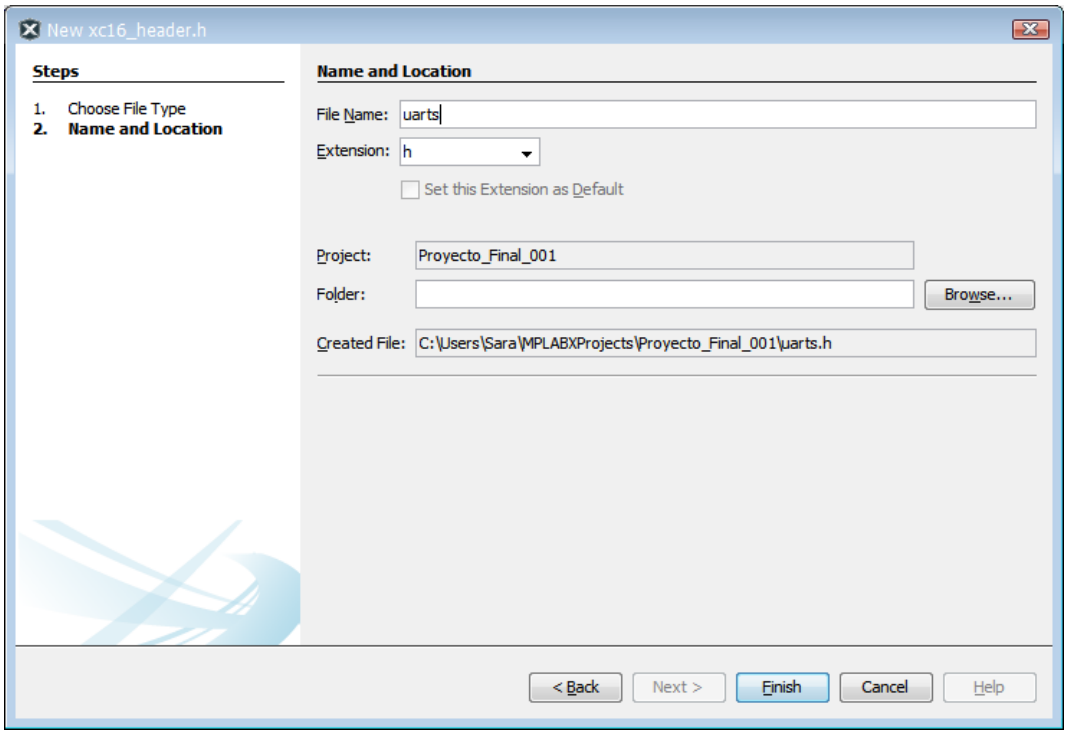


Figura 36. Pantalla de MPLAB X IDE para la creación de un archivo .h

Para nuestro proyecto, se ha optado por crear solo los ficheros cabecera para UART, temporizador, comparador de salida y boton ya que son las funciones principales que necesitaremos para el funcionamiento del mismo.

Por otro lado, para crear la parte concerniente a ficheros .c es muy similar. Primero se ha procedido a crear la subcarpeta, Funciones principales, de igual forma que antes. A continuación, haciendo click derecho sobre ella "New" → mainXC16.c:

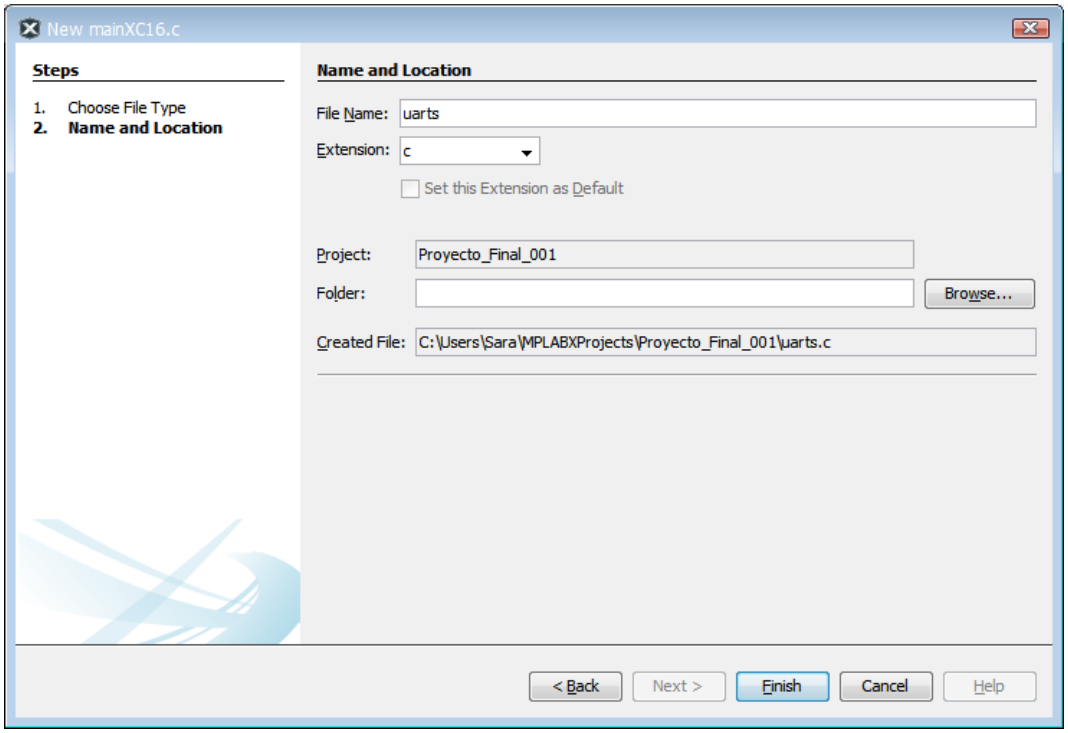


Figura 37. Pantalla de MPLAB X IDE para la creación de un archivo .c

También en este caso se ha optado por crear los mismos cuatro ficheros .c para UART, temporizador, comparador de salida y boton, que van ligados a sus ficheros cabecera.

En el fichero de cabecera "uart" se procede a declarar cada una de las funciones que se van a establecer en el fichero .c que lleva su mismo nombre. Se debe declarar que tipo de función es, el nombre y si necesita de paso de argumentos:

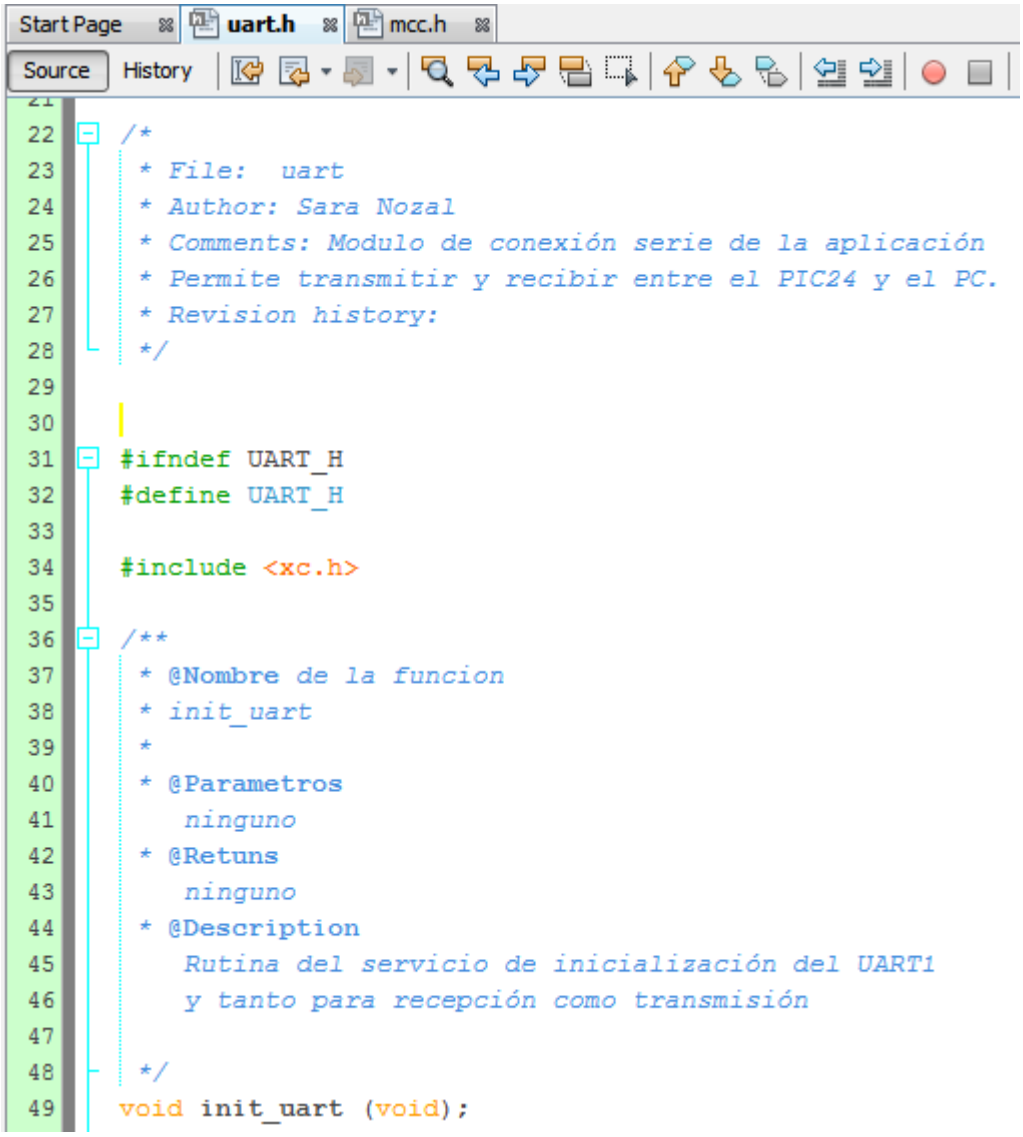


Figura 38. Ejemplo de código de un fichero.h del presente proyecto

Y cada uno de los ficheros .c llevan declaradas las distintas funciones que son necesarias hacer, como por ejemplo, dentro del fichero que contenga la configuración del UART, se declarará de que canales de UART se va a hacer uso, como tiene configurados los terminales, que interrupción tiene asociada, por qué modo de funcionamiento se ha optado…

