CENTRO DE CIENCIAS APLICADAS Y DESARROLLO TECNOLÓGICO
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Manual de Uso

Introducción al modo de bajo consumo de energía de un microcontrolador MSP430FR5969

Miguel Angel Bañuelos Saucedo

Febrero de 2017

Proyecto de desarrollo tecnológico

Institución Patrocinadora

Hospital General "Dr. Manuel Gea González"

Financiamiento externo

Título: Introducción al modo de bajo consumo de energía de un microcontrolador MSP430FR5969

Autor: Miguel Angel Bañuelos Saucedo
Afiliación: Centro de Ciencias Aplicadas y Desarrollo Tecnológico, Universidad Nacional Autónoma de México

Resumen

En este manual se presenta un ejemplo de aplicación de bajo consumo de energía para un microcontrolador de 16 bits MSP430FR5969, que forma parte de una tarjeta de evaluación LaunchPad del fabricante Texas Instruments. El ejemplo consiste en generar una señal de tren de pulsos y se describe en detalle la configuración del sistema de reloj y el temporizador. Como resultado, se obtiene un consumo de 0.95 µA. Se estima que bajo esas condiciones, una pila de litio tipo reloj CR2032 podría mantener operando al sistema por más de 20 años. En un anexo, se incluyen los listados completos de los programas utilizados.

Índice

Nomenclatura

Símbolo	Significado
ADC	*Analog to Digital Converter*
ACLK	*Auxiliary Clock*
CPU	*Central Processing Unit*
CCS	*Code Composer Studio*
DCO	*Digital Controlled Oscillator*
DMA	*Direct Memory Acces*
EEPROM	*Electrically Erasable Programable Read Only Memory*
FRAM	*Ferroelectric Random Access Memory*
GIE	*General Interrupt Enable*
I^2C	*Inter-integrated circuit. Interfaz de comunicación serial entre circuitos integrados*
LED	*Light Emmiting Diode*
LPM	*Low Power Mode*
MCLK	*Main Clock*
MODOSC	*Module Oscillator*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
ROM	*Read Only Memory*
RTC	*Real Time Clock*
RISC	*Reduced Instruction Set Computer*
HFXT	*Se utiliza para identificar a un cristal oscilador de alta frecuencia*
LFXT	*Se utiliza para identificar a un cristal oscilador de baja frecuencia*
SMCLK	*Secondary Master Clock*
FLASH	*Tipo de memoria EEPROM que se borra por bloque*
UART	*Universal Asynchronous Receiver Transmitter*
USB	*Universal Serial Bus*
VLO	*Very Low power Oscillator*
WTD	*WaTchDog*

Introducción

Existen muchos microcontroladores de gran capacidad y con muchas funciones; sin embargo, la potencialidad de estos dispositivos a menudo no va respaldada por una cantidad adecuada de libros donde se detalle su manejo. Por otro lado, las características de estos dispositivos los hace tan versátiles, que es difícil poder cubrir todas las aplicaciones y configuraciones posibles. Es así que la información para la aplicación de los microcontroladores puede estar esparcida en varios manuales y en cientos o miles de páginas. Por otro lado, un área importante de aplicación de los microcontroladores la constituye los sistemas portátiles de bajo consumo de energía. En este manual se presenta el desarrollo de un ejemplo de bajo consumo de energía a partir de un microcontrolador de 16 bits MSP430FR5969 instalado en una tarjeta de evaluación LaunchPad del fabricante Texas Instruments. Se describe el proceso de configuración del complejo sistema de reloj y de un temporizador que genera una señal tren de pulsos. También se aplican algunas técnicas para reducir el consumo de energía y se muestran los resultados obtenidos.

Aunque el autor ha intentado describir el proceso de configuración de manera detallada, se sugiere que el lector cuente con antecedentes generales de programación en lenguaje C para microcontroladores, experiencia en el manejo del ambiente de programación Code Composer Studio, y conocimientos avanzados de arquitectura de microcontroladores.

1 Características del microcontrolador MSP430FR5969

El microcontrolador MSP430FR5969 es un miembro de la familia de microcontroladores de 16 bits de la empresa Texas Instruments, lanzado al mercado en junio de 2014 (Texas Instruments, 2014). Las letras 'FR' en la matrícula del microcontrolador resaltan el empleo de memoria FRAM (*Ferroelectric random access memory*), e identifican a una familia de microcontroladores que la citada empresa inició en marzo de 2012 con el modelo MSP430FR57XX. La familia MSP430 tiene más de 20 años en el mercado y se han producido miles de millones de unidades (Texas Instruments, 2017). La familia MSP430 son microcontroladores de conjunto reducido de instrucciones (RISC, *Reduced Instruction Set Computer)* con juego de registros ortogonal, es decir, que cualquier registro interno puede funcionar como registro acumulador. Las principales características del microcontrolador MSP430FR5969 (Texas Instruments, 2015c) se muestran en la Tabla **1**.

CARACTERÍSTICA	VALOR O INTERVALO
Frecuencia de operación	0 a 16 MHz
Voltaje de operación	1.8 V a 3.6 V
Consumo en modo activo	100 µA/MHz
Consumo en modo de espera	0.4 µA (Típico)
Memoria RAM ferroeléctrica	64 kB
Memoria RAM estática	2 kB
Convertidor analógico-digital	12 bits y hasta 16 canales, 300 kS/s
Sistema de reloj flexible	Reloj interno, externo, y de bajo consumo
Multiplicador en hardware	32 bits
Temporizadores	5 con funciones de captura-compara
Comunicación serial	UART (Nota: No cuenta con I^2C)

Tabla 1 Características principales del microcontrolador MSP430FR5969.

Tradicionalmente los microcontroladores de 16 bits, están limitados a un espacio de direccionamiento de memoria de 64 kB; sin embargo, el microcontrolador MSP430FR5969 se basa en la nueva arquitectura MSP430X, que cuenta con un bus de direcciones de 20 bits lo que le permite tener un espacio de direccionamiento de memoria de 1 MB sin necesidad de paginación. Adicionalmente los registros internos son ahora de 20 bits (con excepción del registro de *estado*), según se muestra en la Fig. 1.

Una de las características que distingue a este microcontrolador de otras familias y de otros microcontroladores en general, es el uso de memoria ferroeléctrica (FRAM) para el almacenamiento del programa y de los datos. Normalmente los microcontroladores cuentan con memoria del tipo EEPROM o FLASH para estas funciones; sin embargo, se reporta que la memoria ferroeléctrica consume menos energía en operaciones de lectura y escritura (Texas Instruments, 2016). La memoria tipo EEPROM, por ejemplo, requiere de un circuito elevador de voltaje durante el proceso de escritura. Es así que el microcontrolador MSP430FR5969 es una opción para el desarrollo de aplicaciones que requieran de bajo consumo de energía. La memoria FRAM, al igual que la memoria EEPROM, no consume energía para almacenar la información; sin embargo, la memoria FRAM tiene ciclos de escritura 100 veces más rápidos que la memoria FLASH.

