ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE CONTROL DE PROPÓSITO GENERAL USANDO EL MICROCONTROLADOR F2013 Y LA HERRAMIENTA DE DESARROLLO "EZ430-F2013" DE TEXAS INSTRUMENTS

PROYECTO PREVIO A LA OBTENCIÓN DEL TITULO DE INGENIERO EN ELECTRÓNICA Y REDES DE LA INFORMACIÓN

RANGLES CÓRDOVA HERNÁN PATRICIO
rangles_asoc@yahoo.com

DIRECTOR: ING. OSWALDO BUITRÓN
oswaldo.buitron@epn.edu.ec

Quito, Agosto 2009

DECLARACIÓN

Yo, Hernán Patricio Rangles Córdova, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Hernán Patricio Rangles Córdova

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Hernán Patricio Rangles Córdova, bajo mi supervisión.

Ing. Oswaldo Buitrón
DIRECTOR DE PROYECTO

DEDICATORIA Y AGRADECIMIENTO

Este proyecto está dedicado a Dios, ser supremo que me dio el regalo más bello, preciado y amado en esta vida, mi familia.

A mis padres Pato y Jenny por su apoyo incondicional y por ser el ejemplo de seres humanos que quiero siempre imitar, además de ser directamente responsables de la finalización de este proyecto.

A mis hermanos San y Riki por siempre estar pendientes de mi.

A mis sobrinos Nicole y Santi y a mi comadre Dany, por darle más vida al hogar que compartimos.

A mi Director de Tesis, por apoyarme en la idea de realizar este tema y guiarme en su realización.

A mi primo Wladimir por toda la ayuda prestada desinteresadamente.

A mis abuelos papá Carlitos y mamá Rosita, a mis tíos Hernán y Cesi, Héctor y Mariani, Henry y Geovy, Rami y Xime por ser participes del camino recorrido para conseguir mi objetivo.

A todos mis familiares, amigos y personas que de una u otra manera brindaron su granito de arena muy necesario para culminar con éxito esta etapa de mi vida.

Y a todas aquellas personas que hicieron el camino más difícil y con tropiezos, lo que hizo que llegar al final fuera mucho más satisfactorio.

A todos ellos dedico este proyecto y sobre todo les estaré eternamente agradecido.

Pato's
2009-08-26

RESUMEN

Los microcontroladores MSP430 contienen una CPU de 16-bits de arquitectura RISC de señal combinada de menor consumo energético y proporciona la mejor solución para una amplia gama de dispositivos portátiles que requieren un bajo consumo de energía. Texas Instruments proporciona toda la información requerida sobre las caracteristicas del Microcontrolador MSP430 incluyendo documentos técnicos, instrumentos, y el software de desarrollo correspondiente.

La herramienta de emulación Flash (FET) no solo permite programar al microcontrolador si no que también permite verificar el código de usuario en tiempo de ejecución, de esta manera se puede comprobar el funcionamiento del diseño implementado con el uso del microcontrolador MSP430F2013, además se puede efectuar la depuración del código de programa utilizando la función de “debug” que esta herramienta incluye.

El presente documento está conformado por cuatro capítulos e incluye la sección de anexos. En el primer capítulo se analiza y detalla las características más importantes del microcontrolador MSP430F2013 que será utilizado como dispositivo central del prototipo de propósito general, además de una breve explicación de la herramienta de desarrollo a utilizarse, en este caso el eZ430-F2013, finalmente se presenta y analiza el diseño de cada uno de los módulos que serán implementados como parte del prototipo final.

En el segundo capítulo se describe el software requerido para el funcionamiento del prototipo, es decir, se detalla el software creado para la interacción entre el PC y el Prototipo y también se detalla el software requerido por cada uno de los módulos de hardware diseñados anteriormente. Además

se describe el ejemplo aplicativo utilizado para ilustrar el desarrollo del prototipo.

En el tercer capítulo se detalla la construcción del prototipo de acuerdo a los diseños desarrollados en el capitulo uno así como todas las pruebas ejecutadas de acuerdo con el protocolo de pruebas propuesto.

En el cuarto capítulo se presentan las conclusiones y recomendaciones que se obtienen de este proyecto.

Finalmente en los anexos se incluye la información que describe en mayor detalle algunas características del microcontrolador o del software relacionado con el mismo, así como de información relacionada con este proyecto.

PRESENTACIÓN

La razón por la cual se ha desarrollado este proyecto es la de mostrar las nuevas tecnologías que están disponibles en el mercado y de las cuales se pueden echar mano para implementar diseños que se requieren tanto en la industria como en el hogar.

Otra razón del proyecto es la de crear un sistema que permita el control de un proceso sin llegar a un costo muy elevado, utilizando herramientas que ofrecen mayores facilidades para el desarrollo de cualquier proyecto.

El objetivo del proyecto es construir un prototipo de propósito general utilizando el microcontrolador MSP430F2013 de la empresa Texas Instruments, el cual permita manipular un proceso que el usuario determine de acuerdo a la programación que se establezca sobre el mismo.

Para facilitar esta tarea se ha integrado en el prototipo un Display que indica el estado del prototipo y también un teclado con el cual se puede acceder a los menús integrados en el.

Para todo ello, se ha desarrollado el software, que permite el manejo de cada uno de los módulos del prototipo, y otra parte la compone un programa que sirve de interfaz entre el usuario y el prototipo desde un computador permitiéndole acceder a muchas más opciones de interacción que simplemente utilizar el prototipo como un dispositivo “solitario”

El prototipo tiene 4 entradas analógicas y 4 salidas digitales acopladas a relés. El programa en PC puede acceder a la información de cada entrada y también puede alterar el estado de las salidas, así como, guardar registros de toda

actividad del prototipo. El prototipo permite el almacenamiento de datos de 16 bits cada 10 minutos, lo que puede resultar muy útil en procesos que requieran vigilancia de estado de un sensor.

El programa en el PC permite no solo acceder al prototipo, también permite que el prototipo controle el estado del PC, gracias a comandos que el programa puede ejecutar de acuerdo al valor que tenga cada entrada y a la configuración que el usuario haya realizado sobre cada una.

El prototipo implementado demuestra las amplias posibilidades de aplicación de los microcontroladores MSP430F2012 / MSP430F2013 y se espera que constituya una guía para aquellos interesados en aprender y desarrollar nuevas aplicaciones basados en esta tecnología.

CONTENIDO

CAPITULO 2. DESARROLLO DE LOS PROGRAMAS

ANEXOS

CAPITULO 1

CAPITULO 1. DISEÑO CIRCUITAL Y HERRAMIENTAS DE HARDWARE

Para el desarrollo del presente trabajo se ha tomado como base la documentación técnica del fabricante, en especial los siguientes archivos provistos por la empresa Texas Instruments; razón por la que en el desarrollo del trabajo no se citan estas referencias:

- slau176.pdf - eZ430-F2013 Development Tool - User's Guide
- slau012a.pdf – MSP430x3xx User’s Guide
- slau144e - MSP430x2xx User’s Guide
- slas491a.pdf

1.1 CARACTERÍSTICAS DE LOS MICROCONTROLADORES DE TEXAS INSTRUMENTS

1.1.1 DESCRIPCIÓN GENERAL

Los MCUs (MicroController Unit – Unidad Microcontroladora) que están basados en procesadores tipo RISC (Reduced Instruction Set Computer – Set Reducido de Instrucciones para Computadoras) de 16-bits, son la solución industrial de menor consumo entre microcontroladores de 8 o 16 bits alimentados por baterías, su principal aplicación incluyen la medición, la instrumentación portátil y los sensores inteligentes.

La familia de microcontroladores MSP430F2xx (MSP430F2013) permiten procesar hasta 16 Millones de Instrucciones Por Segundo (16 MIPS) usando voltajes que van entre 1.8 a 3.6 V. Dentro de sus opciones incluyen un oscilador de bajo poder integrado en el chip con una variación de ±1 %, resistencias internas pull-up/pull-down y opción de chips con un número reducido de terminales.

La familia del MSP430 de Texas Instruments son microcontroladores de muy bajo consumo de energía que constan de varios modelos presentando sets diferentes de dispositivos periféricos dirigidos a sectores específicos para aplicaciones diversas. La arquitectura esta combinada con cinco modos de bajo consumo de energía que es optimizada para lograr una vida extendida de la batería en aplicaciones portátiles de medición. El oscilador controlado digitalmente (DCO – Digitally Controller Oscillator) permite “despertar” al microcontrolador desde los modos de bajo consumo de energía al modo de ejecución en menos de 1 us. El dispositivo contiene una CPU (Central Processing Unit – Unidad de Procesamiento Central) de 16 bits de arquitectura RISC, registros de 16 bits, y generadores de constantes lo que permite mejor eficiencia de código.

1.1.2 CARACTERÍSTICAS TÉCNICAS GENERALES

La familia de microcontroladores MSP430x20xx y en especial el microcontrolador F2013 poseen las siguientes características técnicas, que se indican a continuación y que se presentan en mayor detalle en los literales siguientes:

- Rango de consumo de alimentación entre 1.87 a 3.6 Voltios.
- Tiene un consumo muy bajo de corriente y por lo tanto de potencia:
    - De 220uA en Modo Activo con un voltaje de 2.2 Voltios a una frecuencia de trabajo de 1Mhz.
    - De 0.5uA en Modo de Espera (Standby).
    - De 0.1uA en Modo Apagado o de Retención de RAM (OFF).
- En total posee 5 modos de ahorro de energía.
- Paso al Modo Activo del Modo de Espera en menos de 1us.
- Arquitectura del CPU tipo RISC de 16 bits con un Tiempo de Ciclo de Instrucción de 62.5ns.
- Modulo de configuraciones básica de reloj:
    - Reloj interno de hasta 16MHz con 4 frecuencias calibradas en +/-1%.

  - Oscilador LF (Low Frecuency – Baja Frecuencia) interno de bajo consumo.
  - Cristal Oscilador externo de 32KHz.
  - Fuente externa de reloj digital.
- Contador de 16 bits con 2 registros de captura o comparación.
- Detector de sobre-voltaje (Brownout Detector)
- Permite ser programado simultáneamente con otros microcontroladores gracias al uso de “tableros” especiales para producción en serie.
- No se requiere voltaje externo para la programación del microcontrolador
- Protección del código programado mediante un fusible de seguridad
- Lógica de emulación integrada en el chip con la interfaz Spy-Bi-Wire
- Memoria Flash incluida: 2KB + 256B, 128B de Memoria RAM
- Disponible en un empaquetado de 14 terminales plástico Small-Outline de empaquetado fino (TSSOP)
- Para el microcontrolador MSP430F2013 (F2013) se tiene:
  - Un conversor Análogo / Digital tipo Sigma – Delta con entradas PGA diferenciales y referencias internas.
  - Interface Serial Universal (USI – Universal Serial Interface) que soporta SPI (Serial Peripherical Interface) e I2C (Inter-Integrated Circuit)

1.1.3 DESCRIPCION DE LA CPU.

Todas las operaciones, así como las instrucciones, son realizadas por el CPU del microcontrolador MSP430 como operaciones de registro en conjunción con siete modos de direccionamiento para el operando de origen y cuatro modos de direccionamiento para el operando de destino. La CPU está integrada con 16 registros que proveen reducción del tiempo de ejecución de cada instrucción. El tiempo que lleva la operación registro-a-registro es de un ciclo del reloj de la CPU. Los dispositivos periféricos están conectados a la CPU mediante los buses de datos, de dirección y de control, y pueden ser accedidos como si se tratara de una dirección de memoria cualquiera, es decir, se puede usar cualesquiera de las instrucciones disponibles.

1.1.4 REGISTROS DEL MICROCONTROLADOR

El microcontrolador incorpora dieciséis registros de 16 bits. R0 a R3, son dedicados como contador de programa (PC - Program Counter), Puntero del Stack (SP - Stack Pointer), Registro de estado (SR - Status Register) y un Generador de Constantes (CG - Constant Generator) respectivamente. Los registros restantes R4 a R15 funcionan como registros de uso general. En la figura 1.1 se muestra un diagrama de los registros de la CPU y su interconexión.

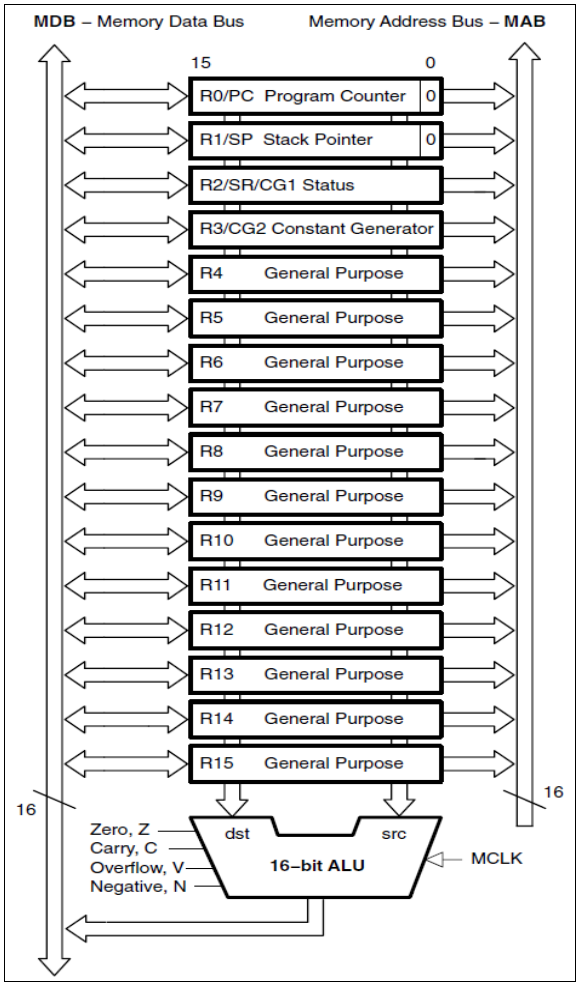


Figura 1.1: Registros del CPU. (Fuente: Texas Instruments)

1.1.4.1 El Contador de Programa (PC)

El Contador de Programa de 16 bits (PC / R0) contiene la dirección de memoria donde se encuentra la siguiente instrucción a ser ejecutada. Cada instrucción

requiere un número par de bytes (dos, cuatro, o seis), y el PC es incrementado en ese número de bytes.

1.1.4.2 Puntero de Pila (SP)

El puntero de la pila (SP/R1) es usado por el microcontrolador para almacenar las direcciones de retorno de llamado de subrutinas e interrupciones. Usa un esquema de pre-decremento, post-incremento, en el cual, su mayor ventaja es que el elemento que se encuentra en la parte más alta de la pila está disponible. Además, el SP puede ser usado por software. El SP es inicializado en la RAM por el usuario, y está alineado en direcciones pares.

1.1.4.3 El Registro de Estado (SR)

El registro de estado (SR) contiene los siguientes bits de estado del CPU, los cuales son detallados en la tabla 1.1. La figura 1.2 muestra la distribución de los bits dentro del registro R2.

- V bit de Desbordamiento
- SCG1 bit 1 de control del generador de reloj del sistema
- SCG0 bit 0 de control del generador de reloj del sistema
- OscOff bit de “Apagado” del Cristal Oscilador
- CPUOff bit de “Apagado” de la CPU
- GIE bit de “Habilitación” de la Interrupción General (GIE)
- N bit “Negativo”
- Z bit “Cero”
- C bit “Carry”

Bit	***Descripción***
V	Bit de Desbordamiento: Activado si el resultado de una operación excede el rango de la variable (incluido el signo), válido para operaciones “Byte” y “Word”.
SCG1, SCG0	Estos bits controlan 4 estados de actividad del reloj de sistema y por lo tanto influye en el funcionamiento del procesador del sistema.
OSCOFF	Si es puesto a 1, el oscilador externo entra en modo de “apagado”, todas las actividades cesan, aunque el contenido de la RAM, los

	registros de los periféricos y los registros en general se mantienen. Salir de este modo es posible solo si se habilita las interrupciones externas cuando se ha habilitado el bit GIE o usando interrupciones del tipo NMI (no enmascarable).
CPUOFF	Si es puesto a 1, la CPU entra en modo de “apagado”, la ejecución de programa se detiene, el contenido de la memoria RAM, los registros de los periféricos y especialmente los periféricos habilitados permanecen activos. Salir de este modo es posible usando cualquiera de las interrupciones habilitadas.
GIE	Si esta puesto a 1, todas las interrupciones enmascarables son controladas. Si esta deshabilitado, todas las interrupciones enmascarables son ignoradas. El bit GIE es borrado por la interrupción y puesto a 1 por la instrucción RETI o por cualquier otra instrucción que habilite este bit
N	Puesto a 1 si el resultado de una operación es negativo. Operación WORD: N es igual al valor del bit 15 del resultado Operación BYTE: N es igual al valor del bit 7 del resultado
Z	Puesto a 1 si el resultado de cualquier operación es “0”
C	Puesto a 1 si el resultado de una operación produce un “Carry”. Algunas instrucciones modifican este bit usando el “inverted zero bits”

Tabla 1.1: Descripción de los bits de estado (SR). (Fuente: Texas Instruments)

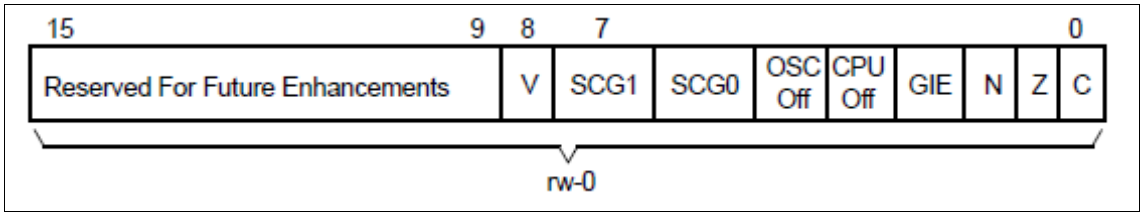


Figura 1.2: Ubicación de los bits de estado en el registro (SR/R2). (Fuente: Texas Instruments)

Nota: *los bits de estado V, N, Z y C son cambiados únicamente por ciertas instrucciones. En el anexo B.6 se encuentra información adicional de las instrucciones y los bits que estas modifican.*

1.1.4.4 Los Registros Generadores de Constantes CG1 y CG2

Las seis constantes comúnmente usadas son generadas con los Registros Generadores de Constantes R2 y R3, sin requerir una palabra de 16 bits

adicional de código de programa. Las constantes son seleccionadas con los modos de direccionamiento del registro de origen (As), como se describe en la tabla 1.2.

Register	As	Constant	Remarks
R2	00	– – – – –	Register mode
R2	01	(0)	Absolute address mode
R2	10	00004h	+4, bit processing
R2	11	00008h	+8, bit processing
R3	00	00000h	0, word processing
R3	01	00001h	+1
R3	10	00002h	+2, bit processing
R3	11	0FFFFh	–1, word processing

Tabla 1.2: Valores de los Generadores de Constantes CG1, CG2. (Fuente: Texas Instruments)

Las ventajas de los Generadores de Constantes son:

- No requiere de instrucciones especiales
- No requiere de código adicional para las seis constantes
- No requiere de código de acceso a memoria para obtener la constante

El ensamblador usa el generador de constantes automáticamente si una de las seis constantes es utilizada como un operando inmediato de Origen. Los registros R2 y R3, cuando son usados en el modo de constantes, no pueden ser direccionados explícitamente.

1.1.4.5 Registros de propósito general R4 a R15

Los doce registros, R4 a R15, son registros de propósito general. Todos ellos pueden ser utilizados como registros de datos, punteros de dirección, o pueden indexar valores y pueden ser accedidos mediante instrucciones de byte o palabra como se muestra en la figura 1.3.

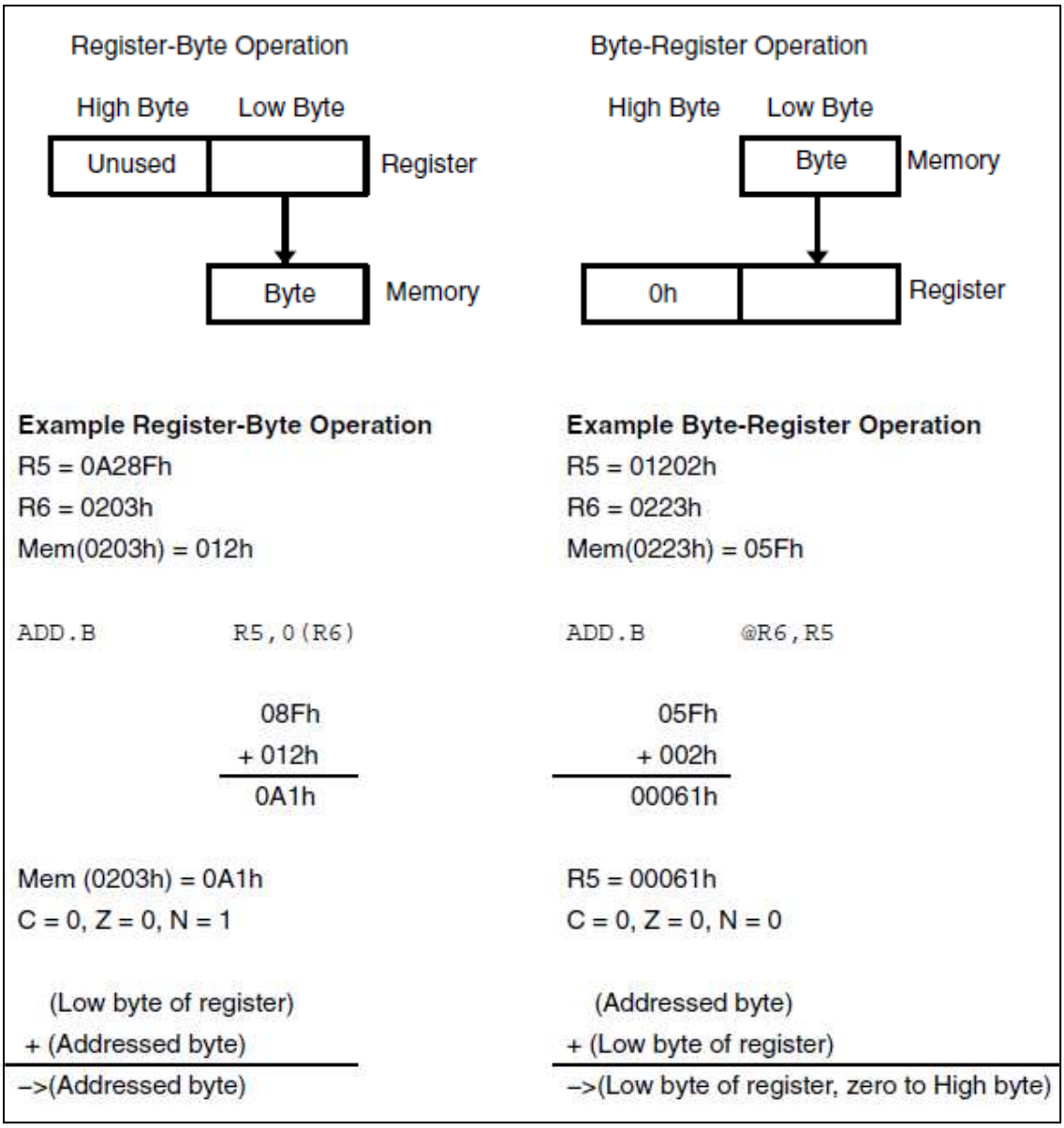


Figura 1.3: Operaciones Registro-Byte y Byte-Registro. (Fuente: Texas Instruments)

1.1.5 MODOS DE OPERACIÓN

El MSP430 tiene un modo activo y cinco modalidades de operación de bajo consumo seleccionables mediante software. Una petición de interrupción puede activar el dispositivo desde cualquiera de los cinco modos de bajo consumo, puede dar servicio a la petición y puede regresar al modo de bajo consumo en el que se encontraba el microcontrolador antes de realizarse la interrupción.

Los siguientes seis modos de operación pueden ser configurados por software:

- Modo Activo (AM - Active Mode);
    - Todos los relojes están activos

- Modo de bajo consumo 0 (LPM0 – Low Power Mode 0);
    - El CPU esta deshabilitado
    - ACLK (Auxiliary Clock) y SMCLK (Sub-Main Clock) permanecen activos. MCLK (Master Clock) esta deshabilitado

- Modo de bajo consumo 1 (LPM1- Low Power Mode 1);
    - El CPU esta deshabilitado
    - ACLK y SMCLK permanecen activos. MCLK esta deshabilitado
    - El generador DC del DCO (Digitally Controller Oscillator) esta deshabilitado si el DCO no se usa en el Modo Activo

- Modo de bajo consumo 2 (LPM2- Low Power Mode 2);
    - El CPU esta deshabilitado
    - MCLK y SMCLK están deshabilitados
    - El generador DC del DCO está habilitado
    - ACLK permanece activo

- Modo de bajo consumo 3 (LPM3- Low Power Mode 3);
    - El CPU esta deshabilitado
    - MCLK y SMCLK están deshabilitados
    - El generador DC del DCO está deshabilitado
    - ACLK permanece activo

- Modo de bajo consumo 4 (LPM4- Low Power Mode 4);
    - El CPU esta deshabilitado
    - ACLK esta deshabilitado
    - MCLK y SMCLK están deshabilitados
    - El generador DC del DCO está deshabilitado

- El Cristal Oscilador externo es detenido

1.1.6 LA MEMORIA FLASH

La memoria Flash puede ser programada a través del puerto Spy-Bi-Wire/JTAG, o por el microcontrolador usando su CPU. La CPU puede realizar escrituras de un solo byte o una sola palabra en la memoria Flash. Las características de la memoria Flash incluyen:

- La memoria Flash tiene n segmentos de memoria principal y cuatro segmentos de memoria de información (de A hasta D) de 64 bytes cada uno. Cada segmento en la memoria principal tiene un tamaño de 512 bytes. EL número de segmentos n depende de cada microcontrolador y en el caso del MSP430F2013 n es igual a 4 (2 KB).

- Los segmentos 0 hasta n puede ser borrados en un paso, o cada segmento puede ser individualmente borrado.

- Los segmentos de A hasta D pueden ser borrados individualmente, o como un grupo con segmentos de 0 a n. Segmentos de A hasta D son llamados también como memoria de información.

- El segmento A contiene datos de calibración. Después de un reset el segmento está protegido nuevamente contra escritura y borrado. Puede ser desbloqueada pero se debe tener cuidado de no borrar este segmento si los datos específicos de calibración del dispositivo son requeridos.

1.1.7 LOS DISPOSITIVOS PERIFÉRICOS

Los dispositivos periféricos están conectados al CPU a través de buses de datos, de dirección, y de control y pueden ser controlados directamente como si se tratara de una dirección de memoria RAM cualquiera. A continuación se describen los periféricos existentes en el microcontrolador MSP430F2013.

1.1.7.1 El oscilador y el reloj del sistema

El sistema de reloj está controlado por el módulo básico del reloj que incluye soporte para un oscilador de cristal de 32.768Hz, un oscilador interno de bajo poder de consumo y baja frecuencia, y un oscilador interno digitalmente controlado (DCO). El módulo básico de reloj es diseñado para cumplir con ambos requisitos: bajo costo del sistema y bajo poder de consumo. El DCO interno provee una fuente de reloj de rápido encendido y se estabiliza en menos de 1us. Para su correcta calibración se deben usar los valores provistos de fábrica que se encuentran en la memoria Flash del segmento A. Para una frecuencia específica se necesitan configurar 2 valores, las direcciones de memoria que contienen estos valores de calibración específicos para cada frecuencia del DCO están indicadas en la tabla 1.3. El módulo básico de reloj provee las siguientes señales del reloj:

- El reloj auxiliar (ACLK), Alimentado tanto de un cristal del reloj de 32768Hz o del oscilador LF interno.

- El reloj principal (MCLK), que es el reloj de sistema usado por el CPU.

- El reloj sub-principal (SMCLK), que es el reloj del sub-sistema usado por los módulos periféricos.

Datos de Calibración del DCO (provisto de fabrica en la memoria Flash en el segmento A)			
Frecuencia del DCO	**Registro de calibración**	**Tamaño**	**Dirección**
1 MHz	CALBC1_1MHz	Byte	010FFh
	CALDCO_1MHz	Byte	010FEh
8 MHz	CALBC1_8MHz	Byte	010FDh
	CALDCO_8MHz	Byte	010FCh
12 MHz	CALBC1_12MHz	Byte	010FBh
	CALDCO_12MHz	Byte	010FAh
16 MHz	CALBC1_16MHz	Byte	010F9h
	CALDCO_16MHz	Byte	010F8h

Tabla 1.3: Datos de Calibración del DCO. (Fuente: Texas Instruments)

1.1.7.2 El circuito de brownout

El circuito del brownout es implementado para proveer la señal interna correcta del reset al dispositivo durante la secuencia de encendido y apagado del microcontrolador. En el anexo A.1 se encuentra una descripción del funcionamiento de este circuito.

1.1.7.3 Entradas/Salidas Digitales

Hay un puerto de entrada/salida de 8 bits implementado en P1 y otro puerto de 2 entradas/salidas implementado en P2, sus características son:

- Los bits de todas las entradas/salidas pueden ser programados independientemente.

- Cualquier combinación de entrada, salida o condición de interrupción es posible.

- Capacidad de selección del flanco para la interrupción en los 8 bits del puerto 1 y los 2 bits del puerto 2.

- Acceso de lectura/escritura a los registros de control del puerto es soportado por todas las instrucciones.

- Cada entrada/salida tiene una resistencia individual de pull-up/pull-down programable.

Mayor información de la descripción técnica del microcontrolador y sus puertos, se presenta en el anexo B.1 y el anexo B.2.

1.1.7.4 El Temporizador del perro guardián "Watchdog Timer" (WDT+)

La función primaria del módulo watchdog timer (WDT+) es realizar un reset controlado del sistema cuando ocurra un problema en la ejecución del software. Si el espacio de tiempo seleccionado caduca, entonces un reset al sistema es generado. Si la función del perro guardián no es necesaria en una aplicación, entonces el módulo puede ser deshabilitado o configurado como un contador de intervalos y puede generar interrupciones en los espacios de tiempo seleccionados.

1.1.7.5 El Timer_A2

El Timer_A2 es un temporizador/contador de 16 bits con dos registros de captura/comparación. Puede dar soporte a la captura/comparación múltiple, salidas PWM, y medidas de intervalos de tiempo. También posee capacidad de interrupción. Las interrupciones pueden ser generadas desde el contador en condiciones de desborde y de cada uno de los registros de captura/comparación.

1.1.7.6 USI (Universal Serial Interface - Interface Serial Universal)

El módulo Interface Serial Universal (USI) sirve para comunicación de datos seriales y provee el hardware básico para los protocolos síncronos de comunicación como SPI e I2C.

1.1.7.7 Conversor Análogo/Digital Sigma/Delta SD16_A (MSP430x20x3)

El módulo Sigma/Delta SD16_A soporta conversiones análogas/digitales de 16 bits. El módulo implementa un núcleo sigma/delta de 16 bits y un generador de referencias. Además de las entradas analógicas externas, un sensor de VCC interno y un sensor de temperatura están también disponibles. El MSP430F2013 no posee sensor de temperatura.

Un mayor detalle de los periféricos disponibles se puede encontrar en el anexo B.2.

1.2 EXPLICACIÓN DE LA HERRAMIENTA DE DESARROLLO EZ430-F2013 Y EL MICROCONTROLADOR F2013

1.2.1 BREVE DESCRIPCIÓN DE LA HERRAMIENTA EZ430-F2013

El eZ430-F2013 es una herramienta completa que provee todo el hardware y software que permite evaluar el MSP430F2013 y además puede completar un proyecto entero en base al mismo. El eZ430-F2013 usa el IAR Embedded Workbench Integrated Development Environment (IDE) que provee la programación y la emulación completa con la opción de diseñar un sistema autónomo o separar la tarjeta que contiene al microcontrolador para integrarlo a un diseño existente. El puerto USB del computador provee la energía para alimentar el MSP430 de ultra-bajo consumo.

Todos los 14 terminales del MSP430F2013 son accesibles en la tarjeta de pruebas MSP-EZ430D para una corrección de errores fácil e interactiva con los dispositivos periféricos. Adicionalmente, uno de los terminales digitales de I/O está conectado a un diodo emisor de luz para una indicación visual. La figura 1.4 muestra una fotografía de la herramienta eZ430-F2013, incluido el microcontrolador MSP430F2013.

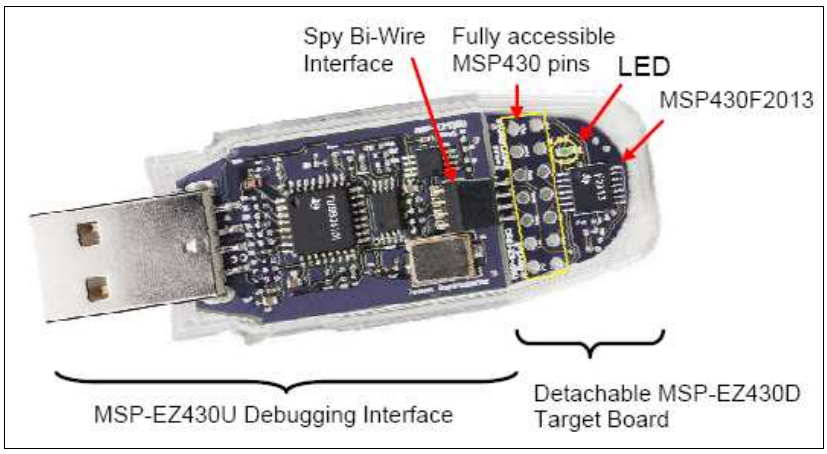


Figura 1.4: Imagen del eZ430-F2013. (Fuente: Texas Instruments).

1.2.2 CONTENIDO DEL KIT EZ430-F2013

Tiene un eZ430-F2013. El hardware es alojado al interior de un encapsulado plástico que puede ser abierto para separar la tarjeta de pruebas MSP-EZ430D de la interfaz de depuración MSP-EZ430U.

Un MSP430 Development Tool CD-ROM que contiene varios documentos incluyendo los relacionados con el eZ430-F2013:

- El MSP430x2xx Family User's Guide, SLAU144A
- El MSP-FET430 FLASH Emulation Tool User's Guide, SLAU138C
- La Guía de Errata MSP-FET430 Emulation Tool User's Guide Errata, SLAZ027
- El EZ430-F2013 User's Guide.
- IAR Embedded Workbench (Version Kickstart)
- Herramientas de desarrollo poderosas y fáciles de usar que incluyen:
    - El simulador (incluyendo simulación de los dispositivos periféricos y las interrupción)
    - El compilador C
    - El ensamblador
    - El editor de enlace
    - Los emuladores (ICE y JTAG)
    - Kits de evaluación
    - El programador del dispositivo
- Las notas aplicativas
    - Código de ejemplo

Es por todo este contenido que el kit permite diseñar cualquier sistema sin necesidad de herramientas o programas adicionales.