Start Page | uart.c | main.c

Source | History

```
10  #include "uart.h"
11
12  // Rutina del servicio de inicialización del UART1
13  // y tanto para recepción como transmisión
14
15  void init_uart (void)
16  {
17      U1MODEbits.UARTEN = 0; //limpia los registros
18      U1MODEbits.RTSMD = 1; //modo simplex
19      U1MODEbits.PDSEL0 = 0; //8 bits y no paridad
20      U1MODEbits.PDSEL1 = 0;
21      U1MODEbits.STSEL = 0; //1 bit parada
22
23
24      U1STAbits.UTXISEL0 = 0;
25      U1STAbits.UTXISEL1 = 0;
26
27      IFS0bits.U1RXIF = 0; //Baja la bandera de interrupción
28      IFS0bits.U1TXIF = 0; //Baja la bandera de interrupción
29      IEC0bits.U1RXIE = 1; //Habilita la interrupción de Recepción
30      IEC0bits.U1TXIE = 1; //Habilita la interrupción de Transmisión
31      IPC2bits.U1RXIP = 6;  //Nivel de interrupción del receptor del UART1 =6
32      IPC3bits.U1TXIP = 6; //Nivel de interrupción del transmisor del UART1 =6
33
34      U1BRG = 25;
35
36
37      U1MODEbits.UARTEN = 1; //Activamos la UART1
38      U1STAbits.UTXEN = 1; // Activamos la transmision
```

Figura 39. Ejemplo de código de un fichero .c del presente proyecto

3.2. CONFIGURACIÓN DE LOS FICHEROS FUNCIÓN .C

Una vez que se tiene clara la distribución a seguir, se procede a explicar de forma específica cada uno de los ficheros de configuración de los distintos módulos.

Algunas de las funciones principales que se han visto anteriormente, como pueden ser las interrupciones, la configuración del oscilador o de los terminales de entrada/salida no tienen un fichero de configuración propio porque están incluidos indirectamente en el resto de los ficheros.

3.2.1. MAIN

Este módulo es el cuerpo principal de la aplicación. Desde esta función principal se llaman al resto de funciones de inicialización, primero se llama al init_T2 para establecer el funcionamiento del temporizador. A continuación, se llama a init_uart, para dejar activo el canal de transmisión y recepción de datos entre la placa y el ordenador. Por último, se establece un sencillo menú para determinar cuándo se genere el pulso.

Además, en este módulo, se determinan las palabras de configuración. Para ello, se accede desde la pestaña superior de MPLAB X IDE a "Production" → "Set Configuration Bits". Desde esa pestaña se puede determinar que configuración se desea y cuando se ha seleccionado se debe pulsar sobre el botón inferior "Generate Source Code to Output", que crea el código en lenguaje C y se puede pegar directamente en tu código.

Para este trabajo se determina el siguiente formato de las palabras:

Palabra de configuración II:

- pragma config POSCMOD = XT; Selección del Oscilador Primario, más concretamente, el modo XT, ya que se emplea el cristal de cuarzo del que dispone la placa.
- pragma config OSCIOFNC = OFF; La función de la salida generada por el oscilador primario será FOSC/2.
- pragma config FCKSM = CSDCMD; La conmutación del reloj con el monitor por si se genera fallos se encuentra deshabilitada.
- pragma config FNOSC = PRI; Si se reinicia el microcontrolador la configuración que se seleccionará para el oscilador será la del Oscilador primario.
- pragma config IESO = OFF; Arranque en dos velocidades desactivado.

Palabra de configuración I:

- pragma config WDTPS = PS32768; Post-escala del temporizador del perro guardián será de 1:32768.
- pragma config FWPSA = PR128; Pre-escala del temporizado del perro guardián será de 1:128.
- pragma config WINDIS = ON; Ventana del perro guardián activada.
- pragma config FWDTEN = OFF; Temporizador del perro guardián desactivado.
- pragma config ICS = PGx2; Se selecciona los terminales EMUC2/EMUD2 para el depurador.
- pragma config GWRP = OFF; Se permite escribir en el código para modificar la memoria volátil.
- pragma config GCP = OFF; El código se puede modificar sin que se borre en caso de intentarlo.

- pragma config JTAGEN = OFF; El puerto JTAG se encuentra deshabilitado ya que se hace uso del puerto serie.

3.2.2. OSCILADOR EXTERNO

Para este proyecto, se prevé en el futuro el uso de un oscilador externo de cuarzo con fuente de señal de reloj, que el usuario podrá modificar a su antojo, a través de la interfaz que posee. Con dicha interfaz se puede regular la frecuencia a la que trabaja la fuente. Para este modus operandi, el oscilador puede operar entre 0 - 32 MHz para cumplir con las limitaciones del fabricante.

Algunos de los puntos por los que se vaticina su uso son:

Mayor versatilidad para la placa, debida a que, al emplear este tipo de fuente, sola es necesario conectarla al terminal de entrada OSC1. De esta manera, queda la patilla OSC2 como E/S de propósito general o como salida de reloj a una frecuencia de Fosc/2.

Es posible sincronizar el funcionamiento del microcontrolador con los demás componentes incorporados en el dispositivo.

En este modo el microcontrolador se pone a funcionar inmediatamente después de encenderlo. No se requiere esperar para estabilizar la frecuencia.

Al deshabilitar temporalmente la fuente de reloj externa, se detiene el funcionamiento del dispositivo, dejando todos los datos intactos. Después de reiniciar el reloj externo, el dispositivo sigue funcionando como si no hubiera pasado nada.

Para que el oscilador quede configurado correctamente desde el punto de vista de software, como mínimo debe cumplir estos requisitos:

- Registro OSCON
  - Bits COSC = 010; Para indicar el modo EC, Oscilador Externo.
  - Bits NOSC = 010; Si se resetea el microcontrolador la configuración que se cargará será la de EC.

Por otro lado, desde el punto de vista hardware, el oscilador debe ir conectado al terminal de entrada OSC1, quedando OSC2 libre.

El usuario debe tener en cuenta a la hora de establecer tiempos, que la frecuencia a la que funcione el oscilador no es a la que van a ejecutarse las instrucciones, sino que será a mitad de esta. Es decir, que, si establece la frecuencia máxima de funcionamiento, 32 MHz, cada ciclo de instrucción se

tardará en completar 16MHz. Este podrá regular la frecuencia a través de una ruleta que dispone el sistema.

En caso de que se desee implementar una post-escala, desde la configuración que dispone el Temporizador, la cual se verá a continuación, se puede aplicar.

3.2.3. TEMPORIZADOR

Se ha optado por configurar el "Timer2", ya que el módulo Comparador de Salida, solo acepta trabajar con este temporizador o con el "Timer3". En el caso, más avanzado, de querer trabajar con un temporizador de 32 bits haríamos uso de la unión de estos dos para que sea compatible con todos los módulos que necesitamos emplear.

Para este módulo, se dispone de una función de inicialización del "Timer2", así como de una función de control de la interrupción.

Este módulo se ha configurado de la siguiente forma:

- T2CONbits.TCS = 0; Con este registro se declara con que fuente de reloj se va a trabajar, en nuestro caso se hará con el reloj interno Fosc/2. En un futuro, se deberá cambiar a un 1 para que trabaje con una fuente externa, que será el oscilador del laboratorio.
- T2CONbits.TCKPS1 = 0;
- T2CONbits.TCKPS0 = 0; Los dos bits del registro TCKPS establece la pre-escala que se desea emplear. Inicialmente se opta por una pre-escala de 1:1, en caso de que se desee modificar, en la función "main" se podría variar dependiendo del funcionamiento que se necesita para cada momento.
- T2CONbits.TGATE = 0; Este registro se mantiene a nivel lógico 0 ya que queremos que el modo de funcionamiento sea el de un temporizador.
- TMR2 = 0X0000; Este registro es donde se guarda el conteo del temporizador. Como es una inicialización, lo establecemos a nivel lógico 0 para que no haya problemas cuando inicie la temporización.
- PR2 = 0x0008; Periodo de T2.
- IFS0bits.T2IF = 0; Baja la bandera de interrupción del temporizador "Timer2", para cuando se produzca la interrupción se active a 1.
- IEC0bits.T2IE = 1; Habilita la interrupción del "Timer2".
- IPC1bits.T2IP = 5; Este registro indica el nivel de prioridad de la interrupción del "Timer2". Es el nivel más bajo, debido a que, aunque

es una parte importante del funcionamiento de la aplicación, la interrupción por la generación del pulso o la transmisión de datos debe ser más alta ya que es más primordial que se avise de estas creaciones a la información de reseteo del contador.

Por otro lado, se encuentran la función de interrupción que es bastante sencilla. Con ella se limpia la bandera que activa y desactiva la interrupción. Además, tiene hace uso de un LED, D5, de control para informar de cada vez que se produce el reseteo del contador del temporizador, TMR2. Esta función se ha designado como **_T2Interrupt** y el registro en el que se indica que se desea limpiar la bandera de interrupción es el IFS0bits.T2IF.

3.2.4. COMPARADOR DE SALIDA

Dado que los tipos de pulso que se desean generar son:

- Un pulso por una patilla de ancho variable (a voluntad del usuario)
- Dos pulsos por la misma patilla y la diferencia de tiempo entre ellos variable (a voluntad del usuario).
- Dos pulsos, cada uno por una patilla, la diferencia de tiempo entre ellos variable (a voluntad del usuario).

Se ha optado por la generación de pulsos duales por comparación de dos registros, a partir de los Comparadores de salida OC1 y OC2. El usuario podrá seleccionar la duración del pulso, es decir, que podrá determinar cuando quiere generar el flanco de subida y el de bajada, haciendo la aplicación más versátil.

Además de poder elegir el tipo de pulso que desea generar.

Para el funcionamiento de este módulo ha sido necesario la creación de una función de inicialización y otra de interrupción para controlar cuando se genera el pulso.

La función **init_uart** contiene la configuración de funcionamiento del módulo a partir de los siguientes registros:

- _TRISD0= 0; Con este registro se establece el terminal 0 del puerto D como salida para el Comparador de Salida.
- OC1CON = 0x0000; Este es el registro de control del Comparador de Salida, se inicializa a 0 como modo de seguridad por si se ha quedado anteriormente registrada alguna otra configuración.
- OC1R y OC1RS son los registros que determinan el flanco de subida y de bajada. Estos se establecen en el módulo UART para que el usuario pueda seleccionar el valor del ancho de pulso.