Fig. 1 Mapa de registros de propósito general.

Una de las características más importantes a estudiar de un microcontrolador es su mapa de memoria, debido a que ahí se identifica la ubicación de algunos recursos importantes, por ejemplo: la dirección inicial de programa al encender el microcontrolador, las direcciones de los vectores de interrupciones, el almacenamiento de la pila *(stack*), y el mapa de puertos. En la Tabla **2** se muestra el mapa de memoria del microcontrolador MSP430FR5969. Se puede observar que se tienen asignadas direcciones de memoria por encima de los 64 kB (localidades superiores a FFFFh), de manera que se pueden completar los 64 kB de memoria de programa a pesar de tener incrustados 2 kB de memoria RAM dentro de los primeros 64 kB del mapa de memoria. Es importante mencionar que muchos de los detalles del mapa de memoria se vuelven transparentes para el programador que usa un lenguaje de alto nivel como el lenguaje C.

Otra característica que define la capacidad de un microcontrolador es el diagrama de bloques de sus recursos internos. En la Fig. **2**, se muestra el diagrama de bloques interno del microcontrolador MSP430FR5969; entre ellos, los bloques de temporizadores (*timers*), el reloj de tiempo real (*RTC)*, el convertidor analógico-digital, y el sistema de reloj.

		MSP430FR59x9
Memoria FRAM Principal: Vectores de interrupción Principal: Memoria de programa		63 kB 00FFFFh - 00FF89h 013FFFh - 004400h
RAM		2 kB 0023FFh - 001A00h
Información de descripción de dispositivos		256 B 001AFFh - 001A00h
Información de memoria (FRAM)	Info A	128 B 0019FFh - 001980h
	Info B	128 B 00197Fh - 001900h
	Info C	128 B 00187fh - 001880h
	Info D	128 B 00187Fh - 001800h
Cargador (Bootstrap loader, ROM)	BSL 3	512 B 0017FFh - 001600h
	BSL 2	512 B 00115FFh - 001400h
	BSL 1	512 B 0013FFh - 001200h
	BSL 0	512 B 0011FFh - 001000h
Periféricos		4 kB 000FFFh - 0h

Tabla 2 Mapa de memoria del microcontrolador MSP430FR59659.

Para el desarrollo de aplicaciones de bajo consumo de energía, es vital tener diferentes posibilidades de configuración del reloj del microcontrolador; ya que a mayor frecuencia de operación se tiene un mayor consumo de energía, por lo que es deseable utilizar una fuente de reloj de baja frecuencia, y tener control sobre la señal de reloj que se aplica a los diversos periféricos del sistema. Por ejemplo, es recomendable tener la posibilidad de alimentar mediante un reloj de baja frecuencia a un periférico (temporizador, o convertidor analógico-digital) y desconectar la señal de reloj de la unidad central de proceso (*CPU*, por sus siglas en inglés), con la finalidad de reducir el consumo de energía. Posteriormente, cuando el periférico termine su tarea, se deberá reactivar el reloj del CPU, y dependiendo de la aplicación, el tiempo de reactivación puede ser un parámetro importante. En la sección 2 se describe la operación del sistema de reloj.

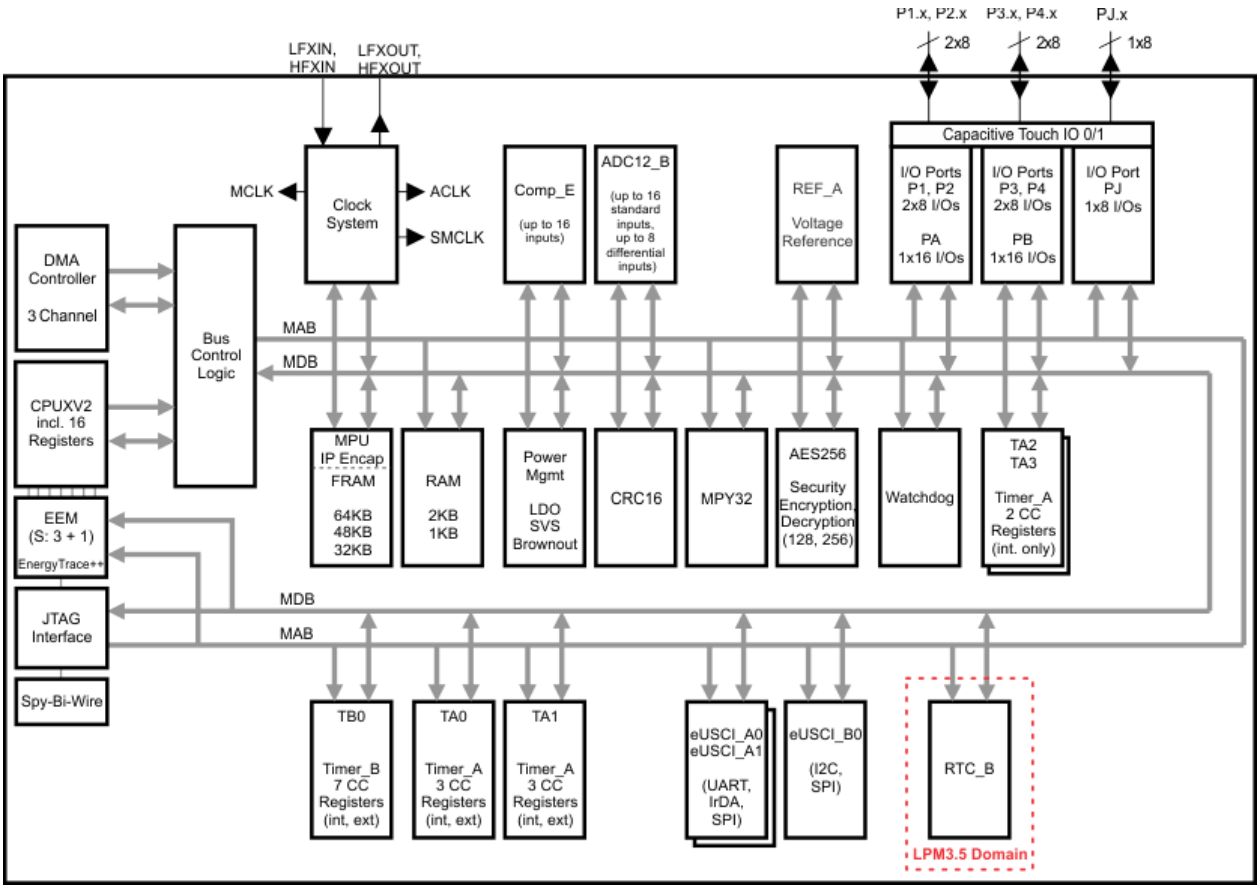

Fig. 2 Diagrama de bloques de los recursos del microcontrolador MSP430FR5969 (Texas Instruments, 2015c).