1.2.3 DESCRIPCION TÉCNICA DEL MICROCONTROLADOR

1.2.3.1. Arquitectura

La arquitectura del MSP430 incluye una CPU tipo RISC de 16 bits que se interconecta al resto de dispositivos mediante dos buses: El primero es el bus de direccionamiento de memoria común (llamado MAB - Memory Address Bus) y el segundo es el bus común para datos (llamado MDB - Memory Data Bus), estos buses permiten transmitir y recibir datos desde cualquiera de los dispositivos integrados en el microcontrolador, tales como: La memoria RAM, La memoria Flash, Pórticos 1 y 2, Timer_A2, USI, etc.. Además posee un sistema de reloj muy flexible que permite seleccionar de manera muy sencilla la fuente de reloj para la CPU y para los dispositivos periféricos. Todos los dispositivos periféricos son modulares y están correlacionados entre sí mediante los buses MAB y MDB. Las características más importantes de la familia MSP430x2xx son las siguientes:

- Tecnología de bajo consumo que extiende la vida de la batería.
- 0.8 uA en modo de reloj en tiempo real
- 250 uA al activarse el micro (ejecuta *millones de instrucciones por segundo* "MIPS").
- Sistema analógico de gran rendimiento ideal para mediciones de precisión.
- Temporizadores/Comparadores con capacidad de interrupción, es decir que pueden provocar la generación de interrupciones al alcanzar un valor determinado.
- CPU de arquitectura RISC de 16-bits que permite nuevas aplicaciones usando una fracción del tamaño de código.

- El diseño compacto del núcleo principal reduce el consumo de energía y el costo.

- Optimizado para la programación moderna de alto nivel.

- Solamente 27 instrucciones de núcleo y siete modos de direccionamiento.

- Capacidad de interrupción-vectorial extensiva.

- La Flash programable en sistema permite cambios al código, actualizaciones in situ y adquisición de datos de estado.

En las figura 1.5 se muestra la arquitectura en forma general de los microcontroladores de la familia MSP430 y en la figura 1.6 se muestra el esquema funcional en bloques del microcontrolador MSP430x20x3, con los periféricos específicos de este modelo.

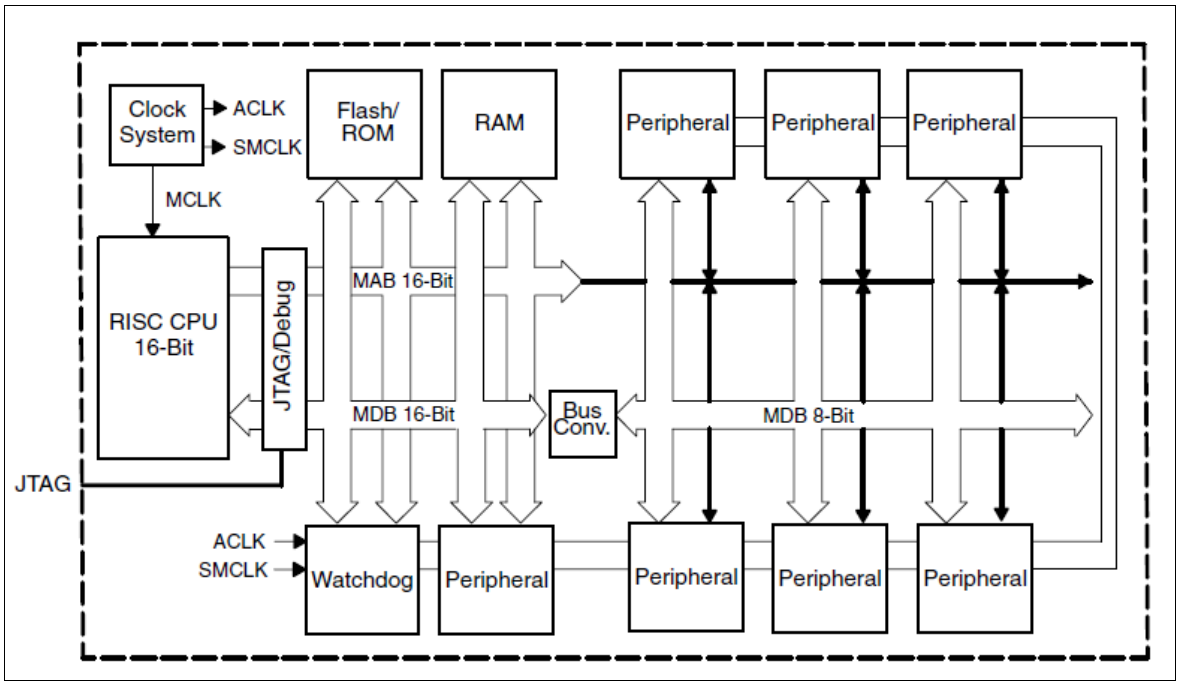


Figura 1.5 Arquitectura del MSP430. (Fuente: Texas Instruments)

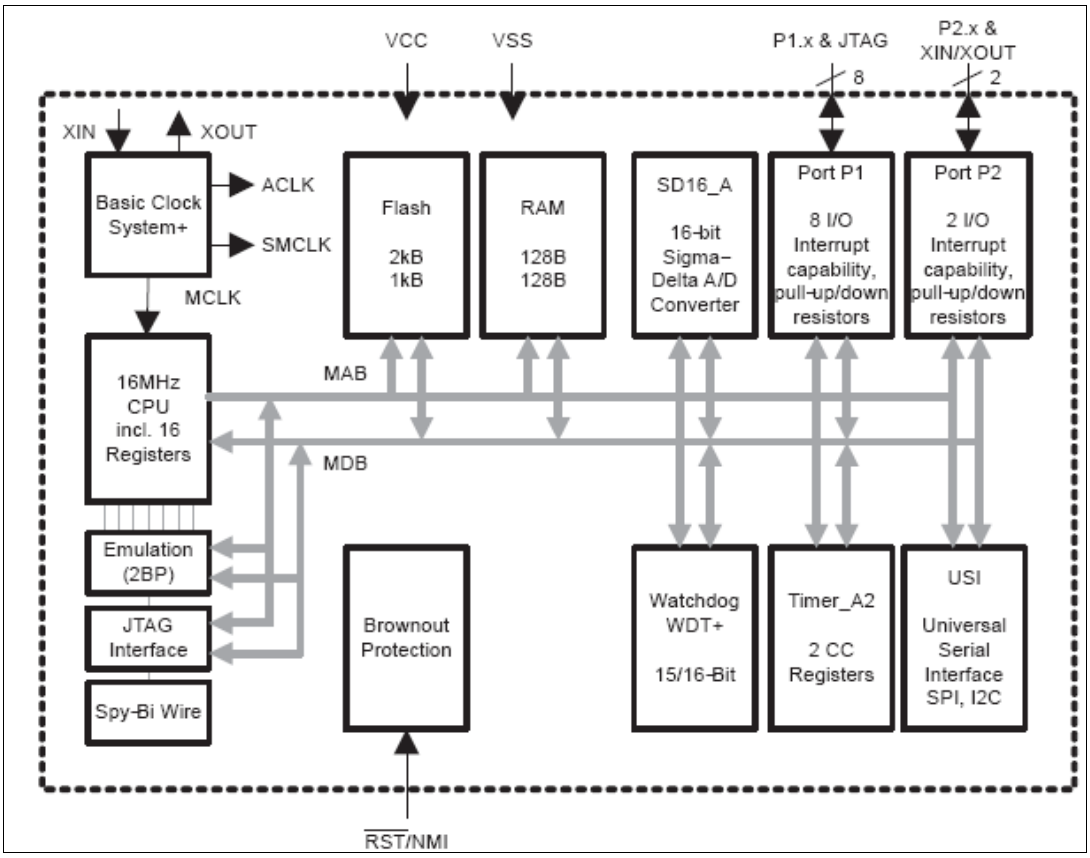


Figura 1.6: Esquema funcional en bloques, MSP430x20x3. (Fuente: Texas Instruments)

1.2.3.2. Sistema de reloj

El sistema de reloj es diseñado específicamente para aplicaciones que usan baterías. Un reloj auxiliar de baja-frecuencia (ACLK) de 32 KHz es controlado directamente. El ACLK puede ser usado tanto para una función de reloj en tiempo real como para la función de activado del microcontrolador. Un oscilador digitalmente controlado de alta velocidad (DCO) integrado puede generar la señal de reloj principal (MCLK) utilizado por la CPU y los dispositivos periféricos de alta velocidad. El DCO está activo y estable en menos de 2 µs a 1 MHz.

Gracias a todas estas características del sistema de reloj, el microcontrolador puede operar en dos velocidades diferentes, lo que permite implementarlo en aplicaciones tipo “burst”, es decir, aplicaciones que solo en ciertos instantes de

tiempo requieren la máxima velocidad de procesamiento, con lo cual se obtiene un ahorro significativo de energía al mantener al CPU en estado pasivo el resto de tiempo. Las señales de reloj pueden aplicarse de la siguiente manera:

- Reloj auxiliar de baja frecuencia (ACLK) = Modo de espera de muy bajo consumo.
- Reloj principal de alta velocidad (DCO) = Procesamiento de señales de alto rendimiento.

1.2.3.3. Emulación integrada

La lógica de emulación reside en el dispositivo mismo y es accedido vía JTAG sin necesidad de usar ningún recurso adicional del sistema.

Los beneficios de la emulación integrada son:

- Desarrollo y depuración con ejecución a máxima velocidad, puntos de parada, depuración paso a paso en la aplicación son soportados.
- El desarrollo en el sistema está sujeto a las mismas características de la aplicación final.
- La integridad de las señales se conserva y no está sujeta a interferencia por cableado externo.

1.2.3.4. Espacio de Direccionamiento

Los circuitos MSP430 tienen un espacio de direccionamiento compartido con los registros de función especiales (SFRs), dispositivos periféricos, la RAM, y la memoria Flash/ROM como se muestra en la figura 1.7. Los datos pueden ser accedidos como bytes o palabras. El acceso del código es llevado a cabo siempre sobre direcciones pares. El espacio de memoria direccionable es actualmente de 128 KB.

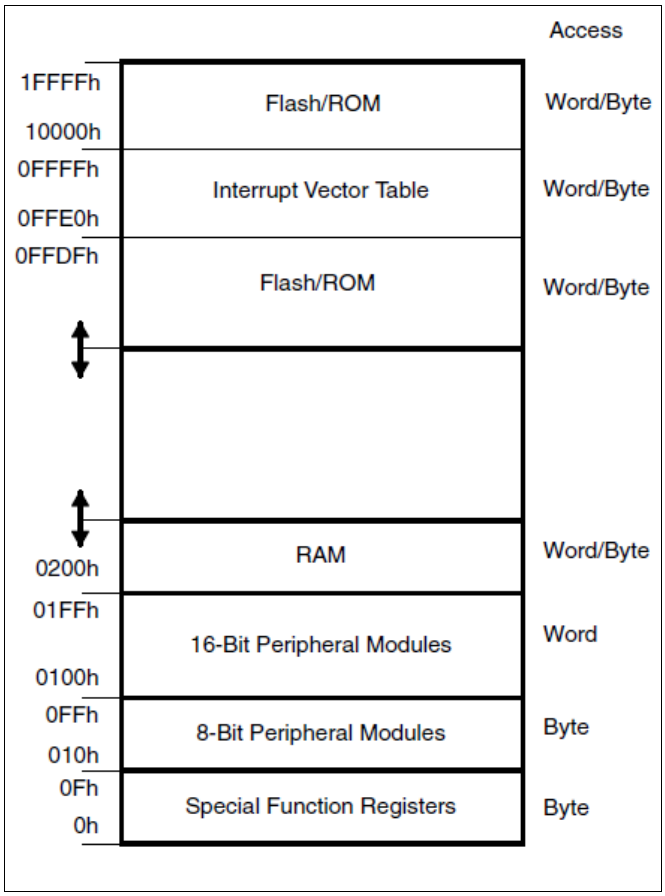


Figura 1.7: Mapa de memoria. (Fuente: Texas Instruments)

1.2.3.7.1 Memoria Flash/ROM

La dirección inicial de la Flash/ROM depende de la cantidad de memoria Flash/ROM presente y varía en cada dispositivo. La dirección final de la memoria Flash/ROM es 0x1FFFF. La memoria Flash/ROM puede ser usada tanto para código de programa como para datos, pudiendo leerse tablas almacenadas directamente de la Flash/ROM sin necesidad de copiarlas a la memoria RAM.

La tabla de los vectores de interrupción está colocada dentro de los 16 bytes altos de la memoria Flash/ROM, con el vector de mayor prioridad de interrupción ubicado en la dirección más alta que es la 0x1FFFF. Un mayor detalle de los vectores de interrupción y sus fuentes se puede encontrar en el anexo B.3

1.2.3.7.2 Memoria RAM

La memoria RAM inicia en 0x0200h. La dirección final de la memoria RAM depende de la cantidad de memoria RAM presente en el dispositivo. Esta memoria puede ser usada tanto para código de programa como para datos.

1.2.3.7.3 Módulos Periféricos

Los módulos periféricos están ubicados dentro del mismo espacio de memoria que el resto de dispositivos. El espacio de direcciones 0x0100h a 0x01FFh está reservado para los módulos periféricos de 16 bits. Estos módulos deben ser accedidos usando instrucciones tipo “word”. Si se usa instrucciones tipo “byte” solo son accesibles las direcciones pares y el resultado del byte alto es siempre “0”.

El espacio de memoria entre 0x0010h y 0x00FFh está reservado para los módulos periféricos de 8 bits. Estos módulos deben ser accedidos siempre con instrucciones tipo “byte”. En caso de lectura de estos módulos usando instrucciones tipo “word” resultaría en datos impredecibles en el byte alto. En cambio, en caso de escritura con instrucciones tipo “word”, solo el byte bajo será escrito en el registro del modulo periférico, ignorándose el byte alto. Mayor información sobre los módulos periféricos y sus registros puede ser encontrado en el anexo B.4.

1.2.3.7.4 Registros de funciones especiales

Algunas funciones de los dispositivos periféricos son configuradas en los SFRs. Los SFRs están ubicados en los 16 bytes más bajos del espacio de memoria RAM y son organizados por byte. Los SFRs deben ser accedidos usando instrucciones de byte solamente. En las hojas de datos del archivo slau144e.pdf se puede observar los bits de los SFR aplicables para cada dispositivo periférico.

1.2.3.5. Terminales del dispositivo MSP430x20x3

La figura 1.8 muestra la configuración de los terminales del MSP430x20x3, que coincide con el microcontrolador usado en este proyecto, que es el MSP430F2013

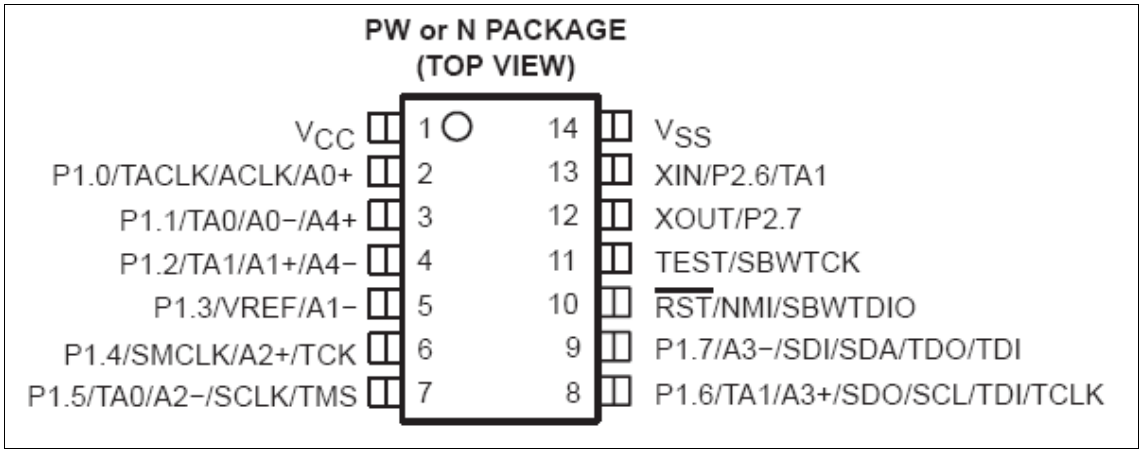


Figura 1.8: Detalle de los terminales del MSP430F2013. (Fuente: Texas Instruments).

1.2.3.6. Mejoras de la familia MSP430x2xx

La tabla 1.4 resalta las mejoras realizadas a la familia de microcontroladores MSP430x2xx.

Tema	Mejora
Reset (Reinicio)	• Circuito de Brownout que permite reiniciar el dispositivo en caso de falla en el voltaje se incluye en todos los dispositivos MSP430x2xx • Las banderas PORIFG y RSTIFG han sido añadidas a IFG1 para indicar la causa de una re-inicialización. • Una ejecución de instrucción desde las direcciones de memoria 0x0000 a 0x01FF reiniciara el dispositivo.
Watchdog Timer (Temporizador de "Perro guardián")	• Todos los dispositivos MSP430x2xx integran el modulo Watchdog Timer+ (WDT+). El WDT+ se asegura que la fuente de reloj nunca sea deshabilitada.

Sistema básico de reloj	• El oscilador LFXT1 tiene condensadores de carga seleccionables en el modo de LF • El LFXT1 soporta cristales de hasta 16 MHz en el modo de HF • El LFXT1 incluye detección de falla de oscilador en el modo de LF • los terminales XIN y XOUT tienen funciones compartidas sobre los terminales 20 y 28 del dispositivo • El ROSC externo que le caracteriza al DCO no es soportado por todos los dispositivos. El software no debe fijar el LSB del registro BCSCTL2 en este caso. • La frecuencia operativa del DCO ha sido incrementada significativamente • La estabilidad de temperatura del DCO ha sido mejorada significativamente
Memoria Flash	• La memoria de información tiene 4 segmentos de 64 bytes cada una. • El segmento A es bloqueado individualmente con el bit LOCKA • Toda la información es protegida contra un borrado masivo con el bit LOCKA • El borrado de segmento puede ser interrumpido por una "interrupción" • Las actualizaciones en la memoria flash pueden ser suspendidas por una interrupción • El voltaje necesario para la programación de la memoria flash ha sido bajado a 2.2 V • El tiempo necesario para programar/borrar la memoria ha sido reducido • Un error de reloj suspende la actualización de la memoria flash
Entradas y salidas Digitales	• Todos los puertos tienen integrado resistores pull-up/pull-down • Las funciones de P2.6 y P2.7 han sido añadidas a dispositivos de 20 y 28 patillas. Éstas son funciones compartidas con XIN y XOUT. El software no debe borrar los bits del P2SELx para estos terminales si la operación de cristal oscilador externo es requerida.
Comparador A	• Han sido expandidas las capacidades de entrada del Comparador A con un nuevo multiplexor de entrada.
Bajo Consumo	• El consumo típico en modo LPM3 ha sido reducido casi en un 50 % a 3 V • El tiempo de inicio del DCO ha sido reducido significativamente
Frecuencia operativa	• La frecuencia operativa máxima es de 16 MHz a 3.3 V
BSL	• Una contraseña incorrecta provoca un borrado masivo de la memoria • La secuencia de entrada de BSL es más robusta para prevenir un ingreso accidental y un borrado accidental.

Tabla 1.4: Mejoras a la familia MSP430x2xx. (Fuente: Texas Instruments)

1.2.3.7. Reinicio e inicialización del sistema

La circuitería de re-inicialización de sistema mostrada en la figura 1.9 obtiene las señales de re-inicialización POR y PUC (Power-On Reset y Power-Up Clear) del microcontrolador. Diferentes eventos provocan estas señales de re-inicialización y las diferentes condiciones iniciales existentes dependen de qué señal fue generada.

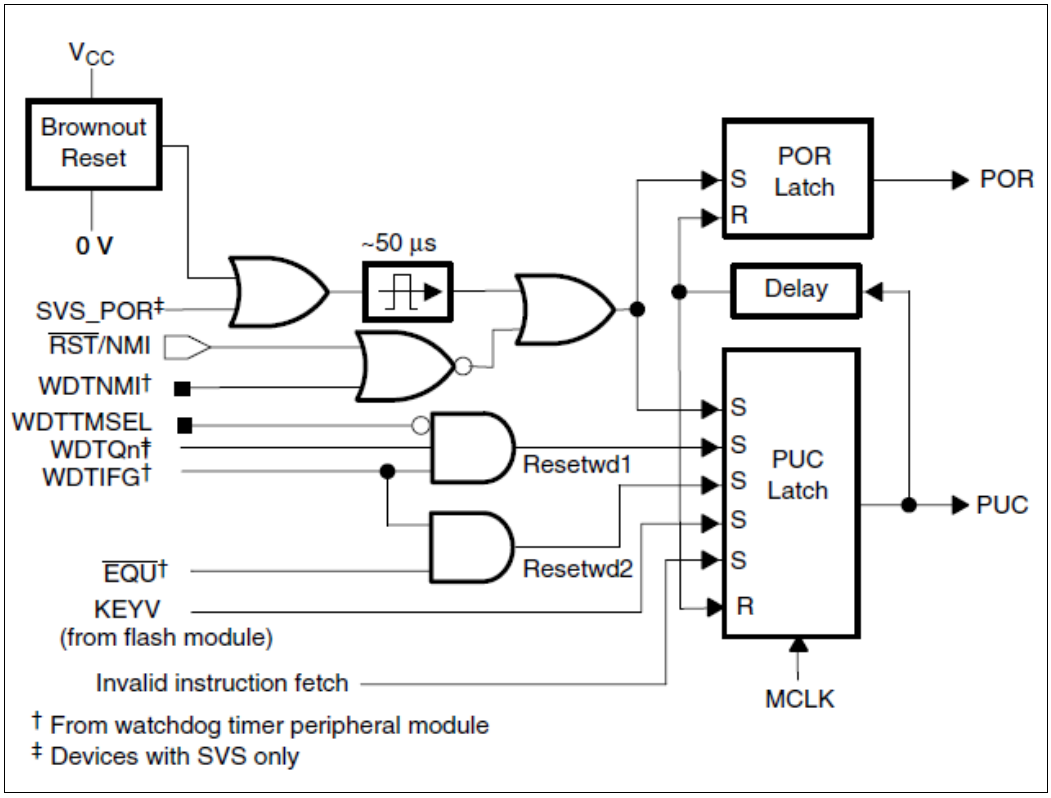


Figura 1.9 Circuito esquemático de la generación de las señales POR y PUC. (Fuente: Texas Instruments)

La señal de “POR” indica la re-inicialización del dispositivo. La señal “POR” es solamente generada por uno de los tres eventos siguientes:

- Suministrar energía al dispositivo (Encender el dispositivo)
- Una señal de bajo voltaje sobre el terminal RST/NMI cuándo está configurado en el modo de re-inicialización

- Una condición de bajo voltaje proveniente del Supervisor De Voltaje SVS cuando PORON = 1.

La señal "PUC" es generada siempre que una señal "POR" sea generada, por el contrario, una señal "POR" no es generada por una señal "PUC". La señal “PUC” es provocada por los siguientes eventos:

- Una señal “POR”
- Expiración del tiempo de espera del Watchdog cuando se está en modo de perro guardián únicamente
- Violación de la llave de seguridad del temporizador “Watchdog”
- Violación de la llave de seguridad de la memoria Flash
- La ejecución de una instrucción desde la sección de memoria de periféricos, que se encuentra entre las direcciones de memoria 0h a 01FFh

1.2.3.7.1 Circuito de Brownout (Brownout Reset - BOR)

El circuito de Brownout detecta la existencia de niveles de voltaje bajos como los que se encuentran al alimentar el microcontrolador por primera vez. El circuito de brownout reinicia el dispositivo provocando una señal “POR” cuando la alimentación es aplicada o removida. Los niveles operativos son mostrados en la Figura 1.10.

La señal "POR" se pone activa cuando el voltaje de alimentación cruza el nivel de VCC inicial - VCC(Start). Este queda activo hasta que el voltaje de alimentación cruza el umbral de Voltaje V(B_IT+) y el tiempo de retraso t(BOR) transcurre. El tiempo de retraso t (BOR) es adaptable y puede ser más largo para una rampa de VCC. El voltaje de histéresis Vhys (B_ IT-) asegura que el voltaje de alimentación debe caer por debajo del voltaje V(B_IT-) para generar otra señal POR desde el circuito de reset de brownout.

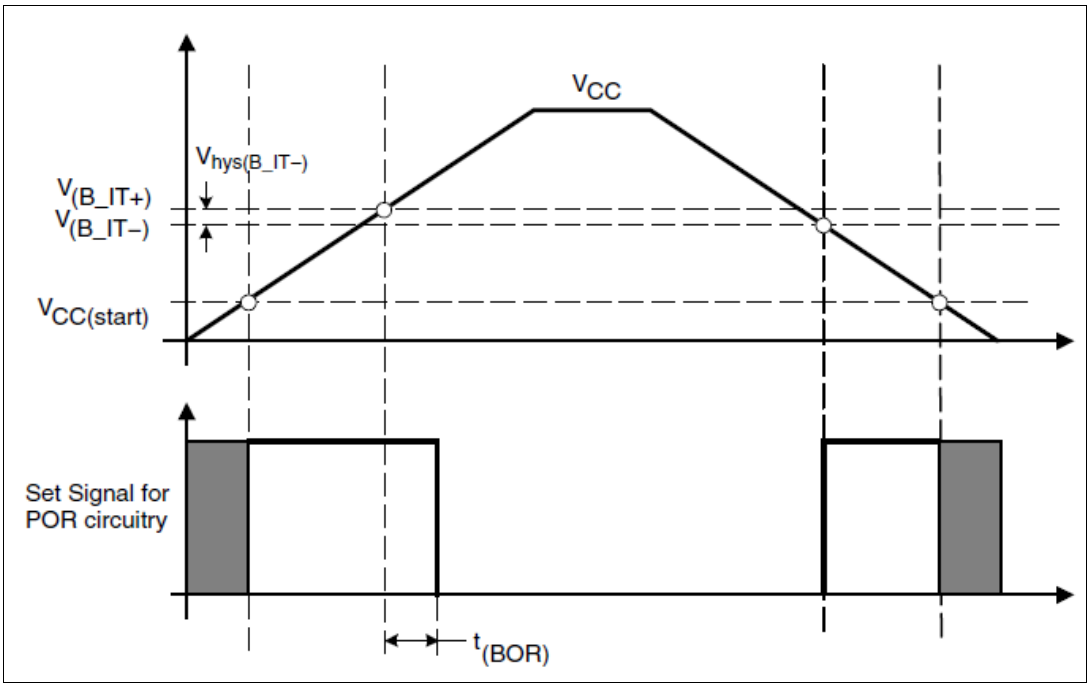


Figura 1.10: Temporización del Brownout. (Fuente: Texas Instruments)

Como el nivel voltaje V(B_IT-) está significativamente encima del nivel de voltaje VCC(start) del circuito de POR, el BOR provee la señal de reset en caso de existir un corte de energía y VCC no caiga por debajo de VCC(start). En las hojas de datos del dispositivo que se encuentran en el anexo B.5 se puede encontrar mayores detalles.

1.2.3.7.2 Condiciones iníciales del dispositivo después de un reset de sistema

Después de una señal "POR", las condiciones iníciales del MSP430 son:

- El terminal RST/NMI es configurado en el modo de reseteo.
- Los terminales de E/S son cambiados al modo de entrada.
- Los módulos periféricos y registros son inicializados tal y como se describe en el respectivo manual.
- Registro de estado (SR) es inicializado.
- EL modulo del temporizador Watchdog es activado en modo de "Perro Guardián".
- Se carga al contador de programa (PC) la dirección de memoria donde se encuentra el vector de reset (0FFFEh). Si el contenido de memoria del

vector de reset es el valor 0FFFFh, el dispositivo será desactivado inmediatamente para que el consumo de energía sea mínimo.

1.2.3.7.2.1 Inicialización de software

Después de un reset del sistema, el software del usuario debe inicializar al circuito para cumplir con los requisitos de la aplicación, por lo que debe realizar, entre otras, las siguientes tareas:

- Inicializar el SP, típicamente con la dirección de inicio de la memoria RAM que es 0200h.
- Inicializar el Watchdog según los requisitos de la aplicación.
- Configurar los módulos periféricos a los requisitos de la aplicación.

Adicionalmente, las banderas del temporizador Watchdog, falla del oscilador, y memoria Flash pueden ser evaluadas para determinar el origen de la reinicialización.

1.2.3.8. Condiciones operativas

1.2.3.8.1 Valores absolutos máximos

Tensiones que sobrepasen los valores indicados a continuación podrían causar el daño permanente del dispositivo. Estos son valores “extremos” solamente, y la operación funcional del dispositivo en éstos o cualquier otra condición superior a aquellas mostradas en la sección “Condiciones operativas recomendadas" no están implicadas. La exposición del dispositivo a los valores máximos absolutos por períodos prolongados podría afectar la confiabilidad de dispositivo.

La siguiente lista muestra los valores máximos absolutos:

- Voltaje aplicable entre VCC y VSS: de 0.3 V hasta 4.1 V

- Voltaje aplicable en cualquier terminal, referenciado con VSS va desde -0.3 V hasta VCC + 0.3 V, a excepción del voltaje necesario para quemar el fusible del JTAG (*VFB*) que si puede ser excedido.
- Corriente de diodo en cualquier terminal de dispositivo: 2 mA.
- Temperatura de almacenamiento, Tstg sin programar el dispositivo, va desde - 55 ºC hasta 150 ºC
- Temperatura de almacenamiento, Tstg con el dispositivo programado, va desde −40ºC hasta 85ºC.

La máxima temperatura puede ser aplicada durante el proceso de soldado de acuerdo con la especificación JEDEC J-ETS-020, las temperaturas de reflujo máximas no deben ser más altas que las mostradas en las etiquetadas o cajas.

1.2.3.8.2 Condiciones Operativas Recomendadas

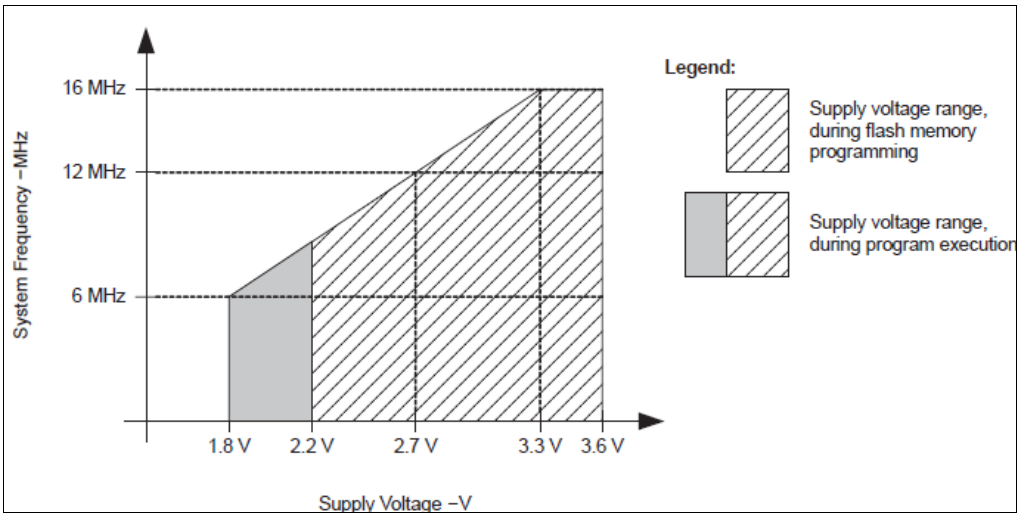
La tabla 1.4 muestra una tabla que contiene los valores para permitir que el microcontrolador trabaje en optimas condiciones, si alguna de ellas no fuera respetada, podría provocar daños personales o daños irreparables al microcontrolador. La figura 1.11 muestra el área operativa segura del microntrolador con sus correspondientes niveles de voltajes y frecuencias.

Parámetro	Condición	Min	Nom	Max	Unidad
Voltaje de alimentación VCC durante la ejecución de programa		1.8		3.6	V
Voltaje de alimentación VCC durante la programación/borrado de la memoria Flash		2.2		3.6	V
Voltaje de alimentación VSS			0		V
Rango de temperatura operativo al aire libre, TA		-40º		85º	ºC
Frecuencia del procesador FSYSTEM (frecuencia de MCLK máxima)	VCC=1.8V, Ciclo de trabajo= 50% ±10%	dc		6	MHz
	VCC=2.7V, Ciclo de trabajo= 50% ±10%	dc		12	
	VCC ≥3.3V, Ciclo de trabajo= 50% ±10%	dc		16	

NOTAS: *1. El CPU del MSP430 es alimentada con la señal de reloj directamente de MCLK. Tanto la fase alta como la fase baja del MCLK no deben exceder la anchura de pulso de la frecuencia máxima especificada.*

2. Los módulos podrían tener una especificación de reloj de entrada máxima diferente. Haga referencia a la especificación del módulo respectivo en la hoja de datos.

Tabla 1.5: Valores recomendados para una operación optima. (Fuente: Texas Instruments)



Nota: *La frecuencia mínima del procesador está definida por el reloj del sistema. La operación de programado o borrado de la memoria flash requiere un mínimo de 2.2V en VCC.*

Figura 1.11: Área operativa segura. (Fuente: Texas Instruments)

En el anexo B.5 se adjunta las características eléctricas del microcontrolador MSP430F2013.

1.3 DEFINICIÓN DE CARACTERÍSTICAS DEL PROTOTIPO.

El prototipo tendrá las siguientes características tanto en Hardware como en Software:

1.3.1 CARACTERISTICAS DE HARDWARE:

1.3.1.1 Entradas Análogas/Digitales:

El dispositivo contara con 4 entradas Analógicas y su funcionamiento estará basado en el conversor A/D Sigma/Delta provisto internamente en el microcontrolador MSP430F2013, que para la aplicación será configurado como conversor de un solo paso con un voltaje de referencia interno de 1.2V, lo que permite ingresar en el terminal únicamente voltajes entre 0 y 0.6V (Vmax = Vref / 2).

Al disponer de una solo entrada, se usara un multiplexor bidireccional de tecnología CMOS que nos permita la multiplicación de las 4 entradas análogas respectivas además de permitir reducir el voltaje y hacerlo proporcionalmente, consiguiendo a la salida rangos aproximados de entre 0 y 0.6 Voltios para voltajes de entrada de 0 a 5 Voltios. Este circuito será detallado más adelante. Las entradas serán seleccionadas progresivamente y si se necesita hacerlas trabajar como entradas digitales (ON / OFF), se debe realizar la discriminación del nivel de transición mediante código de programa, también estas entradas no podrán ser usadas mediante peticiones de interrupción.

1.3.1.2 Salidas Digitales para carga AC/DC

El prototipo tendrá 4 salidas independientes programables con posibilidad de manejar cargas tanto AC como DC gracias al uso de relés electromecánicos, la carga máxima estará determinada por la capacidad del relé, en este caso será de:

- AC 10A 125V
- AC 6A 250V
- DC 6A 28V

1.3.1.3 Comunicación serial con el PC

El prototipo podrá comunicarse e interactuar con un PC gracias a una interfaz RS232 integrada y programada en el microcontrolador, esta interfaz usará el puerto serial disponible del PC o en su ausencia podrá utilizar un adaptador USB-RS232, el cual permita el manejo de la interfaz serial requerida.

1.3.1.4 Display LCD y Teclado Matricial

Para facilitar el uso del prototipo por parte del usuario, se dispondrá de un display tipo LCD de 2 líneas por 16 caracteres cada una (32 caracteres en total) y de un teclado matricial de 4 por 4 caracteres (16 teclas en total), lo que proporcionara al usuario la facilidad de ingresar a las funciones internas del prototipo sin necesidad de usar un PC.