- OC1CONbits.OCM2 = 1;
- OC1CONbits.OCM1 = 0;
- OC1CONbits.OCM0 = 0; Con estos tres bits OCM se determina el modo de funcionamiento que se desea para el Comparador de Salida. En este caso, se ha seleccionado el modo dual de generación de pulso por comparación con dos registros.
- OC1CONbits.OCTSEL = 0; Este registro determina cuál de los dos temporizadores disponibles se va a emplear. Se ha optado por el "Timer2".
- IPC0bits.OC1IP0 = 1;
- IPC0bits.OC1IP1 = 1;
- IPC0bits.OC1IP2 = 1; Con estos tres bits OC1IP se determina el nivel de prioridad de la interrupción del Comparador de Salida 1. Se ha elegido el nivel más alto, ya que la generación del pulso es la parte fundamental de este trabajo.
- IFS0bits.OC1IF = 0; Baja la bandera de interrupción del generador de pulsos OC1, para cuando se produzca la interrupción se active a 1.
- IEC0bits.OC1IE = 1; Activa la interrupción del OC1.
- T2CONbits.TON = 1; Activamos el "Timer2" para que empiece a contar. Aunque esto no es parte del control del Comparador de Salida, se activa en ese instante porque es cuando se necesita que se inicie el proceso del temporizador.

Así mismo, se han configurado estos mismos registros para el generador de pulsos OC2.

Por otro lado, se encuentran las funciones de interrupción **_OC1Interrupt** y **_OC2Interrupt**. Estas se emplean para limpiar la bandera y para llevar un control mediante el LED D4 que se enciende cuando se termina de generar el pulso completo, es decir, cuando se produce el flanco de bajada.

3.2.5. UART

Se ha optado por esta forma de comunicación porque es sencilla y rápida. La transmisión secuencial de datos es idónea para nuestro trabajo, pues se desea principalmente para informar del estado del pulso que se va a generar, es decir, si está a nivel alto o bajo. Y ello se indica con un solo bit, por lo cual para el trabajo que se necesita el costo es realmente bajo con esta forma de trabajar. Además, actualmente prácticamente todos los computadores y

dispositivos disponen de un puerto serie, por lo que no habría problema para implementarlo con cualquier sistema del laboratorio.

Para el módulo de UART se necesita hacer uso de las funciones de transmisión y recepción de datos. Además, se crean las funciones de interrupción para controlar esta comunicación.

Se ha creado la función **init_uart**, que establece la inicialización de este módulo, indicando por qué características se va a regir. Otro punto a favor de este módulo es que con la misma función se inicializa el modo de transmisión de datos y el de la recepción.

Este módulo se ha configurado de la siguiente forma:

- U1MODEbits.UARTEN = 0; Este registro es el de habilitación del módulo UART, de primeras se inicializa para limpiar los registros.
- U1MODEbits.RTSMD = 1; Este registro lo activamos para hacer uso del "Modo Simplex". Este modo de funcionamiento sirve como "aviso" al microcontrolador para indicar cuando está listo para transmitir datos.
- U1MODEbits.PDSEL0 = 0;
- U1MODEbits.PDSEL1 = 0; Estos dos bits PDSEL indican cuantos bits se van a transmitir y el tipo de paridad del que va a hacer uso. En este caso, con los dos bits programándose a 0 se indica que se va a transmitir 8 bits y no tiene paridad.
- U1MODEbits.STSEL = 0; En este registro se indica cuantos bits de parada se van a emplear, es ente caso solo se emplea uno.

Se ha elegido este formato de actuar, 8 bits sin paridad con un bit de parada, porque es el formato estándar de comunicación serie de los ordenadores.

- U1STAbits.UTXISEL0 = 0;
- U1STAbits.UTXISEL1 = 0; Estos dos bits del registro UTXISEL, como se ha explicado en anteriores apartados, sirven para elegir en qué momento se genera la interrupción. En nuestro caso, se ha optado por que se active la bandera de interrupción cuando se transfiere un carácter del UxTXREG al UxTSR. Así, se puede controlar que cuando se transmita el dato de si el pulso está a nivel alto o bajo se ha recibido correctamente. Esta lógica se aplica igualmente cuando se hace la recepción de datos en el microcontrolador.
- IFS0bits.U1RXIF = 0; Baja la bandera de interrupción de la recepción, para cuando se produzca la interrupción se active a 1.
- IFS0bits.U1TXIF = 0; Baja la bandera de interrupción la transmisión, para cuando se produzca la interrupción se active a 1.

- IEC0bits.U1RXIE = 1; Habilita la interrupción de Recepción.
- IEC0bits.U1TXIE = 1; Habilita la interrupción de Transmisión.
- IPC2bits.U1RXIP = 6; Nivel de prioridad de la interrupción del receptor del UART1. Se ha elegido un nivel alto, ya que la transmisión es función primordial en esta aplicación.
- IPC3bits.U1TXIP = 6; Nivel de interrupción del transmisor del UART1. Al igual que con la anterior el nivel de prioridad debe ser alto.
- U1BRG=25; Este registro el nivel de velocidad de transmisión/recepción de bits. Este valor se debe calcular para las características del microcontrolador. Para ello se puede hacer uso de la fórmula o consultando la hoja de características se informa en una tabla. Siguiendo la fórmula:

$$UxBRG = \frac{Fcy}{4 \cdot Baud\ Rate} - 1$$

Figura 40. Fórmula general para el cálculo del registro UxBRG

En nuestro caso, Fcy (ciclo de instrucción), es la mitad de la frecuencia de oscilación del microcontrolador (8MHz), por lo tanto, tiene un valor de 4 MHz.

$$U1BRG = \frac{4000000\ (Hz)}{4 \cdot 9600} - 1$$

Figura 41. Fórmula para el cálculo del registro U1BRG

- U1MODEbits.UARTEN = 1; Para finalizar, se debe activar el módulo UART1.
- U1STAbits.UTXEN = 1; Y una vez que se ha activado el módulo ya se puede activar la transmisión. Esta activación la incluimos en la inicialización del módulo debido a que desde el inicio se va a necesitar transmitir datos. Además, hasta que el registro de transmisión no contenga el dato que se desea informar no va a enviar información al computador.

Por otro lado, se encuentran las funciones de interrupción. Estas se emplean para limpiar la bandera y para llevar un control mediante LEDS que indican que se ha enviado un dato o recibido.

Se las ha designado por **_U1RXInterrupt** y **_U1TXInterrupt**, y sus registros en los que se indica que se limpia la bandera de interrupción son respectivamente IFS0bits.U1RXIF y IFS0bits.U1TXIF.

A mayores, en la interrupción de la recepción de datos se encuentra una rutina de control para determinar cuándo se crea el pulso. Si el usuario envía una “S” se llamará a la rutina de inicialización del Comparador de Salida para

crear el pulso en el momento que se desee. En caso de que no se quiera crear el pulso se redirige al menú inicial.

A continuación, te permite indicar el valor de ancho de pulso que se desea generar. Se puede introducir un valor o establecer uno por defecto, OC1=0x3000 y OC1RS=0x3003.

```
switch (U2RXREG){

    case 'S':

        printf("\n");
        _LATA0 = 1;     // Enciende el LED D3
        printf("Indique valores para los tiempos, pulse 'R' para establecer OC1 y 'T' para OCRS");
        printf("\n");
        printf("Sino se establecera el valor por defecto pulsando 'D'");
        printf("\n");
        break;
```

Figura 42. Menú para establecer los tiempos

```
switch (aux){
    case 1:
            printf("Valor del registro OC1: ");
            OC1R = U2RXREG;
            U2TXREG = OC1R;
            printf("\n");
            aux = 3;
            printf("Para indicar el valor de OC1RS pulse T");
            printf("\n");
            break;
```

Figura 43. Fragmento de código para la introducción del valor del registro OC1 por el usuario

```
    case 2:
            printf("\n");
            printf("Valor para registro OC1RS");
            printf("\n");
            printf("Indique que tipo de pulso desea crear");
            printf("\n");
            printf("Pulso por una patilla de ancho variable: Marca A");
            printf("\n");
            printf("Dos pulsos por una misma patilla y diferencia entre ellos variable: Marca B");
            printf("\n");
            printf("Dos pulsos, cada uno por una patilla y diferencia entre ellos variable: Marca C");
            printf("\n");
            aux = 3;

            break;
```

Figura 44. Fragmento de código para la introducción del valor del registro OC1RS por el usuario

```
    case 0:

            OC1R = 0x3000; // Iniciaiza el registro del comparador
            OC1RS = 0x3003; //Inicializa el registro del segundo comparador
            aux = 3;

            break;
    case 3:
        break;
```

Figura 45. Fragmento de código para el establecimiento de valores por defecto para los registros OC1 y OC1RS

Una vez que se han fijado los valores de tiempo que se quieren, se debe seleccionar que tipo de pulso se va a generar:

- Si se introduce una "A", se crea un único pulso por la patilla OC1.
- Si se introduce una "B", se crean dos pulsos, con distancia de tiempo variable por el usuario, y por la misma patilla OC1.
- Si se introduce una "C", se crean dos pulsos, uno por la patilla OC1, y otro por la patilla OC2.Con distancia de tiempo variable por el usuario.

Para la creación de los segundos pulsos, cuando desee el usuario deberá pulsar el botón S3. Una vez que lo pulse se procede a crear el nuevo pulso por la misma patilla que el anterior o por una distinta dependiendo del modo seleccionado inicialmente.

```
case 'C':
    printf("\n");
    init_OutputCompare ('1');
    printf("\n");
    printf("Pulsa el boton S3 cuando se desee crear el segundo pulso");
    printf("\n");
    while (!boton_presionado( )) ;
    init_OutputCompare ('2');
    break;
```

Figura 46. Fragmento de código para la creación de dos pulsos, cada uno por una patilla distinta y con distancia de tiempo variable

3.2.1. BOTONES

Como se ha indicado en el anterior apartado, ha sido necesario la creación de un módulo para configurar los botones. Este módulo es muy sencillo, solo consta de una función de inicialización **init_Botones**, para establecer el puerto D6 como entrada ya que es el asociado a este botón.