1.1 La tarjeta de evaluación MSP430FR5969 LaunchPad

Para la evaluación del microcontrolador y su configuración en modo de bajo consumo se utilizó la tarjeta de evaluación MSP-EXP430FR5969 LaunchPad de Texas Instruments (Texas Instruments, 2015b), la cual tiene un costo de $15.99 dólares americanos más gastos de envío, y se puede conseguir a través del fabricante o de distribuidores. Ésta es la única tarjeta de evaluación que tiene el microcontrolador MSP430FR5969, aunque hay otras disponibles con otros microcontroladores de las familias MSP430 y MSP430FR. La ventaja de utilizar la tarjeta LauchPad, es que ya incluye un programador e interfaz de conexión USB para usar con una computadora personal. La tarjeta LaunchPad cuenta con pines para conectar módulos de expansión (booster pack), y para realizar pruebas básicas tiene dos botones de presión y dos LED. Para evaluar el modo de operación de bajo consumo cuenta con un súper capacitor, conectores para utilizar una fuente de poder externa y terminales para medir el consumo de corriente. En la Fig. **3** se muestra una imagen de la tarjeta de evaluación. En la parte inferior se observa el microcontrolador, el cual tiene unas dimensiones de 7 mm x 7 mm. Esta tarjeta no cuenta con un cristal de alta frecuencia, pero su reloj interno puede funcionar hasta 16 MHz. Adicionalmente, existe instalado un cristal de 32.768 kHz como opción para operación en modo de bajo consumo.

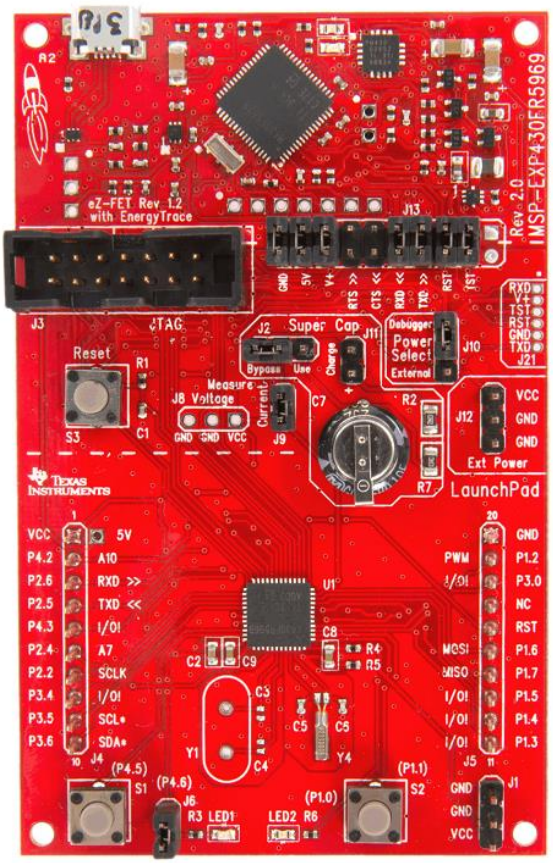

Fig. 3 MSP-EXP430FR5969 LaunchPad (Texas Instruments, 2015b).

2 El sistema de reloj

El sistema de reloj del microcontrolador MSP430FR5969 es muy versátil y cuenta con fuentes de reloj internas y externas. Las externas pueden ser un reloj de baja frecuencia (LFXT), el cual soporta un cristal de 32.768 kHz, o alta frecuencia (HFXT) de 4 MHz a 24 MHz (dependiendo de la versión). Las fuentes internas pueden ser un oscilador de muy baja potencia (VLO) de 10 kHz típicos, un oscilador controlado digitalmente (DCO), y un oscilador de baja potencia (MODOSC) con frecuencia típica de 5 MHz. A partir de estas 5 fuentes de reloj, se pueden generar seis diferentes señales de reloj, según se muestra en la Fig. **4**. Las señales de reloj principales son el reloj principal MCLK, el reloj secundario SMCLK, y el reloj auxiliar ACLK. En términos generales el sistema de reloj puede considerarse como un sistema de interconexión entre las fuentes de reloj y las señales de reloj, que además incluye módulos de división de frecuencia. El diagrama de bloques completo del sistema de reloj se muestra en la Fig. **5**.

El microcontrolador permite seleccionar modos de operación de baja potencia de acuerdo con la Tabla **3**. Por ejemplo, el modo LPM3, sólo utiliza el reloj auxiliar (ACLK) para alimentar periféricos tales como temporizadores, convertidor analógico-digital o reloj de tiempo real (RTC), mientras que el CPU permanece desconectado.

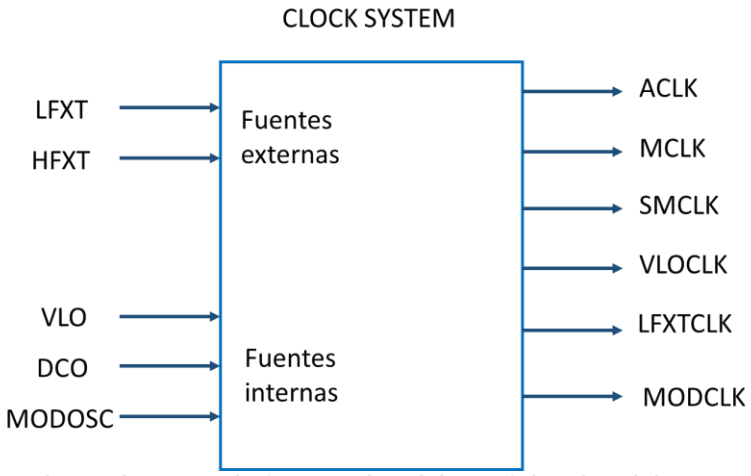

Fig. 4 Diagrama de fuentes de reloj y señales de reloj**.**

Low-Power Modes

Operating Mode	CPU (MCLK)	SMCLK	ACLK	RAM Retention	BOR	Self Wakeup	Interrupt Sources
Active	☒	☒	☒	☒	☒		Timers, ADC, DMA, WDT, I/0, External Interrupt, COMP, Serial, RTC, other
LPM0		☒	☒	☒	☒	☒	
LPM1		☒	☒	☒	☒	☒	
LPM2			☒	☒	☒	☒	
LPM3			☒	☒	☒	☒	
LPM3.5					☒	☒	External Interrupt, RTC
LPM4				☒	☒		External Interrupt
LPM4.5					☒		External Interrupt

LPM is great, but waking up...

Tabla 3 Diferentes tipos de modo de operación de baja potencia (Texas Instruments, 2015a).