1.3.2 CARACTERÍSTICAS DEL SOFTWARE

En lo referente al software, el prototipo dispondrá para el usuario de todos los comandos disponibles tanto en lenguaje ensamblador como en lenguaje C, tendrá bloques de programa especiales para la configuración de los periféricos instalados en su construcción, estos son las 4 entradas, las 4 salidas, la comunicación serial, la adquisición de datos, la visualización de información en el display y el escaneo del teclado, entre otras funciones internas especificas de este prototipo.

Además se proveerá de un programa diseñado para interactuar con el prototipo desde el PC, permitiendo al PC adquirir datos de las entradas o accionar los relés en sus salidas, como ejemplo básico de aplicación.

1.4 DISEÑO DEL PROTOTIPO, A NIVEL CIRCUITAL.

A continuación se presentan los esquemas circuitales de cada sección del prototipo, además de la explicación de cada una de ellas. En el CD que complementa este proyecto se encuentra el directorio “Esquemas Electronicos” que contiene todos los archivos creados dentro del programa Altium Designer con los diseños de todos los circuitos aquí mencionados.

1.4.1 ESQUEMA CIRCUITAL DE LAS ENTRADAS ANALOGAS

El esquema de la figura 1.12 muestra el circuito multiplexor y reductor de voltaje de las 4 entradas análogas, en este esquema se indica respectivamente la salida multiplexada que va conectada a la entrada del conversor A/D Sigma/Delta del microcontrolador, las 4 entradas análogas y los terminales de selección de entrada.

Está basado en el integrado NTE4529B tipo CMOS, Selector de datos Doble de 4-Canales (8 entradas) Análogas. La reducción de voltaje se la realiza con un divisor de tensión formado por 2 resistencias, una de 1K ohmio y otra de 150 ohmios conectadas en serie a la salida del multiplexor.

Las salidas del microcontrolador son conectadas directamente a las entradas del multiplexor, aunque el voltaje máximo de funcionamiento del microcontrolador es de 3V, los circuitos TTL detectan un nivel alto (1L) entre los 2 y 5 voltios, por lo tanto no se requieren realizar adaptaciones o utilizar circuitos de interface en estos valores de salida.

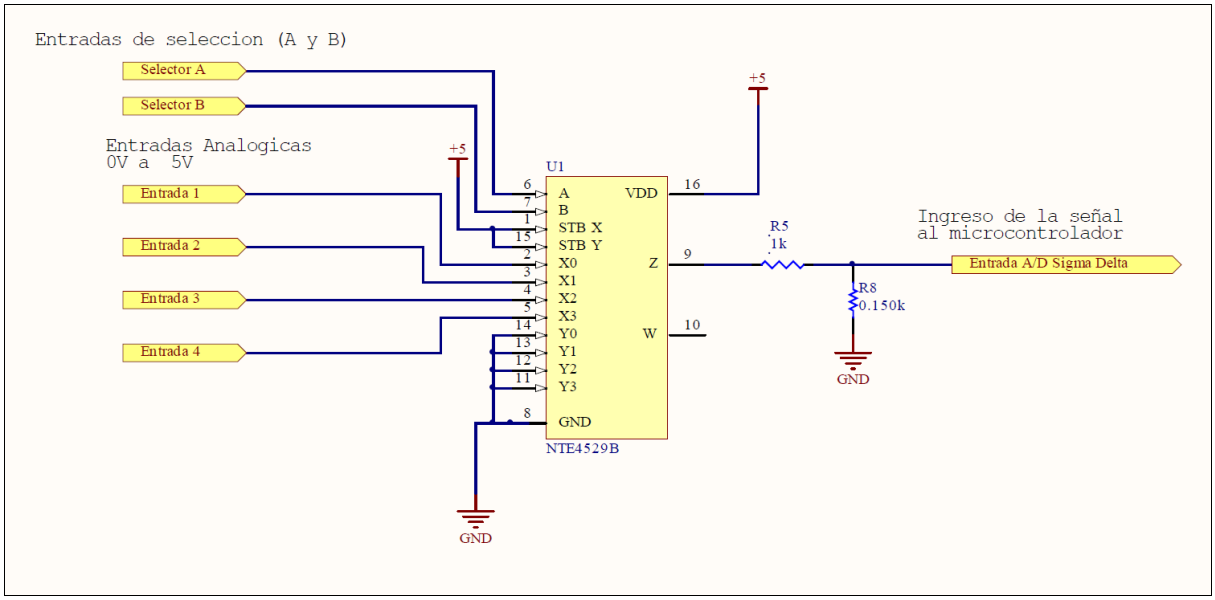


Figura 1.12: Esquema electrónico del selector/seguidor de voltaje para las entradas Análogas. (Elaborado por el Autor)

1.4.2 ESQUEMA CIRCUITAL DE LAS SALIDAS DIGITALES

En este circuito se muestran las 4 salidas digitales que podrían manejar cargas tanto en AC como en DC. Para evitar los problemas de diferencia de voltaje y la poca corriente permitida en los terminales del microcontrolador, los terminales manejan relés conectados a los mismos por medio de transistores que trabajen en corte y saturación, es decir, que trabajan como interruptores para accionar los relés. La corriente es limitada por la resistencia de 10K ohmios.

En la figura 1.13 se muestra el esquema eléctrico incluido los valores de los elementos usados.

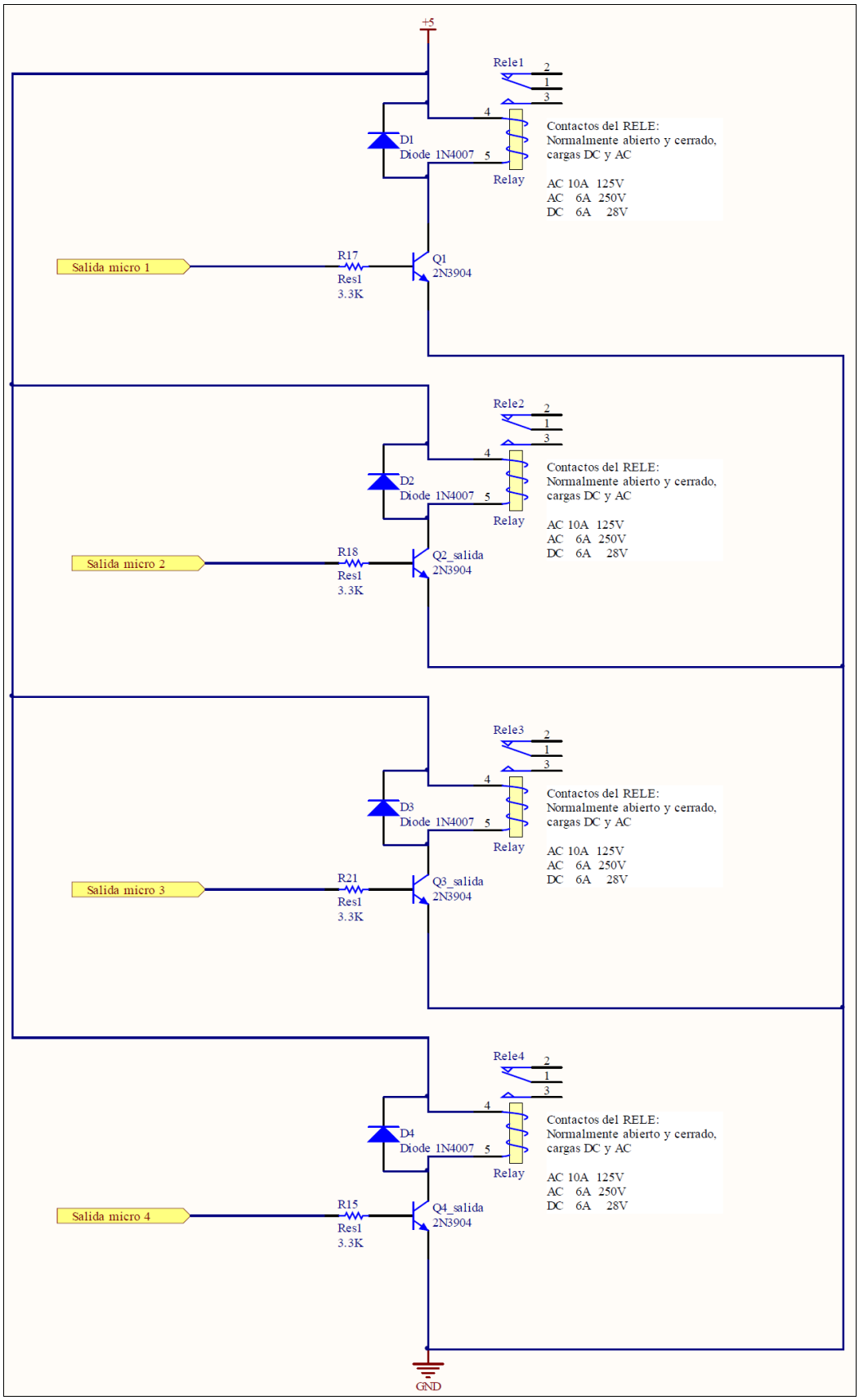


Figura 1.13: Esquema eléctrico de las salidas digitales. (Elaborado por el Autor)

1.4.3 ESQUEMA CIRCUITAL DEL CONVERSOR TTL A RS232, COMUNICACIÓN SERIAL

El circuito mostrado en la figura 1.14 es el que realiza la conversión de niveles de tensión entre TTL y RS232, para cumplir con la norma para la comunicación serial.

La señal de salida del microcontrolador (0 a 3V) será aplicada directamente al terminal del MAX232 aprovechando que el voltaje mínimo que debe ser ingresado para detectarse un nivel alto es de 2V (Ver en el anexo C.1 las especificaciones técnicas del integrado MAX232, además del diagrama de conexión), la salida TTL del MAX232 será controlada con una resistencia de 10k ohmios para permitir la conexión al terminal del microcontrolador y de esta manera controlar la corriente y el voltaje máximo que a él llega. Las señales RS-232 son conectadas directamente con el puerto serial del PC.

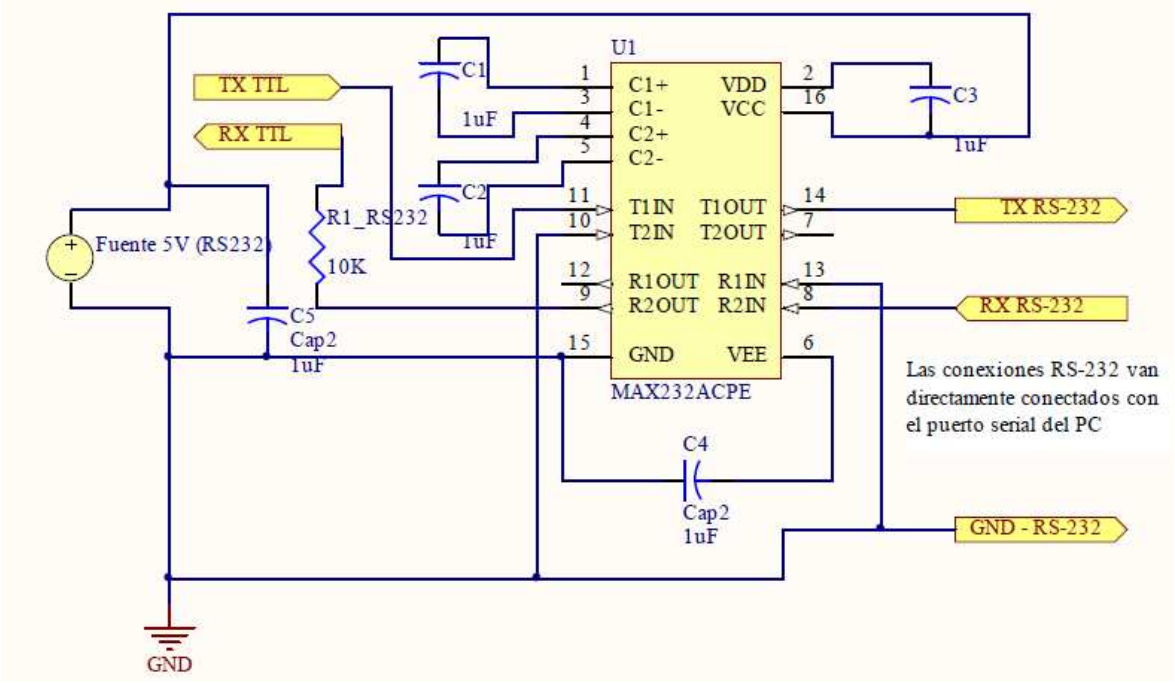


Figura 1.14: Circuito Conversor TTL-RS232. (Elaborado por el Autor)

1.4.4 DIAGRAMA DE INTERCONEXIÓN DE CADA PERIFÉRICO A CADA UNO DE LOS MICROCONTROLADORES

La figura 1.15 muestra los terminales usados de cada uno de los microcontroladores y la interconexión realizada con cada uno de los circuitos externos necesarios para implementar el prototipo. Se indican también por bloques los circuitos anteriormente explicados y la circuitería que cada dispositivo requiere para su correcto funcionamiento.

La comunicación entre cada microcontrolador se la lleva a cabo a través del protocolo I2C, que, mediante la utilización de 2 líneas de comunicación permite enlazar más un dispositivo al mismo tiempo.

Para el diseño se ha utilizado 2 familias de microcontroladores, la familia MSP430F2013 y la familia MSP430F2012.

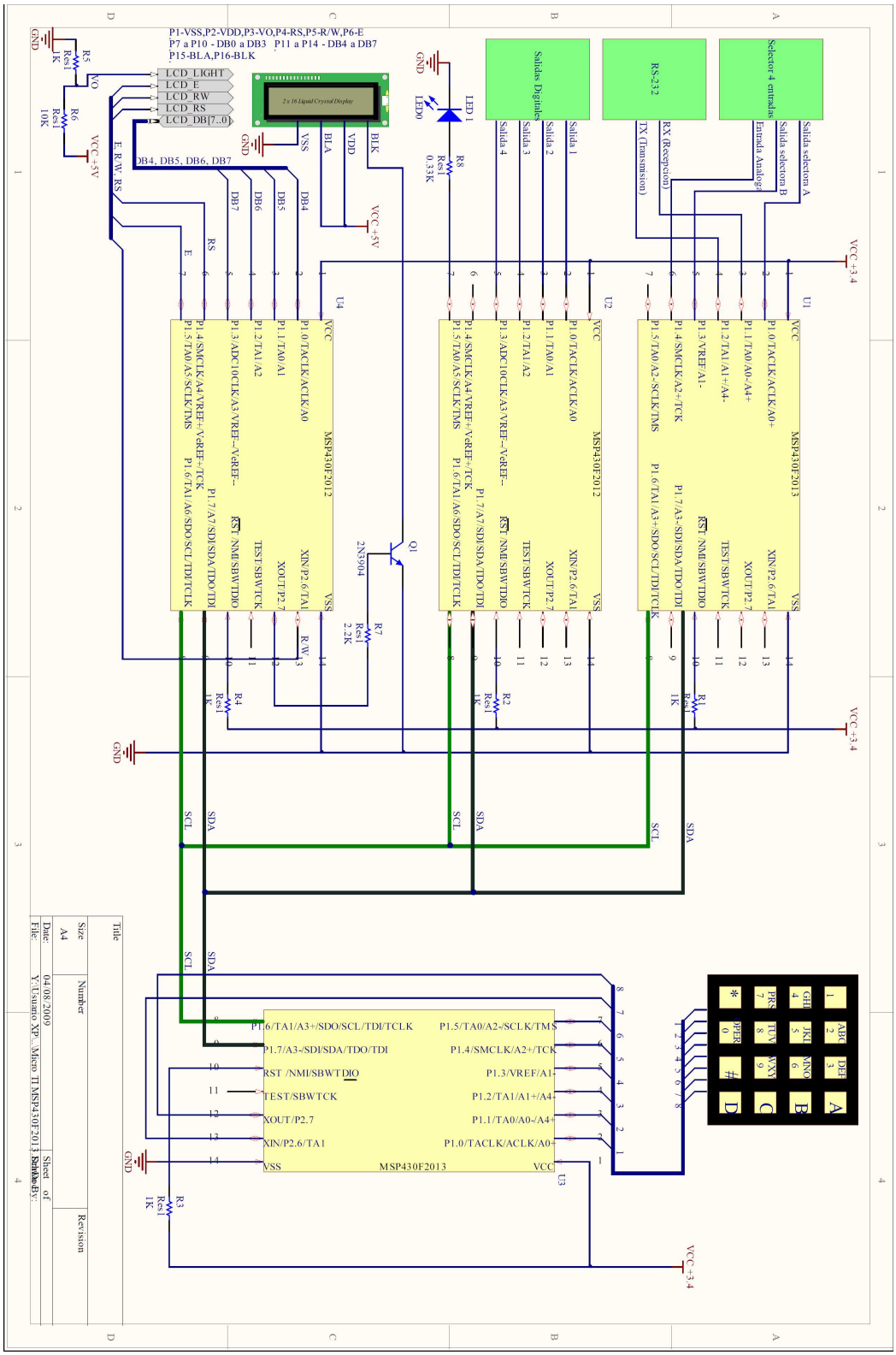


Figura 1.15: Interconexión de periféricos a los microcontroladores. (Elaborado por el Autor)

1.4.5 FUENTE DE ALIMENTACIÓN

Como su nombre lo indica, la figura 1.16 muestra el circuito utilizado para la fuente de alimentación del prototipo, esta fuente entrega los 2 voltajes que se requiere tanto para alimentar a los microcontroladores como para alimentar al resto de elementos (relés, CI's, Display, Teclado, etc.). El sistema se alimenta con el voltaje de la red pública.

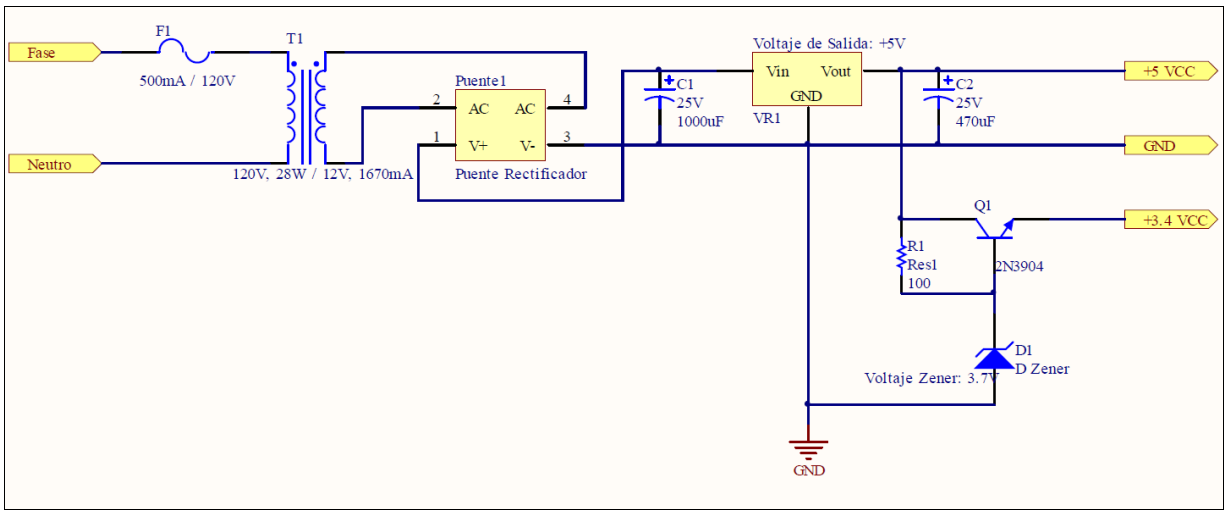


Figura 1.16: Fuente de alimentación del prototipo. (Elaborado por el Autor)

CAPITULO 2

CAPITULO 2. DESARROLLO DE LOS PROGRAMAS

2.1 DESCRIPCIÓN Y CARACTERÍSTICAS GENERALES DE CADA LENGUAJE

Para el desarrollo del presente capitulo se han tomado como base la documentación técnica del diseñador de los lenguajes de programación, en este caso Microsoft, además de otras fuentes de consulta como www.wikipedia.org. Mayor detalle de las fuentes consultadas se encuentra en la bibliografía.

2.1.1 DESCRIPCIÓN Y CARACTERÍSTICAS GENERALES DE VISUAL STUDIO 2005

Visual Studio 2005 se empezó a comercializar a través de Internet a partir del 4 de Octubre de 2005 y llegó a los comercios a finales del mes de Octubre en inglés. En castellano no salió hasta el 4 de Febrero de 2006. Microsoft eliminó el término ".NET", pero eso no indica que se alejara de la plataforma .NET, de la cual se incluyó la versión 2.0.

La actualización más importante que recibieron los lenguajes de programación fue la inclusión de tipos genéricos, similares en muchos aspectos a las plantillas de C#. Con esto se consigue encontrar muchos más errores en el momento de compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

Se incluye un diseñador de implantación, que permite que el diseño de la aplicación sea validado antes de su implantación. También se incluye un entorno para publicación web y pruebas de carga para comprobar el rendimiento de los programas bajo varias condiciones de carga.

Visual Studio 2005 también añade soporte de 64-bit. Aunque el entorno de desarrollo sigue siendo una aplicación de 32 bits Visual C++ 2005 soporta compilación para x86-64 (AMD64 e Intel 64) e IA-64 (Itanium). El SDK incluye compiladores de 64 bits así como versiones de 64 bits de las librerías.

Visual Studio 2005 tiene varias ediciones radicalmente distintas entre sí: Express, Standard, Professional, Tools for Office, y 5 ediciones Visual Studio Team System. Éstas últimas se proporcionaban conjuntamente con suscripciones a MSDN cubriendo los 4 principales roles de la programación: Architects, Software Developers, Testers, y Database Professionals. La funcionalidad combinada de las 4 ediciones Team System se ofrecía como la edición Team Suite.

Tools for the Microsoft Office System está diseñada para extender la funcionalidad a Microsoft Office.

Las ediciones Express se han diseñado para principiantes, aficionados y pequeños negocios, todas disponibles gratuitamente a través de la página de Microsoft se incluye una edición independiente para cada lenguaje: Visual Basic, Visual C++, Visual C#, Visual J# para programación .NET en Windows, y Visual Web Developer para la creación de sitios web ASP.NET. Las ediciones express carecen de algunas herramientas avanzadas de programación así como de opciones de extensibilidad.

Se lanzó el Service Pack 1 para Visual Studio 2005 el 14 de Diciembre de 2006.
La versión interna de Visual Studio 2005 es la 8.0, mientras que el formato del archivo es la 9.0.

La ventaja de utilizar este entorno de desarrollo es que Microsoft provee mucha información y ejemplos que permiten utilizarlo sin mayores dificultades, además permite la reutilización de ensamblados o librerías creadas en versiones anteriores, lo cual resultará de mucha utilidad en la implementación del programa de interfaz para el usuario.

2.1.2 DESCRIPCIÓN Y CARACTERÍSTICAS GENERALES DE VISUAL BASIC 2005

El 13 de febrero de 2002, nació oficialmente Visual Basic .NET, junto con el resto de la familia de Visual Studio. Visual Basic .NET no era un simple "upgrade" del antiguo Visual Basic, era un cambio realmente profundo y radical, convirtiéndolo en uno de los lenguajes más poderosos, con características avanzadas, como verdadera orientación a objetos, multi-threading, y la posibilidad de crear Web Services, entre otras cosas.

La plataforma .NET, base de este nuevo lenguaje, se gesto en Microsoft durante algunos años, y formó parte de una estrategia impulsada por esta empresa para conquistar el mercado del desarrollo y de internet, y de seguir creciendo.

En estos años, la comunidad de Visual Basic ha crecido hasta ser la mayor comunidad de desarrolladores de software del mundo. Durante ese tiempo, una industria entera de vendedores de componentes creció alrededor de este producto. Éste, combinado con la sencilla forma de desarrollar aplicaciones para Windows, fueron la base fundamental de la realización de la visión de Microsoft para la programación basada en Windows.

Los cambios en Visual Basic .NET están proyectados para:

- Simplificar el lenguaje y hacerlo más coherente.
- Agregar nuevas características solicitadas por usuarios.
- Hacer el código más sencillo de leer y mantener.
- Ayudar a los programadores a evitar los errores de programación.
- Crear aplicaciones más sólidas y más sencillas de depurar.

Visual Basic .NET ofrece numerosas características nuevas y mejoradas, como herencia, interfaces y sobrecarga, que lo convierten en un eficaz lenguaje de programación orientado a objetos. Ahora se puede crear aplicaciones multiproceso y escalables utilizando subprocesamiento múltiple explícito. Otra característica nueva de Visual Basic .NET incluye el control estructurado de

excepciones, atributos personalizados y compatibilidad con CLS (Common Language Specification, Especificación de lenguajes comunes).

CLS es un conjunto de reglas que estandariza cosas como tipos de datos y el modo en que se exponen e inter-operan los objetos. Visual Basic .NET agrega varias características que aprovechan las ventajas de CLS. Cualquier lenguaje compatible con CLS puede utilizar las clases, los objetos y los componentes que se crean en Visual Basic .NET, y a su vez Visual Basic puede tener acceso a las clases, los componentes y los objetos desde otros lenguajes de programación compatibles con CLS sin tener en cuenta diferencias específicas del lenguaje como los tipos de datos. Las características de CLS que utilizan los programas de Visual Basic .NET son los ensamblados, espacios de nombres y atributos.

Visual Basic .NET ofrece numerosas características de lenguaje orientado a objetos nuevas o mejoradas como la herencia, la sobrecarga, la palabra clave “Overrides”, interfaces, miembros compartidos y constructores.

También se incluyeron el control estructurado de excepciones, delegados y varios tipos de datos nuevos.

A continuación en la figura 2.1 se indican las mejoras y las características nuevas que se incluyen en esta versión de Visual Basic para Visual Studio 2005.

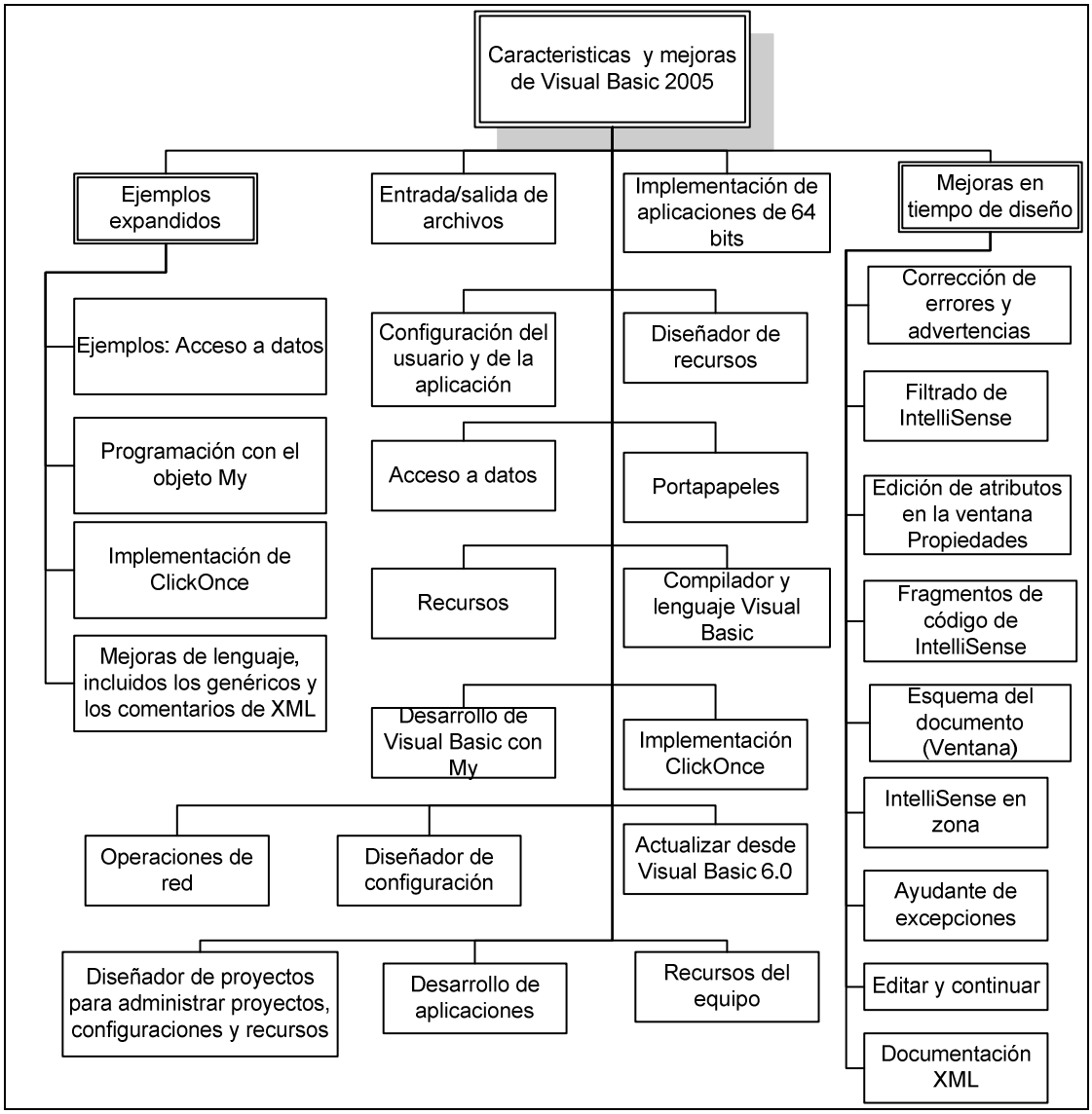


Figura 2.1: Características y mejoras de Visual Basic 2005. (Elaborado por el autor)

En síntesis, el uso de Visual Basic 2005 en el desarrollo de este proyecto está contemplado debido a la facilidad del lenguaje, permite la reutilización de ensamblados o librerías escritas en cualquier otro lenguaje compatible, y principalmente a que anteriormente ya se han desarrollado proyectos utilizando versiones anteriores del mismo lenguaje, lo que facilita el desarrollo de este proyecto.

2.2 DESCRIPCIÓN Y DESARROLLO DEL CÓDIGO FUENTE DEL EJEMPLO DE APLICACIÓN EN EL MICROCONTROLADOR

2.2.1 DESCRIPCION TÉCNICA DE LOS MÓDULOS PERIFERICOS USADOS PARA LA APLICACIÓN PRÁCTICA

2.2.1.1 Set de instrucciones

El set de instrucciones de la familia MSP430 consta de 51 comandos, estos a su vez tienen tres tipos de formatos y pueden usar hasta siete modos de direccionamiento. Cada instrucción puede operar con datos de tamaño de byte o de tamaño de una palabra (2 bytes).

En la tabla 2.1 se muestra un ejemplo de cada formato existente para el uso de las instrucciones.

Operandos duales, Origen-Destino	Ej: ADD R4,R5	R4 + R5 → R5
Operandos individuales, Solo Destino	Ej: CALL R8	PC → (TOS), R8 → PC
Salto Relativo, In/condicional	Ej: JNE	Salte si el bit es igual a 0

Tabla 2.1: Formatos para las instrucciones (usando “Palabras”, 2 bytes).
(Fuente: Texas Instrumens)

La lista completa de instrucciones que soporta el CPU del MSP430 se incluye en el anexo B.6.

2.2.1.1.1 Modos de direccionamiento

Se dispone de varias maneras de acceder a la dirección de memoria donde se ubica cada operando, es así que en cualquier instrucción donde se requiera los operandos de Origen y/o Destino, se los puede acceder de varios modos diferentes; 7 modos para el operando de Origen y 4 modos para el operando de destino. Cualquier combinación de modo de direccionamiento entre el Origen y

el Destino es posible en una instrucción. Todo el espacio de memoria es cubierto con cualquiera de los modos disponibles.

Los diferentes modos de direccionamiento son seleccionados mediante los bits As para el Origen (Addressing mode used for the source - modo de direccionamiento elegido para el origen) y los bits Ad para el Destino (Addressing mode used for the destination - modo de direccionamiento elegido para el destino), estos bits están ubicados en la instrucción que será ejecuta por la CPU del microcontrolador. La Tabla 2.2 muestra el valor de los bits As y Ad y el modo de direccionamiento respectivo, en cambio, en la tabla 2.3 se muestra una lista de los modos de direccionamiento posibles.

As/Ad	Modo de direccionamiento	Sintaxis	Descripción
00/0	Registro	Rn	El contenido de los registros son operandos
01/1	Indexado	X(Rn)	(Rn + X) apunta al operando. X es almacenado en la siguiente palabra
01/1	Simbólico	ADDR	(PC + X) apunta al operando. X es almacenado en la siguiente palabra. El modo indexado X(PC) es usado.
01/1	Absoluto	&ADDR	La palabra a continuación de la instrucción contiene la dirección absoluta. X es almacenada en la siguiente palabra. El modo indexado X(SR) es usado.
10/-	Registro indirecto	@Rn	Rn es usado como puntero al operando
11/-	Auto incremental indirecto	@Rn+	Rn es usado como puntero al operando. Rn es incrementado en 1 para instrucciones byte (.B) y en 2 para instrucciones palabra (.W).
11/-	Inmediato	#N	La palabra siguiente a la instrucción contiene la constante inmediata N. El modo de autoincremento indirecto @PC+ es usado.

Tabla 2.3: Modo de operación de direccionamiento Origen/Destino. (Fuente: Texas Instrumens)

Modo de Direccionamiento	O	D	Sintaxis	Ejemplo	Operación
Registro	X	X	MOV Ro, Rd	MOV R10, R11	R10 → R11
Indexado	X	X	MOV X(Rn), Y(Rm)	MOV 2(R5), 6(R6)	M(2+R5) → M(6+R6)
Simbólico (Relativo al Contador de Programa)	X	X	MOV EDE, TONI		M(EDE) → M(TONI)
Absoluto	X	X	MOV &MEM, &TCDAT		M(MEM) → M(TCDAT)
Indirecto	X		MOV @Rn, Y(Rm)	MOV @R10, Tab(R6)	M(R10) → M(Tab+R6)
Indirecto auto-incremental	X		MOV @Rn+,Rm	MOV @R1+,R11	M(R10) → R11 R10 + 2 → R10
Inmediato	X		MOV #X, TONI	MOV #45,TONI	#45 → M(TONI)

NOTA: O = Origen, D = Destino

Tabla 2.2: Descripción de los modos de direccionamiento. (Fuente: Texas Instrumens)

2.2.1.2 Las Interrupciones

Las prioridades de interrupción son fijas y están definidas de acuerdo al orden de los módulos en la cadena de conexión como se muestra en la Figura 2.2. Mientras más cercano está un módulo de la CPU/NMIRS, más alta es la prioridad. Las prioridades de interrupción definen qué interrupción es ejecutada cuando más de una interrupción se genera simultáneamente.