Por otro lado, se cuenta con la función booleana**, boton_presionado**, que indica cuándo el botón es presionado por el usuario y cuándo se encuentra en su estado de reposo.

```
/*************** Comprobacion de si se pulsa el boton S3*******************/
bool boton_presionado (void){
    return ( ( S3_PORT == 0 ) ? true : false ) ;
}
```

Figura 47. Función botón_presionado

4. PRUBEAS

Para la realización de las distintas pruebas a lo largo de la realización del trabajo ha sido necesario la implementación de algunos drivers y programas.

Para comenzar, al emplear el puerto serie ha sido necesario el uso del programa Hercules, que proporciona una interfaz sencilla de comunicación no solo por puerto serie, aunque en este caso es de la que se ha hecho uso. Además, se ha necesitado la descargar de un driver para implementar la lógica de comunicación serie RS-232 que se requería.

Una vez se ha tenido instaladas ambas cosas, se ha procedido a configurar el puerto USB que iba a hacer de conector con el computador para la comunicación. En este caso, se ha optado por establecer uno de los puertos USB como puerto COM6, que recibiría 8 bits sin paridad y con un bit de parada. Esto se ha hecho así, ya que la interfaz del programa Hercules pide seleccionar para que puerto se va a realizar la comunicación y el protocolo que va a seguir, y ambas configuraciones deben coincidir.

A continuación, se muestra la interfaz del programa Hercules:

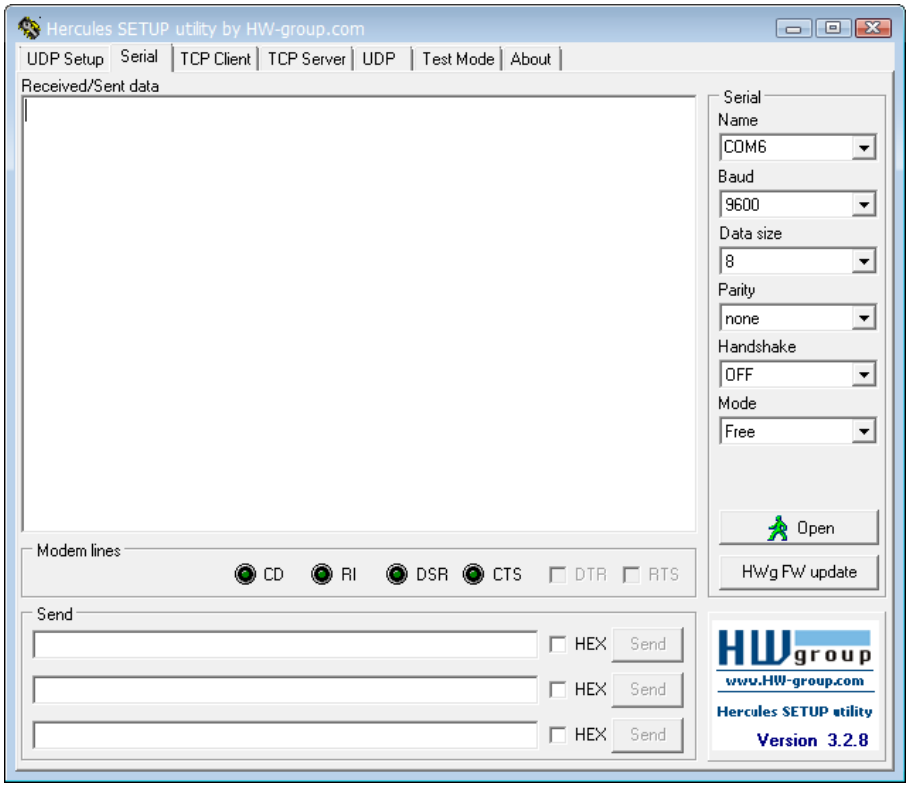


Figura 48. Interfaz del programa Hercules

Como se puede apreciar, en la parte superior se puede seleccionar el tipo de comunicación que se va a realizar, es este caso, en serie. En la banda izquierda, se puede seleccionar la configuración que se va a seguir, como ya se ha comentado, se emplea el puerto COM6, con una lógica 8N1 y con un

ratio de baudios de 9600. Este ratio es importante, ya que como se mostró en el apartado de UART, al calcular el valor del registro UxBRG se emplea ese ratio para su cálculo.

En la parte principal de la interfaz, se encuentra la zona en la que el usuario puede ver que mensajes llegan y se transmiten. En caso de querer enviar un dato al microcontrolador, desde las barras inferiores se puede escribir el carácter que se desea transmitir. También permite seleccionar el lenguaje en que se quiere mandar, optando en nuestro caso por el hexadecimal.

Como dato de interés, a la hora de hacer las pruebas era tedioso encontrar tantos mensajes de depuración en la pantalla. Como solución, desde la pantalla en la que se muestran los datos recibidos y transmitidos, si se hace click con el botón derecho te permite limpiar todo el registro que se tiene guardado, dejando la interfaz limpia de mensajes.

Por otro lado, como apuesta por interfaz que se pueda emplear en un futuro se ha optado por Visual Basic Express 2010, ya que es una herramienta de uso libre que dispone de muchos tutoriales para la creación de interfaces.

Como inicio, se propone dos pantallas, una primera de inicio en la que a través de un botón se accede a la aplicación:

Figura 49. Pantalla inicial de la propuesta de Interfaz en Visual Basic Express 2010

Y una segunda pantalla a la que se accede una vez que se ha pulsado el botón “Disparador de pulsos”. Esta pantalla es sencilla, dispone de 4

generadores de pulso, para las distintas opciones que desee implementar el usuario, pudiéndose crear más o menos botones. Si seleccionamos el Pulso 3 se comprueba que sale un mensaje de información mostrando que aún no ha llegado datos. Por otro lado, también posee funciones para mostrar recursos gráficos, por ejemplo, si se pulsa el botón de Pulso 1 te permite cargar una imagen.

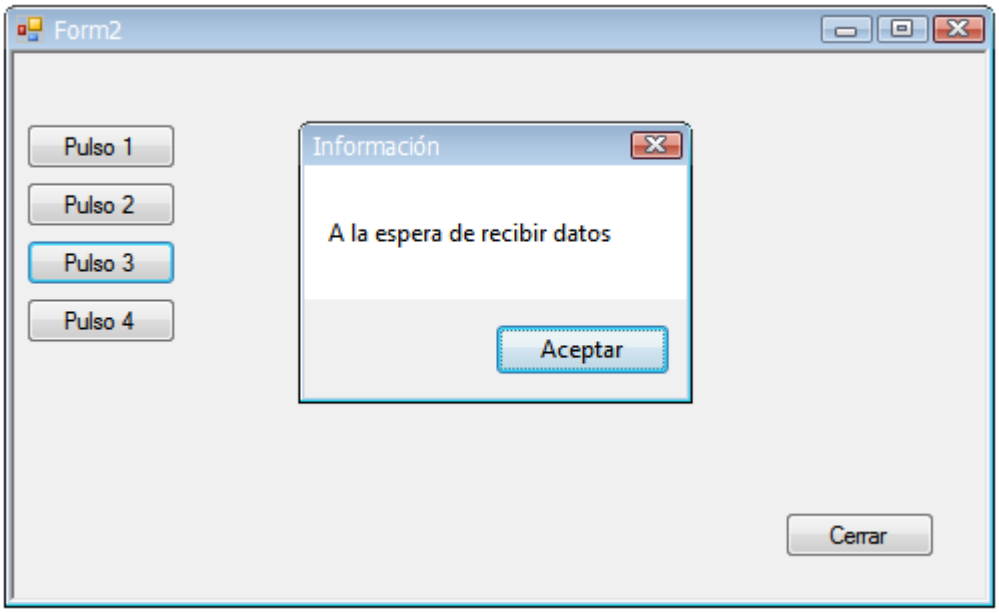


Figura 50. Pantalla secundaria de la propuesta de Interfaz en Visual Basic Express 2010

Evidentemente, en esta interfaz se podrían implementar más funciones, como, por ejemplo, poder seleccionar las pre-escalas y post-escalas que se quieren para los distintos módulos que hacen usos de ellas. O distintos modos de generación de pulsos, como pueden ser generar pulsos continuos, o con modulación del ancho de pulso... Además, dado que se puede trabajar con distintos recursos gráficos se podría incluir una interfaz en la que se muestre como se genera el pulso y la escala de tiempo en la que se está trabajando, así como el valor del temporizador que está corriendo. Para ello, habría que investigar como anexar ambos entornos para un correcto funcionamiento.

Por último, se adjunta una explicación sencilla de cómo realizar las pruebas y el traspaso de la codificación al microcontrolador.

1. Guardar el proyecto una vez que se ha configurado las funciones.
2. Establecer el proyecto como proyecto principal. Para ello, si hacemos click derecho sobre el nombre del proyecto aparece la opción “Set as Main Project”.

3. Determinar cómo se quiere depurar el proyecto. Para ello, se accede a la pestaña superior "Production" → "Set Project Configuration" → "Customize". A continuación, aparece una nueva ventana en la que se debe elegir el PICkit3 y el compilador XC16:

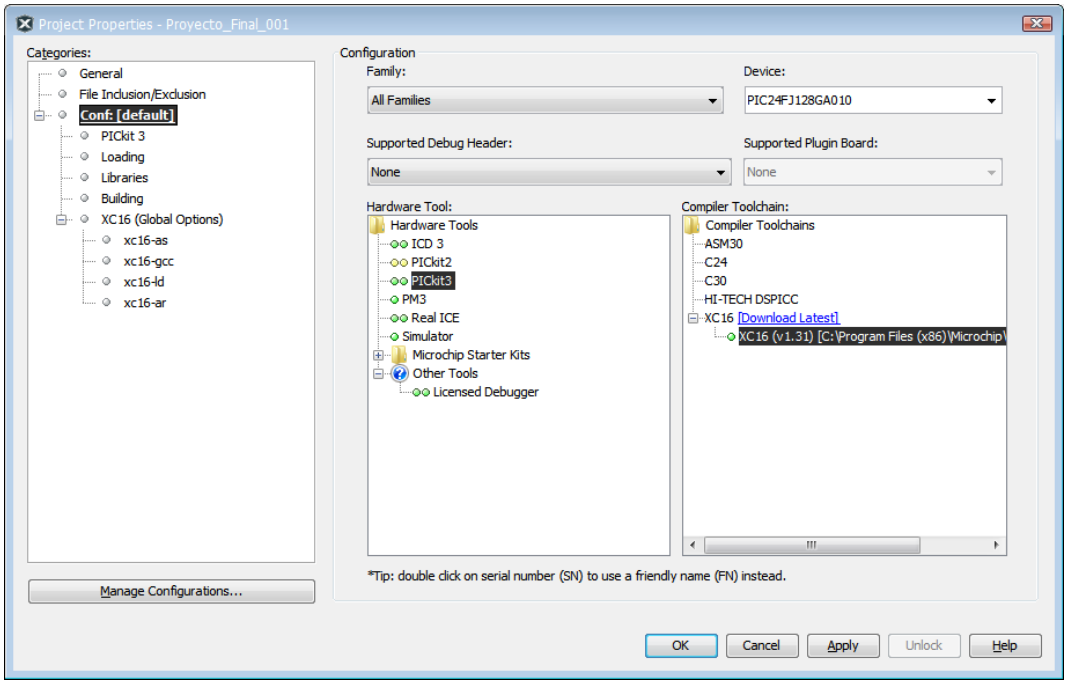


Figura 51. Pantalla de MPLAB X IDE para selección de las características de depuración

4. Una vez que está todo configurado, y se tiene conectada la placa a la fuente de alimentación y el PICkit3 a los terminales correspondientes, se pulsa el botón de traspasar la información al microcontrolador.