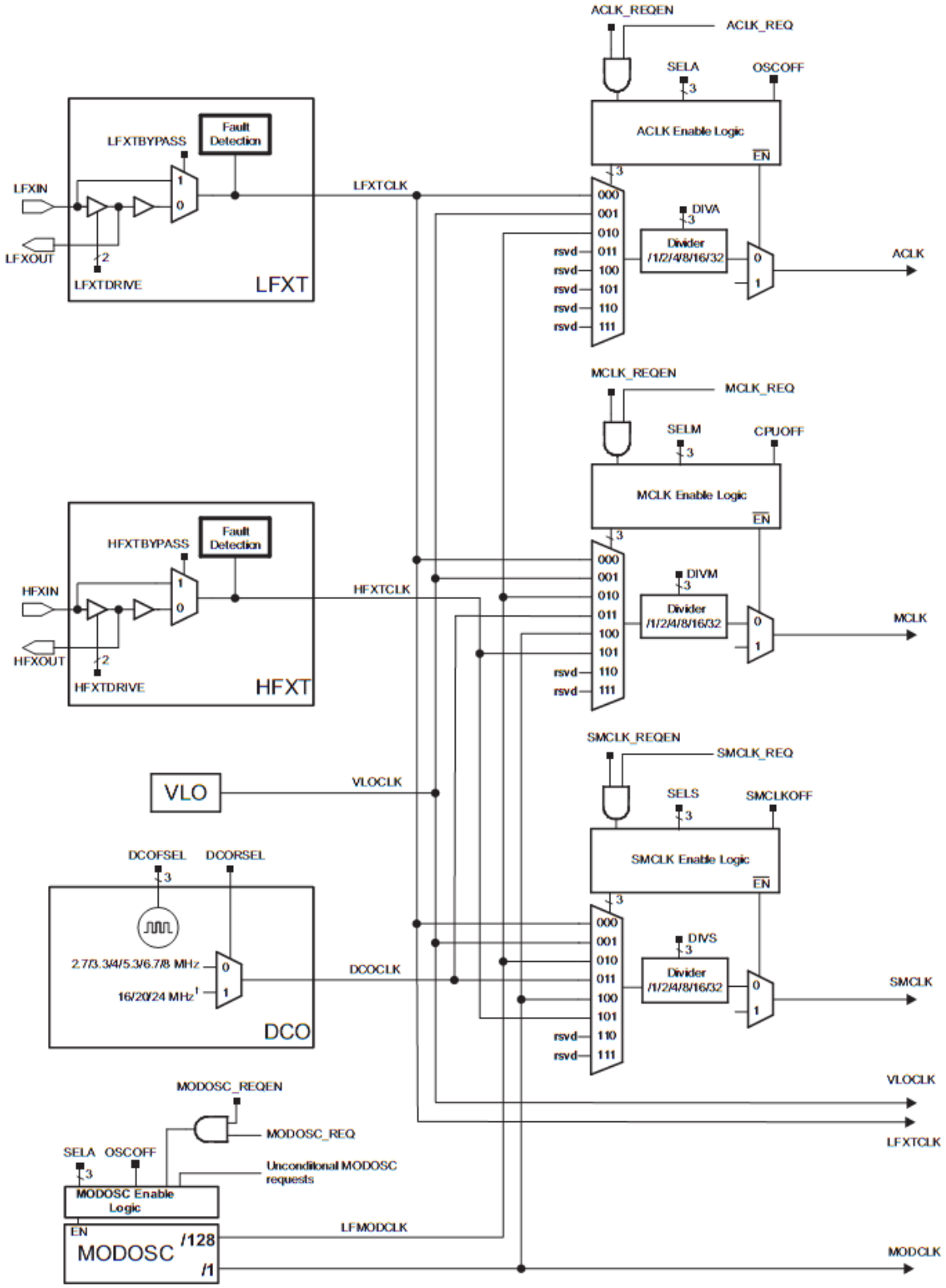

Fig. 5 Diagrama de bloques detallado del sistema de reloj (Texas Instruments, 2015c).

Al inicializarse el microcontrolador, el sistema de reloj toma la siguiente configuración por defecto:

- Se selecciona LFXT como fuente de reloj de la señal LFXTCLK, y esta última se selecciona como señal de reloj auxiliar (ACLK), sin subdividir.
- DCOCLK se selecciona para generar las señales de reloj principal y secundaria (MCLK y SMCLK) subdividido por 8.
- Los pines LFXIN y LFXOUT se configuran como entradas-salidas digitales de propósito general. Esto deshabilita el uso de la fuente de reloj de baja potencia LFXT hasta que dichos pines sean reconfigurados para su uso como reloj.
- Los pines HFXIN y HFXOUT se configuran como entradas-salidas digitales de propósito general, lo cual deshabilita la señal de reloj HFXT.

Es decir, al iniciar el microcontrolador su operación, sólo se generarán las señales de reloj MCLK y SMCLK a partir del oscilador interno DCO.

Para la configuración del reloj, y en general para la utilización de los recursos del microcontrolador, existe una biblioteca de funciones llamada MSP430Ware, misma que se puede descargar desde la página web de Texas Instruments. El procedimiento sugerido de instalación consiste en abrir el ambiente de desarrollo Code Composer Studio (CCS): Dar clic en en CCS App Center, seleccionar MSP430Ware y dar clic en instalar software. Sin embargo, utilizando el CCS 6.2.0 este procedimiento no instaló la versión más nueva de la biblioteca y se optó por una instalación manual. Para ello, se descargó y ejecutó el archivo MSP430Ware_3_60_00_10_setup.exe y se seleccionó como directorio de instalación la ruta por defecto (c:\ti\msp). Entonces, al abrir el programa CCS, éste detecta las bibliotecas y las carga. Para utilizar la biblioteca se requiere añadir la siguiente línea al código al programa principal:

```
#include <driverlib.h>
```

2.1 Configuración del reloj en modo de bajo consumo

Como se mencionó anteriormente, la operación de un microcontrolador en modo de bajo consumo requiere la utilización de un reloj de baja frecuencia. En este caso, se utiliza el reloj de 32.768 kHz que está incluido en la tarjeta LaunchPad. El procedimiento de configuración consiste en tres etapas: Selección del reloj, arranque del reloj y configuración de las señales de reloj internas (ver Fig. 6). Para ello, se utilizan tres funciones de la biblioteca MSP430Ware, que se describen a continuación.
En primer lugar, la selección del reloj externo se hace mediante la siguiente función:

```
CS_setExternalClockSource(
LF_CRYSTAL_FREQUENCY_IN_HZ,
HF_CRYSTAL_FREQUENCY_IN_HZ
 );
```

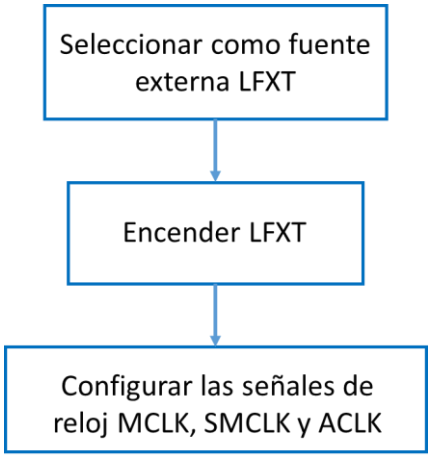

Fig. 6 Etapas de configuración del reloj del sistema.

donde LF_CRYSTAL_FREQUENCY_IN_HZ, y HF_CRYSTAL_FREQUENCY_IN_HZ son etiquetas que deben definirse en el encabezado del archivo de programa, y deben indicar la frecuencia del cristal a utilizar. Para el caso que nos atañe, esto se realiza con las siguientes líneas de código:

```
#define LF_CRYSTAL_FREQUENCY_IN_HZ     32786
#define HF_CRYSTAL_FREQUENCY_IN_HZ     0
```

Así se define la frecuencia del oscilador de baja frecuencia como igual a 32 768 Hz, y dado que la tarjeta LaunchPad no cuenta con cristal de alta frecuencia, se define la frecuencia del oscilador de alta frecuencia como igual a cero.