Hay tres tipos de interrupciones:

- Inicialización del Sistema (Reset)
- NMI No Enmascarable (**N**o **M**askable **I**nterrupt)
- Enmascarable

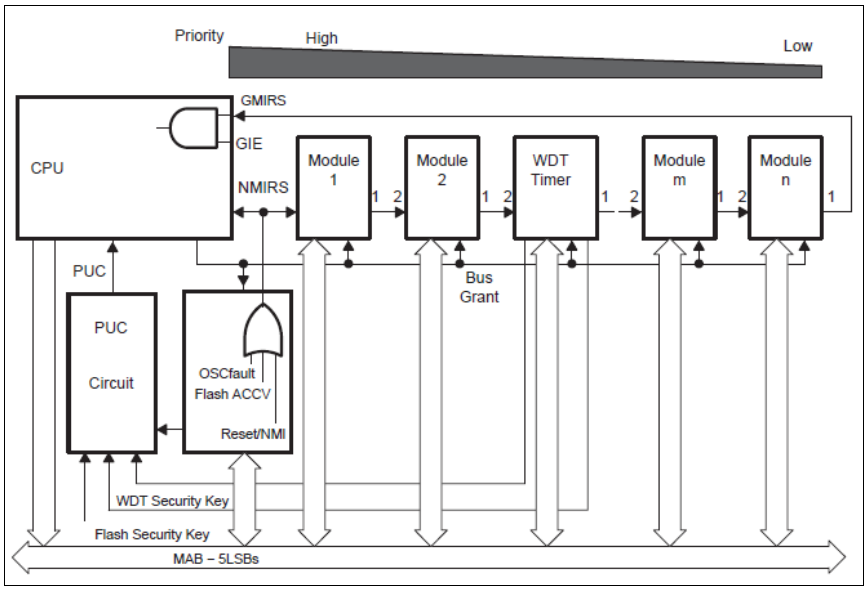


Figura 2.2: Prioridad de interrupción. (Fuente: Texas Instruments)

A continuación, en la figura 2.3 se muestra un resumen de los tipos de interrupciones existentes en el microcontrolador,

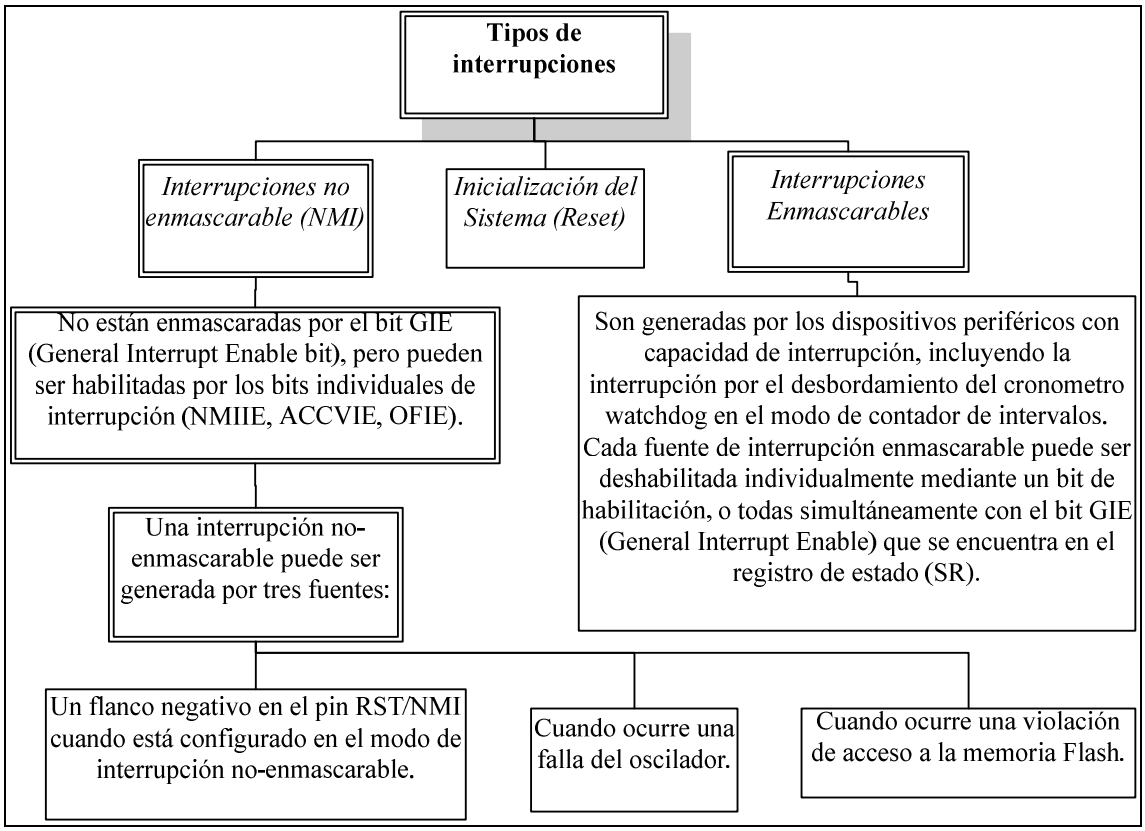


Figura 2.3: Resumen de los tipos de interrupciones existentes en el microcontrolador. (Elaborado por el autor)

Los vectores de interrupción, así como el vector de reset del microcontrolador están ubicados en el rango de direcciones de 0FFFFh a 0FFC0h. Cada vector de interrupción contiene la dirección de 16 bits de la rutina de servicio de interrupción respectiva.

Si el vector de reset (localizado en dirección 0FFFEh) contiene el valor 0FFFFh (por ejemplo, porque la memoria flash no está programada), entonces el microcontrolador entrará en el modo de bajo consumo LPM4 inmediatamente después de ser energizado.

La tabla completa con las fuentes de interrupción (Vectores de interrupción) del MSP430F2013 se incluye en el anexo B.3.

Cuando un dispositivo periférico genera una interrupción y los bits tanto de habilitación de interrupción del dispositivo como el bit GIE están habilitados, la rutina de servicio de interrupción es ejecutada, esto para las interrupciones enmascarables. Para las interrupciones no-enmascarables solamente debe setearse el bit de habilitación respectivo para que la interrupción correspondiente pueda ser llamada y ejecutada. En la figura 2.4 se muestra un resumen de la secuencia requerida para realizar el procesamiento de cada interrupción.

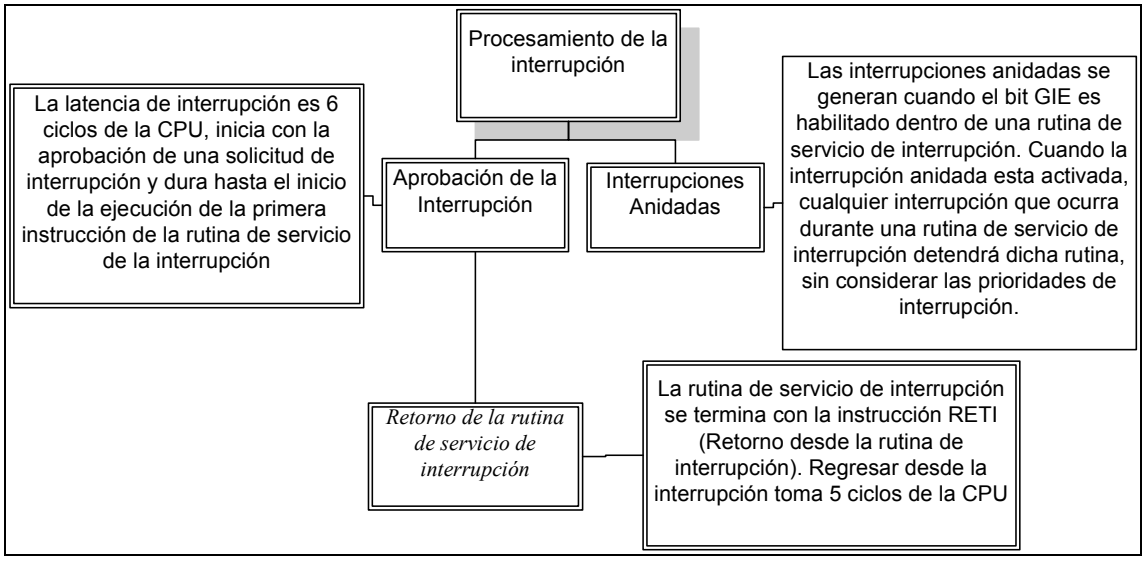


Figura 2.4: Procesamiento de la interrupción. (Elaborado por el autor)

2.2.1.3 Espacio de Direccionamiento

2.2.1.3.1 La memoria Flash / ROM

La dirección de inicio de la memoria Flash/ROM depende de la cantidad de memoria presente y varía dependiendo del dispositivo. La dirección final de la memoria es 0x0FFFF. La memoria Flash puede ser usada tanto para código de programa o para datos. Tablas de "Palabras" (2 bytes) o tablas de 1 byte pueden ser almacenadas y usadas en la memoria Flash/ROM sin la necesidad de copiar estas tablas en la memoria RAM antes de usarlas.

En la Tabla 2.4 se muestran las direcciones de memoria para los dispositivos de la familia MSP430f201X

Memoria	Tamaño	Direcciones
Principal: Vector de interrupción	Flash	32 bytes 0FFFFh – 0FFC0h
Principal: Memoria para código (Memoria Flash Total: 2KB menos 32 bytes usados para vectores de interrupción)	Flash	0FFFFh – 0F800h
Memoria de información	Tamaño	256 Byte 010FFh – 01000h
RAM	Tamaño	128 Byte 027Fh – 0200h
Periféricos	De 16-bits De 8-bits De 8-bits (SFR)	01FFh – 0100h 0FFh – 010h 0Fh – 00h

Nota: SFR (Special Function Register – Registro de función especial)

Tabla 2.4: Organización de la memoria (MSP430F201x). (Fuente: Texas Instruments)

2.2.1.3.2 La memoria RAM

La memoria RAM empieza en la dirección 0x0200. La dirección final de la RAM depende de la cantidad presente en cada dispositivo. La memoria RAM puede ser usada tanto para el código de usuario (programas) como para datos y variables.

Para los dispositivos MSP430F201X la memoria RAM disponible es de 128 bytes, siendo la dirección final 0x027F.

2.2.1.3.3 Dispositivos Periféricos Modulares

Los dispositivos periféricos modulares están asignados dentro del mismo espacio de direccionamiento que el resto de la memoria.

El espacio de direcciones de 0x0100 a 0x01FF es reservado para los dispositivos periféricos modulares de 16 bits. Cada módulo debe ser accedido utilizando instrucciones de palabra. Si, al contrario, se utilizaran instrucciones de byte, solamente las direcciones pares están permitidas (acceso solo al byte bajo), en este caso el byte alto del resultado es siempre 0.

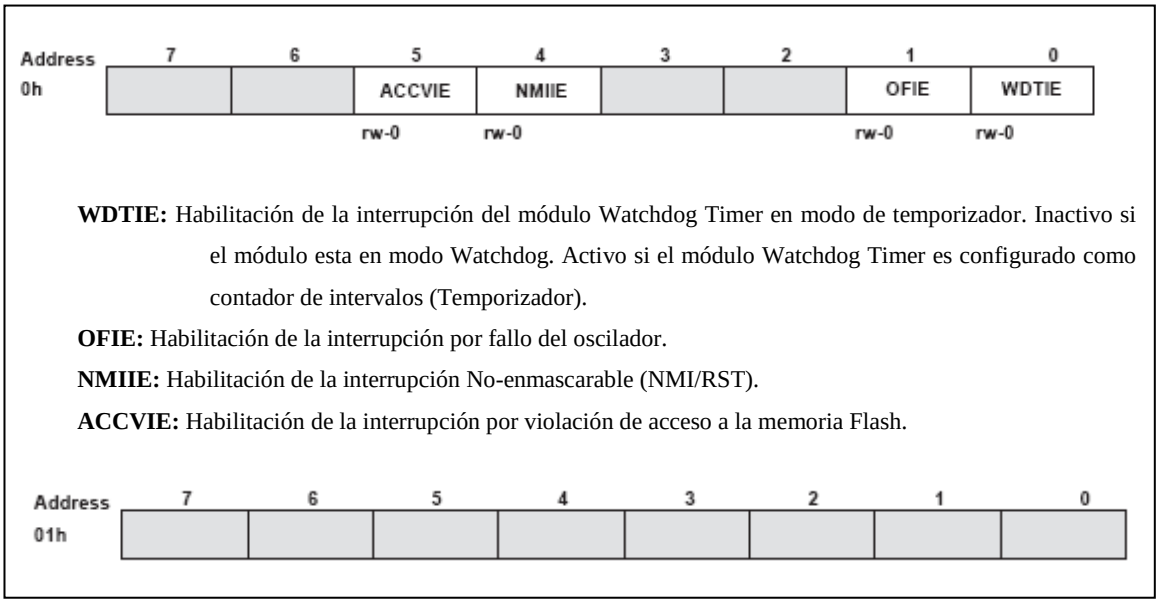
El espacio de direcciones de 0x0010 a 0x00FF está reservado para los dispositivos periféricos modulares de 8 bits. Estos módulos deben ser accedidos con instrucciones de byte. Peticiones de lectura a estos módulos usando instrucciones de palabra resulta en datos inesperados en el byte alto. Si un dato de palabra (16 bits) es escrito en un módulo de 1 byte (8 bits), solamente el byte bajo es escrito en el registro del dispositivo periférico, haciendo caso omiso del byte alto.

2.2.1.3.4 Registros de función especial (SFRs)

La mayoría de los bits de habilitación de las interrupciones correspondientes a cada dispositivo modular están reunidos en la parte más baja del espacio de memoria, esto es de 0x0000 a 0x000F. Los bits en estos registros no asignados a un propósito funcional muestran que ciertos módulos no están físicamente presentes en el dispositivo.

En la Figura 2.5 se encuentran los registros especiales de habilitación de interrupción (0x0000, 0x0001) y en la Figura 2.6 están los bits que representan las banderas de interrupción (0x0002, 0x0003), de los dispositivos

MSP430F201X. En la figura 2.7 esta la leyenda para interpretar la simbología que se encuentra en cada gráfico.



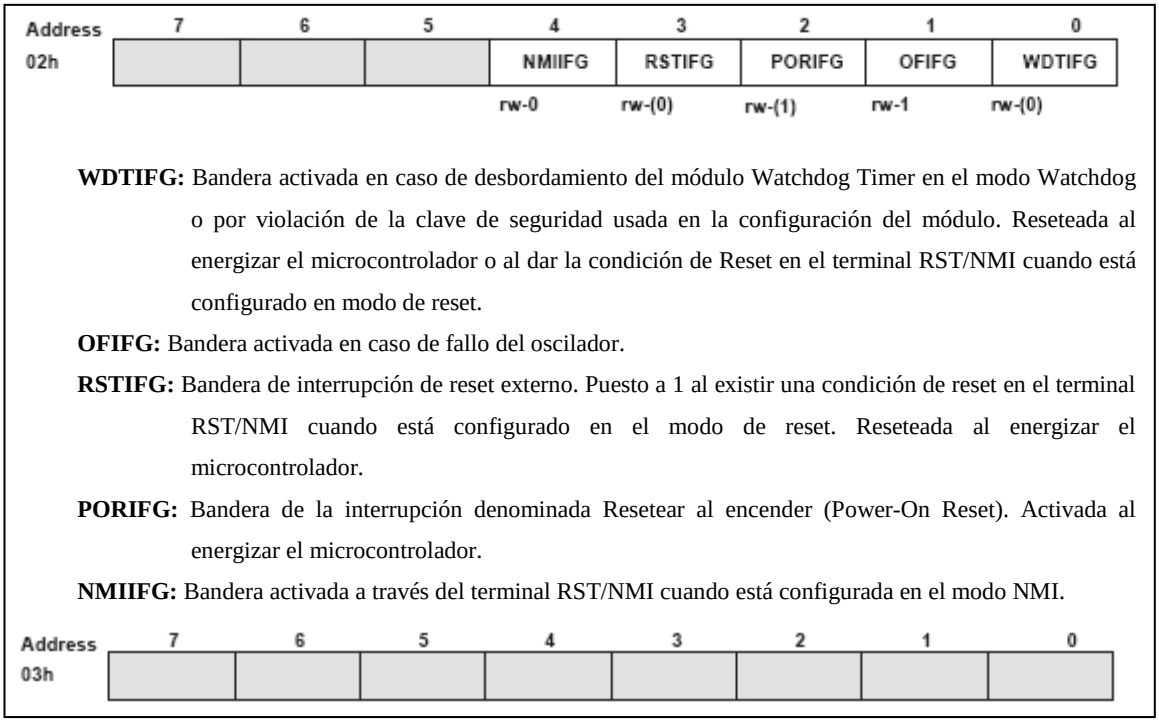
WDTIE: Habilitación de la interrupción del módulo Watchdog Timer en modo de temporizador. Inactivo si el módulo esta en modo Watchdog. Activo si el módulo Watchdog Timer es configurado como contador de intervalos (Temporizador).

OFIE: Habilitación de la interrupción por fallo del oscilador.

NMIIE: Habilitación de la interrupción No-enmascarable (NMI/RST).

ACCVIE: Habilitación de la interrupción por violación de acceso a la memoria Flash.

Figura 2.5: Registros de habilitación de interrupción (direcciones 0h y 01h). (Fuente: Texas Instruments)



WDTIFG: Bandera activada en caso de desbordamiento del módulo Watchdog Timer en el modo Watchdog o por violación de la clave de seguridad usada en la configuración del módulo. Reseteada al energizar el microcontrolador o al dar la condición de Reset en el terminal RST/NMI cuando está configurado en modo de reset.

OFIFG: Bandera activada en caso de fallo del oscilador.

RSTIFG: Bandera de interrupción de reset externo. Puesto a 1 al existir una condición de reset en el terminal RST/NMI cuando está configurado en el modo de reset. Reseteada al energizar el microcontrolador.

PORIFG: Bandera de la interrupción denominada Resetear al encender (Power-On Reset). Activada al energizar el microcontrolador.

NMIIFG: Bandera activada a través del terminal RST/NMI cuando está configurada en el modo NMI.

Figura 2.6: Banderas de interrupción (direcciones 02h y 03h). (Fuente: Texas Instruments)

Leyenda:	
rw:	El bit puede ser leído y escrito.
rw-0,1:	El bit puede ser leído y escrito. Es puesto a 0 o 1 por un evento “PUC”.
rw-(0,1):	El bit puede ser leído y escrito. Es puesto a 0 o 1 por un evento “POR”.
	SFR (Special Function Register) No presente en el dispositivo.

Figura 2.7: Leyenda para la interpretación de la simbología descrita en las Figuras 2.5 y 2.6. (Fuente: Texas Instruments)

2.2.1.3.5 Organización de datos y valores en la memoria

Los bytes pueden ser ubicados en las direcciones pares o impares. Las palabras (2 bytes) pueden ser ubicadas solamente en direcciones pares como se muestra en la Figura 2.7. Cuando se usa instrucciones de palabra, solamente deben usarse direcciones pares. El byte bajo de la palabra esta siempre en una dirección par mientras el byte alto está en la próxima dirección impar.

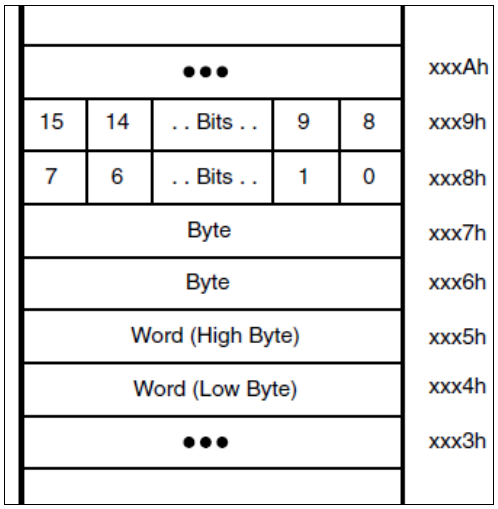


Figura 2.7: Organización de bits, bytes, y palabras en la memoria Flash. (Fuente: Texas Instruments)

2.2.1.4 Configuración y Funcionamiento de los terminales de Entrada/Salida Digitales y sus registros

Los terminales de Entrada/Salida digitales deben ser configurados por el programa en ejecución. El tipo de configuración, el uso de los registros y la operación de estos terminales son discutidos en la figura 2.8, presentando un resumen de todos los registros existentes y su significado.

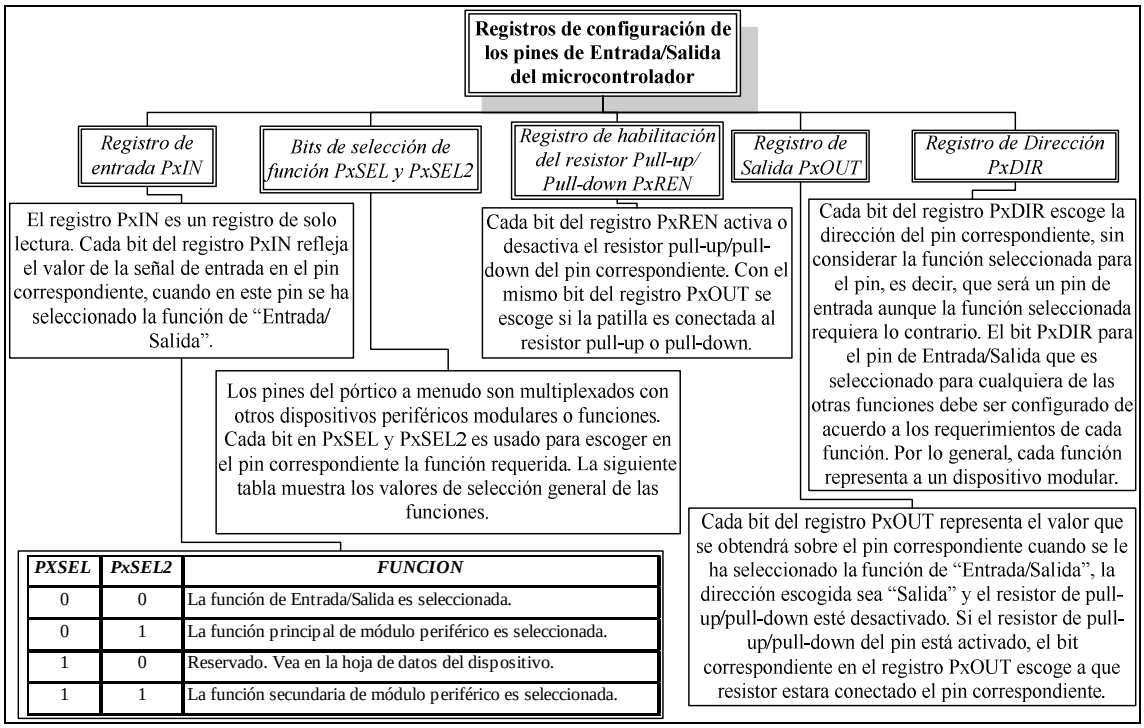


Figura 2.8: Registros de configuración de los terminales de Entrada/Salida del microcontrolador. (Elaborado por el autor).

Cada terminal en los puertos P1 y P2 tienen la capacidad de interrupción, se la puede configurar mediante los registros PxIFG (Port x Interrupt Flag Register), PxIE (Port x Interrupt Enable Register), y PxIES (Port x Interrupt Edge Select Register). Existe un solo vector de interrupción para todos los terminales de un pórtico, y un vector de interrupción por cada pórtico. El registro PxIFG puede ser evaluado para determinar el o los terminales que generaron la interrupción. En la figura 2.9 se muestra un resumen de los registros utilizados para la configuración de las interrupciones en los pórticos 1 y 2.

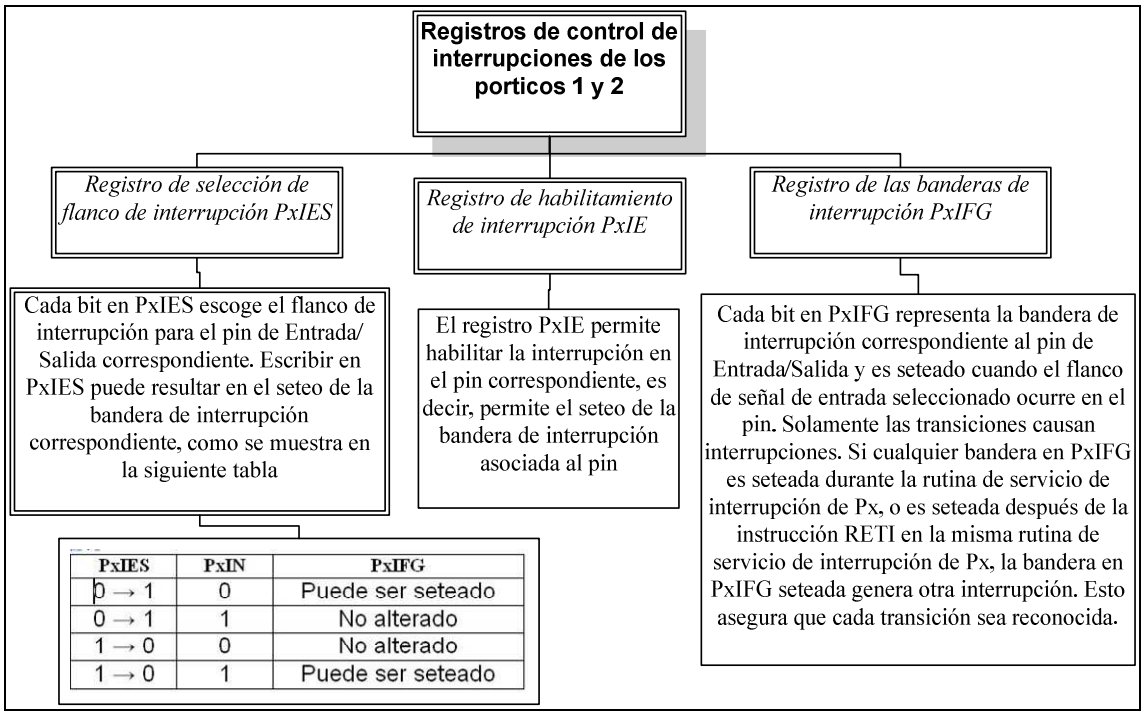


Figura 2.9: Registros utilizados para la configuración de las interrupciones en los pórticos 1 y 2. (Elaborado por el autor).

2.2.1.4.1 Configuración de los terminales sin uso

Los terminales de Entrada/Salida sin usar en el microcontrolador deben ser configurados de la siguiente manera:

- Función seleccionada: “Entrada/Salida”
- Dirección: Salida
- Conexión física del terminal: ninguna

Así se previene una entrada flotante y se reduce el consumo de energía. El valor del bit PxOUT (Port x Output Register) es irrelevante, ya que el terminal está desconectado.

Por otra parte, el resistor integrado de pull-up/pull-down puede ser habilitado configurando el bit PxREN (Port x Pullup/Pulldown Resistor Enable Register) del terminal sin uso para prevenir una entrada flotante.

2.2.1.5 El Timer_A2 (Temporizador/Contador)

El Timer_A2 es un temporizador / contador de 16-bits con dos registros de captura/comparación. Es implementado en los dispositivos de la familia MSP430x20xx. Soporta múltiple captura/comparación, también puede generar salidas del tipo PWM (*Pulse-Width Modulation- Modulación por ancho de pulsos*) e intervalos de temporización.

Además tiene una amplia capacidad de generar interrupciones por diferentes eventos, ya sea por el desbordamiento del contador o por las condiciones de cada uno de los registros de captura/comparación.

Entre las características principales del Timer_A2 tenemos:

- Temporizador/contador Asincrónico de 16-bits con cuatro modos de funcionamiento
- La fuente de reloj es seleccionable y configurable
- Dos registros configurables de captura/comparación
- Salidas configurables con capacidad PWM
- Enclavamiento asíncrono de las entradas y salidas
- Un solo registro de interrupción para una rápida decodificación de todas las interrupciones del Timer_A.

En la Figura 2.10 se muestra el diagrama de bloques del Timer_A.

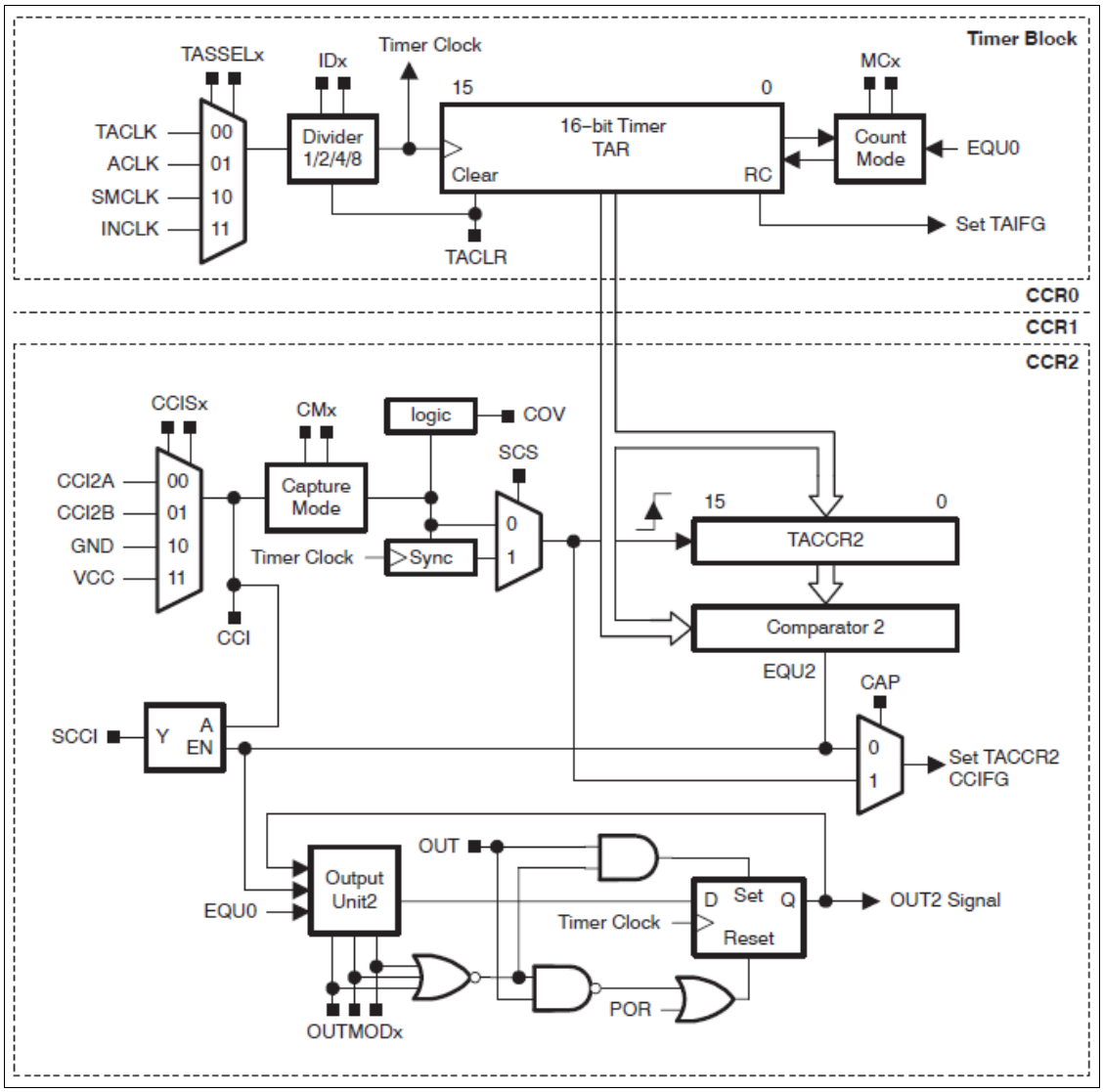


Figura 2.10: Diagrama de bloques del Timer_A. (Fuente: Texas Instruments)

2.2.1.6 La memoria Flash

La memoria Flash de la familia MSP430 es direccionable y programable a nivel de bit, byte, y palabra. Este módulo posee un controlador integrado que controla las operaciones de programado y borrado de la memoria. El controlador entre sus características tiene:

- 4 Registros
- Un generador de sincronización “timming”
- Un generador de voltaje para alimentar las operación de borrado y programado de memoria.

Entre las características del módulo de Memoria Flash del MSP430 tenemos:

- Generación del voltaje interno necesario para las operaciones de programado y borrado de la memoria
- Programación a nivel de bit, byte o palabra
- La operación de la memoria Flash requiere de un bajo consumo de energía
- Borrado de la memoria por segmentos o masivamente

2.2.1.6.1 Segmentación de la Memoria Flash

La memoria Flash del MSP430 esta particionada en segmentos. Aunque es direccionable y programable a nivel de bit, byte, y palabra, el segmento es la porción más pequeña de memoria que puede ser borrada.

La memoria Flash de cada dispositivo está particionada en 2 secciones: la sección de memoria principal (Main Memory) y la sección de memoria de información (Information Memory). No existe diferencia alguna en la operación de ambas secciones, código de programa o datos de usuario pueden ser grabados sin problema. La única diferencia es el tamaño total de memoria flash y el tamaño del segmento de cada uno. En la familia de microcontroladores MSP430F2012 / MSP430F2013 la sección de memoria de información contiene 4 segmentos de 64 bytes cada uno con un total de 256 bytes, mientras que la sección de memoria principal contiene 4 segmentos de 512 bytes cada uno, dando un total de 2K bytes de memoria. Entonces, la memoria Flash total es de 2048 bytes + 256 bytes = 2304 bytes.

En la Figura 2.11 se muestra un ejemplo de segmentación de la memoria flash de un dispositivo que tiene 4K bytes de memoria principal + 256 bytes de memoria de información, es decir, posee 8 segmentos de memoria principal y 4 segmentos de memoria de información.

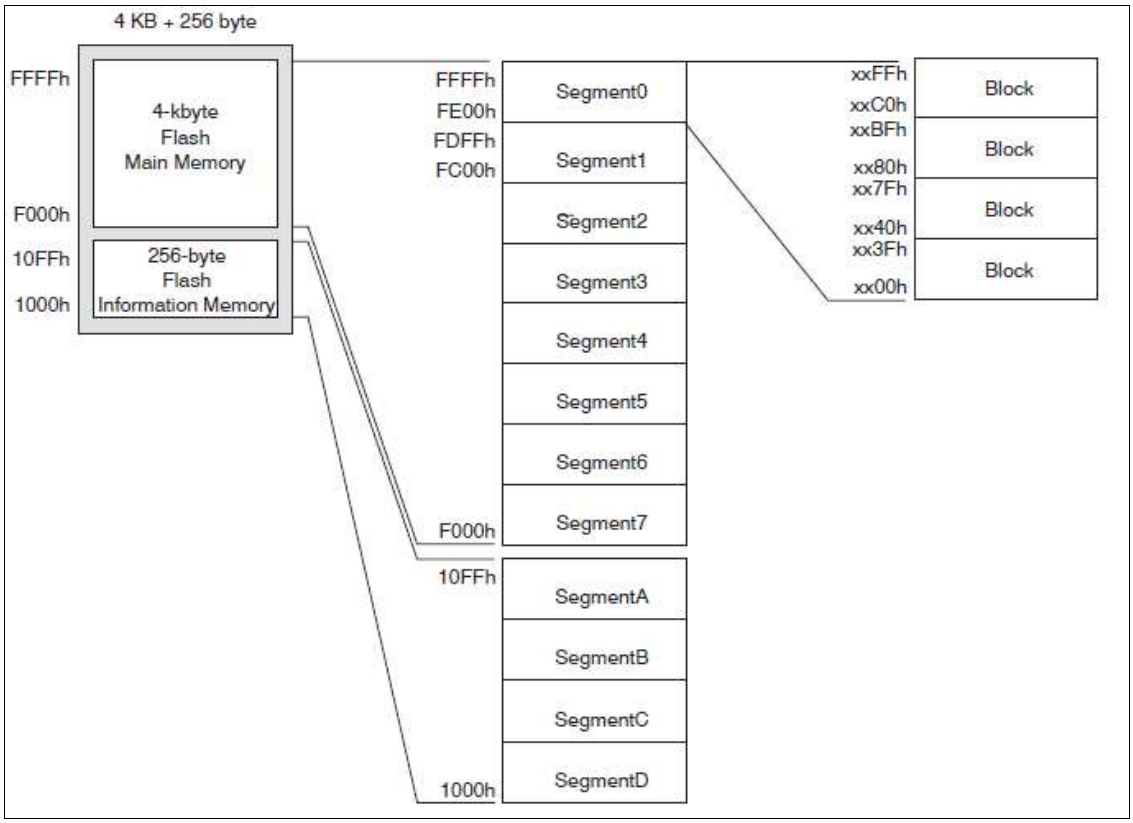


Figura 2.11: Ejemplo de segmentación de la memoria flash de un dispositivo con 4K bytes de memoria principal + 256 bytes de memoria de información. (Fuente: Texas Instruments)

2.2.1.6.2 Modo de operación de la memoria Flash

EL modo de operación por default de la memoria Flash es de solo lectura. En este modo, las operaciones de borrado o escritura no pueden ser iniciadas debido a que el generador de sincronización y el generador de voltaje para el módulo de la memoria flash están apagados, de esta manera opera idénticamente igual a una memoria ROM.