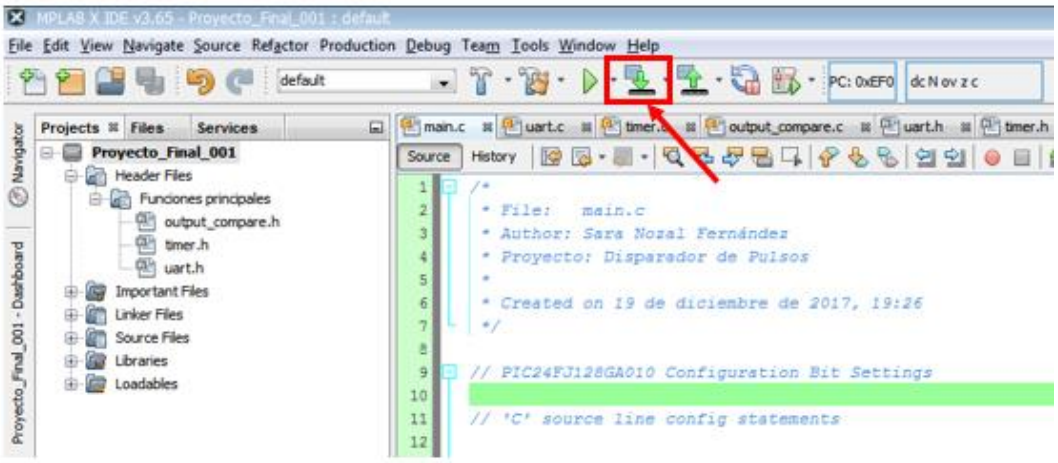


Figura 52. Icono de MPLAB X IDE para transferencia de datos del ordenador personal al PIC24

Cuando se pulsa este botón, ya se hace la depuración, la función de ensamblar y el proceso de enlazado. Se genera el fichero ensamblado que se

implementa en el micro, y aparece en pantalla el resultado de la depuración y de la transferencia de datos.

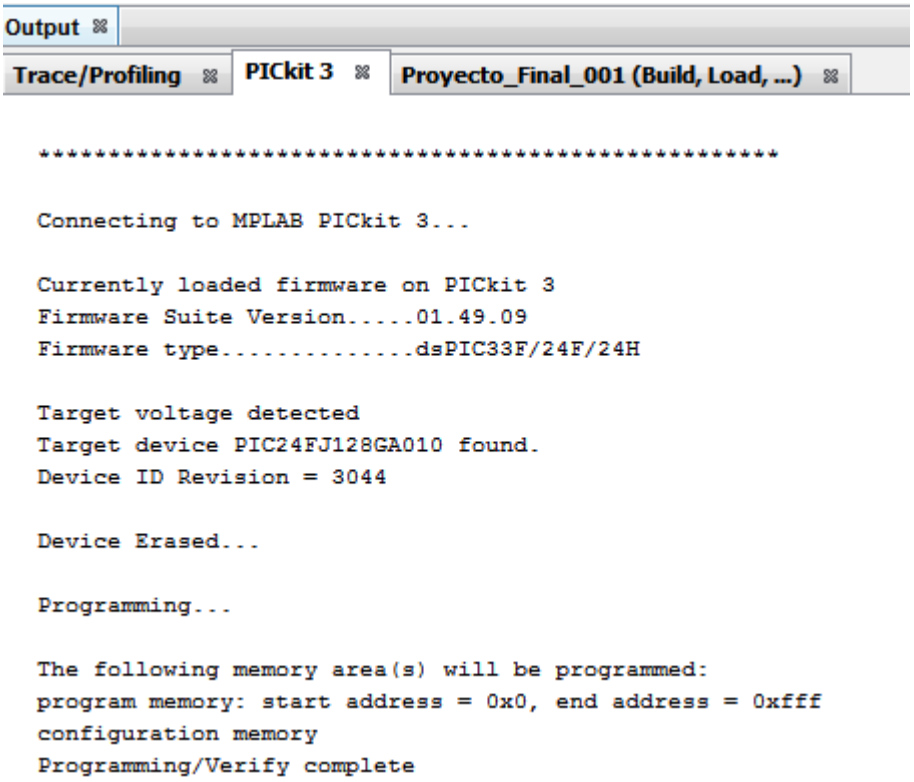


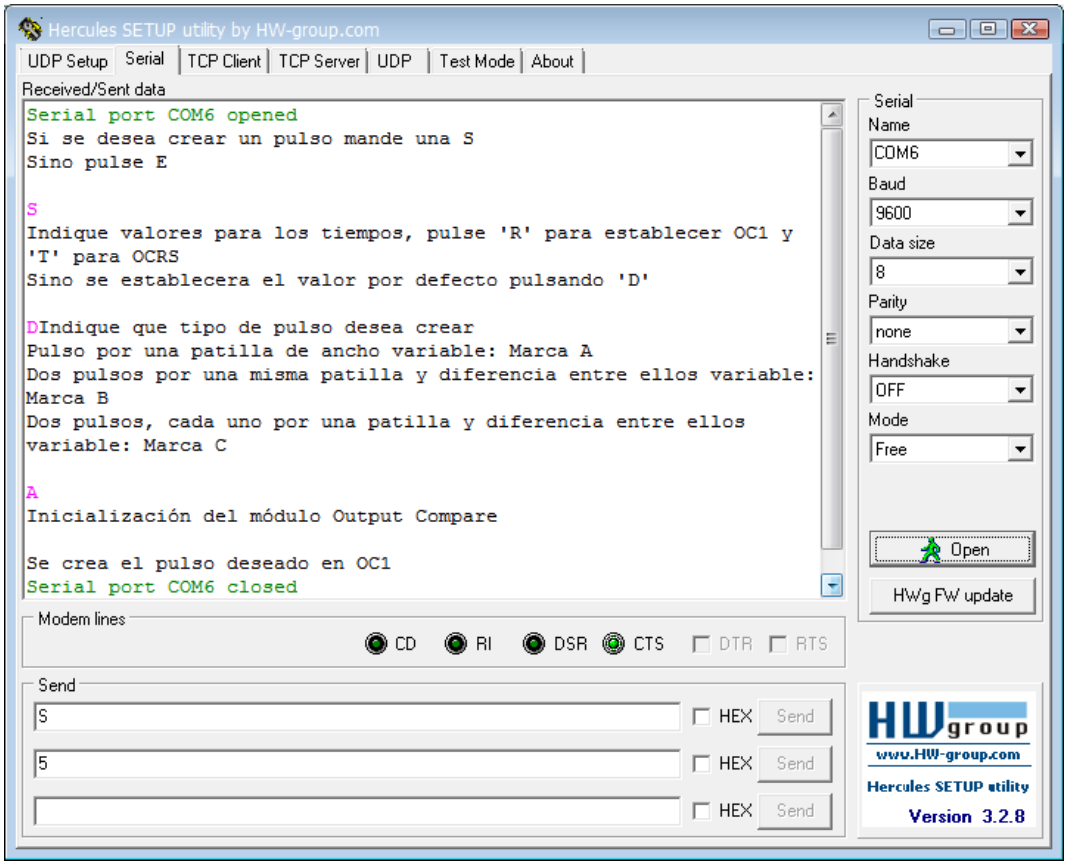
Figura 53. Mensaje de MPLAB X IDE para indicar la finalización de transferencia de datos

Una vez realizados los pasos, ya se tendría la tarjeta programada para comprobar el funcionamiento de las funciones que se han implementado.