Posteriormente, el oscilador de baja frecuencia se activa utilizando la siguiente función:

```
CS_turnOnLFXT(CS_LFXT_DRIVE_3);
```

La corriente requerida por el oscilador puede ajustarse en uno de 4 niveles ascendentes: DRIVE_0, DRIVE_1, DRIVE_2, y DRIVE_3 (Texas Instruments, 2015c). Es decir, la función acepta cuatro valores distintos, según se lista en la Tabla **4**.

CS_turnOnLFXT (lfxtdrive)

DRIVER CURRENT	
CS_LFXT_DRIVE_0	Mínima fuerza y consumo
CS_LFXT_DRIVE_1	
CS_LFXT_DRIVE_2	
CS_LFXT_DRIVE_3*	Máxima fuerza y consumo

* valor por omisión

Tabla 4 Parámetros de configuración del reloj LFXT.

Para configurar las señales de reloj MCLK, SMCLK y ACLK se utiliza la función:

```
CS_initClockSignal ( selectedClockSignal,  clockSource,  clockSourceDivider)
```

Esta función recibe tres parámetros: selectedClockSignal, que define cuál señal de reloj se va a configurar; clockSource, que indica cuál es la fuente del reloj; y clockSourceDivider, que selecciona un factor de división de la frecuencia de la fuente. Los valores que pueden tomar estos parámetros se enlistan en la Tabla **5**.

CS_initClockSignal (selectedClockSignal, clockSource, clockSourceDivider)

selectedClockSignal	clockSource	clockSourceDivider
CS_ACLK	CS_VLOCLK_SELECT	CS_CLOCK_DIVIDER_1[+]
CS_MCLK	CS_DCOCLK_SELECT*	CS_CLOCK_DIVIDER_2
CS_SMCLK	CS_LFXTCLK_SELECT	CS_CLOCK_DIVIDER_4
CS_MODOSC	CS_HFXTCLK_SELECT*	CS_CLOCK_DIVIDER_8[x]
	CS_LFMODOSC_SELECT	CS_CLOCK_DIVIDER_16
	CS_MODOSC_SELECT*	CS_CLOCK_DIVIDER_32

* No disponible para ACLK [+] Por omisión para ACLK
[x] Por omisión para SMCLK y MCLK

Tabla 5 Parámetros de configuración de la señal de reloj.

Por lo tanto, la señal de reloj principal se puede configurar con la siguiente instrucción:

```
CS_initClockSignal(
        CS_MCLK,
        CS_LFXTCLK_SELECT,
        CS_CLOCK_DIVIDER_1   );
```

De esta manera, se selecciona como fuente al oscilador de baja frecuencia LFXT, sin subdividir, es decir, funcionará a 32.768 kHz. Instrucciones similares se pueden emplear para configurar los relojes secundario y auxiliar quedando:

```
CS_initClockSignal(
        CS_SMCLK,
        CS_LFXTCLK_SELECT,
        CS_CLOCK_DIVIDER_1   );

CS_initClockSignal(
        CS_ACLK,
        CS_LFXTCLK_SELECT,
        CS_CLOCK_DIVIDER_1   );
```

Los pines asociados al uso del oscilador externo LFXT son el PJ.4 y el PJ.5. Al arrancar su operación el microcontrolador, estos pines están configurados como líneas de entrada/salida digital, por lo que se deben reconfigurar utilizando las siguientes funciones:

```
GPIO_setAsPeripheralModuleFunctionOutputPin(
        GPIO_PORT_PJ,
        GPIO_PIN5,
        GPIO_PRIMARY_MODULE_FUNCTION );

GPIO_setAsPeripheralModuleFunctionInputPin(
        GPIO_PORT_PJ,
        GPIO_PIN4,
        GPIO_PRIMARY_MODULE_FUNCTION );
```

Esto define al pin PJ.4 como entrada del oscilador LFXT, y al pin PJ.5 como salida del oscilador LFXT. La estructura de la función puede habilitar la función primaria, secundaria o terciaria del pin simplemente indicando cualquiera de los modos

```
GPIO_PRIMARY_MODULE_FUNCTION
GPIO_SECONDARY_MODULE_FUNCTION
GPIO_TERNARY_MODULE_FUNCTION
```

Para determinar cuál es la función primaria, secundaria o terciaria, nos basamos en la secuencia que aparece en el diagrama de pines del microcontrolador. Por ejemplo, si aparece indicado

```
P1.3/TA1.2/UCB0STE/A3/C3
```

entonces la función primaria es el temporizador TA1, pin 2; la función secundaria es la comunicación serial SPI STE(Serial Transmit Enable), etc.

Finalmente, la escritura en los puertos de entrada/salida está deshabilitada al arranque del sistema, esto evita problemas por posibles fallas en la operación del microcontrolador. Para habilitar la escritura en los puertos se debe ejecutar la instrucción

```
PMM_unlockLPM5();
```

La instrucción hace referencia al módulo de administración de potencia (power managment module), liberando (un-locking) el bit LPM5 y permitiendo que los puertos salgan de su estado por defecto de alta impedancia.

3 Ejemplo de aplicación: Generación de un tren de pulsos

Para probar la configuración en modo de bajo consumo de energía, a manera de ejemplo, se generará una señal de tren de pulsos formada por un pulso de 3 ms y una frecuencia de 10 Hz, tal como se muestra en la Fig. **7**.

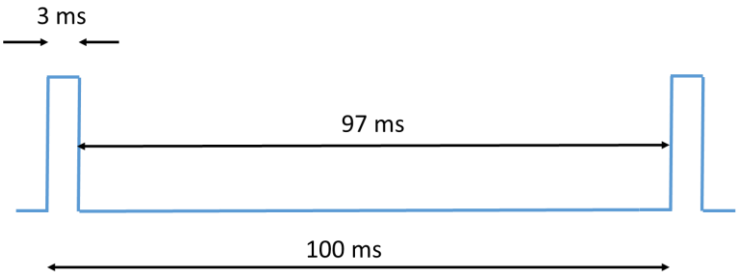

Fig. 7 Señal de tren de pulsos propuesta.

Para la generación de esta señal se puede utilizar cualquiera de los temporizadores con que cuenta el microcontrolador. Se opta por utilizar el temporizador de 16 bits TA1, por tener conexión de salida en la tarjeta LaunchPad. Dado que la frecuencia de reloj es de 32768 Hz, el período debe entonces durar 3277 pulsos de reloj para producir una frecuencia de 10 Hz, y el ancho de pulso debe ser de 98 pulsos para durar aproximadamente 2.991 ms.