La memoria Flash del MSP430 puede ser programada por el mismo CPU y no necesita de voltajes externos adicionales. Para seleccionar los modos de escritura o borrado se utilizan los bits BLKWRT (Block Write mode), WRT (Write), MERAS (Mass erase), y ERASE. Estos modos pueden ser:

- Escritura por byte o palabra
- Escritura por bloque

- Borrado de segmento
- Borrado masivo (solo los segmentos de memoria principal)
- Borrado Total (todos los segmentos de memoria)

Mientras la memoria Flash es programada o borrada se prohíbe la lectura o escritura de la misma. Si se requiere que el CPU trabaje mientras se realiza una de estas operaciones, se debe ubicar el código en la memoria RAM. Cualquier actualización de la memoria Flash puede ser iniciada desde la memoria Flash o desde la RAM.

2.2.1.6.2.1 Programación (Escritura) de la Memoria Flash

Los modos de escritura son seleccionados mediante los bits BLKWRT y WRT, estos modos están descritos en la Tabla 2.5

BLKWRT	WRT	Modo de escritura
0	1	Escritura de byte o palabra
1	1	Escritura en Bloque

Tabla 2.5: Modos de escritura de la memoria Flash. (Fuente: Texas Instruments)

Ambos modos de escritura usa una secuencia de peticiones individuales de escritura, pero si se usa el modo de escritura en bloque es por lo menos 2 veces más rápido escribir un bloque completo que escribirlo en modo de byte o palabra, debido a que el generador de voltaje permanece encendido todo el tiempo que se escribe el bloque y en cambio, en el modo byte o palabra el generador continuamente esta encendiéndose y apagándose. Cualquier instrucción que modifique la memoria de destino puede usarse para modificar la memoria Flash.

Una palabra (byte bajo + byte alto) no debe ser escrita más de 2 veces entre ciclos de borrado debido a que esta acción puede provocar daños en la memoria Flash.

El bit BUSY es puesto a 1 mientras la operación de escritura esta activa y es puesto a 0 cuando la operación ha finalizado. Si la operación de escritura es iniciada desde la RAM, el CPU no debe acceder a la memoria Flash mientras el bit BUSY esta activado, de lo contrario, una violación de acceso se producirá, el bit ACCVIFG (Access Violation Interrupt Flag) es activado y el resultado de la escritura es indeterminado. La operación de escritura puede ser iniciada también desde la memoria Flash, en este caso, la sincronización está controlada por el módulo de la memoria Flash y la CPU es detenida hasta que la petición de escritura finalice. Una vez que la petición ha finalizado, la CPU continúa la ejecución del programa con la instrucción siguiente a la petición de escritura.

2.2.1.6.3 Controlador de la memoria Flash

Los registros FCTLx (Flash Memory Control Register) son registros de 16 bits de lectura/escritura protegidos por contraseña, cualquier acceso sobre estos debe usar instrucción de palabra y la petición de escritura debe incluir la contraseña 0A5h en el byte alto de la palabra a setear. Cualquier intento de escritura en cualquiera de los registros FCTLx con cualquier valor diferente al valor 0A5h, es una violación a la clave de seguridad, lo que activa la bandera KEYV (Flash Security Key Violation) y dispara el evento PUC de reseteo del sistema.

Cualquier intento de lectura en cualquiera de los registros FCTLx dará como resultado el valor 096h en el byte alto.

Cualquier intento de escritura en el registro FCTL1 durante una operación de borrado o escritura provoca una violación de acceso y activa el bit ACCVIFG. Escribir sobre FCTL1 está permitido en el modo de escritura en bloque cuando el bit WAIT esta puesto a 1, pero cualquier intento de escritura sobre FCTL1 en el mismo modo de escritura en bloque cuando el bit WAIT esta puesto a 0 produce una violación de acceso y activa el bit ACCVIFG. De igual manera, cualquier intento de escritura en FCTL2 cuando el bit BUSY esta puesto a 1 es

una violación de acceso. Cualquiera de los registros FCTLx pueden ser leídos mientras el bit BUSY esta puesto a 1, esto no provoca una violación de acceso.

2.2.1.6.4 Interrupciones del controlador de la memoria Flash

El controlador de la memoria Flash dispone de dos Fuentes de interrupción, KEYV y ACCVIFG.

La bandera ACCVIFG es activada cuando una violación de acceso ocurre. Cuando al bit ACCVIE (Flash Memory Access Violation Interrupt Enable) se lo habilita consecutivamente después de una operación de borrado o escritura de la Flash, la activación de la bandera ACCVIFG genera una petición de interrupción. La bandera ACCVIFG alimenta el vector de interrupción no enmascarable, es por esto que no es necesario que el bit de interrupción general este activado para que se ejecute la interrupción. También puede ser verificado por el programa del usuario para chequear si existe o no una violación de acceso a la memoria Flash, pero siempre debe ser puesta a 0 por el programa en ejecución.

La bandera de violación a la clave de seguridad KEYV es activada cuando cualquiera de los registros del controlador de la memoria Flash son escritos con una contraseña incorrecta, cuando esto ocurre, el evento PUC es generado reiniciando de inmediato el microcontrolador.

2.2.1.7 Interfaz de comunicación serial universal, trabajando en modo I2C

La Interfaz de comunicación serial universal (**U**niversal **S**erial **C**omunication **I**nterface) es un módulo que soporta múltiples modos de comunicación serial. Diversos módulos USCI pueden soportar varios modos de comunicación serial. Cada módulo USCI es denominado con una letra diferente, así, el módulo USCI_A es diferente al módulo USCI_B. Si en un microcontrolador se dispone de más de un módulo USCI del mismo tipo, se diferenciaran entre sí por un número, por ejemplo si un dispositivo tiene dos módulos USCI_A, el primer

módulo será el USCI_A0 y el segundo será el USCI_A1. Los modos de comunicación soportados por cada tipo de módulo USCI son:

Módulo USCI_Ax:

- Modo UART
- Tipo pulso para comunicaciones IrDA
- Detección automática de la velocidad de transmisión en comunicaciones LIN
- Modo SPI

Módulo USCI_Bx:

- Modo I2C
- Modo SPI

2.2.1.7.1 Introducción al módulo USCI: Modo I2C

En el modo I2C (**I**nter-**I**ntegrated **C**ircuit), el módulo USCI provee una interfaz entre el microcontrolador MSP430 y los dispositivos compatibles con I2C conectados mediante el método de bus serial I2C de 2 líneas. Componentes externos enlazados al bus I2C transmiten o reciben datos seriales desde y hacia el módulo USCI a través de la interfaz I2C de 2 líneas.

Las características del modo I2C incluyen:

- Cumple con las especificaciones descritas en la Philips Semiconductors I2C specification v2.1.
- Soporta los modos de direccionamiento de 7 bits y de 10 bits.
- Llamado general.
- Señales de START, RESTART, STOP.
- Modo Multi-Master para la transmisión y recepción de datos.
- Modo Esclavo para la transmisión y recepción de datos.
- Soporta el modo estándar que provee velocidades de hasta 100kbps o el modo “fast” que soporta hasta 400kbps.
- Frecuencia programable en el Modo Master.

- Diseñado para un bajo consumo de energía.
- En modo esclavo, se detecta la señal de START que permite salir de cualquier modo de bajo consumo de energía.
- Permite la operación del esclavo en modo LPM4.

El modo I2C soporta cualquier dispositivo Master o Esclavo I2C compatible. En la Figura 2.12 se muestra un ejemplo de un bus I2C, cada dispositivo es reconocido por una dirección única y puede operar tanto como trasmisor y como receptor. Un dispositivo enlazado al bus I2C puede ser considerado como Master o Esclavo cuando realiza una transferencia de datos. El dispositivo Master inicia la transferencia de datos y genera la señal de reloj SCL (Serial Clock). Cualquier dispositivo direccionado por el Master puede ser considerado un Esclavo.

Los datos en el bus I2C son comunicados usando la línea SDA (Serial Data) con la señal de reloj generada en SCL. Estas líneas son bidireccionales y deben estar conectadas a una referencia positiva mediante una resistencia de pull-up. Cabe anotar que los terminales SDA y SCL de los microcontroladores MSP430 no deben ser conectados a voltajes de referencia mayores al voltaje de alimentación del dispositivo.

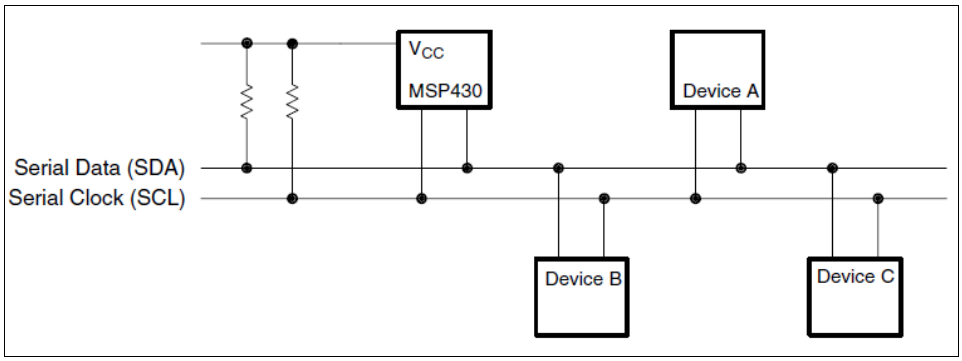


Figura 2.12: Diagrama de conexión del bus I2C. (Fuente: Texas Instruments)

2.2.1.7.2 Datos seriales en el bus I2C

El dispositivo Master genera un pulso de reloj para cada bit que transfiere. El modo I2C opera con bytes (8 bits) y transfiere el bit más significativo primero,

como se muestra en la figura 2.13. El primer byte que sigue a la condición START consiste en la dirección de 7 bits del esclavo y el bit de Lectura/Escritura (R/W). Cuando el bit R/W es cero, el Master transmite datos al esclavo, en cambio, cuando R/W es uno, el Master recibe datos desde el Esclavo. El bit ACK (Acknowledgement) es enviado desde el receptor después de cada byte en la novena señal de reloj en la línea SCL.

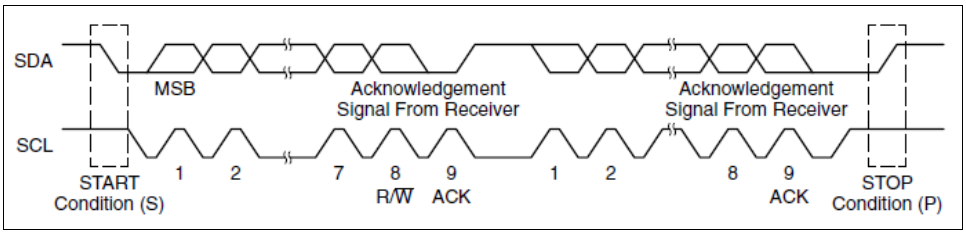


Figura 2.13: Transferencia de datos en el módulo I2C. (Fuente: Texas Instruments)

Las condiciones de START y STOP son generadas por el dispositivo Master, la condición START es una transición de alto a bajo en la línea SDA mientras la línea SCL está en nivel alto y la condición de STOP es una transición de bajo a alto en la línea SDA mientras la línea SCL está en nivel alto. El bit de bus ocupado, UCBBUSY (Bus Busy), se activa después de la condición START y se lo borra luego de la condición STOP. Los datos en la línea SDA deben estar estables durante el periodo en que la señal de reloj SCL está en nivel alto como se muestra en la Figura 2.14. Los estados de nivel alto y bajo solo pueden ser cambiados cuando la línea SCL se encuentra en nivel bajo, de otro modo las condiciones de START o STOP pueden generarse.

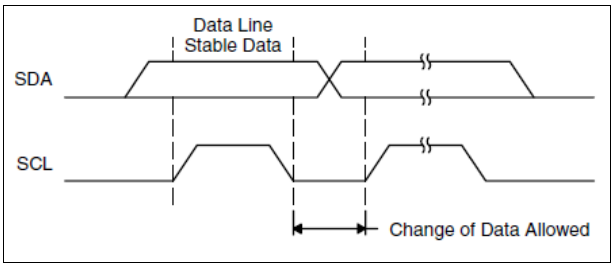


Figura 2.14: Transferencia de bits en el bus I2C. (Fuente: Texas Instruments)

En el modo I2C el módulo USCI puede operar de 4 maneras, en la figura 2.15 se muestra un resumen de los modos de operación posibles en el módulo I2C.

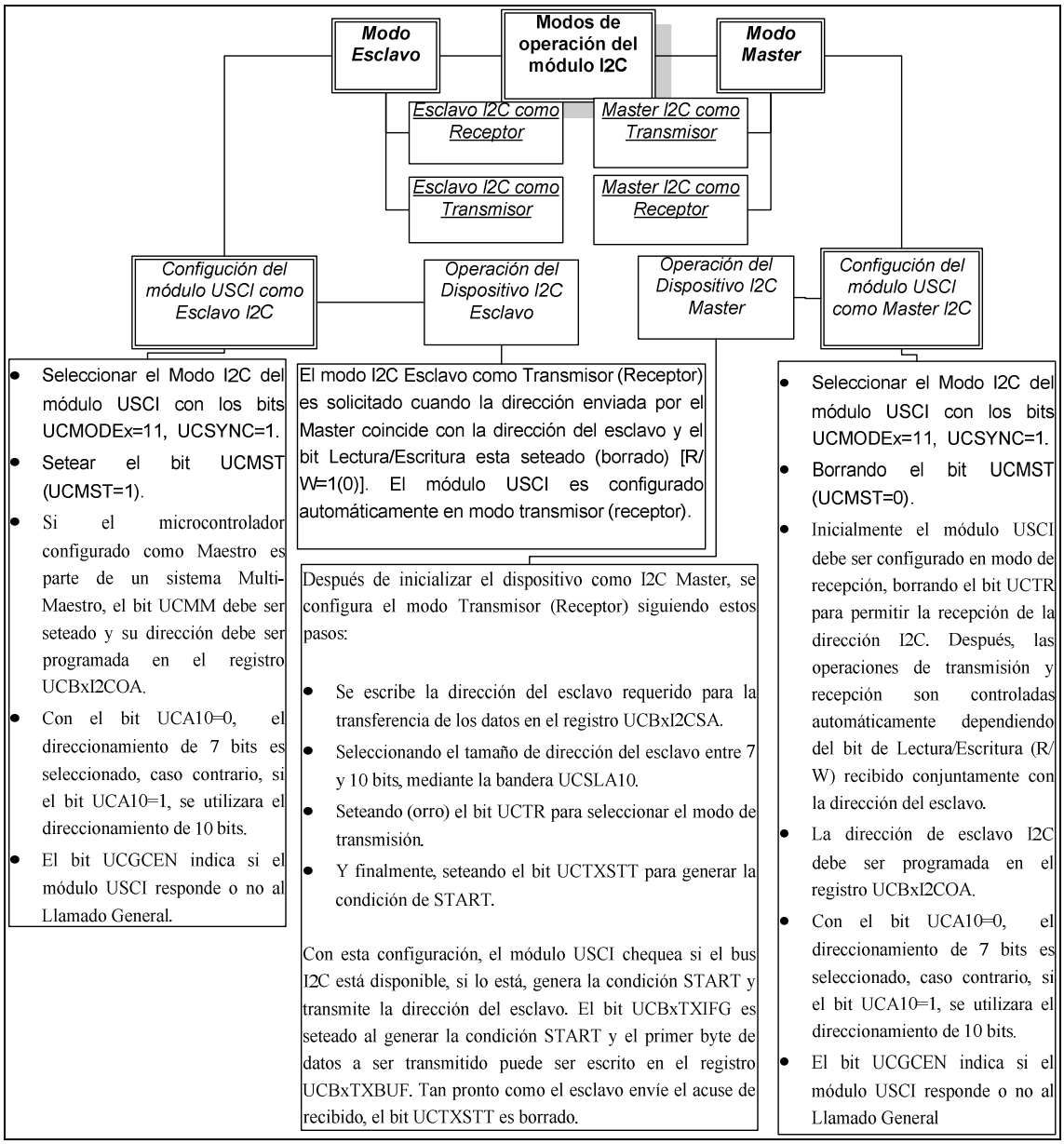


Figura 2.15: Modos de operación del módulo I2C. (Elaborado por el autor)

Si dos o más dispositivos Master transmiten la condición de START simultáneamente sobre el bus I2C, el procedimiento de arbitraje es invocado. En la Figura 2.16 se muestra el procedimiento de arbitraje entre 2 dispositivos Master. Este procedimiento usa el dato que se encuentra presente en cada uno de los buffers de salida de los dispositivos contendores. El primer Master que necesitara setear la línea SDA en nivel alto será sobrescrito por aquel que

generara un nivel bajo en la misma línea. El procedimiento de arbitraje da la ventaja al dispositivo que transmita los datos binarios de menor valor. El dispositivo Master que pierde la contienda pasa automáticamente a modo de I2C Esclavo como Receptor y activa la bandera de “Perdida del arbitraje” UCALIFG (Arbitration Lost Interrupt Flag). En el caso de que 2 o más dispositivos envíen los mismos bytes iniciales, el arbitraje continua con los subsecuentes bytes transmitidos.

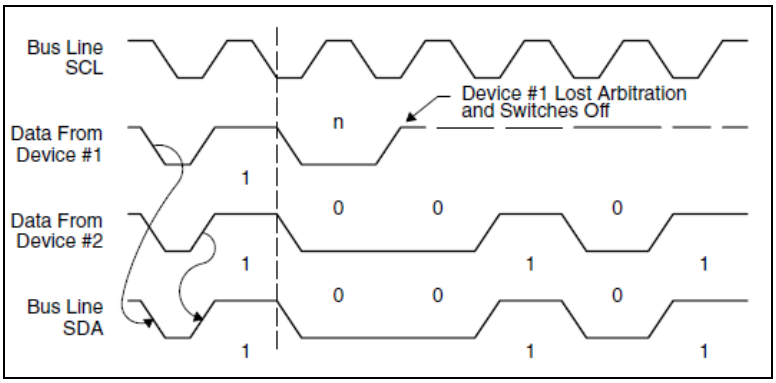


Figura 2.16: Procedimiento de arbitraje entre dos dispositivos Master. (Fuente: Texas Instruments)

Si el procedimiento de arbitraje esta en progreso y uno de los dispositivos Master genera la condición de START repetitivo o de STOP en la línea SDA, los dispositivos que están involucrados en el arbitraje deben enviar la condición de START repetitivo o de STOP en la misma posición. El arbitraje no está permitido entre:

- Una condición de START repetitivo y un bit de datos.
- Una condición de STOP y un bit de datos.
- Una condición de START repetitivo y una condición de STOP.

2.2.1.7.3 Interrupciones del módulo USCI en modo I2C

El modo USCI en modo I2C tiene 2 vectores de interrupción. El primero está asociado con las banderas de interrupción de transmisión y recepción y el segundo vector de interrupción está asociado con las banderas de interrupción de los cuatro tipos estados. Cada bandera de interrupción tiene su propio bit de habilitamiento de la interrupción. Cuando una interrupción es habilitada y el bit

GIE esta activado, la bandera de interrupción generara un pedido de interrupción. Las transferencias que realice el controlador DMA (Direct Memory Access) son controladas por las banderas UCBxTXIFG (Transmit Interrupt Flag) y UCBxRXIFG (Receive Interrupt Flag) en los dispositivos que poseen el controlador DMA.

2.2.2 DESCRIPCION Y DESARROLLO DE LA APLICACIÓN PRÁCTICA Y LOS MÓDULOS NECESARIOS PARA SU FUNCIONAMIENTO

El desarrollo de la aplicación busca mostrar todas las capacidades que el prototipo posee, pero para que la aplicación pueda funcionar se necesita que el prototipo tenga una serie de programas y funciones que permiten al prototipo comunicarse entre sus componentes y evitan que el usuario tenga que realizar las mismas configuraciones necesarias para el funcionamiento del mismo.

En la primera parte se explicará y detallará el funcionamiento de cada uno de los módulos requeridos por el prototipo para su funcionamiento, los protocolos creados para las comunicaciones internas mediante I2C y la comunicación al PC mediante RS-232 estará incluida en el anexo E.

La segunda parte mostrará el ejemplo que se utilizó mientras se desarrollaba el prototipo, con el cual se pudo comprobar el correcto funcionamiento de cada uno de los módulos.

Debido a que el código fuente de cada módulo es demasiado extenso, no se encuentran incluidos en los anexos, se encontraran ubicados en el CD-ROM o soporte magnético adjunto al material impreso de este proyecto.

2.2.2.1 Módulos del prototipo: explicación de los programas

El prototipo consta de 4 módulos y cada uno de ellos realiza una función bien definida. Debido al número de entradas/salidas que posee cada integrado y al

número de periféricos que se necesitan integrar al prototipo, se dividió el funcionamiento de cada dispositivo.

Para la comunicación entre cada uno de ellos se usará el protocolo I2C, que es un protocolo de comunicación serial que utiliza dos líneas, la primera envía desde un dispositivo Master la señal de reloj y la segunda línea es usada para transmitir/recibir los bytes correspondientes.

Se escoge implementar este protocolo debido a que los microcontroladores usados en el prototipo tienen el hardware necesario para llevar a cabo la comunicación I2C.

Los módulos implementados son:

1. Comunicación RS-232, Conversión Análogo/Digital y multiplexacion de las entradas analógicas (Master)
2. Manejo del Display LCD (Esclavo Display)
3. Decodificación del Teclado (Esclavo Teclado)
4. Salidas Digitales y almacenamiento de datos (Esclavo Salidas)

Las direcciones I2C esclavo utilizadas por el dispositivo Master para la comunicación son:

- Dirección I2C del esclavo Teclado: 0x011h, en decimal: 17
- Dirección I2C del esclavo Display: 0x012h, en decimal: 18
- Dirección I2C del esclavo Salidas 0x013h, en decimal: 19

En la figura 2.17 se resumen los módulos que integran el prototipo de acuerdo a la comunicación I2C que utiliza.

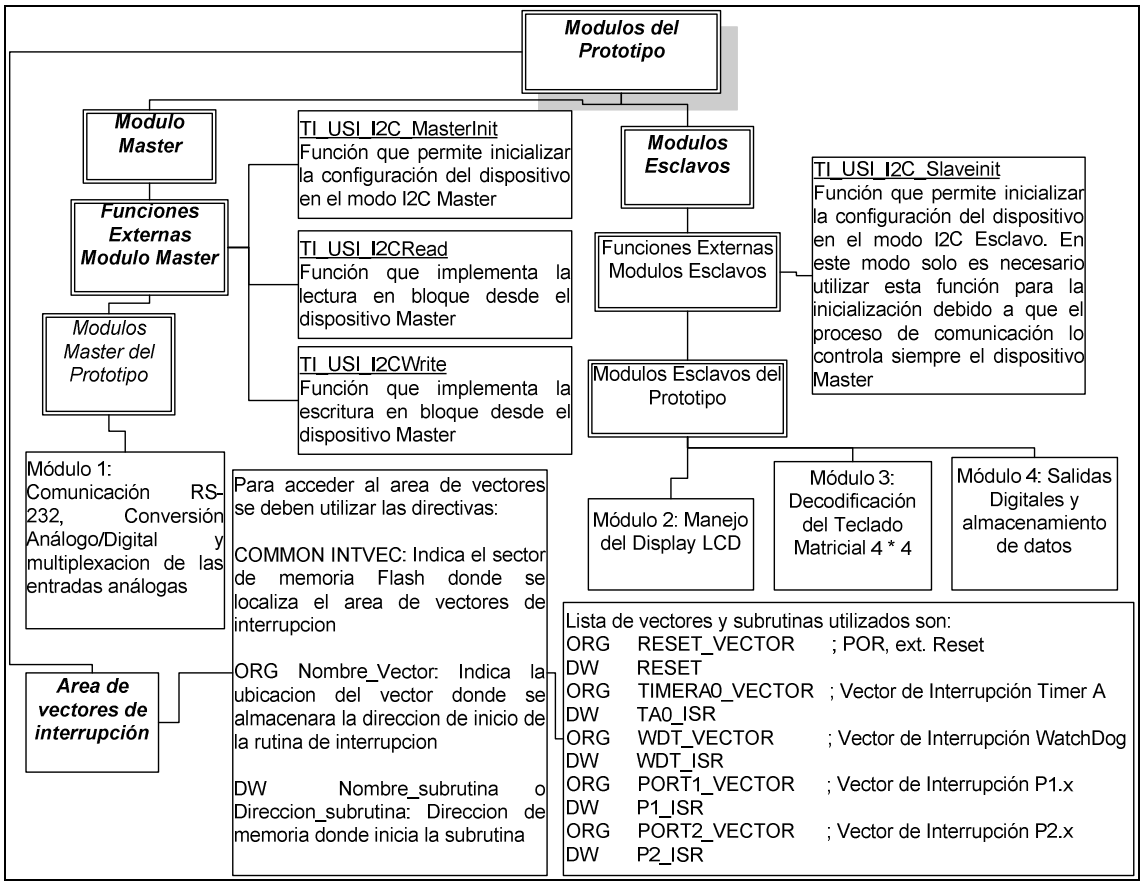


Figura 2.17: Resumen de los módulos que conforman el prototipo. (Elaborado por el autor)

2.2.2.1.1 Módulo 1: Comunicación RS-232, Conversión Análogo/Digital y multiplexación de las entradas análogas

La idea principal de este módulo es la de proporcionar al prototipo la capacidad de comunicarse con el PC mediante la interfaz RS-232, así mismo, permite el uso de 4 entradas análogas multiplexadas y seleccionadas secuencialmente por el microcontrolador. Debido a que este módulo tendrá que interactuar directamente con el PC se lo seleccionó como el dispositivo Master para la comunicación I2C. La librería utilizada tanto para el dispositivo Master como para los dispositivos Esclavos fue encontrada en la página web del fabricante. Cabe señalar que la librería proporcionada por el fabricante para los dispositivos Esclavos fue modificada por que la misma contenía un error en el código fuente. En la figura 2.18 se resumen las partes más importantes del código fuente diseñado para este módulo.

Figura 2.18: Resumen del modulo 1: Comunicación RS-232, Conversión Análogo/Digital y multiplexación de las entradas análogas. (Elaborado por el autor)

2.2.2.1.2 Módulo 2: Manejo del Display LCD

EL manejo del display se lo realiza mediante 4 líneas de datos y 3 líneas de control, de esta manera podemos trabajar con el mismo microcontrolador sin necesidad de reemplazarlo con uno de mayores prestaciones. En este modo de trabajo, el dato de 8 bits se envía en 2 secuencias de 4 bits.

EL módulo permite inicializar el display, controlar el apagado o encendido del Led de Backlight del display para permitir el ahorro de energía además de que está diseñado para comunicarse con otros dispositivos mediante el uso del protocolo I2C en modo Esclavo.

Los comandos que dispone mediante el protocolo I2C sirven para ejecutar la gran mayoría de comandos propios del display y también otros propios del funcionamiento interno del prototipo.

Debido a que está configurado en modo I2C esclavo, cualquier dispositivo externo que desee controlarlo simplemente deberá conocer su dirección y los comandos respectivos.

Para el funcionamiento del prototipo, este módulo incluye en su memoria mensajes que son mostradas al usuario dependiendo de los eventos que ocurran en el prototipo.

En la figura 2.19 se resumen las partes más importantes del código fuente diseñado para este módulo.

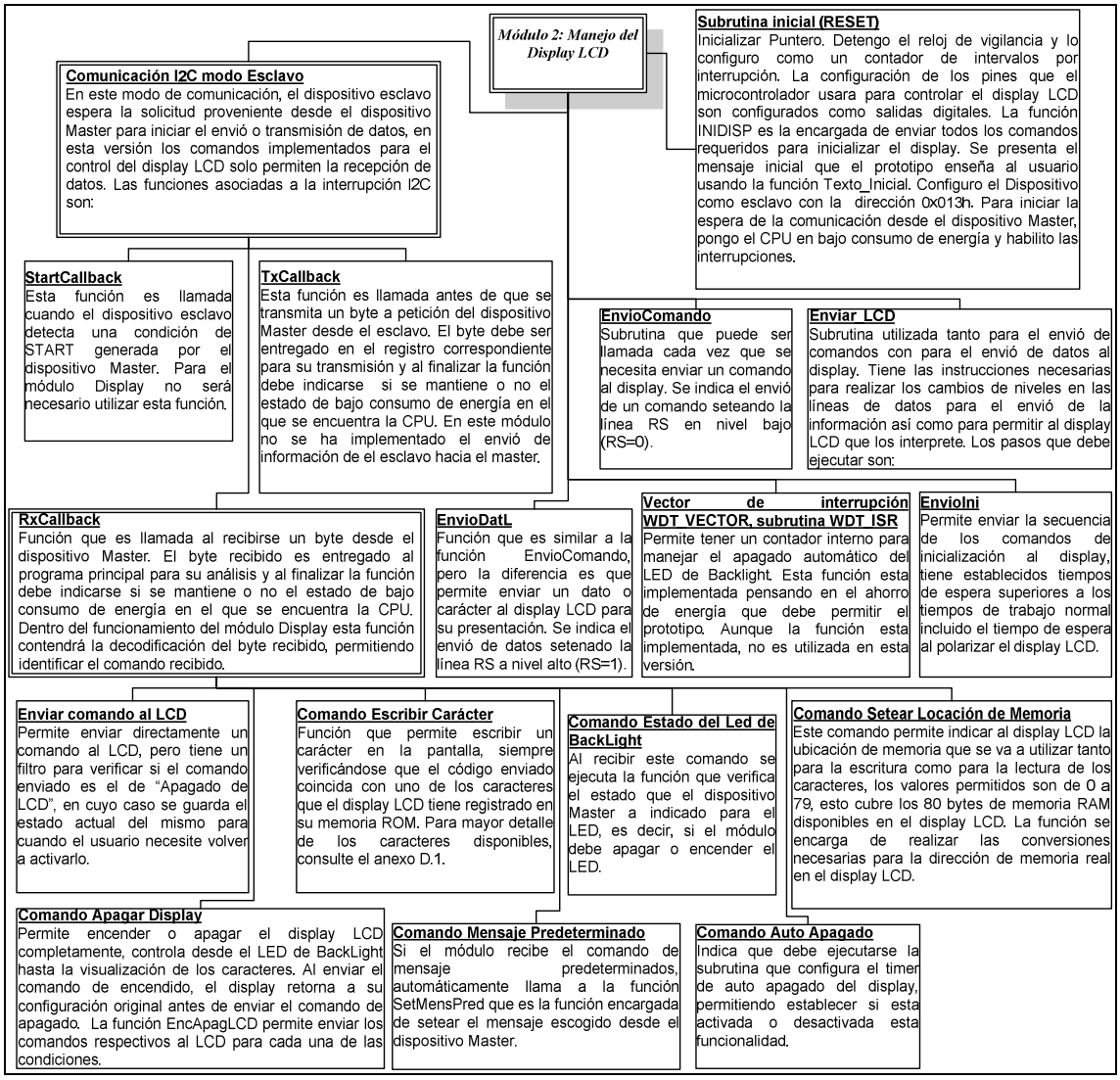


Figura 2.19: Resumen del modulo 1: Módulo 2: Manejo del Display LCD. (Elaborado por el autor)

*2.2.2.1.3 Módulo 3: Decodificación del Teclado Matricial 4 * 4*

Este módulo se encarga de la decodificación del teclado, detecta la presión de 1 o más teclas simultáneamente y queda en espera hasta que el dispositivo Master pida la(s) tecla(s) presionada(s). El escaneo se realiza por barrido primero de las filas y después de las columnas, permitiendo detectar varias teclas presionadas simultáneamente. Cada tecla presionada es codificada con valores ASCII que representan los números y letras graficados en el teclado, a excepción de la tecla asterisco (*) y numeral (#), que se representan con las letras E y F respectivamente.

El módulo está configurado en modo I2C Esclavo, de esta manera se puede acceder a este módulo si se conoce su dirección I2C esclavo. Para el envió de las teclas presionadas, primero envía el numero de teclas detectadas y a continuación envía una tras otra las teclas decodificadas.

Al igual que el anterior módulo configurado en modo I2C esclavo, solo existe una función externa para el uso de la librería de comunicación I2C, esta es TI_USI_I2C_SlaveInit que configurara el dispositivo con la dirección I2C esclavo 0x011h. En la figura 2.20 se muestra un resumen de las principales características del programa que compone el módulo.