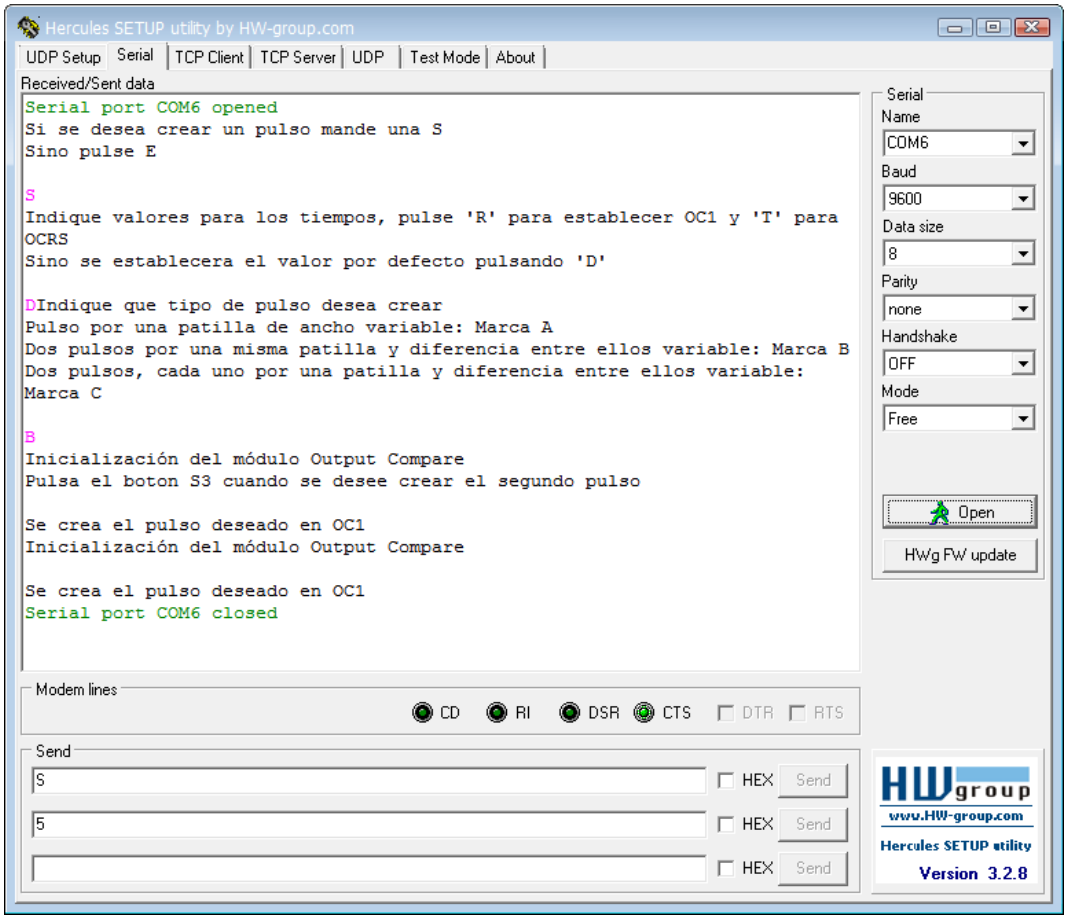
Para hacer pruebas con el polímetro de los distintos valores que adquirían las patillas del microcontrolador, se creó una placa de expansión que se ensambla con la placa Explorer 16 por medio del conector J2. A su vez, este estaba conectado al microcontrolador por medio de pistas. En el anexo A se adjunta un esquema de la placa de expansión.

Para terminar, a continuación se muestran una serie de pruebas con la interfaz de Hercules para comprobar que se generan bien los tres tipos de pulso:

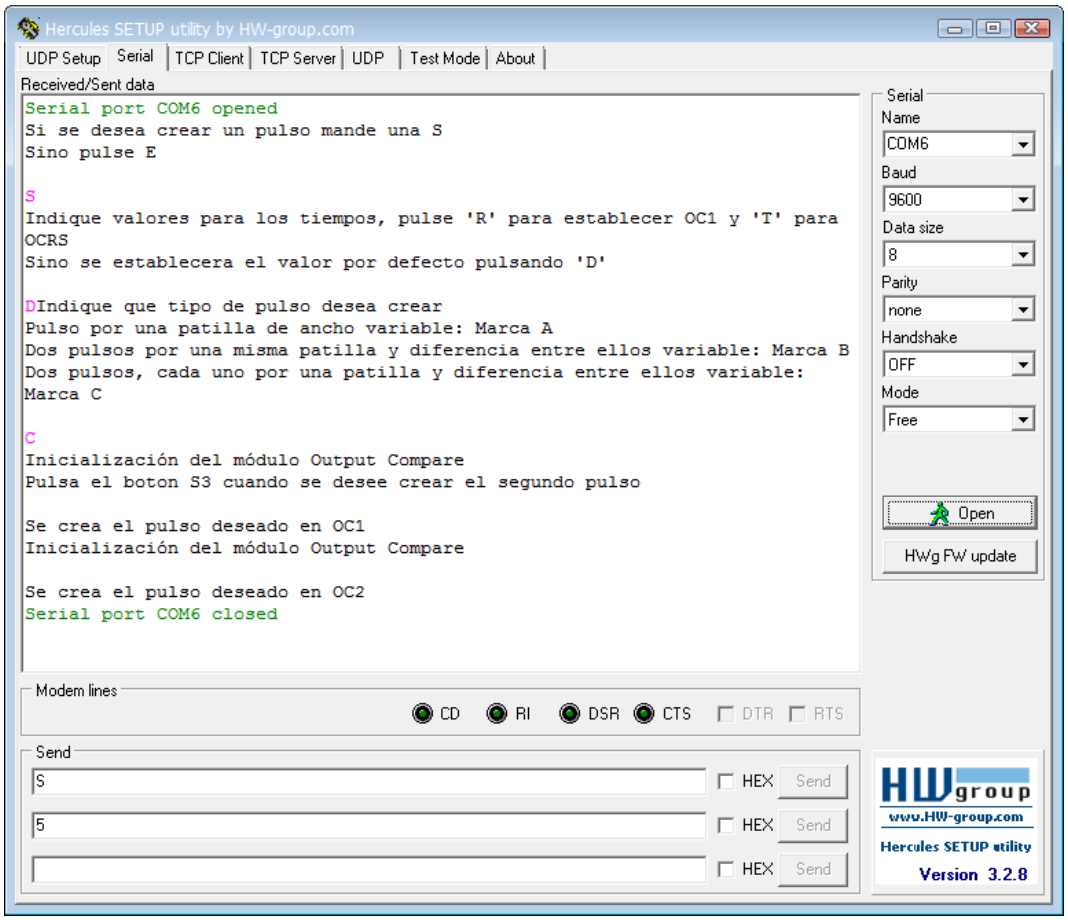
4.1. MODO A: UN ÚNICO PULSO DE ANCHO A ELECCIÓN DEL USUARIO



4.2. MODO B: DOS PULSOS, CON DISTANCIA DE TIEMPO VARIABLE POR EL USUARIO, Y GENERADOS EN LA MISMA PATILLA.



4.3. MODO C: DOS PULSOS, CON DISTANCIA DE TIEMPO VARIABLE POR EL USUARIO, Y GENERADOS EN DISTINTAS PATILLAS.



5. PROPUESTA DE DISEÑO HARDWARE

A pesar de haber trabajado a lo largo de este proyecto con la placa de Microchip Explorer 16, uno de los objetivos propuestos al inicio del trabajo era evaluar la posibilidad de migrar el sistema actual a uno más pequeño, rápido y fiable.

Debido a esto, se propone un modelo de placa adecuado a las necesidades de este estudio, omitiendo aquellas funciones de la placa Explorer 16 de las que no se ha hecho uso.

A continuación, se muestra un esquema general de los módulos que contendrá la futura placa:

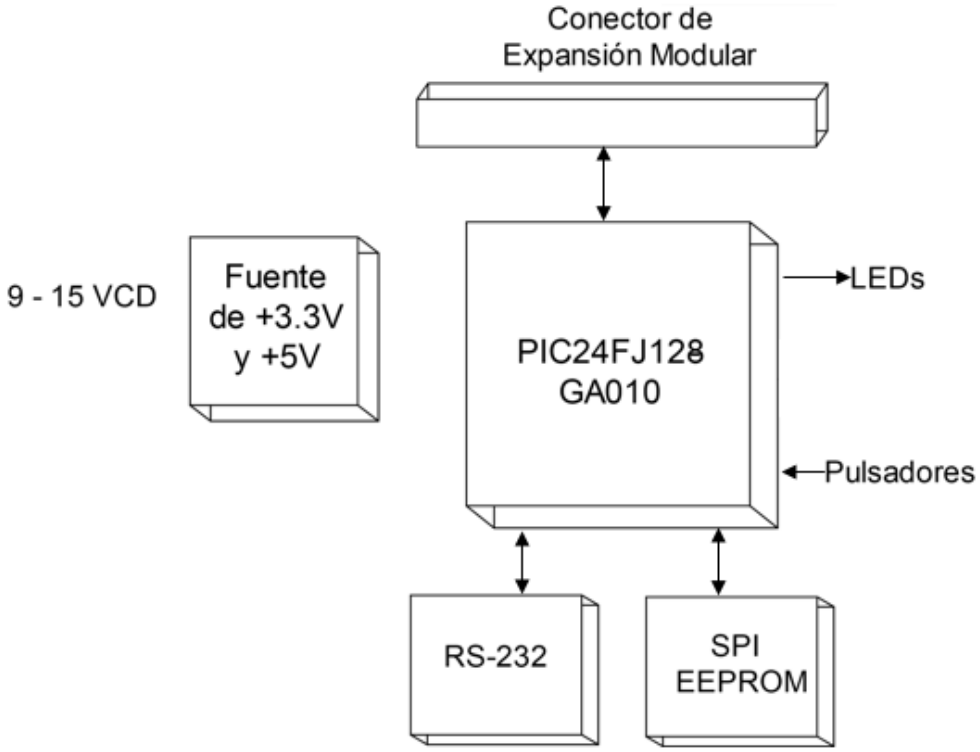


Figura 54. Esquema general de propuesta de placa

Como se puede comprobar, con un diseño más sencillo se pueden cubrir las necesidades funcionales.

Será necesario que la placa cuente con los siguientes módulos:

Primeramente, se cuenta con el microcontrolador PIC24FJ128GA010 montado de forma permanente. No se ha optado por adaptarlo con un procesador PIM de 100 patillas, debido a que estos se emplean para poder cambiar más fácilmente de modelo de mico controlador con mismo número de terminales,

y no se plantea la necesidad de estar cambiando asiduamente de microcontrolador.

Por otro lado, se implementa un adaptador de la fuente de tensión para conseguir voltajes con valor de +5 V y +3.3V. Esto se consigue mediante la puesta en paralelo de un par de capacitores para la obtención de cada voltaje.

También, será indispensable contar con un módulo de memoria EEPROM, ya que presenta las ventajas de ser una memoria ROM que puede ser programada, borrada y reprograma eléctricamente tantas veces como se desee. Algo muy útil, en caso de necesitar realizar modificaciones en el futuro de una manera sencilla. El componente de microchip elegido para realizar esta función es el 25LC256.

Otro módulo importante que se debe tener es un oscilador de cuarzo de 8MHz, porque a pesar de que se va a emplear un oscilador externo, en caso de que este falle o se rompa, solo sería necesario cambiar la palabra de configuración POSCMOD = XT para que el sistema siguiera funcionando aunque con alguna limitación. Por ejemplo, no se podría modificar la frecuencia a la que trabaja por medio de una interfaz.

Asimismo, se contará con un módulo para poder realizar las comunicaciones series mediante un UART con protocolo RS-232. El componente elegido de la marca Texas Instruments será el MAX3232.