La señal deseada se puede obtener configurando el temporizador para operar en modo de modulación por ancho de pulso (PWM, por sus siglas en inglés). Para ello se utiliza la estructura

```
Timer_A_outputPWMParam param
```

la cual recibe, además del período y el ciclo de trabajo, los parámetros listados en la Tabla **6**. Es decir, se debe indicar cuál es la fuente de reloj, el factor de división del reloj, el número de registro de captura-comparación que se utilizará, y el modo de salida. En este caso, se selecciona como fuente de reloj al reloj auxiliar (ACLK), debido a que este puede seguir operando al entrar en modo de bajo consumo de energía; es decir, puede seguir funcionando después de desconectar la fuente de reloj del CPU (MCLK), y así se reduce el consumo. Para este ejemplo se utilizó el registro de captura-compara número dos. Finalmente, como modo de salida se utiliza el modo Reset-Set. En esta modalidad, el temporizador cuenta los pulsos de reloj hasta un valor definido por el registro TAxCCR0, y luego de manera descendente hasta cero. Al mismo tiempo, se utiliza el valor definido por el registro TAxCCR2 como umbral de comparación para generar una salida alta, sólo en la sección descendente de la cuenta, según se muestra en la Fig. **8**.

param.clockSource=
TIMER_A_CLOCKSOURCE_EXTERNAL_TXCLK (por defecto)
TIMER_A_CLOCKSOURCE_ACLK
TIMER_A_CLOCKSOURCE_SMCLK
TIMER_A_CLOCKSOURCE_INVERTED_EXTERNAL_TXCLK
param.clockSourceDivider =
TIMER_A_CLOCKSOURCE_DIVIDER_1(por defecto)
Valores aceptados: 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56, 64
param.compareRegister =
TIMER_A_CAPTURECOMPARE_REGISTER_0
Valores posibles: 0, 1, 2, 3, 4, 5, 6
param.compareOutputMode =
TIMER_A_OUTPUTMODE_OUTBITVALUE (por defecto)
TIMER_A_OUTPUTMODE_SET
TIMER_A_OUTPUTMODE_TOGGLE_RESET
TIMER_A_OUTPUTMODE_SET_RESET
TIMER_A_OUTPUTMODE_TOGGLE
TIMER_A_OUTPUTMODE_RESET
TIMER_A_OUTPUTMODE_TOGGLE_SET
TIMER_A_OUTPUTMODE_RESET_SET

Tabla 6 Parámetros de configuración del temporizador TA.

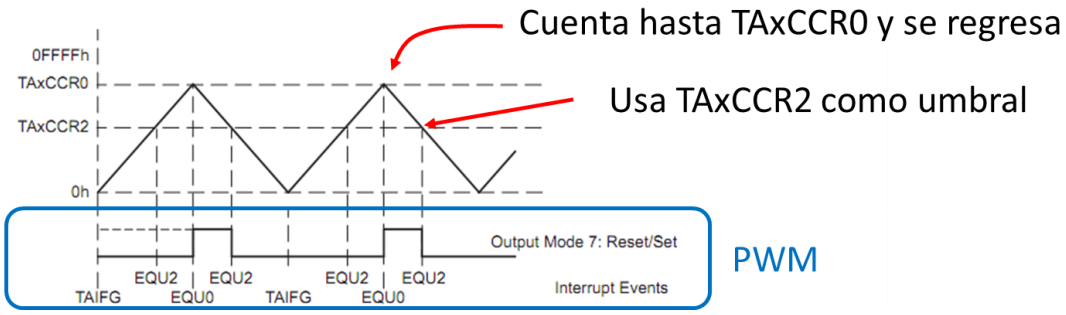

Fig. 8 Operación del modo Reset_Set (Texas Instruments, 2015c).

La configuración del temporizador queda entonces definida mediante el siguiente código:

```
Timer_A_outputPWMParam param = (Instruments);
        param.clockSource = TIMER_A_CLOCKSOURCE_ACLK;
        param.timerPeriod = myPeriod;
        param.compareRegister = TIMER_A_CAPTURECOMPARE_REGISTER_2;
        param.compareOutputMode = TIMER_A_OUTPUTMODE_RESET_SET;
        param.dutyCycle = myDutyCycle;
Timer_A_outputPWM( TIMER_A1_BASE, &param );
```

Para habilitar el pin P1.3 como la salida del temporizador, se requiere de la siguiente instrucción

```
GPIO_setAsPeripheralModuleFunctionOutputPin(
        GPIO_PORT_P1,
        GPIO_PIN3,                                  // TA1.2
        GPIO_PRIMARY_MODULE_FUNCTION
);
```

4 Pruebas

Después de configurar los relojes para operar con el cristal de 32.768 kHz y fijar los parámetros de operación del temporizador, lo que resta es activar un modo de bajo consumo de energía. En este caso utilizaremos el modo LPM3, el cual desconecta el CPU pero mantiene la señal de reloj ACLK que es utilizada por el temporizador TA1.2. Para hacer que el microcontrolador entre en modo de bajo consumo se utiliza la siguiente instrucción:

```
_bis_SR_register(LPM3_bits + GIE);
```

Con la cual se modifica directamente el contenido del registro de estado.

El diagrama de flujo del programa principal se muestra en la Fig. 9. Ahí se indica que el primer proceso a realizar es la desconexión del *Watchdog*, debido a que no se utiliza en este ejemplo. La desconexión se hace mediante la instrucción:

```
WDT_A_hold( WDT_A_BASE );
```

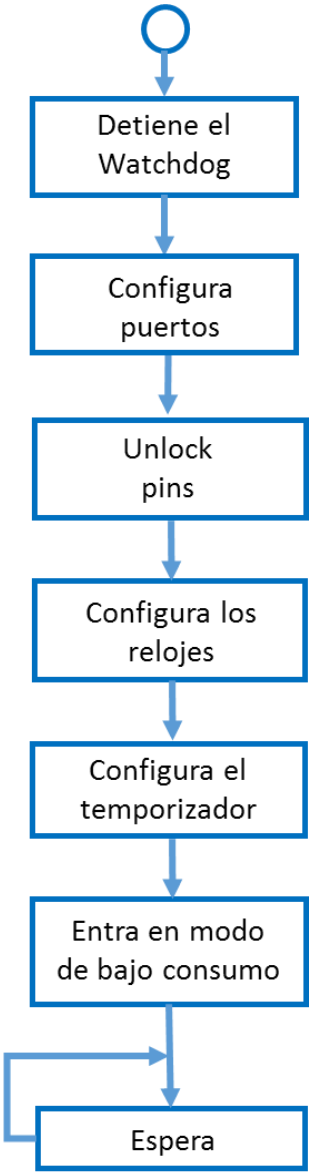

Fig. 9. Diagrama de flujo del programa principal main.c.