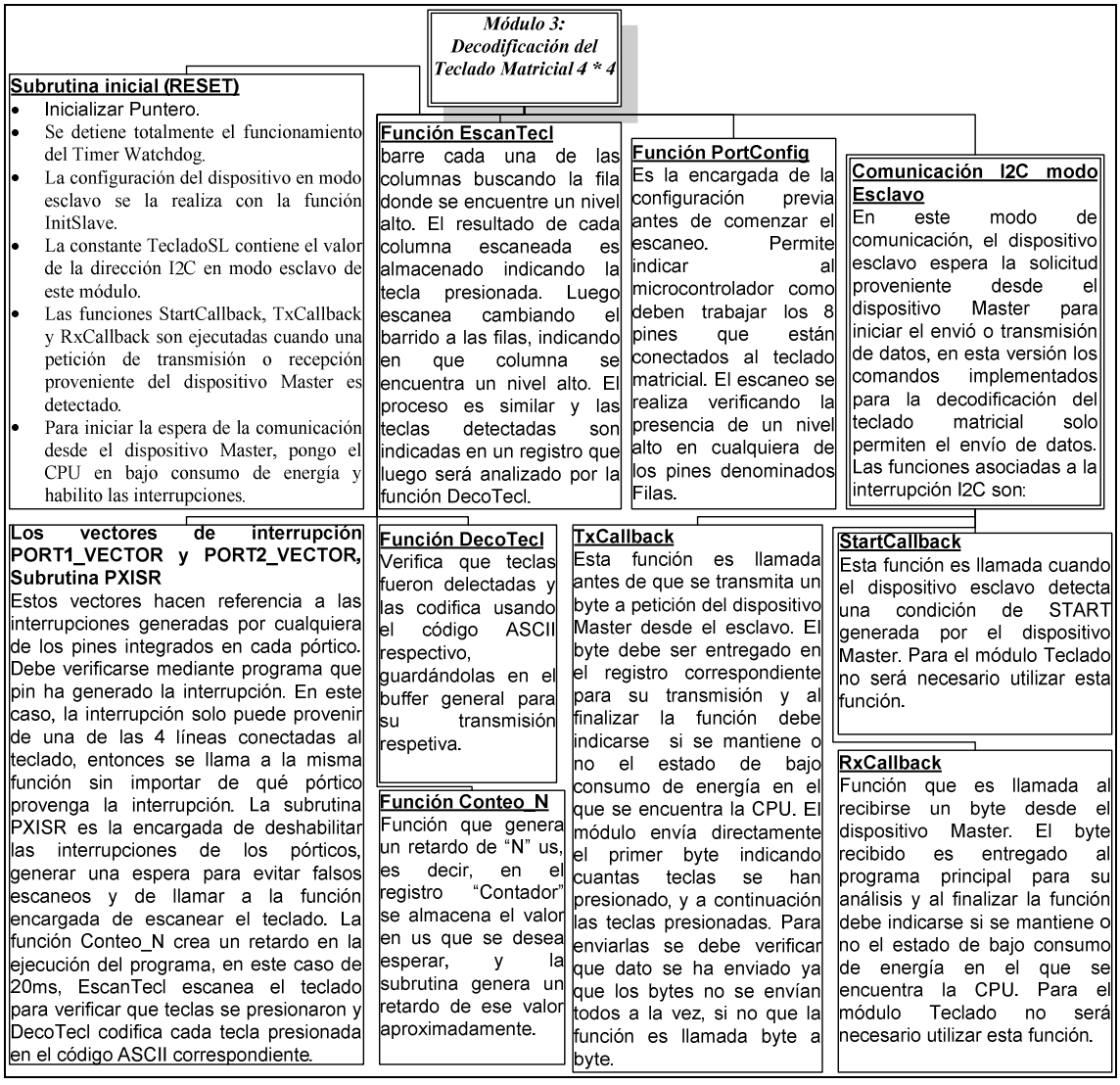


Figura 2.20: Resumen de la principales características del programa que integra el módulo. (Creado por el autor)

2.2.2.1.4 Módulo 4: Salidas Digitales y almacenamiento de datos

Este módulo permite controlar las salidas digitales y además permite almacenar datos en la memoria flash interna del microcontrolador. Está configurado en modo I2C esclavo para poder comunicarse con los otros módulos existentes en el prototipo. Los comandos I2C que puede recibir permiten: El control del estado de cada una de las salidas, almacenar un dato de 2 bytes en memoria, recuperar la información almacenada en memoria, borrar la memoria Flash de datos del dispositivo y verificar la cantidad de bytes utilizados por los datos almacenados en la memoria Flash.

En las figuras 2.21.a y 2.21.b se resumen las partes más importantes del código fuente diseñado para este módulo.

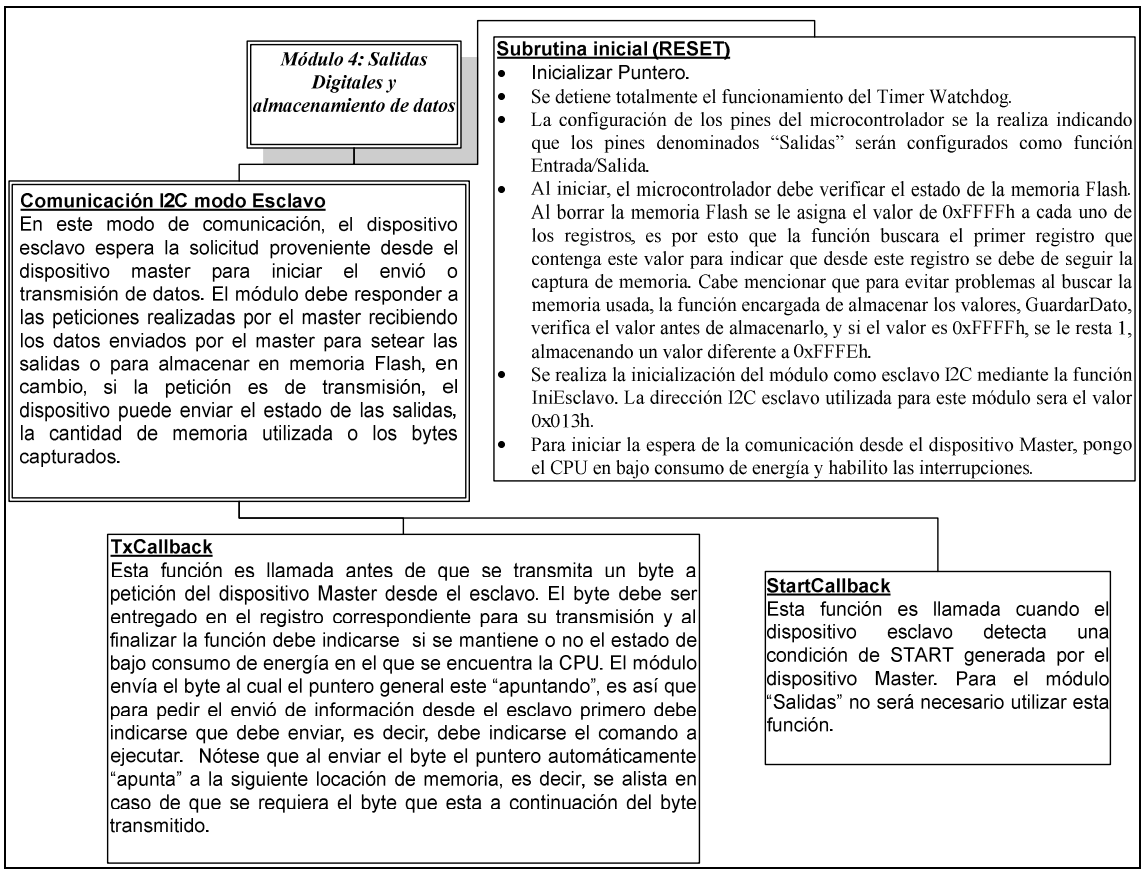


Figura 2.21.a: Resumen del Módulo 4: Salidas Digitales y almacenamiento de datos – Subrutinas RESET, TxCallback y StartCallback. (Elaborado por el autor).

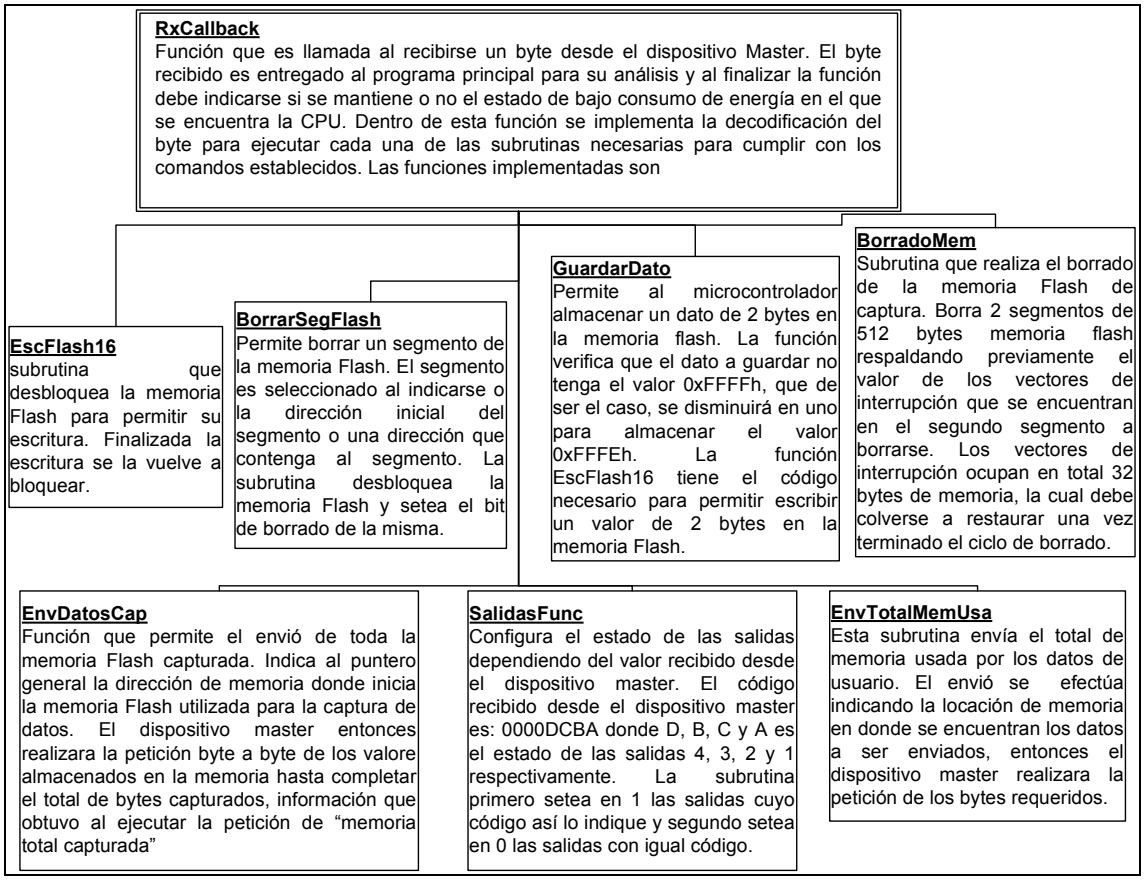


Figura 2.21.b: Resumen del Módulo 4: Salidas Digitales y almacenamiento de datos – Subrutina RxCallback. (Elaborado por el autor).

2.2.2.2 Ejemplo usado en el desarrollo del prototipo

El ejemplo utilizado en el desarrollo del prototipo, permite en forma general verificar el funcionamiento de cada una de las entradas y salidas existentes en el prototipo, la idea principal es utilizar cada una de las entradas como un interruptor para cada una de las salidas.

El encendido o apagado de cada salida se la realiza discriminando el valor obtenido luego de la conversión, este valor digital puede ser:

- A 0 Voltios de entrada, el valor digital será 0
- A 5 Voltios de entrada, el valor digital será 65535

Con estos datos se estableció un valor de escala media, 32767, con el cual se indicaba que a valores de entrada menores o iguales la salida correspondiente estará apagada y valores superiores la salida estará encendida, con esto las entradas estarán trabajando como si se trataran de entradas digitales.

El ejemplo también determina que el valor a ser almacenado cada 10 minutos en la memoria Flash será el valor que la entrada 4 tenga en ese momento.

2.2.2.2.1 Variables, Constantes, Registros y Banderas que el usuario puede utilizar

Aunque el código interno de trabajo del prototipo consume gran cantidad de recursos del microcontrolador, los recursos que se han logrado liberar están disponibles para que el usuario los utilice. A continuación se entrega una lista con todos los recursos disponibles para el usuario.

- Variables con valores que el usuario puede utilizar

  - La tabla 2.5 muestra el nombre de las variables con el resultado de la conversión digital de las entradas

Nombre	***Longitud***	***Descripción***
&ENTRADA_A	2 bytes	Valor de la entrada A
&ENTRADA_B	2 bytes	Valor de la entrada B
&ENTRADA_C	2 bytes	Valor de la entrada C
&ENTRADA_D	2 bytes	Valor de la entrada D

Tabla 2.5: nombre de las variables con el resultado de la conversión digital. (Elaborada por el autor).

  - La Variable “DatGuardar” debe contener el valor a ser almacenado en la memoria Flash, este valor debe tener una longitud de 2 bytes (1 palabra)

- La variable &Temporal1 puede ser utilizada como una variable temporal, cuyo contenido será válido mientras no se termine la ejecución del código de usuario.

- Memoria RAM disponible para el usuario: 8 bytes.

- Los registros de la CPU que el usuario puede utilizar sin ninguna restricción como registros no-volátiles, son los registros R5, R6, R7, R8, R9, R10, R11. Estos registros de 2 bytes de longitud serán alterados solo cuando el prototipo sea desconectado de la red eléctrica.

- El usuario puede afectar el estado de las salidas directamente, para ello, debe utilizar la variable &Banderas y establecer el estado que requiera sobre una o más entradas, la manera de hacerlo es mediante las instrucciones “BIS”, “BIC”, “BIT” y el nombre de la salida. He aquí algunos ejemplos:

  - Activar la Salida 1 y la Salida 3

    ```
    BIS     #SALIDA_1+ SALIDA_3, &Banderas
    ```

  - Desactivar la Salida 2

    ```
    BIC     #SALIDA_2, &Banderas
    ```

  - Verificar el estado de la Salida 4

    ```
    BIT     #SALIDA_4, &Banderas
    JC      Salida_activada   ; La salida esta activada, ejecuto el código respectivo
    ; Salida desactivada
    ```

- El usuario también tiene acceso a 4 banderas de estado dentro de la variable &Banderas, estas banderas son: BandUsu_1, BandUsu_2, BandUsu_3, BandUsu_4, cuyas equivalencias son:

```
; BandUsu_1      EQU     1000h
; BandUsu_2      EQU     2000h
; BandUsu_3      EQU     4000h
```

```
; BandUsu_4      EQU     8000h
```

2.2.2.2.2 Programa demostrativo

Como se indico antes, el programa permite utilizar las entradas como interruptores del estado de la respectiva salida, el código fuente está indicado a continuación:

```
;////////////////////////////////////////////////////////////////////////
CODIGO_USER;            Código de Usuario (INICIO)
;////////////////////////////////////////////////////////////////////////
                mov.w &ENTRADA_D,DatGuardar
                cmp   #32767,&ENTRADA_A    ; comparo la entrada con la mitad de la escala
                jc    SET_1_A
                bic.b #SALIDA_1,Banderas   ; apago la salida
                jmp   ENT_B
SET_1_A         bis.b #SALIDA_1,Banderas   ; enciendo la salida
ENT_B           cmp   #32767,&ENTRADA_B    ; comparo la entrada con la mitad de la escala
                jc    SET_1_B
                bic.b #SALIDA_2,Banderas   ; apago la salida
                jmp   ENT_C
SET_1_B         bis.b #SALIDA_2,Banderas   ; enciendo la salida
ENT_C           cmp   #32767,&ENTRADA_C    ; comparo la entrada con la mitad de la escala
                jc    SET_1_C
                bic.b #SALIDA_3,Banderas   ; apago la salida
                jmp   ENT_D
SET_1_C         bis.b #SALIDA_3,Banderas   ; enciendo la salida
ENT_D           cmp   #32767,&ENTRADA_D    ; comparo la entrada con la mitad de la escala
                jc    SET_1_D
                bic.b #SALIDA_4,Banderas   ; apago la salida
                jmp   FIN_USER
SET_1_D         bis.b #SALIDA_4,Banderas   ; enciendo la salida
;////////////////////////////////////////////////////////////////////////
FIN_USER    ret;                Código de Usuario (FIN)
;////////////////////////////////////////////////////////////////////////
```

2.3 DESCRIPCIÓN Y DESARROLLO DE CLASES Y LIBRERÍAS PARA EL PROGRAMA ESPECIAL DE CONTROL ENTRE EL PC Y EL PROTOTIPO

La interfaz que el usuario podrá utilizar para interactuar con el prototipo será escrita en su mayor parte en el lenguaje Visual BASIC 2005. La descripción de las librerías creadas específicamente para la implementación de este programa serán analizas y explicadas posteriormente, además, podrán ser escritas en otros lenguajes integrados en la herramienta Visual Studio 2005 u otras versión es anteriores.

Debido a que el código fuente de cada módulo es demasiado extenso, no se encuentran incluidos en los anexos, se encontraran ubicados en el CD-ROM o soporte magnético adjunto al material impreso de este proyecto.

2.3.1 RESUMEN DE LOS ENSAMBLADOS DLL CREADOS PARA LA INTERFAZ DEL USUARIO

Los ensamblados o DLL's que acompañan a la aplicación principal llamada "interfaz" son:

- Dialogos_Folder_File
- Manejo de archivos
- Simulacion Teclas: Este ensamblado consta de 2 DLL, creados en Visual Studio 2002. Estos son:
    - Simulacion CPP.dll
    - DLL_String.dll

2.3.1.1 Ensamblado/Librería Dialogos_Folder_File

Esta librería contienen las ventanas de dialogo más utilizadas por los usuarios, estas ventanas permiten seleccionar archivos para su apertura, para su guardado, selección de carpetas, etc. El nombre de la clase de la cual debe

crearse una instancia es “clsDialogos”. En la figura 2.22 se muestran las funciones que el ensamblado pone a disposición del usuario.

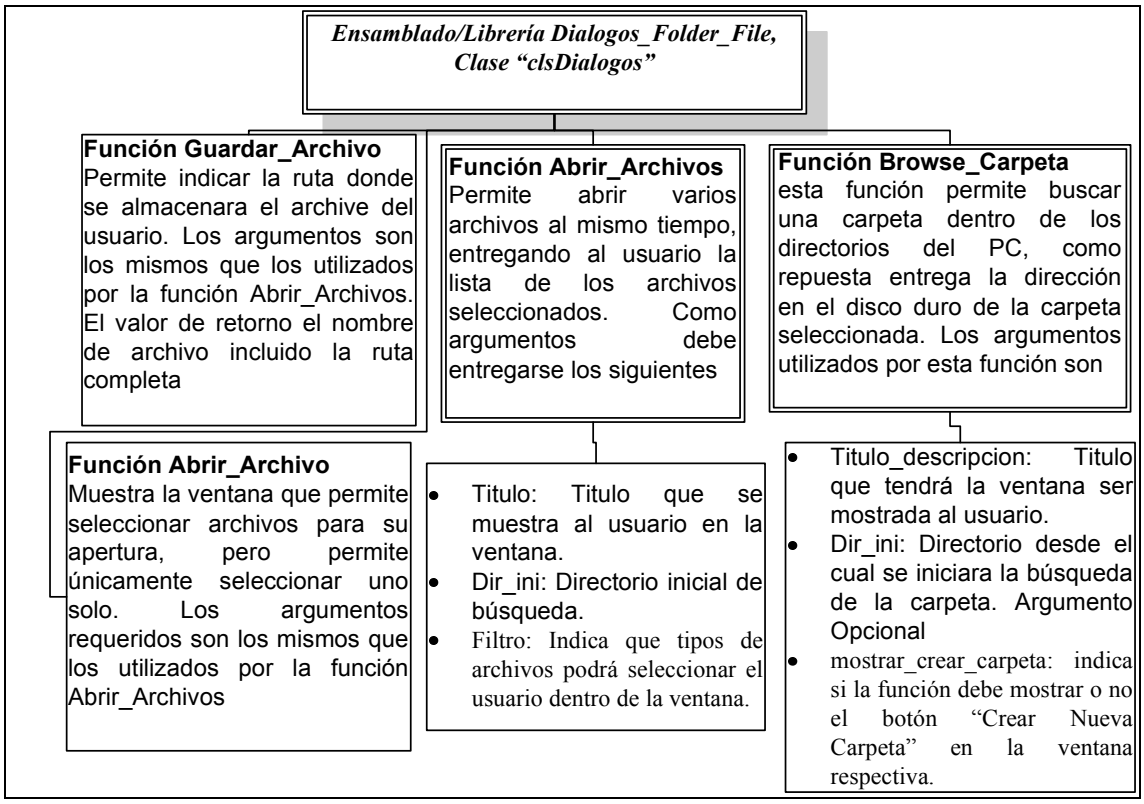


Figura 2.22: Funciones existentes en el ensamblado “Dialogos_Folder_File”. (Elaborado por el autor)

2.3.1.2 Ensamblado/Librería Manejo de archivos

Este ensamblado permite la manipulación de archivos, es decir, permite la escritura o lectura de datos de usuario, datos de programa, guardar o leer configuraciones, etc. También permite realizar la copia de archivos así como verificar la existencia física de un archivo en el disco duro. El nombre de la clase es “clsManejoArchivo”. En la figura 2.23 se describen las funciones existentes en esta clase.

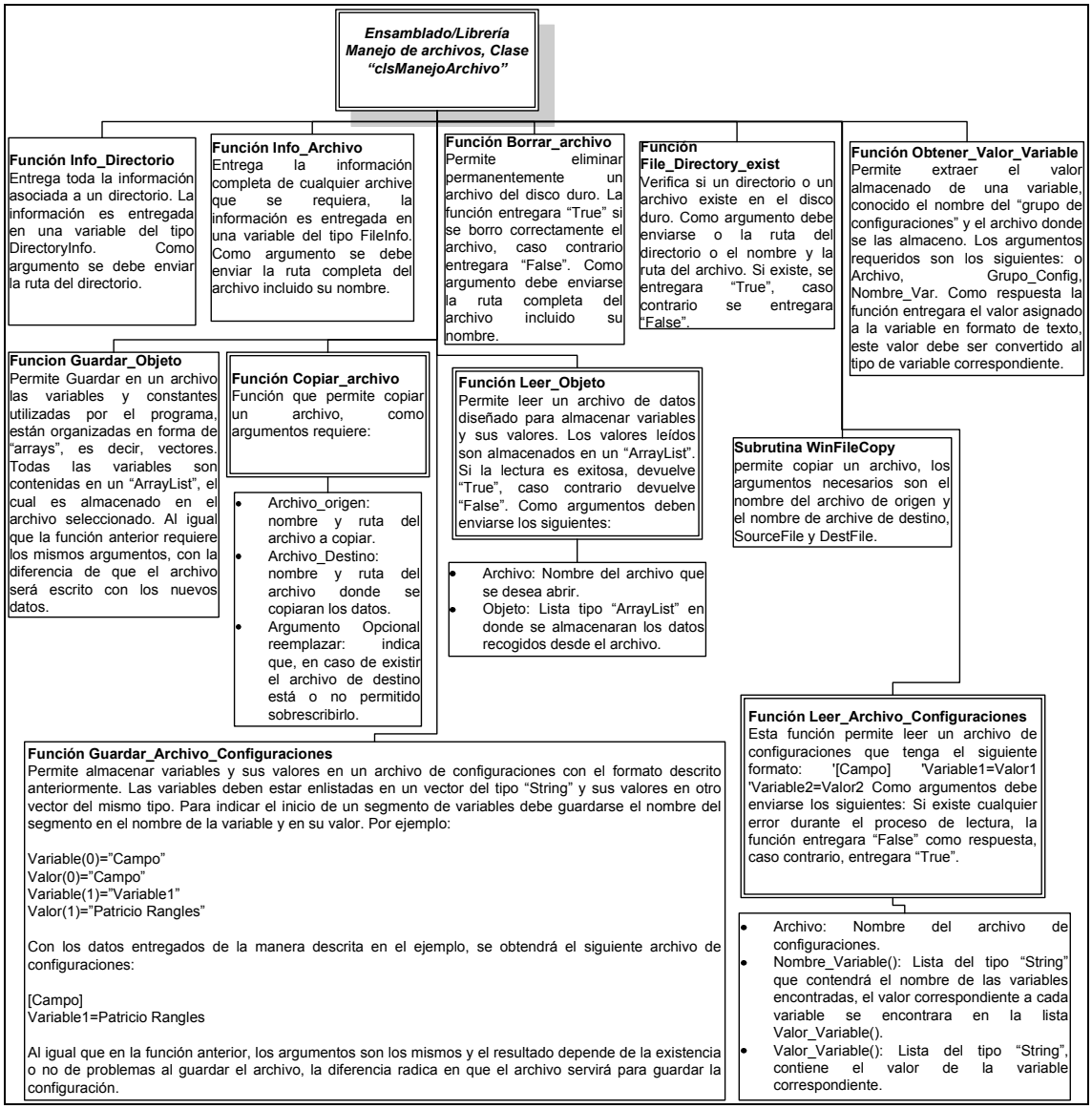


Figura 2.22: Funciones existentes en la librería "clsManejoArchivo". (Elaborado por el autor)

2.3.1.3 Simulación Teclas

Para ejecutar la simulación de teclado, es decir, para permitir que el programa simule la "presión" o pulsación de teclas, se ha creado 2 ensamblados o librerías que entre sí complementan toda la funcionalidad necesaria para que el usuario pueda ejecutar la simulación de presión de una o varias teclas combinadas o no. Debido a que en Visual Studio 2002, los comandos necesarios para llevar a cabo la simulación de teclado en Visual Basic no están disponibles, se recurrió al lenguaje de programación C++ que si los incluye, creándose el ensamblado "Simulacion CPP.dll", también, debido al limitado

número de comandos existentes en C++ para el manejo de cadenas de caracteres, es necesaria la creación del ensamblado DLL_String.dll que facilita en gran medida esta tarea.

Ambos ensamblados están diseñados para trabajar en conjunto, especialmente el ensamblado Simulación CPP.dll, que requiere que el ensamblado DLL_String.dll esté presente siempre. A continuación se detallan las funciones y subrutinas presentes en cada uno de los ensamblados

2.3.1.3.1 Ensamblado/Librería DLL_String

Para crear una nueva instancia de este ensamblado debe utilizarse el nombre de su clase que es “clsString”, esta clase está dividida en 2 secciones: la primera agrupa las funciones que permiten la manipulación de cadenas de texto, mientras que la segunda parte agrupa las funciones que entregan la información sobre las teclas existentes en el sistema para su simulación. En las figuras 2.24.a y 2.24.b se presentan las funciones existentes en este ensamblado.

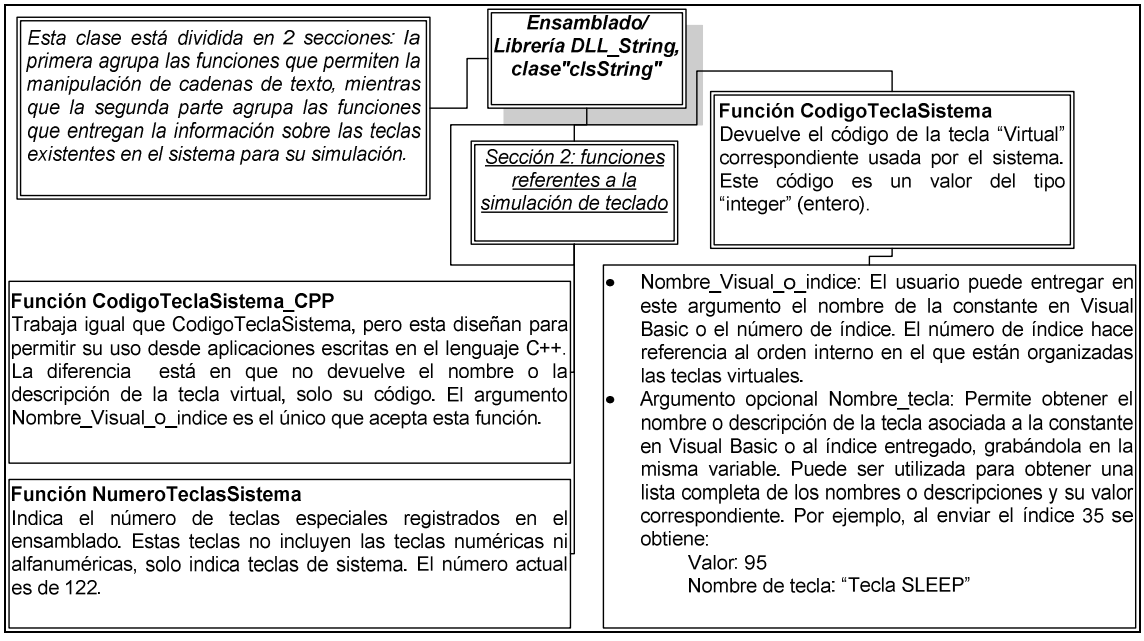


Figura 2.24.a: Funciones existentes en la librería DLL_String, clase clsString. – Sección Manejo de cadenas de texto (Elaborado por el autor)

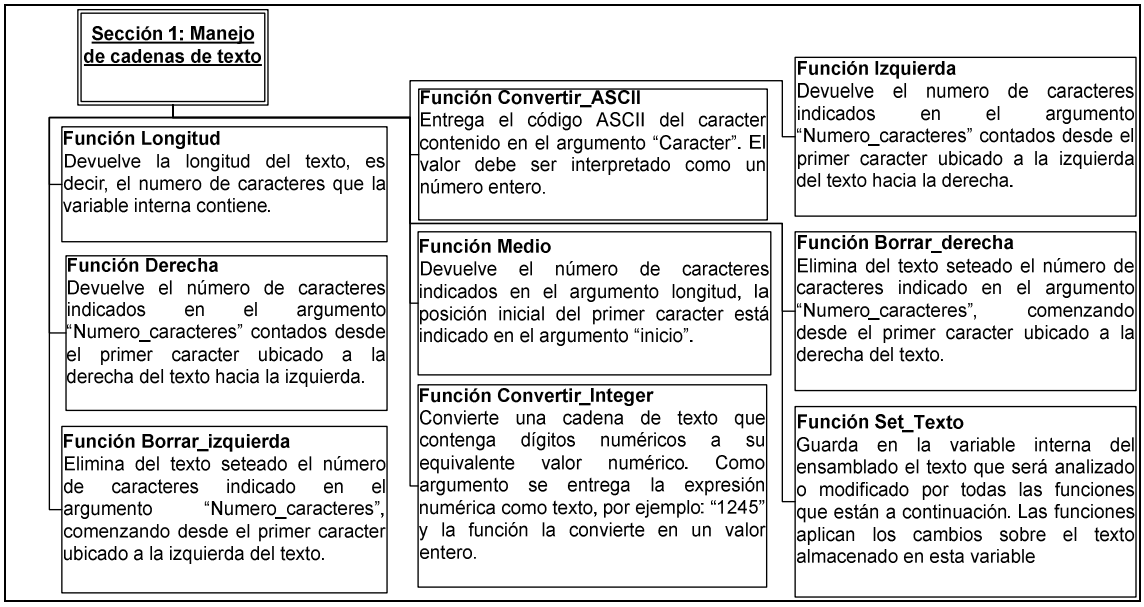


Figura 2.24.b: Funciones existentes en la librería DLL_String, clase clsString. – Sección Manejo de cadenas de texto (Elaborado por el autor)

2.3.1.3.2 Ensamblado/Librería Simulacion CPP.dll

El código del ensamblado esta desarrollado en el lenguaje C++ debido a la facilidad que ofrece el mismo para manipular el hardware del sistema. El nombre de la clase es "clsSend_Keys". En las figuras 2.25.a 2.25.b se muestran las funciones y subrutinas a las que el usuario puede acceder una vez instanciada la clase.

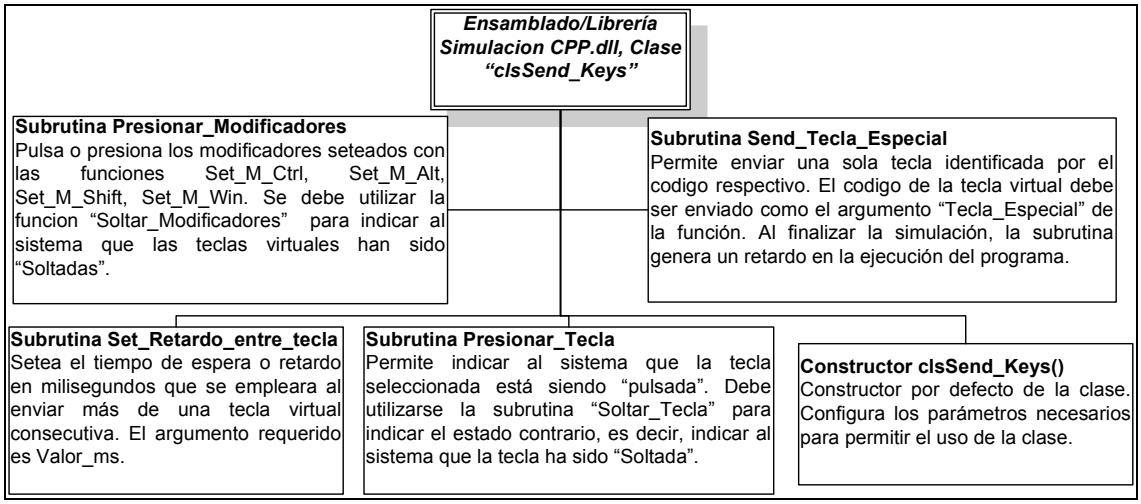


Figura 2.25.a: Funciones y subrutinas a las que el usuario puede acceder una vez instanciada la clase "clsSend_Keys". (Elaborado por el autor)

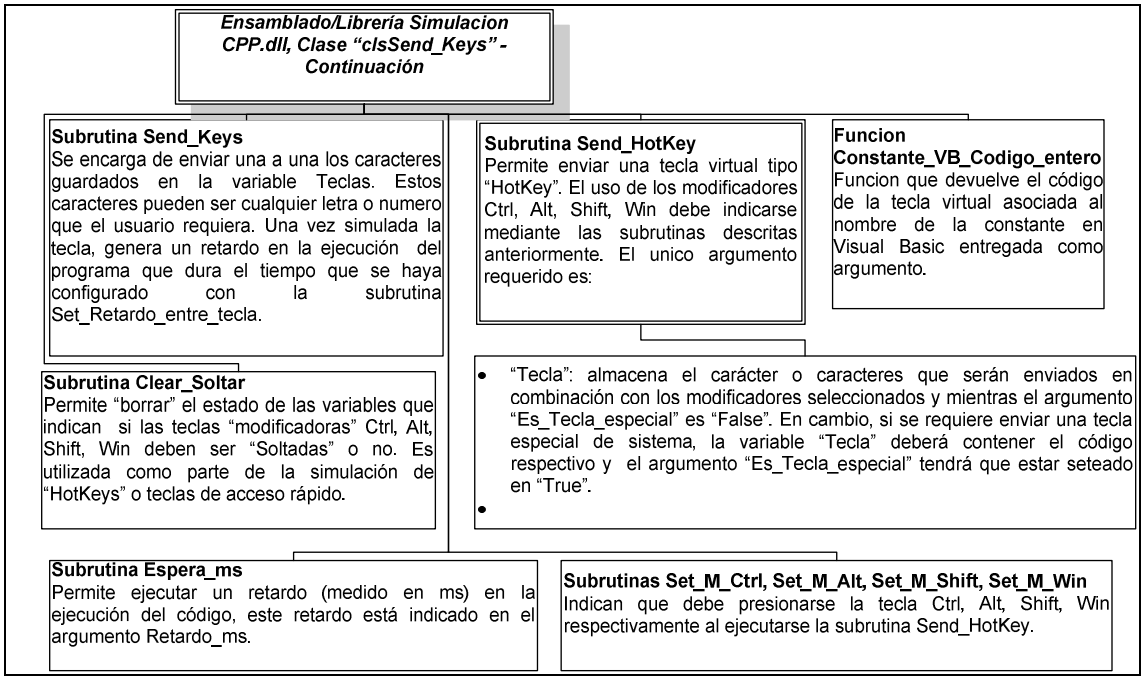


Figura 2.25.b: Funciones y subrutinas a las que el usuario puede acceder una vez instanciada la clase "clsSend_Keys" - Continuación. (Elaborado por el autor)

2.3.2 INTERFAZ PARA LA INTERACCION CON EL USUARIO

Para que el usuario pueda acceder a todas las características implementadas en el prototipo de propósito general, es necesario que disponga de un programa que sea capaz de proporcionarle todas las facilidades de manejo de estas características.