Para terminar se contará con una serie de LEDs que actuarán a modo de aviso cuándo surja un evento. Y de un pulsador conectado a la patilla MCLR del PIC24 para poder resetear el microcontrolador cuando se necesite.

También, se ha optado por incluir un conector de expansión modular, para poder realizar pruebas más cómodamente aprovechando que se dispone de una placa ya creada que ha sido utilizada a lo largo de este proyecto.

Para finalizar, se incluye un conector para poder programar mediante el PICkit3.

Los esquemas eléctricos de montaje para esta placa se pueden encontrar en el anexo B, Esquemas eléctricos de propuesta hardware, al final del trabajo.

6. CONCLUSIONES

El objetivo final del proyecto era crear un sistema capaz de emplear un oscilador externo para crear pulsos a gusto del usuario para emplearlo como sistema de metrología en el laboratorio.

Al inicio de este trabajo se establecieron cuatro puntos básicos para conseguir este objetivo:

- Evaluación de la posibilidad de migrar a un nuevo entorno de desarrollo o IDE, basado en software libre, en lugar del actual sistema en uso.
- Estudio del modo de funcionamiento y lenguaje de los microcontroladores.
- Diseño y prueba de las funciones que permiten controlar los periféricos, así como de las tareas principales del sistema.
- Evaluación de las posibilidades de comunicación con la interfaz del operador.

Los puntos en los que más se han insistido han sido el segundo y el tercero, teniendo en cuenta que se empezó este proyecto sin conocimiento alguno de programación de microcontroladores. Al final de este periodo, se ha conseguido obtener un manejo adecuado de este tipo de programación y de trabajo con el entorno elegido. Por otro lado, el tercer punto es intrínseco al aprendizaje, ya que durante este ha sido necesario el estudio de los periféricos, además de efectuar su programación con sus consecuentes pruebas para la verificación de su funcionamiento.

De igual modo, se ha esclarecido que con el uso del entorno MPLAB X IDE junto con el microcontrolador es posible migrar el sistema actual a uno nuevo, más compacto y rápido.

Por otro lado, se ha hecho una propuesta de posible entorno a emplear para la creación de la interfaz, teniendo en cuenta que quedaría pendiente ver como anexar ambos entornos para que funcionen adecuadamente.

Asimismo, el uso del oscilador externo se ha estudiado y se ha explicado cómo se debería programar y conectar si se decidiera finalmente hacer uso de él. En este aspecto, sería necesario realizar las pruebas pertinentes para que la aplicación funcionara debidamente. Sería aconsejable, implementar programación en lenguaje ensamblador, ya que para el cálculo de tiempos de las instrucciones es más asequible y fiable.

Por último, sería recomendable, una vez conseguido el funcionamiento final con interfaz, implementar el cálculo de errores del dispositivo que se desea

medir respecto del pulso generado, para obtener la información de forma sencilla y rápida.

7. GLOSARIO

"Breakpoints": Es una pausa intencional y controlada durante la ejecución o depuración de un programa.

Búfer: Espacio de la memoria en un disco o en un instrumento digital reservado para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

CPU: Unidad central de procesamiento o unidad de procesamiento central, es el hardware dentro de un ordenador u otros dispositivos programables, que interpreta las instrucciones de un programa informático mediante la realización de las operaciones básicas aritméticas, lógicas y de entrada/salida del sistema

Disparador Schmitt: Es un tipo especial de circuito comparador, se emplea para evitar el ruido.

Depurador: Es un programa usado para probar y depurar (eliminar) los errores de otros programas.

EEPROM: Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente y que no es volátil.

Hardware: Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

Interfaz de usuario: Medio por el que una persona se puede comunicar con una máquina, es este caso, el Disparador de pulsos. Está compuesta por los sistemas de contacto entre el usuario y el equipo, como puede ser, el monitor de la pantalla, el teclado o el ratón.

Memoria ROM: Es una memoria de sólo lectura que se utiliza para almacenar los programas que ponen en marcha el ordenador y realizan los diagnósticos.

Microcontrolador: Circuito integrado el cual será el componente principal de una aplicación embebida, esto implica que está destina para la realización de una tarea específica. Incluye sistemas para controlar elementos de entrada/salida a modo de pequeña computadora. También incluye a un procesador y memoria, tanto flash como RAM, donde se podrá guardar el programa y sus variables.

Oscilador: Generador de corriente alterna a partir de corriente continua. Su principal ventaja es que la frecuencia que genera no es fija, sino que el usuario la puede modificar a su gusto.

PLL: Sistema basado en la retroalimentación de la frecuencia y fases. Se emplea a menudo para crear osciladores muy estables.

Puerto Serie: Interfaz física que permite el envío y recepción de datos digitales mediante la conexión a un periférico. Se caracteriza por transmitir bit a bit, ser de bajo costo y su sencillez.

Señal de reloj: Señal empleada en la coordinación de acciones de dos o más circuitos. Una señal de reloj oscila entre estado alto o bajo, y gráficamente toma la forma de una onda cuadrada.

Software: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

8. BIBLIOGRAFÍA

[1] Adobe (2018). *Creative Cloud Illustrator*. Recuperado de:

https://www.adobe.com/es/products/illustrator.html

[2] Data Sheet Microchip (2003-2013). *256K SPI Bus Serial EEPROM.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/21822G.pdf

[3] Data Sheet Microchip (2005-2012). *PIC24FJ128GA010 Family. 64/80/100-Pin, General Purpose, 16-Bit Flash Microcontrollers.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39747F.pdf

[4] Data Sheet Texas Instruments (1989-2014). *MAX232x Dual EIA-232 Drivers/Receivers.* Recuperado de:

http://www.ti.com/lit/ds/symlink/max232.pdf

[5] Manual Microchip (2006). *Interrupts.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39707a.pdf

[6] Manual Microchip (2006). *Output Compare.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39706a.pdf

[7] Manual Microchip (2006). *Timers.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39704a.pdf

[8] Manual Microchip (2007). *Data Memory.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39717a.pdf

[9] Manual Microchip (2007). *Real-Time Clock and Calendar (RTCC).* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39696b.pdf

[10] Manual Microchip (2007). *UART.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/en026583.pdf

[11] Manual Microchip (2008). *Input Capture with Dedicated Timer.*

Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/InputCapture39722a.pdf

[12] Manual Microchip (2009). *Oscillator.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39700c.pdf

[13] Manual Microchip (2009). *PICkit™ 3 Programmer/Debugger User's Guide.* Recuperado de:

https://www.sparkfun.com/datasheets/Programmers/PICkit_3_User_Guide_51795A.pdf

[14] Manual Microchip (2011*). Data EEPROM.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39720b.pdf

[15] Manual Microchip (2012). *PIC24FJ128GA010 Family Data Sheet.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/39747F.pdf

[16] Manual Microchip (2014). *Explorer 16 Development Board User's Guide.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/50001589b.pdf

[17] Manual Microchip (2016). *MPLAB® XC16 ASSEMBLER, LINKER AND UTILITIES User's Guide.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/50002106C.pdf

[18] Manual Microchip (2016). *MPLAB® XC16 C Compiler User's Guide.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/50002071F.pdf

[19] Manual Microchip (2016). *MPLAB® XC16 User's Guide for Embedded Engineers.* Recuperado de:

http://ww1.microchip.com/downloads/en/DeviceDoc/50002446B.pdf

[20] Microsoft (2015). *Visual Basic Express 2010. Crear e Implementar Interfaces*. Recuperado de:

https://docs.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/interfaces/walkthrough-creating-and-implementing-interfaces

[21] Microsoft (2015). *Visual Basic Express 2010. Definir Clases.* Recuperado de:

https://docs.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/objects-and-classes/walkthrough-defining-classes

[22] MikroElectronika (2018). *Microcontroladores PIC –Programación en C con ejemplos.* Recuperado de:

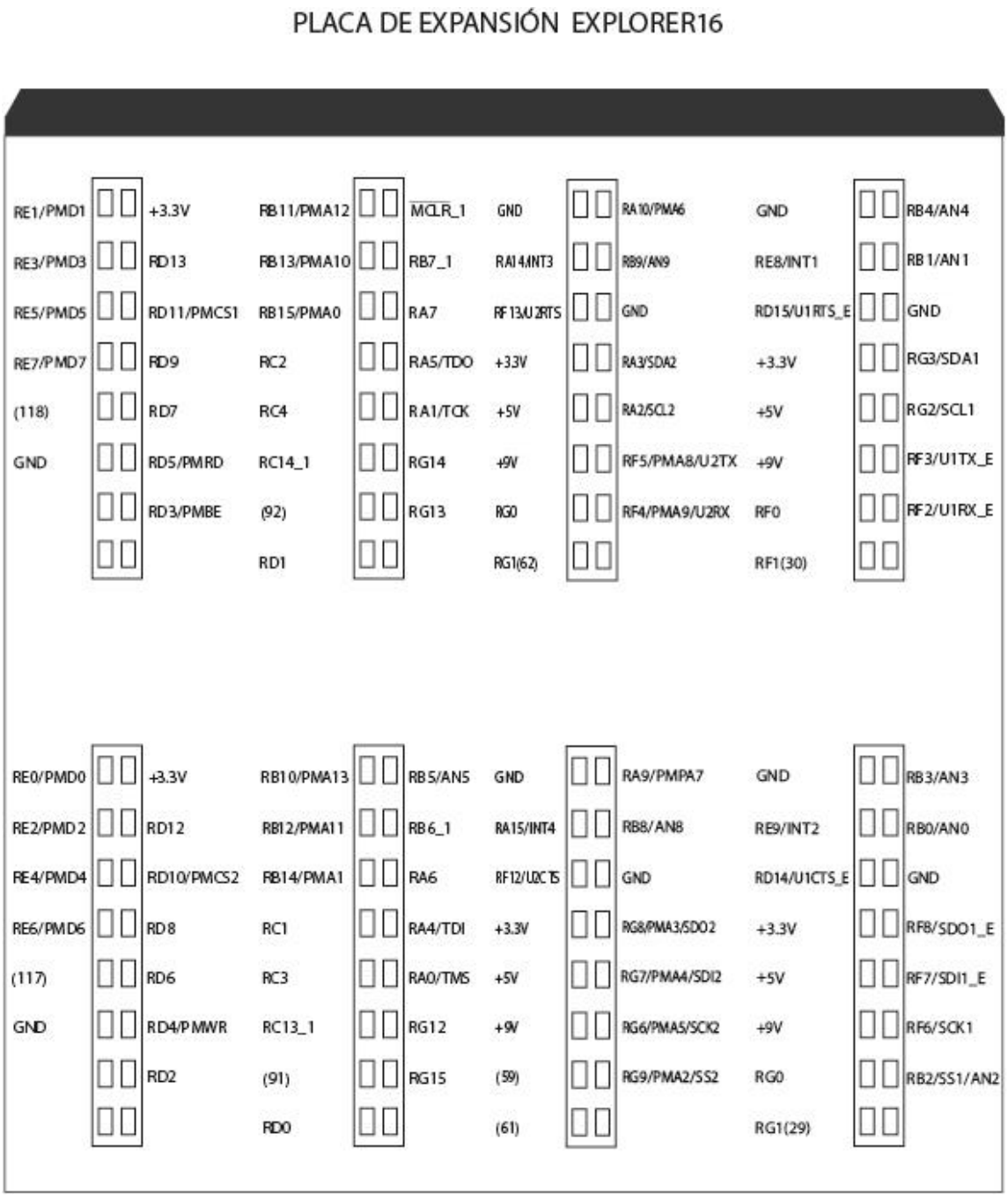
https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/ejemplo-3