Como fuente de energía se utilizó el súper capacitor (10 Farads @ 5 VCD) que está instalado en la tarjeta. Para ello, primero se alimentó la tarjeta mediante su conexión mediante un cable USB a una computadora. Esto carga el capacitor mediante el puente de conexión J11 (*jumper*). Posteriormente se modifica la configuración de los puentes de conexión para utilizar como alimentación el capacitor (puentes J2 y J11). También se requiere desconectar los puentes de conexión hacia el área del programador; es decir, se desmontan todos los puentes del conector J13. Finalmente, para poder medir el consumo de corriente, la tarjeta LaunchPad cuenta con dos pines diseñados con ese fin.

Las primeras pruebas fueron desalentadoras, pues se midió un consumo de aproximadamente 250 µA. Entonces, con la finalidad de reducir el consumo de energía se procedió a configurar todos los puertos como salida, y a escribir un cero en todos ellos. Esta es una recomendación del fabricante para evitar que los puertos puedan estar cambiando de estado. Después de esta modificación, el consumo disminuyó a 40 µA. El código para realizar esta configuración es el siguiente:

```
P1DIR = 0xFF;       // PUERTO 1
P1OUT = 0x00;
P2DIR = 0xFF;      // PUERTO 2
P2OUT = 0x00;
P3DIR = 0xFF;     // PUERTO 3
P3OUT = 0x00;
P4DIR = 0xFF;      // PUERTO 4
P4OUT = 0x00;
PJDIR = 0xFF;     // PUERTO J
PJOUT = 0x00;
```

Obsérvese que la convención para definir el sentido de operación del puerto es inversa a la de otros microcontroladores. Para el MSP430FR5969, el bit de dirección igual a '1' indica salida y si es igual a '0' indica entrada.

Un consumo de 40 µA, ya se consideraba aceptable, pues una pila de tipo reloj CR2032 (3V) podría mantener operando el sistema durante 212 días. Por otro lado, después de utilizar la utilería llamada EnergyTrace para intentar rastrear posibles fuentes de consumo de energía no obtuvimos información útil.

Finalmente, observamos que el propio programa Code Composer Studio mantenía operando el depurador (debugger) para monitoreo y control de la tarjeta LaunchPad, cuando deshabilitamos el depurador, logramos reducir el consumo hasta 0.95 µA. Este valor fue determinado mediante un multímetro de 4-4/5 dígitos (50,000 cuentas) Sanwa modelo PC5000A, el cual tiene una resolución de 0.01 µA para corriente directa y un error de exactitud de 0.15% + 20 cuentas, en la escala de 500 µA (ver Fig. 10). Con este nivel de consumo, la misma pila CR2032 podría mantener operando el sistema por más de 20 años.

El listado completo del programa final se encuentra en el Anexo A.

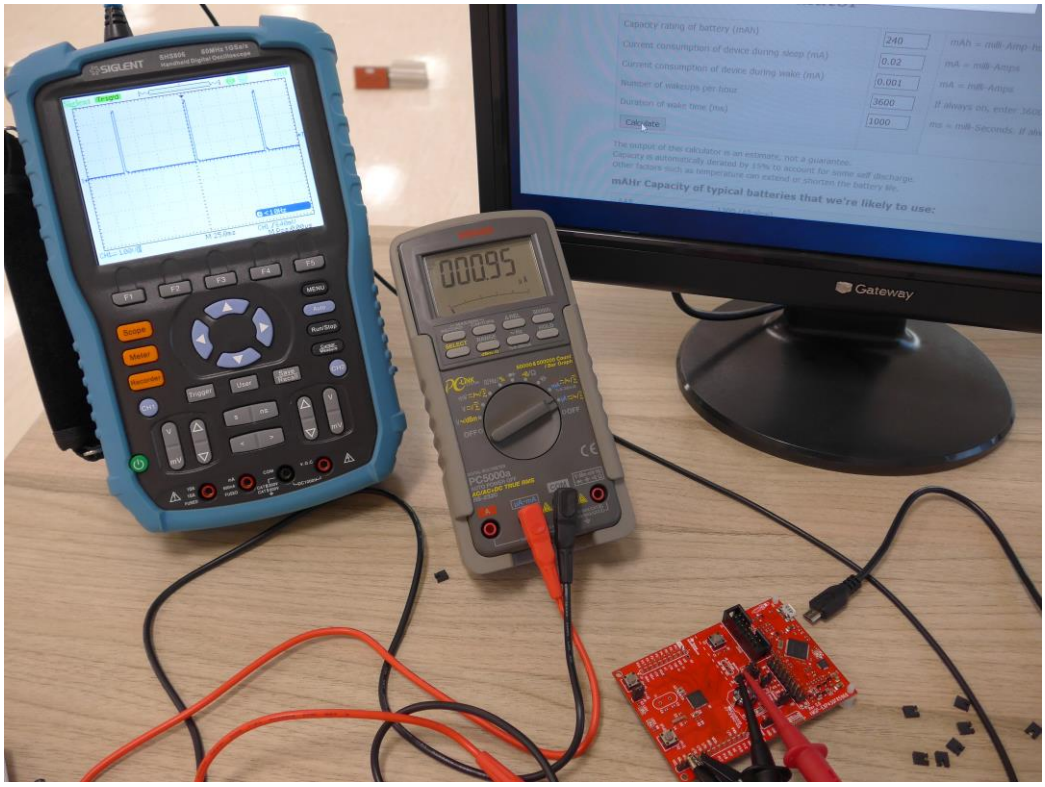

Fig. 10 Fotografía mostrando la operación de la tarjeta LaunchPad basada en el microcontrolador MSP430FR5969, generando una señal de tren de pulsos y consumiendo 0.95 µA.

5 Conclusiones

Muchos de los microcontroladores en la actualidad tienen funciones de administración de energía orientadas a aplicaciones de bajo consumo. En este manual se ha mostrado el proceso de configuración de un caso de bajo consumo de energía utilizando una tarjeta de evaluación LaunchPad con un microcontrolador MSP430FR5969. Utilizando un cristal de 32.768 kHz como fuente de reloj, se ha configurado el microcontrolador para generar una señal tren de pulsos de baja frecuencia (10 Hz) logrando un consumo de 0.95 µA. El procedimiento aquí detallado puede servir como base para desarrollar aplicaciones de bajo consumo que utilicen otros recursos del microcontrolador.