La interfaz Prototipo – PC esta implementada de tal manera que permite al usuario manipular el prototipo y recoger toda la información que este puede entregar con un mínimo de conocimientos de programación. El usuario puede realizar seguimientos al estado de las salidas, como a los valores de las entradas, también puede acceder a la memoria de datos del prototipo permitiéndole extraer toda la información almacenada.

En la figura 2.26 se describen los elementos existentes en el programa desarrollado.

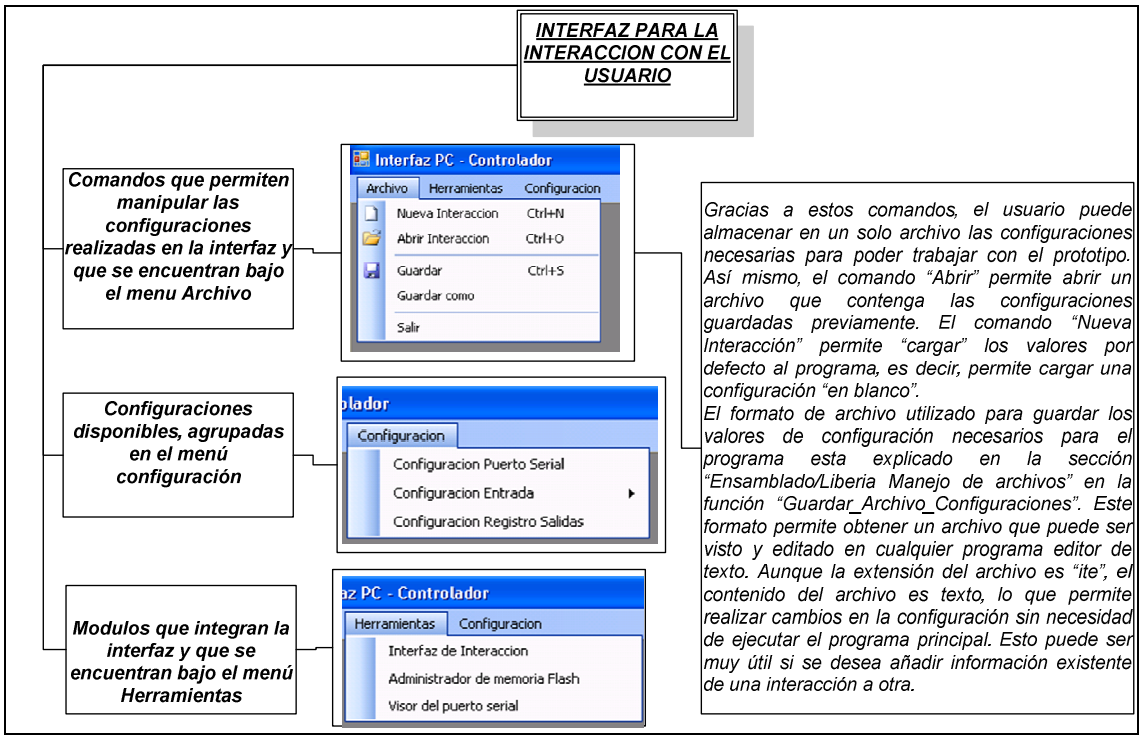


Figura 2.26: Menús y elementos existentes en el programa desarrollado. (Elaborado por el autor)

En el menú Herramientas se integran los siguientes módulos:

- Interfaz de Interacción
- Administrador de memoria Flash:
- Visor del puerto serial

En el menú Configuración, se agrupan las siguientes configuraciones disponibles del programa:

- Configuración Puerto Serial
- Configuración Entrada
- Configuración Registro Salidas

A continuación se describen cada una de las opciones disponibles en el menú principal.

2.3.2.1 Interfaz de Interacción

El módulo interfaz de interacción permite al usuario interactuar directamente con el prototipo. En la figura 2.27 se describen las características que el programa presenta al usuario.

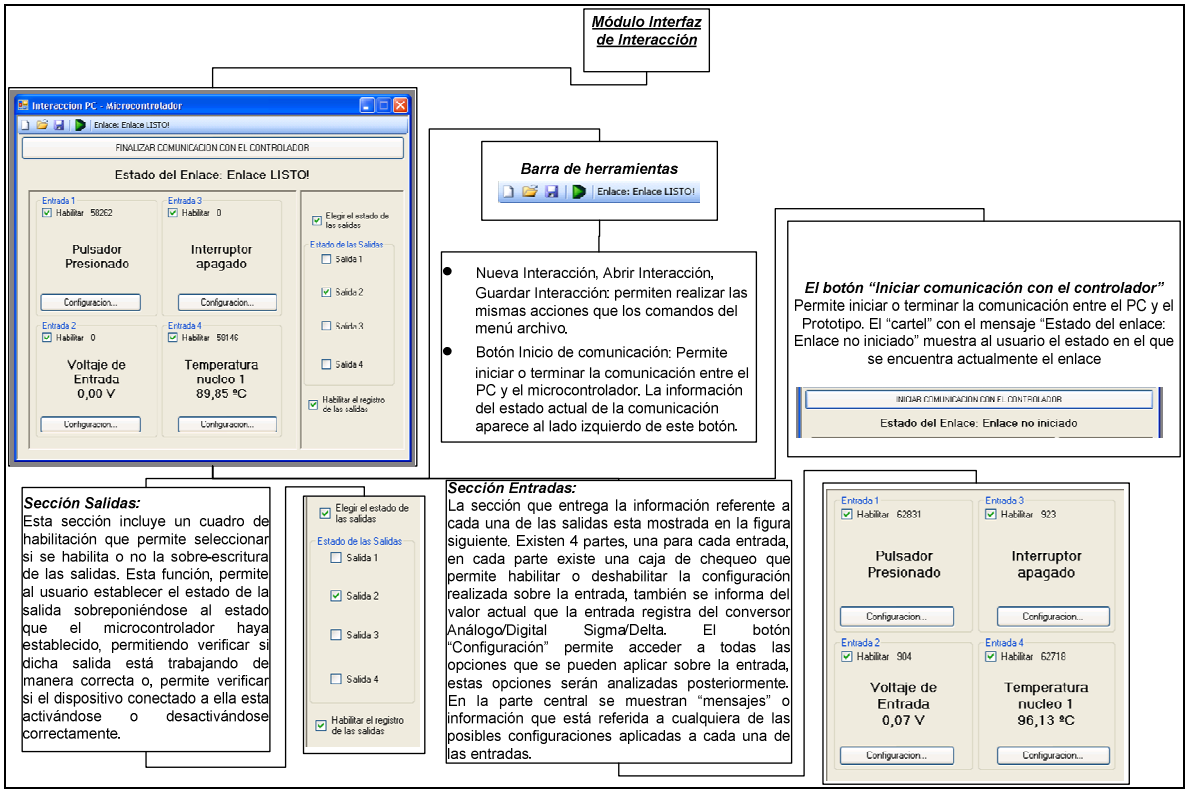


Figura 2.27: Características generales del módulo Interfaz de interacción. (Elaborado por el autor)

2.3.2.2 Administrador de Memoria Flash

Este módulo permite administrar la memoria flash que se está utilizando dentro del prototipo para la adquisición de datos. El módulo además integra las opciones normales para manejar archivos del tipo texto, es decir, permite guardar y abrir archivos de datos almacenados anteriormente, con la única limitante de que no permite editar los datos recogidos desde el prototipo. En la figura 2.28 se resumen las funcionalidades y características de este módulo.

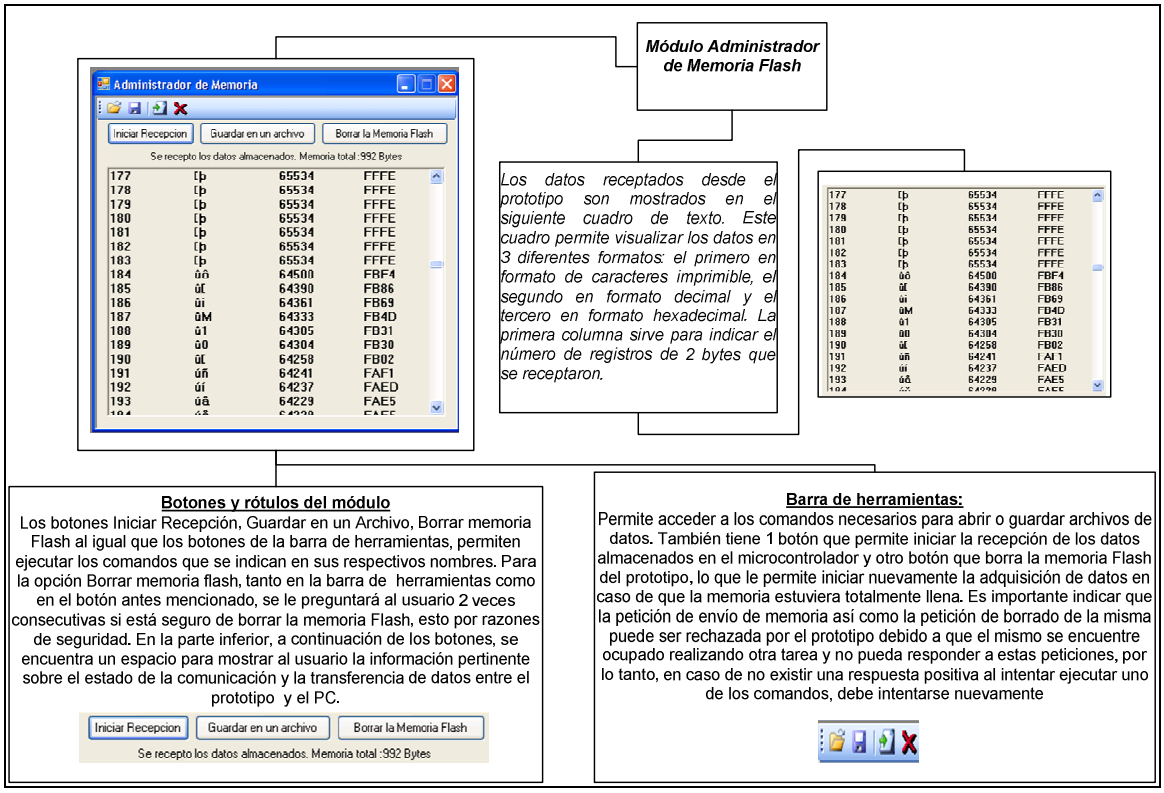


Figura 2.28: Características del módulo Administrador de Memoria Flash. (Elaborado por el autor).

2.3.2.3 Visor del puerto serial

Este módulo permite visualizar los datos enviados desde el programa al prototipo y también los bytes recibidos desde el prototipo. El módulo fue utilizado como una herramienta de depuración mientras se desarrollaban el resto de programas y opciones. Para su uso, debe iniciarse uno de los 2 módulos descritos anteriormente y este debe iniciar la comunicación con el microcontrolador. En la figura 2.29 se muestra el módulo Visor del puerto serial mientras se ejecutaba el módulo “Interacción PC - Microcontrolador”.

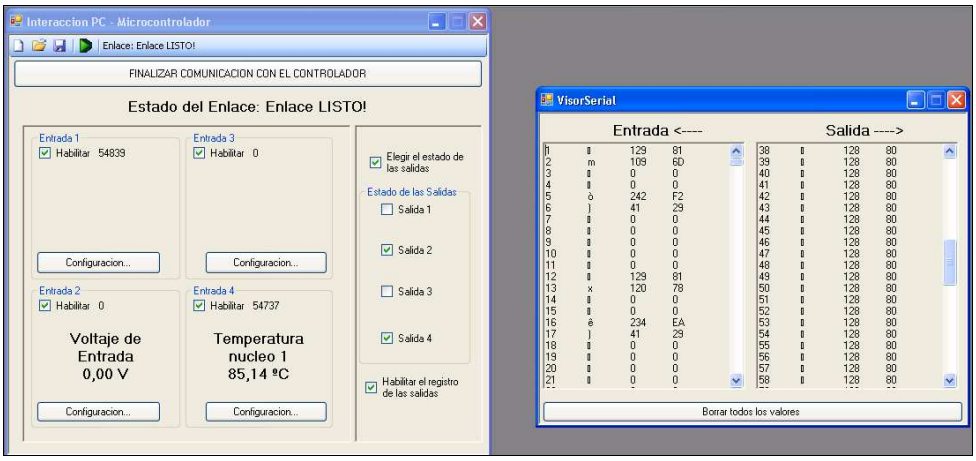


Figura 2.29: Visualización de datos del puerto serial mientras se ejecuta la interacción entre el PC y el prototipo. (Elaborado por el autor)

La ventana de este módulo permite solo la visualización de datos. El único comando disponible para el usuario es el botón “Borrar todos los valores” que inicializa la ventana a su estado inicial. Esta herramienta solo visualiza los datos recibidos y enviados dentro del programa principal, no se puede utilizar para visualizar los datos transferidos en otros programas.

2.3.2.4 Configuración Puerto Serial

Hace referencia a las configuraciones requeridas por el puerto serial. Estas configuraciones serán utilizadas por todos los módulos que existen en la aplicación. En la figura 2.30 se muestra el cuadro de dialogo y las opciones a configurarse

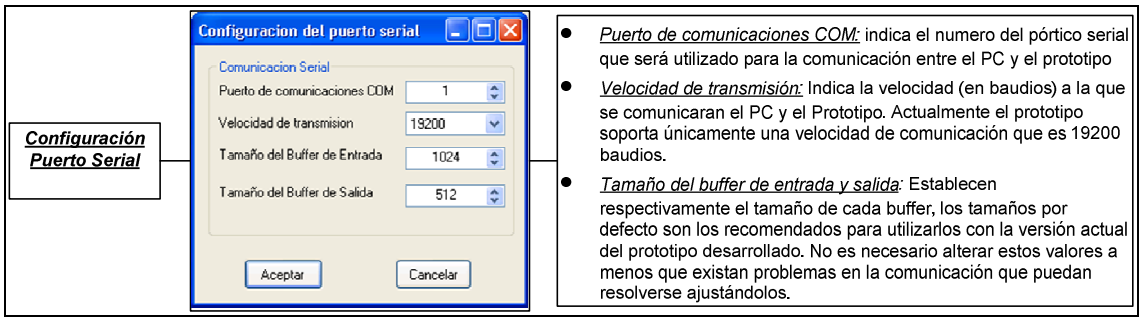


Figura 2.30: Cuadro de dialogo “Configuración del puerto serial” y sus opciones. (Elaborado por el autor)

2.3.2.5 Configuración Entrada

En este menú se puede acceder a la configuración de cada una de las entradas, al igual que lo permite el botón “Configuración” que existe en la sección “Entradas” del módulo “Interacción PC – microcontrolador”. La configuración de cada entrada permite seleccionar al usuario el tipo de entrada que se utiliza en el prototipo, es decir, permite indicar al programa como debe tratar la información que proviene desde las entradas del prototipo.

Dependiendo del tipo de entrada seleccionado por el usuario, el programa permite acceder a varias opciones, la opción ninguno permite dejar la entrada sin ninguna configuración. Los tipos de entrada existentes son:

- Sensor
- ON/OFF – Pulsador

También existe la posibilidad de registrar en un archivo los valores de cada entrada enviados por el microcontrolador, esto mediante la opción “Registro” del cuadro de dialogo “Configuración de entrada”, las opciones que se pueden encontrar son detalladas en la figura 2.31. En la figura 2.32 se muestra las opciones existentes para el tipo de entrada Sensor y en la figura 2.33 para el tipo de entrada ON/OFF – Pulsador.

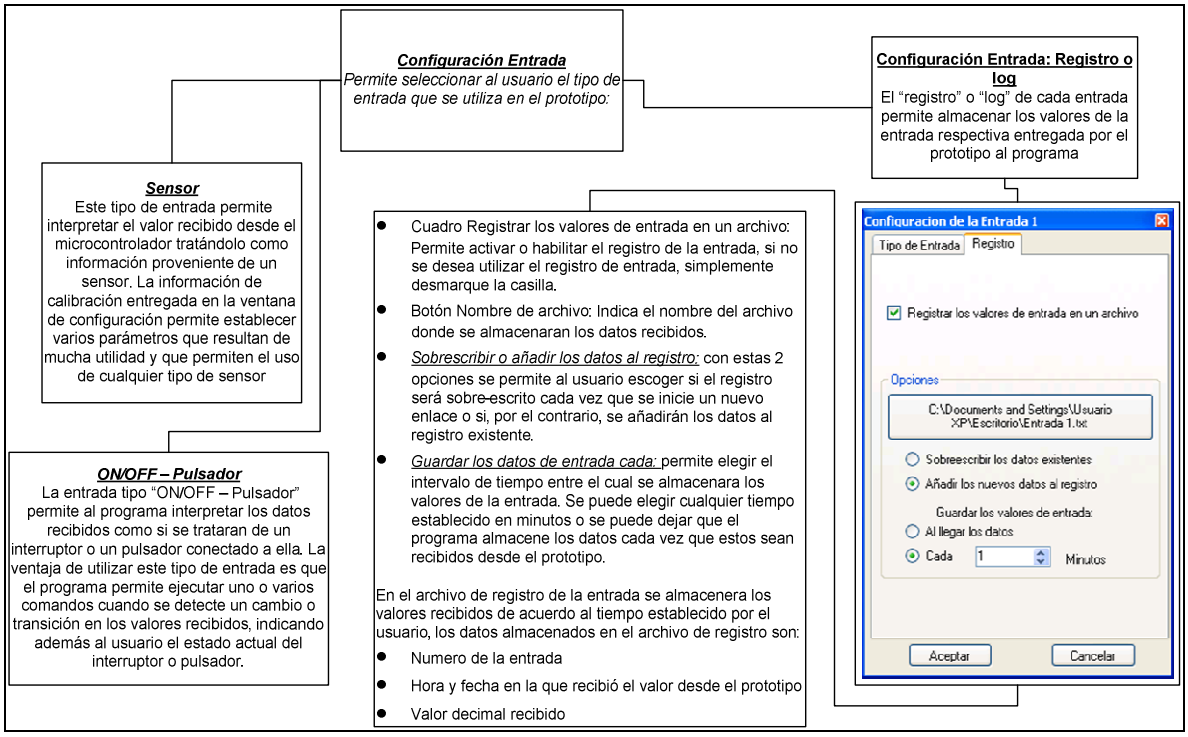


Figura 2.31: Detalle de la configuración de las entradas. (Elaborado por el autor)

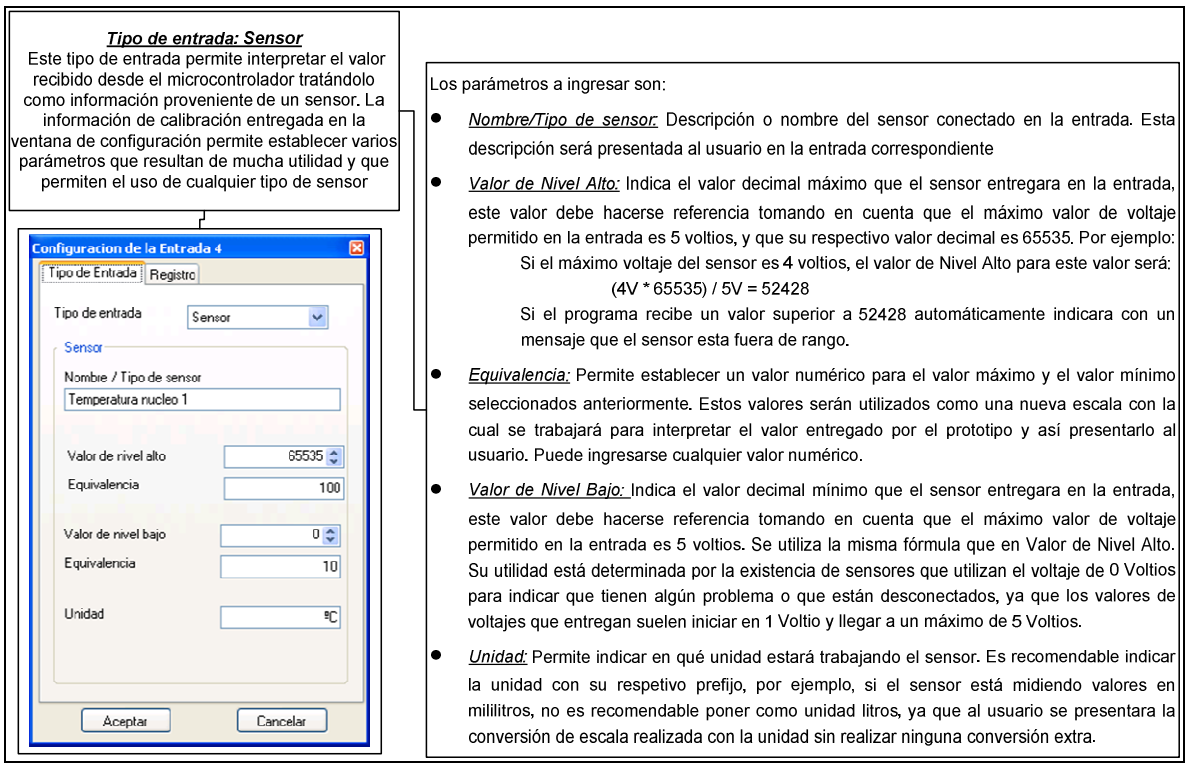


Figura 2.32: Opciones del tipo de entrada "Sensor". (Elaborado por el autor)

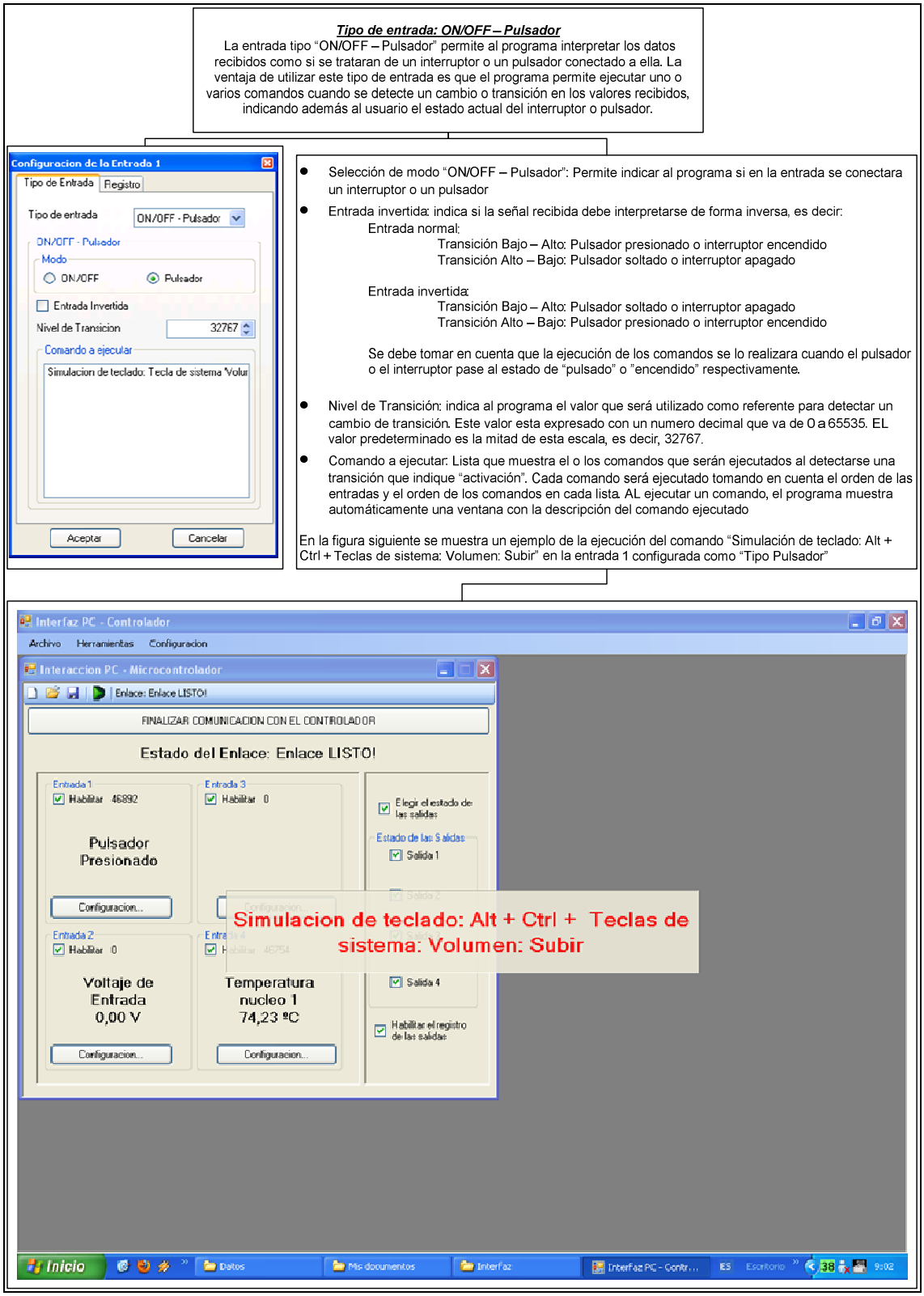


Figura 2.33: Opciones del tipo de entrada "ON/OFF – Pulsador". (Elaborado por el autor)

En la figura 2.34 se muestran los tipos de comandos existentes para el tipo de entrada “ON/OFF – Pulsador”, incluyendo las opciones que posee cada una de ellas. Con estas opciones es posible ejecutar la mayoría de programas existentes en el PC, o, también, es posible ejecutar combinaciones de teclas específicas para ejecutar una acción pre-configurada en el PC.

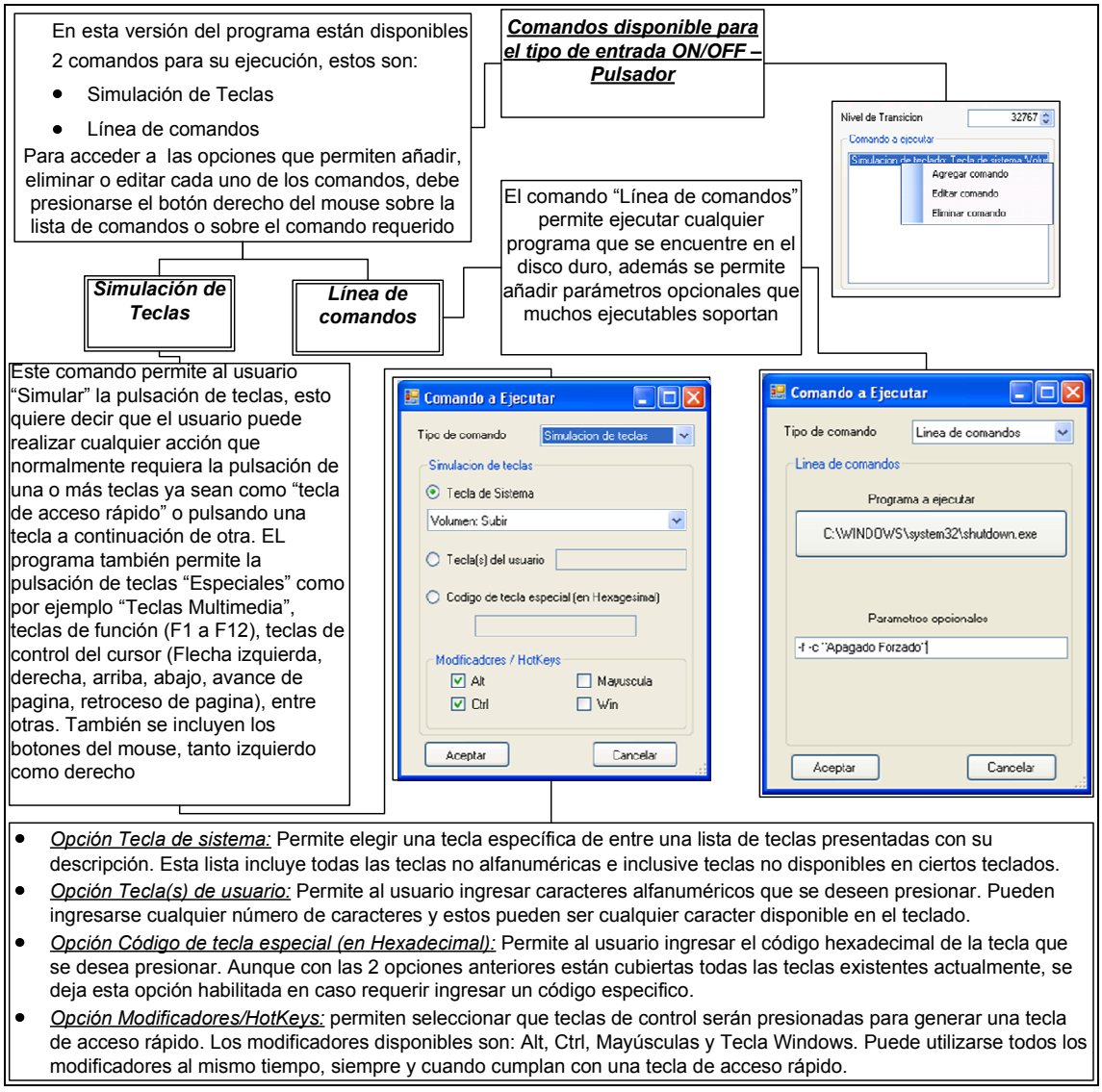


Figura 2.34: Tipos de comandos existentes para el tipo de entrada ON/OFF – Pulsador. (Elaborado por el autor)

2.3.2.6 Configuración Registro Salidas

Esta configuración permite establecer los valores requeridos por el programa para almacenar los cambios de estados de cada una de las salidas. En la figura

2.35 se muestra el cuadro de dialogo “Configuración del registro” con las opciones que el usuario puede cambiar.

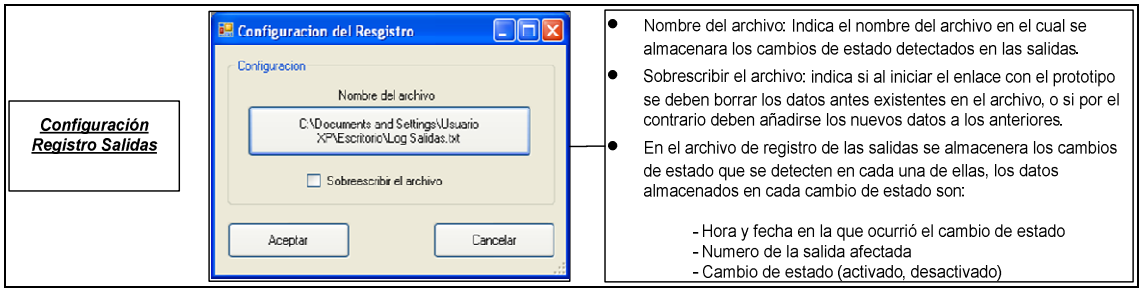


Figura 2.35: de dialogo “Configuración del registro” y sus opciones. (Elaborado por el autor)

2.4 PROGRAMACIÓN DEL MICROCONTROLADOR

2.4.1 DESCRIPCION DE LA HERRAMIENTA DE EMULACION FLASH (FLASH EMULATION TOOL- FET)

Texas Instruments ofrece varias herramientas de emulación Flash según los diferentes requisitos. En la Tabla 2.6 se detallan las herramientas existentes y sus características generales.

Característica	***eZ430-F2013***	eZ430-RF2500	MSP-FET430UIF	MSP-FET430PIF
Soporta todos los dispositivos MSP430 basados en memoria Flash (F1xx, F2xx, F4xx, F5xx)			X	X
Soporta únicamente los dispositivos MSP430F20xx	X			
Soporta únicamente los dispositivos MSP430F20xx/F21x2/F22xx		X		
Permite “quemar” el fusible de protección			X	
Voltaje ajustable en el dispositivo de prueba			X	
Voltaje ajustado a 2.8V en el dispositivo de prueba				X
Voltaje ajustado a 3.6V en el dispositivo de prueba	X	X		
JTAG de 4 hilos			X	X
JTAG de 2 hilos (*)	X	X	X	
Aplicaciones UART		X		

Soportado por CCE	X	X	X	X
Soportado por IAR	X	X	X	X

(*) La interfaz de debug JTAG de 2 hilos también es denominada como interface Spy-Bi-Wire (SBW)

Tabla 2.6: Características de la Herramienta de emulación Flash

2.4.1.1 Instalación del Software

En el CD de instalación se incluye la versión más reciente del software al momento del lanzamiento del producto, pero se recomienda buscar en la página WEB (http://focus.ti.com/docs/toolsw/folders/print/iar-kickstart.html) la última versión disponible para su descarga.

Una vez descargada la última versión disponible (o usando la versión contenida en el CD), se procede con la instalación del programa. Para el desarrollo de este proyecto se usara el IAR Embedded Workbench KickStart for MSP430 V5.0, descargado desde la pagina WEB.

Los pasos a seguir para instalarlo son los siguientes:

1) Ejecutamos el instalador, en este caso ejecutamos el archivo FET_R513.exe que se encuentra comprimido dentro del archivo slac050t.zip, descargado desde la WEB.

2) Al ejecutar este archivo aparece un cuadro de dialogo, en el se debe seleccionar el botón SETUP.

3) Aparece el instalador del entorno IAR. Como siguiente paso, presionamos NEXT en la pantalla que nos presenta el ordenador.

4) Seguimos los pasos necesarios hasta que la instalación sea completada. Al finalizar la instalación, aparecerá una pantalla en la que presionaremos “Finish” y seguido de esto se mostrará una página WEB con notas sobre la versión que acabamos de instalar, así como enlaces útiles con más información sobre esta tecnología.

Esta aplicación y sus controladores son compatibles con las siguientes plataformas:

Aplicación: Windows® 98, Windows 2000, Windows ME, Windows NT 4.0, Windows XP, y Windows Vista ya sea de 32 o 64 bits.

Controladores: La interface FET USB trabaja solamente con Windows 2000, Windows XP, y Windows Vista, todas estas en sus versiones de 32 bits.

2.4.1.2 Instalación del Hardware MSP-EZ430-F2013

1. Conecte la interfaz de depuración MSP-EZ430-F2013 a un puerto USB del PC.

2. Windows debe reconocer el hardware como "MSP-FET430UIF JTAG Tool". El nombre del dispositivo puede variar dependiendo la versión del mismo.

3. El Asistente de Hardware inicia automáticamente y se abre la ventana denominada "Asistente para hardware nuevo encontrado", en esta ventana debe seleccionarse la opción “No por el momento” para evitar que busque el controlador en el internet. Haga clic en siguiente.

4. En la ventana siguiente se debe escoger la opción “Instalar automáticamente el software (recomendado)” Al dar clic sobre el botón "Siguiente" el Asistente de Hardware intentará encontrar el controlador en el sistema. Si el controlador fuese encontrado, entonces se puede continuar con el paso 8. En caso de no encontrarse, presione "Regresar" y continúe con el paso 5.