[23] Real Academia Española (1993-2018). Recuperado de:

http://www.rae.es/

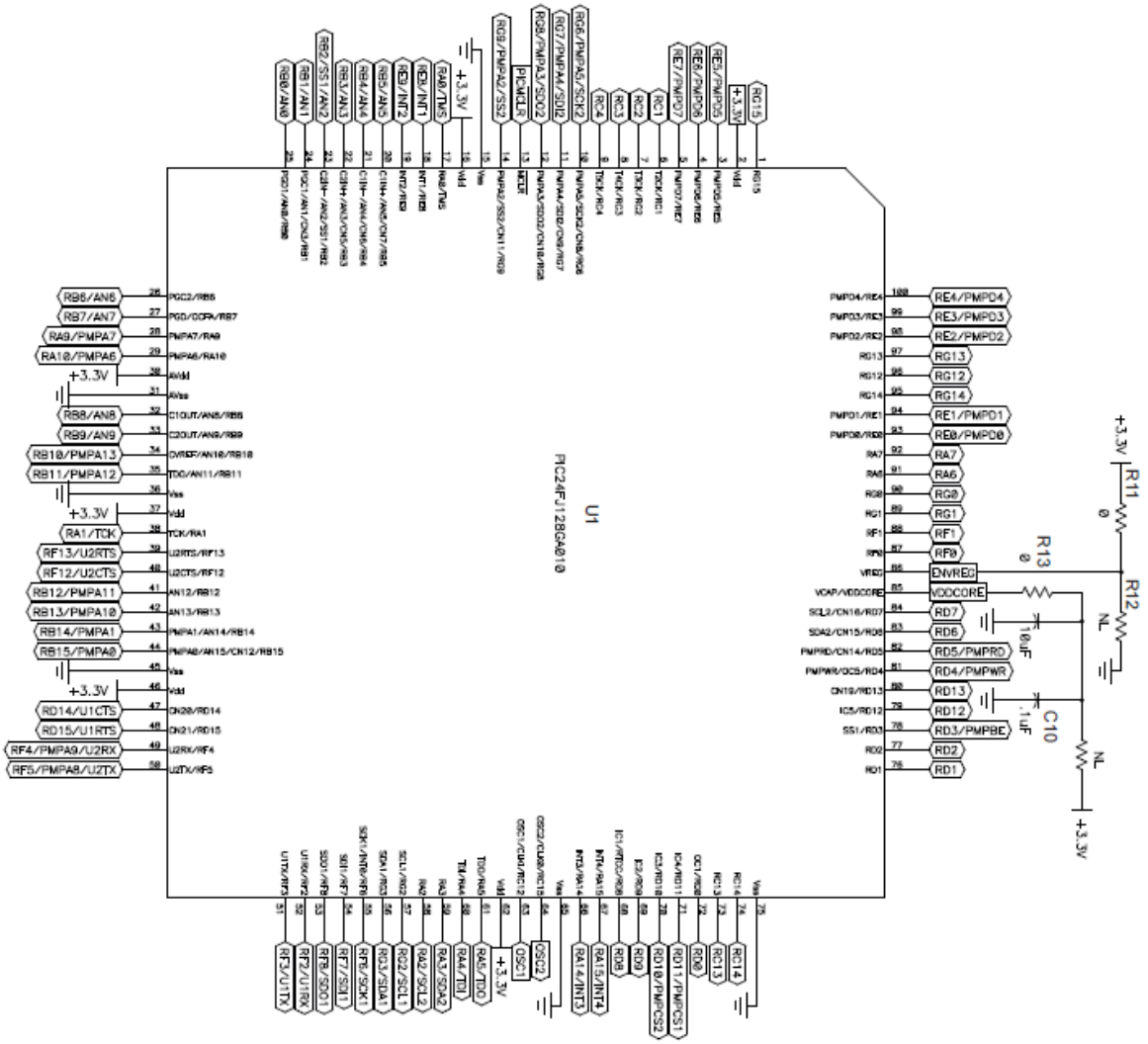
9. ANEXO

9.1 ANEXO A: DISEÑO DE LA PLACA DE EXPANSIÓN

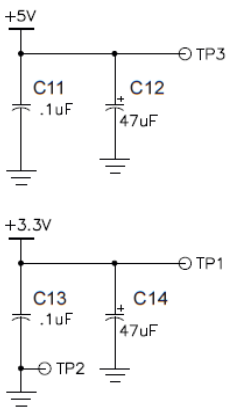


9.2 ANEXO B: ESQUEMA ELÉCTRICO DE PROPUESTA HARDWARE

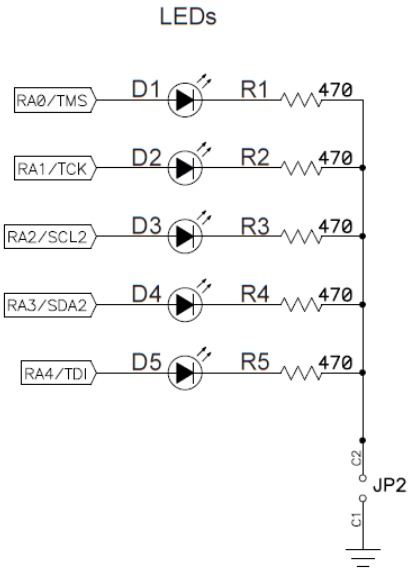
9.2.1 PIC24FJ128GA010 Y FUENTE DE ALIMENTACIÓN

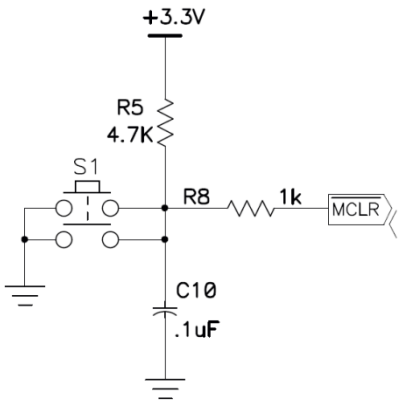


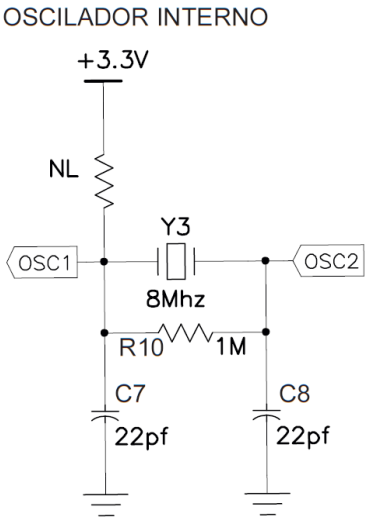
FUENTE DE ALIMENTACIÓN

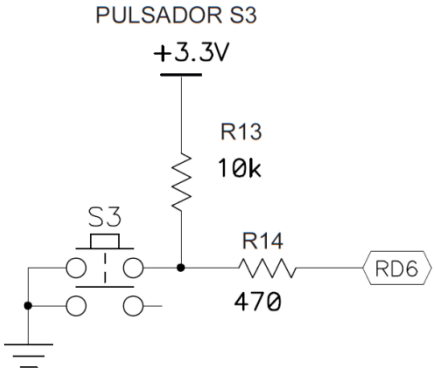


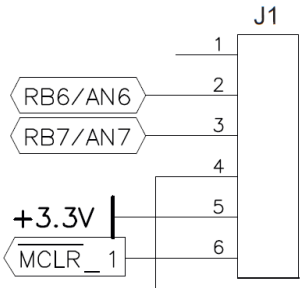
9.2.2 LEDS, PULSADORES, CONRECTOR PICKIT3, OSCILADOR INTERNO MEMORIA EEPROM Y UART

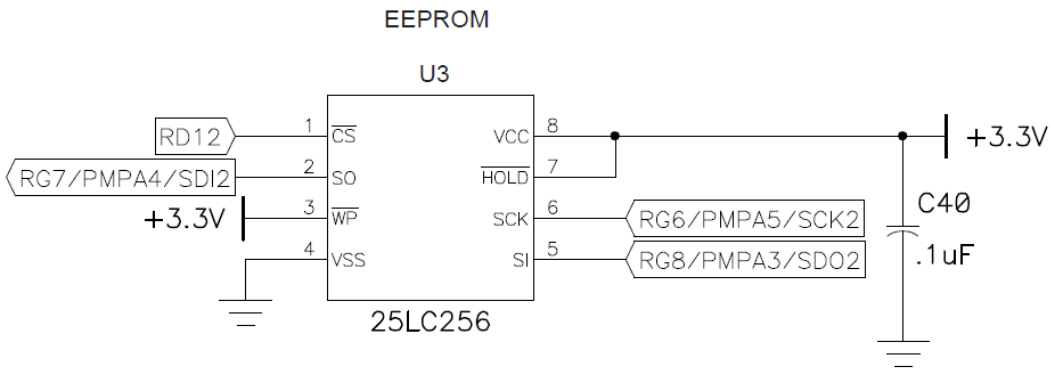












9.2.3 CONECTOR DE EXTENSIÓN MODULAR