Anexo A. Listado de programas

A.1 main.c

```
// -------------------------------------------------------------------------
// main.c   ('FR5969 Launchpad)
// modificación PWM_Test1
// -------------------------------------------------------------------------

//***** Header Files ********************************************************
#include <driverlib.h>
#include "myClocks.h"
#include "myTimers.h"
#include "myPorts.h"

//***** Global Variables ****************************************************
const uint16_t period    = 3277;   // Define el período del temporizador
const uint16_t dutyCycle = 98;              // Define el ciclo de trabajo del temporizador

//*************   MAIN     ***************************************
void main (void)
{
   WDT_A_hold( WDT_A_BASE );  // Detiene el Watchdog
   initPorts();    // Inicializa los puertos
   PMM_unlockLPM5();   // Unlock pins
   initClocks();   // Configura el temporizador
   initTimers( period, dutyCycle );    // Arranca el temporizador
   _bis_SR_register(LPM3_bits + GIE);    // Entra en LPM3 y habilita interrupciones

   while(1) {
         // Ciclo infinito. Queda habilitado el temporizador y desconectado el CPU
      }
}
```

A.2 myClocks.c

```
// -------------------------------------------------------------------------
// myClocks.c    ('FR5969 Launchpad)
//
// Se configuran los relojes del sistema
//    ACLK  = 32.768 kHz
//    SMCLK = 32.768 kHz
//    MCLK  = 32.768 kHz
// -------------------------------------------------------------------------

//***** Header Files ********************************************************
//#include <stdbool.h>
#include <driverlib.h>
#include "myClocks.h"

//***** Defines *************************************************************
#define LF_CRYSTAL_FREQUENCY_IN_HZ     32786   // 32.786 kHz
#define HF_CRYSTAL_FREQUENCY_IN_HZ     0       // No está instalado

//***** Global Variables ****************************************************

//***** initClocks **********************************************************
void initClocks(void) {
        CS_setExternalClockSource(                      // Establece la frecuencia del reloj externo
        LF_CRYSTAL_FREQUENCY_IN_HZ,
        HF_CRYSTAL_FREQUENCY_IN_HZ
   );
         CS_turnOnLFXT(CS_LFXT_DRIVE_3);                // Activa el oscilador con corriente DRIVE=3

         //  ****** Configuración de los relojes MCLK, SMCLK, y ACLK
   CS_initClockSignal(
        CS_ACLK,                        // Configura ACLK
        CS_LFXTCLK_SELECT,              // Fija LFXT como fuente de reloj
        CS_CLOCK_DIVIDER_1              // Divide la fuente por 1
   );

   CS_initClockSignal(
        CS_SMCLK,                       // Configura SMCLK
        CS_LFXTCLK_SELECT,              // Fija LFXT como fuente de reloj
        CS_CLOCK_DIVIDER_1              // Divide por 1
   );

   CS_initClockSignal(
        CS_MCLK,                        // Configura MCLK
        CS_LFXTCLK_SELECT,              // Fija LFXT como fuente de reloj
        CS_CLOCK_DIVIDER_1              // Divide por 1
   );
}
```

A.3. myClocks.h

```
/*
 * myClocks.h
 *
 *  Created on: 24/11/2016
 *      Author: MIKE
 */

#ifndef MYCLOCKS_H_
#define MYCLOCKS_H_

//***************  Prototipo ******************

void initClocks(void);

#endif /* MYCLOCKS_H_ */
```

A.4 myPorts.c

```
/*
 * myPorts.c
 *
 *  Created on: 06/12/2016
 *      Author: MIKE
 */
#include <driverlib.h>
#include "myPorts.h"

void initPorts( void ){   // Todos los puertos como salida = 0
            P1DIR = 0xFF;       // PUERTO 1
            P1OUT = 0x00;
            P2DIR = 0xFF;      // PUERTO 2
            P2OUT = 0x00;
            P3DIR = 0xFF;     // PUERTO 3
            P3OUT = 0x00;
            P4DIR = 0xFF;      // PUERTO 4
            P4OUT = 0x00;
            PJDIR = 0xFF;     // PUERTO J
            PJOUT = 0x00;
}

  // CONFIGURACIÓN ESPECÍFICA
                  // Habilita el uso de los pines del cristal LFXT
            GPIO_setAsPeripheralModuleFunctionOutputPin(          GPIO_PORT_PJ,          GPIO_PIN5,
GPIO_PRIMARY_MODULE_FUNCTION );                 // LFXTCLK_OUT
            GPIO_setAsPeripheralModuleFunctionInputPin(          GPIO_PORT_PJ,          GPIO_PIN4,
GPIO_PRIMARY_MODULE_FUNCTION );                  // LFXTCLK_IN

                  // Habilita el pin de salida del temporizador
            GPIO_setAsPeripheralModuleFunctionOutputPin(
                  GPIO_PORT_P1,
                  GPIO_PIN3,                                          // TA1.2
                  GPIO_PRIMARY_MODULE_FUNCTION
            );
}
```

A.5 myPorts.h

```
/*
 * myPorts.h
 *
 *  Created on: 06/12/2016
 *      Author: MIKE
 */

#ifndef MYPORTS_H_
#define MYPORTS_H_

// Prototypes
void initPorts( void );

#endif /* MYPORTS_H_ */
```

A.6 myTimers.c

```
// -------------------------------------------------------------------------
// myTimers.c
// Autor MIKE
// -------------------------------------------------------------------------

//***** Header Files ********************************************************
#include <driverlib.h>
#include "myTimers.h"

//***** Defines *************************************************************

//***** Global Variables ****************************************************

//*****************************************************************************
// Configura el temporizador
//*****************************************************************************
void initTimers( uint16_t myPeriod, uint16_t myDutyCycle )
{
    Timer_A_outputPWMParam param = { 0 };
      param.clockSource = TIMER_A_CLOCKSOURCE_ACLK;                 // Usa ACLK (activo en modo
LPM3)

      param.timerPeriod = myPeriod;                          // Periodo
      param.compareRegister  =  TIMER_A_CAPTURECOMPARE_REGISTER_2;    // Usa CCR2 como
registro captura/compara
      param.compareOutputMode = TIMER_A_OUTPUTMODE_RESET_SET;     // Modo Reset_Set
      param.dutyCycle = myDutyCycle;                          // Fija el ciclo de trabajo
   Timer_A_outputPWM( TIMER_A1_BASE, &param );
}
```

A.7 myTimers.h

```
/*
 * myTimers.h
 *
 *  Created on: 24/11/2016
 *      Author: MIKE
 */

#ifndef MYTIMERS_H_
#define MYTIMERS_H_

// Prototypes
void initTimers( uint16_t, uint16_t );

#endif /* MYTIMERS_H_ */
```

Referencias bibliográficas

Texas Instruments (2014). **FRAM FAQs**. Texas Instruments, Inc.

Texas Instruments (2015a). **MSP430 Design Workshop Revision 4.01**. Texas Instruments Inc.

Texas Instruments (2015b). **MSP430FR5969 Launch Pad Development Kit (MSP-EXP430FR5969) User's Guide (Rev. B)**. Texas Instruments Inc.

Texas Instruments (2015c). **MSP430FRXX Mixed-Signal Microcontrollers (Rev. E)**. Texas Instruments Inc.

Texas Instruments (2016). **MSP Low-Power Microcontrollers**. Texas Instruments, Inc.

Texas Instruments (2017). **MSP430 ultra-low-power Microcontrollers** [Internet]. Disponible desde: <http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/overview.page> [Acceso el 13 de enero de 2017].