5. Seleccione la opción "Instalar desde una lista o ubicación específica (avanzado)".

6. Seleccione la opción “Incluir esta ubicación en la búsqueda” y seleccione el directorio donde están localizados los controladores, que por defecto es “C:\ Archivos de programa\ IAR Systems\ Embedded Workbench 5.0\ 430\ drivers\ TIUSBFET\ WinXP”.

7. El Asistente muestra un mensaje indicando que el controlador apropiado ha sido encontrado.

8. Note que Windows XP muestra una advertencia de que el controlador que se quiere instalar no está certificado por Microsoft. Ignore esta advertencia y presione "Continuar de cualquier manera”.

9. El Asistente mostrara un mensaje indicando que se ha finalizado la instalación del software para el "Adaptador MSP-FET430UIF (TI USB FET)" o "MSP430 Application UART".

10. Después de cerrar el Asistente de nuevo Hardware encontrado, Windows automáticamente reconocerá otro dispositivo de Hardware llamado "MSP-FET430UIF - Serial Port".

11. Dependiendo de la versión actual del sistema operativo, los controladores correspondientes pueden ser instalados automáticamente o puede ser que el Asistente de Nuevo Hardware encontrado se reabra. Si el Asistente se presentara solicitando información, entonces repita los pasos descritos arriba.

2.4.1.3 Probando el IAR Embedded Workbench y el MSP-ez430-F2013: Parpadeando el LED

En el siguiente ejemplo se demuestra el uso del FET con el programa introductorio equivalente al ejemplo introductorio del lenguaje C "¡Hola Mundo!". Una aplicación que produce el “parpadeo” del LED es escrita y descargada al FET, que finalmente reproducirá la aplicación desarrollada.

A continuación se describen los pasos que deben seguirse para reproducir el ejemplo:

1. Inicie el IAR Embedded Workbench (Menu Inicio → Programas → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V4 → IAR Embedded Workbench).

2. Haga clic en el menú File → Open Workspace con lo que se abre el cuadro de dialogo que permite abrir el siguiente archivo: C:\ Archivos de Programa\ Embedded Workbench 5.0\ 430\ FET_examples\ Flashing the LED.eww. Una vez seleccionado el archivo la ventana del área de trabajo se abre.

3. A continuación de un clic sobre la etiqueta al pie de la ventana del área de trabajo que corresponde a su dispositivo MSP430 (MSP430xxxx) y el lenguaje que se desea usar (ensamblador o C), como se muestra en la Figura 2.36.

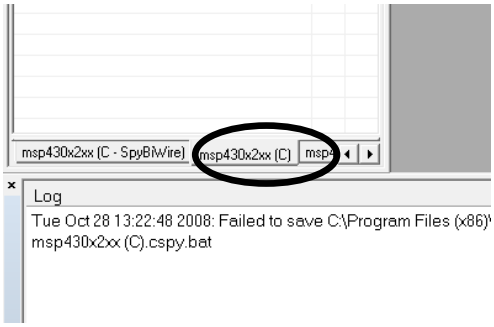


Figura 2.36: Selección del área de trabajo del ejemplo demostrativo “Flashing the LED”, lenguaje seleccionado es C. (Elaborado por el autor).

4. Ahora de un clic en el menú Project → Options → FET Debugger → Setup → Connection para seleccionar el puerto apropiado. Las opciones posibles son:
    a. Usar el Texas Instruments LPT-IF para el FET de interfaz de puerto paralelo (MSP-FET430PIF)
    b. Usar el Texas Instruments USB-IF para el de interfaz USB (MSP-FET430UIF) que también sirve para el MSP-eZ430-F2013.

Para este caso deberá seleccionarse la interfaz USB-IF como se muestra en la Figura 2.37

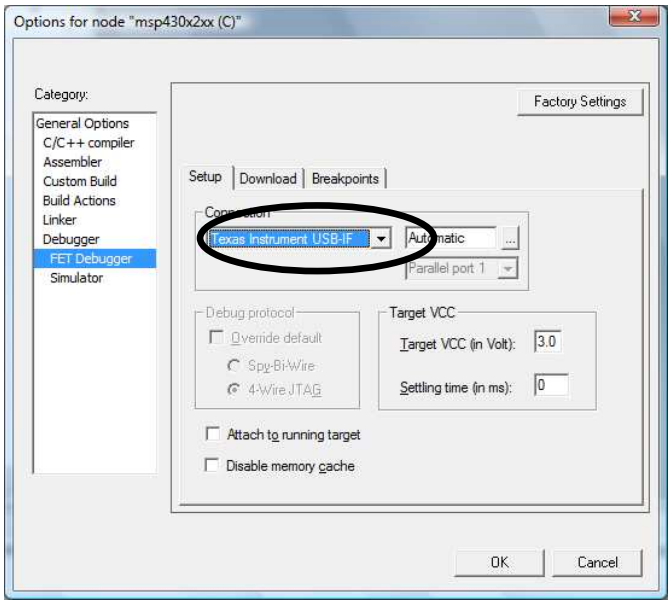


Figura 2.37: Selección del puerto USB-IF para la comunicación con el dispositivo de pruebas ez430-F2013. (Elaborado por el autor).

5. Luego de un clic en el menú Project → Rebuild All para recompilar el código fuente. Se puede mirar el código del ejemplo haciendo doble clic en el nombre del proyecto, y después haciendo doble clic en el archivo fuente requerido. En la Figura 2.38 se muestra la selección del comando “Rebuild All” del menú Project.

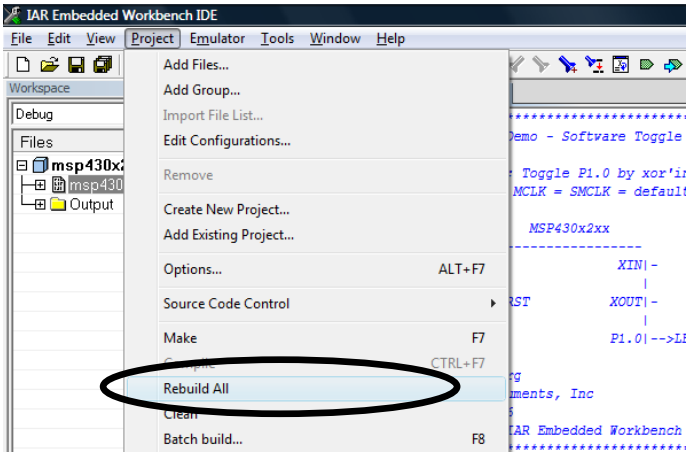


Figura 2.38: Recopilación del proyecto

6. Como siguiente paso, de un clic en el menú Project → Debug para iniciar el depurador C-SPY. C-SPY borra primero la memoria Flash del microcontrolador para luego descargar el programa compilado en la memoria Flash del dispositivo. Cabe indicar que el dispositivo depurador eZ430-F2013 debe estar correctamente conectado al puerto USB del PC para que la descarga del programa sea realizada correctamente. En el anexo B.7 se incluye las preguntas y respuestas a los problemas más comunes (FAQ) en caso de que C-SPY sea incapaz de comunicarse con el dispositivo. En la Figura 2.39 se muestra la selección del comando Debug para iniciar la depuración del ejemplo en el microcontrolador.

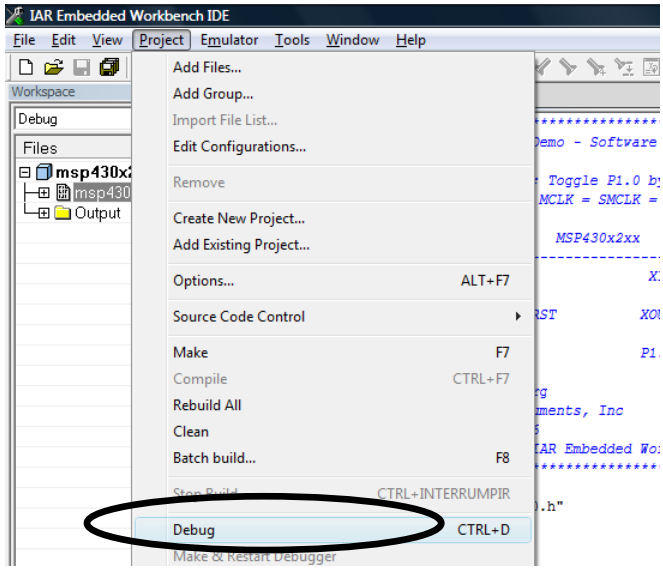


Figura 2.39: Iniciar la depuración mediante C-SPY

7. Una vez cargada la aplicación en la memoria Flash del microcontrolador de un clic en el menú Debug → Go para iniciar la aplicación. El diodo emisor de luz debería brillar intermitentemente.

8. Para detener la depuración del programa de un clic en el menú Debug → Stop Debugging, así se termina con la depuración en C-SPY, y se regresa al IAR Embedded Workbench.

9. Para salir completamente del programa, de un clic en el menú File → Exit.

2.4.1.4 Documentación y fuentes de consulta para los dispositivos de la familia MSP430

Las fuentes primarias de información del MSP430 son los datos específicos del dispositivo y la guía de usuario.

Las versiones más actuales de estos documentos disponibles a la hora de producción son incluidas en el CD-ROM que se entrega conjuntamente con la herramienta de depuración eZ430-F2013. El sitio Web del MSP430 (www.ti.com/msp430) provee a los usuarios de las versiones más recientes de estos documentos.

La documentación en formato PDF que describe cada una de las herramientas del IAR (el Workbench/C-SPY, el ensamblador, el compilador C, el editor de enlace, y las librerías) se encuentran en las carpetas “common\doc” y “430\doc” dentro del directorio de instalación. Los suplementos de estos documentos, por ejemplo la información de última hora, están disponibles en formato HTML en los mismos directorios. El archivo “430\doc\readme_start.htm” provee un punto de partida conveniente para navegar la documentación del IAR.

En los archivos del tipo “.htm” que se encuentran localizados a todo lo largo del árbol de directorios del KickStart, está contenida la mayor parte de información actualizada y suplementa a los archivos PDF. Además, la documentación del KickStart está disponible mediante la ayuda en línea.

Los archivos denominados “Léeme primero” (README FIRST) tanto del IAR como de TI además del documento SLAU138K.pdf pueden ser encontrados en el Menú Inicio $\rightarrow$ Programas $\rightarrow$ IAR Systems $\rightarrow$ IAR Embedded Workbench KickStart for MSP430 V4. En la Tabla 2.7 está el detalle de la información existente al momento del lanzamiento de la herramienta de desarrollo eZ430-F2013

Herramienta	Guía del usuario	Información más actualizada
Workbench/C-SPY	EW430_UsersGuide.pdf	Readme.htm, ew430.htm, cs430.htm, cs430f.htm
Ensamblador	EW430_AssemblerReference.pdf	A430.htm, a430_msg.htm
Compilador	EW430_CompilerReference.pdf	Icc430.htm, icc430_msg.htm
Librería C		Clibrary.htm
Enlazador y Librería	Xlink.pdf	Xlink.htm, xman.htm, xar.htm

Tabla 2.7: Archivos de ayuda e información correspondientes a cada herramienta. (Fuente: Texas Instruments)

2.4.2 FLUJO DE DESARROLLO

El desarrollo de este sub-capitulo está basado en la información obtenida al fabricante, principalmente la información encontrada en el archivo "slau138k.pdf", en el cual esta detallado el flujo de desarrollo que debe seguirse para programar los microcontroladores de la familia MSP430.

2.4.2.1 Visión general

Las aplicaciones pueden ser desarrolladas en lenguaje ensamblador y/o C usando el Workbench y son depuradas usando el C-SPY. El depurador C-SPY trabaja como una sola pieza integrada al Workbench. Sin embargo, es más conveniente hacer la distinción entre el ambiente de desarrollo de código (Workbench) y el ambiente de depuración C-SPY. El C-SPY puede ser configurado para operar con la Herramienta de Emulación Flash (FET – **F**lash **E**mulation **T**ool), es decir con un dispositivo real MSP430 o con un simulador informático del dispositivo.

El término KickStart se usa para referirse al Workbench y al C-SPY colectivamente. Las herramientas de desarrollo del software KickStart son un producto del IAR.

2.4.2.2 Usando el ambiente de desarrollo KickStart

El ambiente de desarrollo KickStart es de función limitada y posee las siguientes restricciones:

- El compilador del lenguaje C no genera un archivo de lista de código de ensamblado.
- El editor de enlace asocia un máximo de 4K bytes de código en lenguaje C, pero puede enlazar una cantidad ilimitada de código ensamblador.
- El simulador introduce un máximo de 4K bytes de código.

Una versión completa y sin restricciones de las herramientas de desarrollo puede ser adquirida en IAR. Un set de función media – llamado "Baseline", con una limitante en el tamaño de código C de 12K bytes y operaciones básicas en coma flotante está disponible en IAR. Consulte el sitio Web de IAR (www.iar.se) para obtener mayor información.

2.4.2.3 Los Ajustes del Proyecto

Los ajustes requeridos para configurar el ambiente de desarrollo (Workbench) y el ambiente de depuración (C-SPY) son numerosos y detallados. Revise los ajustes de proyecto de los ejemplos que se incluyen al instalar el software tanto para lenguaje C como para lenguaje ensamblador, se puede acceder a los ajustes del proyecto haciendo clic en el menú Project $\rightarrow$ Options con el nombre del proyecto seleccionado. Use estos ajustes de proyecto como plantillas al desarrollar sus proyectos. Note que si el nombre de proyecto no se encuentra seleccionado cuando los ajustes son realizados, entonces estos ajustes serán aplicados al archivo que se encuentre seleccionado.

A continuación se explican los ajustes de proyecto recomendados o que son requeridos por la herramienta de desarrollo. El menú de acceso para cada opción se muestra entre paréntesis.

- Especifique el dispositivo o microcontrolador que será usado para las pruebas (General Options → Target → Device).
- Habilite el tipo de proyecto a desarrollar, puede usar solo lenguaje ensamblador o convinar el lenguaje C y ensamblador (General Options → Target → Assembler-only project).
- Posibilite o no la generación de un archivo de salida “ejecutable” (General Options → Output → Output file → Executable).
- Para depurar de manera más sencilla un proyecto hecho en lenguaje C, deshabilite la optimización (C/C++ Compiler → Optimizations → Size → None). Esto provee de mejor soporte para la depuración.
- Habilite la generación de la información de depuración a la salida del compilador (C/C++ Compiler → Output → Generate debug information).
- Especifique la ruta donde se encuentra el preprocesador del lenguaje C (C/C++ Compiler → Preprocessor → Include Paths).
- Habilite la generación de la información de depuración a la salida del ensamblador (Assembler → Output → Generate Debug Info).
- Especifique la ruta donde se encuentra el preprocesador del ensamblador (Assembler → Preprocessor → Include Paths).
- Para depurar el proyecto usando C-SPY, especifique un formato compatible en el menú Linker → Output → Format → Debug information for C-SPY, ya sea que utilice módulos de control de rutinas o use un módulo de emulación de entrada/salida.
- Especifique la ruta donde se encuentre cualquiera de las bibliotecas usadas (Linker → Config → Search paths).
- Especifique el controlador que usara con C-SPY. Seleccione la opción Project → Options → Debugger → Setup → Driver → FET Debugger para usar la depuración en el FET (por ejemplo con el dispositivo MSP430). Seleccione "Simulator" para realizar la depuración en el simulador integrado. Si el FET Debugger es seleccionado, use la opción Project → Options → FET Debugger → Setup → Connection para seleccionar el tipo de puerto apropiado: Texas Instruments LPT-IF para el FET de

Interface paralela (MSP-FET430PIF) o Texas Instruments USB-IF para el de interfaz USB (MSP-FET430UIF) que sirve también para el eZ430.

- Habilite el archivo "Device Description" (Debugger → Setup → Device description file → Override default). Este archivo indica al depurador C-SPY las características específicas del dispositivo que usara para la depuración. Este archivo depende del microcontrolador especificado en el dispositivo de pruebas.
- Habilite el borrado de la memoria principal y la memoria de Información antes de que se envié el código del programa al microcontrolador (FET Debugger → Download → Erase main and Information memory).
- Para maximizar el desempeño durante la depuración, deshabilite los puntos de interrupción Virtuales (FET Debugger → Breakpoints → Use virtual breakpoints) y también deshabilite todos los puntos de interrupción del sistema (FET Debugger → Breakpoints → System breakpoints on).

Para configurar un proyecto rápidamente puede usar las configuraciones de fabrica usando el botón “Factory settings” que permite configurar el proyecto a un estado utilizable de manera rápida.

A continuación se describen los pasos que pueden ser usados para configurar rápidamente un proyecto. Note que en la ficha General Options no se tiene un botón de seteo de valores por defecto.

1. Especifique el dispositivo o microcontrolador de pruebas (General Options → Target → Device).

2. Habilite un proyecto en lenguaje ensamblador o un proyecto que convine lenguaje C y lenguaje ensamblador (General Options → Target Assembler-only project).

3. Habilite la generación del archivo de salida ejecutable (General Options → Output → Output file → Executable).

4. Acepte los ajustes de fábrica para el compilador (C/C++ Compiler → Factory Settings).

5. Acepte los ajustes de fábrica para el ensamblador (Assembler → Factory Settings).

6. Acepte los ajustes de fábrica para el enlazador (Linker → Factory Settings).

7. Acepte los ajustes de fábrica para el depurador C-SPY (Debugger → Factory Settings).

8. Escoja la opción de depuración en el microcontrolador (Debugger → Setup → Driver → FET Debugger).

9. Especifique el puerto activo usado para interconectarse con el FET, si es el depurador de puerto paralelo LPT1 escoja la opción FET Debugger → Setup → Connection → Texas Instruments LPT-IF o especifique el puerto USB (FET Debugger → Setup → Connection → Texas Instruments USB-IF) si se va a utilizar la herramienta de depuración eZ430-F2013.

2.4.2.4 Creando Un Proyecto Desde Cero

A continuación se indican las instrucciones paso a paso necesarias para crear un proyecto ya sea en lenguaje ensamblador o en lenguaje C desde cero, permitiendo luego descargar la aplicación y correrla en el MSP430F2013 usando la herramienta de depuración eZ430-F2013.

1. Inicie el ambiente de desarrollo Embedded Workbench Kickstart, haciendo clic en el Menú Inicio → Programas → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V4 → IAR Embedded Workbench.

2. Cree un nuevo archivo de texto (File → New → File).

3. Ingrese el código de programa en el archivo de texto. Use los archivos del tipo “.h” para simplificar el desarrollo del código de programa. KickStart está provisto de archivos que definen los registros y los nombres de los terminales para cada microcontrolador. Estos archivos pueden simplificar enormemente la tarea de desarrollo del código de programa. Estos archivos están ubicados en el directorio “C:\ Archivos de Programa\ Embedded Workbench 5.0\ 430\ inc”. Para incluir el archivo .h correspondiente al dispositivo de pruebas en el código escrito se utiliza la sintaxis #include “msp430xyyy.h”. Adicionalmente, los archivos .io430xxx.h son provistos y optimizados para ser incluidos en el código de programa cuando se usa lenguaje C.

4. Guarde el archivo de texto con el código de programa (File → Save). Es recomendable que los archivos del tipo ensamblador sean guardados con un sufijo ".s43" y los archivos del tipo C sean guardados con el sufijo ".c".

5. Cree una nueva área de trabajo (File →New →Workspace).

6. Cree un nuevo proyecto (Project → Create New Project). Seleccione la opción Tool chain y a continuación MSP430, luego en Project Templates seleccione Empty project y de un click en OK. Especifique un nombre de proyecto y de clic en "Save".

7. Añada el archivo de texto que contiene el código de programa al proyecto (Project → Add Files). Seleccione el archivo correspondiente y de un clic en "Open". Alternativamente, puede hacer doble clic en el archivo para añadirlo al proyecto. Para añadir archivos de lenguaje ensamblador al proyecto debe cambiar el tipo de archivo presentado por default en el dialogo “Add Files” que es “C/C++”. Para ver archivos del tipo ensamblador “.s43”, seleccione “Assembler Files” en el menú contextual “Files of type”.

8. Guarde toda el área de trabajo (File → Save Workspace). Especifique un nombre para el área de trabajo y de un clic en "Save".

9. Configure ahora las opciones del proyecto (Project → Options). Para cada una de estas sub categorías: General Options, C/C++ Compiler, Assembler, Linker, Debugger, aceptar los valores por defecto con las siguientes excepciones:
   - Especifique el tipo de microcontrolador o dispositivo de pruebas (General Options → Target → Device).
   - Habilite el tipo de proyecto ya sea solo en lenguaje ensamblador o en lenguaje C y ensamblador combinados (General Options → Target → Assembler-only project).
   - Habilite la generación del archivo de salida ejecutable (General Options → Output → Output file → Executable).
   - Para realizar la depuración sobre el FET (por ejemplo el MSP430), de un clic en el menú Debugger → Setup → Driver → FET Debugger.
   - Especifique el puerto que se usara para interactuar con el FET (FET Debugger → Setup → Connection).

10. Compile el proyecto (Project → Rebuild All).

11. Depure la aplicación creada usando el C-SPY (Project → Debug). Esto inicia el depurador C-SPY que toma el control del dispositivo de pruebas, borra su memoria, luego la programa con la aplicación, y finalmente reinicia el microcontrolador o dispositivo de pruebas.

    En el anexo B.7 se encuentra la lista de las preguntas más frecuentes, en la sección A.3 en el FAQ de Debugging se pueden ver casos en que C-SPY no pudiera comunicarse con el dispositivo y que se debe hacer en los mismos.

12. Para iniciar la aplicación haga clic en Debug → Go.

13. Para detener la aplicación haga clic en Debug → Stop Debugging, esto detiene el depurador C-SPY saliendo del mismo y permite regresar al Workbench.

14. Si desea salir del Workbench haga clic en File → Exit.

2.4.2.5 Administración de la pila "stack" del microcontrolador y los archivos .xcl

El tamaño que se reserva para la pila puede ser cambiado a través del cuadro de diálogo que se encuentra en las opciones del proyecto (General Options → Stack/Heap) o a través de la modificación directa de los archivos de control de enlazador ".xcl". Estos archivos son introducidos al enlazador y contienen sentencias que controlan la asignación de memoria del dispositivo (memoria RAM, memoria flash). Para mayor información sobre los archivos ".xcl" vea la documentación XLINK de IAR. Los archivos ".xcl" se suministran con el FET y se encuentran en el directorio C:\ Archivos de Programa\ Embedded Workbench 5.0\ 430\ config\ lnk430xxxx.xcl. En este archivo se define un segmento re-localizable (RSEG) llamado CSTACK. CSTACK es usado para definir la región de la memoria RAM que será usada para la pila de sistema dentro de programas escritos en lenguaje C. El segmento CSTACK también puede ser usado en programas escritos en lenguaje ensamblador (`MOV.W #SFE(CSTACK), SP`). CSTACK está definido para extenderse desde la última ubicación de la memoria RAM por un espacio de 50 bytes.

Existen otras sentencias en el archivo ".xcl" que definen otras regiones de memoria re-localizables que son asignadas desde la primera ubicación de la memoria RAM hacia la parte inferior de la pila. Es crítico notar lo siguiente:

- Los archivos ".xcl" proporcionados con el FET reservan 50 bytes de la memoria RAM para la pila, indistintamente si esta cantidad de espacio de pila es requerido en realidad o si es suficiente para la aplicación.

- No existe ninguna rutina de comprobación de la pila. La pila puede rebosar los 50 bytes reservados y posiblemente sobrescribir los otros segmentos de memoria. Ningún error es presentado en caso de que esto suceda.

Además, los archivos ".xcl" proporcionados pueden ser modificados para cambiar el tamaño de la pila de acuerdo a las necesidades de la aplicación. Para esto modifique la variable D_STACK_SIZE=xx que permite reservar xx bytes para la pila. Note que el archivo ".xcl" también reserva 50 bytes para la pila denominada "heap" en caso de ser requerido, por ejemplo, por la función "malloc()".

2.4.2.6 Cómo generar el archivo .txt de Texas Instruments (y otros formatos de archivo)

El enlazador del Kickstart puede ser configurado para generar archivos en el formato ".txt" de TI para usarlos con los programadores GANG430 y PRGS430. Si desea cambiar el formato, simplemente haga clic en el menú Project → Options → Linker → Output → Format → Other → msp430-txt. Los formatos para Intel™ y Motorola™ también son soportados y pueden ser seleccionados.

Para mayor información, consulte en el anexo B.7, en la sección A.2 del FAQ, bajo el subtítulo de "Program Development".

2.4.2.7 Visión general de los programas de ejemplo

Los programas de ejemplo para los dispositivos MSP430 se encuentran en el directorio C:\ Archivos de Programa\ Embedded Workbench 5.0\ 430\ FET_examples. Cada carpeta contiene códigos de ejemplo en lenguaje ensamblador y lenguaje C.

El archivo denominado "Flashing the LED.eww" organiza el código de demostración convenientemente en un solo espacio de trabajo. Este contiene el código de programa en lenguaje ensamblador y en lenguaje C para cada una

de las familias de dispositivo MSP430. Las versiones de depuración y “release” son proporcionadas en cada uno de los proyectos.

El archivo “contents.htm” organiza y documenta convenientemente los ejemplos que se encuentran en dicha carpeta.

Existen ejemplos adicionales que pueden ser encontrados en la página de inicio del MSP430 bajo el nombre de “Code Examples”.

Tome en cuenta que algunos ejemplos requieren de un cristal oscilador externo de 32KHz conectado en el terminal LXFT1, y que no todos los FETs están provistos de este terminal.

2.4.2.8 Como usar C-SPY

En el anexo B.8 se encuentra una lista completa y la descripción de cada uno de los menús específicos del FET. Aquí se analizara las principales funciones del depurador C-SPY, comenzando por los puntos de interrupción.

2.4.2.8.1 Tipos de puntos de interrupción

El mecanismo de punto de interrupción usa un número limitado de recursos de depuración integrado en el dispositivo, específicamente, n registros de punto de parada como se indica en la Tabla 2.8 Cuando se setean n o menos puntos de parada, la aplicación avanza a la velocidad máxima de dispositivo, es decir, ejecuta el código en tiempo real. Cuando se utiliza mas de los n puntos de parada disponibles en hardware y está activado el uso de puntos de parada virtuales (FET Debugger $\rightarrow$ Breakpoints $\rightarrow$ Use virtual breakpoints), la aplicación se ejecuta con el control del PC anfitrión. El sistema funciona a una velocidad mucho menor pero brinda puntos de parada de software ilimitados, es decir, no se ejecuta la aplicación en tiempo real. Durante este modo, el PC repetidamente ejecuta un paso en el microcontrolador e interroga el dispositivo después de cada operación para determinar si se llego o no a un punto de parada.

Tanto la dirección de memoria del código como el valor de una variable pueden ser usadas como puntos de parada. Ambos requieren de dos puntos de parada físicos del MSP430 cada uno.

JTAG de 4 hilos	*Spy-By-Wire de 2 hilos*	*# de breakpoints*	*Rango de breakpoints*	*Control de reloj*	*Secuenciador de estado*	*Seguidor de memoria*
X	**X**	**2**	**---**	**X**	**---**	**---**

--- No disponible X Disponible

Tabla 2.8: Número de “Breakpoints” en el microcontrolador MSP430-f20xx y otras características de emulación. (Fuente: Texas Instruments)

2.4.2.8.2 Usando los “Breakpoints” o puntos de parada

Si el depurador C- SPY es iniciado con más de los n puntos de parada disponibles en hardware y los puntos de parada virtuales están deshabilitados, un mensaje informa al usuario de que solamente n puntos de parada de tiempo real son posibles y uno o más de los puntos de parada restantes son desactivados. Note que el Workbench permite setear cualquier número de puntos de parada, sin considerar la configuración de los puntos de parada virtuales en C-SPY. Entonces si los puntos de parada virtuales están desactivados, un máximo de n puntos de parada pueden ser seteados dentro del C-SPY.

Para poder resetear el microcontrolador temporalmente se requiere de un punto de interrupción, esto si la opción Project → Options → Debugger → Setup → Run To está activada. Para mayor información revise en el anexo B.7, en la sección de FAQ, Debugging #32.

El comando "Run To Cursor " (Ejecutar hasta el cursor) requiere de un punto de parada temporalmente. Por consiguiente, solamente n - 1 puntos de parada puede estar activos cuando se ejecuta este comando, esto si los puntos de parada virtuales se encuentran desactivados. Mayor información se encuentra

en el anexo B.7, en la sección FAQ, Debugging #33. Entonces, si mientras se procesa un punto de parada, una interrupción se activa, el depurador C-SPY se detiene en la primera instrucción de la rutina de servicio de la interrupción. Sobre este tema, se puede ver más información en el anexo B.7, en la sección FAQ, Debugging #26.

2.4.2.8.3 Usando “Single Step”

Cuando se depura un programa escrito en lenguaje ensamblador:

- Los comandos "Step Over", "Step Out", y "Next Statement" funcionan de la misma manera que "Step Into"; esto es, la instrucción en curso es ejecutada a la máxima velocidad.

- Una operación de un solo paso de una instrucción "CALL" (llamado a subrutina), detiene la ejecución en la primera instrucción de la función solicitada.

- Para realizar el comando Step Over en una llamada de función "CALL" y permitir que se ejecute este comando a la máxima velocidad disponible, se reemplaza este comando, primero ubicando un punto de parada después de la instrucción "CALL" y luego usando el comando GO que permite ir al punto de interrupción en tiempo real.

Cuando se depura un programa escrito en lenguaje C:

- El comando “Single Step” permite ejecutar la próxima instrucción presente en el código de programa. Por lo tanto, es posible ejecutar “Single Step” sobre una referencia de función. Si es posible, un punto de parada en el dispositivo es ubicado después de la referencia de función, y finalmente un se ejecuta el comando GO de manera implícita. Esto causa que la función sea ejecutada en tiempo real. Si ningún punto de parada en hardware está disponible, La función es ejecutada con el control del PC, es decir a menor velocidad.

- El comando “Step Into” es soportado.

- El comando “Step Out” es soportado.

Cuando se trabaja dentro del Modo Desensamblado “Disassembly mode” (View → Disassembly):

- Una operación “Single Step” en una instrucción que no sea CALL ejecuta la instrucción a la velocidad máxima del dispositivo, es decir se ejecuta en tiempo real.

- Una operación “Single Step” en una instrucción que sea CALL coloca, de ser posible, un punto de parada físico en el microcontrolador después de la instrucción de llamada, y a continuación ejecutara el comando GO. La función requerida es ejecutada en tiempo real. Si ningún punto de parada físico en el microcontrolador está disponible antes de ejecutar el comando GO, la función se ejecuta con la ayuda del PC. En cualquiera de los dos casos, la ejecución del programa se detiene en la instrucción siguiente a la instrucción CALL.

Es posible cambiar a “Single Step” solamente cuando las declaraciones de código están presentes. Los puntos de parada deben ser usados cuando se ejecuta una aplicación en la cual no exista cierto código fuente, por ejemplo, poner el punto de parada después de una instrucción CALL que ejecute una función para la que no exista código fuente, y luego ejecutar el comando GO hasta llegar al punto de parada en tiempo real. Si, durante una operación “Single Step”, una interrupción es activada, la instrucción en curso es finalizada y el depurador C-SPY se detiene ante la primera instrucción de la rutina de servicio de interrupción. En el anexo B.7, en la sección FAQ, Debugging #26, se puede encontrar mayor detalle.

2.4.2.8.4 Como usar la ventana de vigilancia de variables "Watch Windows"

El mecanismo de vigilancia de variables del depurador C-SPY permite que las variables en C sean monitoreadas durante la sesión de depuración. Aunque originalmente no fue diseñado para hacerlo, el mecanismo de vigilancia de variables puede ser extendido para monitorear variables del lenguaje ensamblador.

Suponga que las variables a vigilar están definidas en la RAM, por ejemplo:

```
RSEG DATA16_I       ; Segmento de memoria RAM
varword ds 2        ; 2 bytes por palabra.
varchar ds 1        ; 1 byte por carácter.
```

En el depurador C-SPY:

1. Abra la ventana de vigilancia de variables (View → Watch).
2. Haga clic en el menú Debug → Quick Watch.
3. Para vigilar una variable del tipo "varword", escriba en la descripción (Expression box): (__data16 unsigned int *) varword
4. Para vigilar una variable del tipo "varchar", escriba en la descripción (Expression box): (__data16 unsigned char *) varchar
5. Haga clic en el botón "Add Watch".
6. Cierre la ventana de vigilancia rápida "Quick Watch".
7. Para la entrada creada, haga clic en el símbolo "+" para exhibir el contenido (o el valor) de la variable vigilada.

Para cambiar el formato de la variable exhibida (por: default, binario, octal, decimal, hexadecimal, carácter), seleccione la variable deseada, haga clic con el botón derecho del mouse y luego seleccione el formato deseado. El valor de la variable exhibida puede ser cambiado seleccionándolo, y luego ingresando el nuevo valor.

En el lenguaje C, las variables pueden ser vigiladas seleccionándolas y luego arrastrándolas y soltándolas en la ventana de vigilancia de variables.

Ya que los dispositivos periféricos del MSP430 usan mapeo de memoria, es posible extender el concepto de vigilar variables a vigilar dispositivos periféricos. Sea cuidadoso ya que pueden provocarse efectos no deseados cuando los dispositivos periféricos son leídos o escritos por el depurador C-SPY. Para mayor información, consulte el anexo B.7, sección FAQ, Debugging #24.

Los registros del núcleo del CPU pueden ser especificados para vigilancia precediendo su nombre con "#", por ejemplo: #PC, #SR, #SP, #R5.

Las variables vigiladas dentro de la ventana son actualizadas únicamente cuando el depurador C-SPY obtiene el control del microcontrolador, por ejemplo al alcanzar un punto de parada o al ejecutar el comando “Single Step”.

Aunque los registros pueden ser monitoreados usando la ventana de vigilancia de variables, usar la ventana específica para visualizar los registros (View → Register) es el método más recomendado.

CAPITULO 3

CAPITULO 3. CONSTRUCCIÓN Y PRUEBAS

3.1 DESARROLLO DEL PROTOCOLO DE PRUEBAS

3.1.1 DESCRIPCIÓN GENERAL

El protocolo de pruebas implementado para este proyecto tiene como propósito verificar la funcionalidad del prototipo y sus principales características. La verificación inicial se llevará a cabo mientras el prototipo se encuentre implementado en protoboard, una vez comprobado su funcionalidad al igual que la funcionalidad del programa para el PC, se procederá a la construcción del prototipo diseñando los respectivos circuitos impresos y montando todo el sistema en su respectiva caja. Para finalizar, se comprobará que la construcción del prototipo no tenga problemas. Todos los resultados obtenidos serán organizados en el subcapítulo Resultados Experimentales.

3.1.1.1 Pruebas del prototipo, hardware y software

Las pruebas del prototipo comprenden la verificación del funcionamiento de cada uno de los módulos que lo componen con su respectivo código de programa. En el presente caso, existen 4 módulos, 3 de ellos utilizan la comunicación I2C en modo esclavo mientras que el dispositivo llamado “Entradas” trabaja en modo I2C Máster. Debido al tipo de comunicación y a la forma como están configurados, las pruebas deben realizarse con todos los módulos interconectados, caso contrario el prototipo no funcionará correctamente.

Los módulos han sido desarrollados pensando en la implementación de futuras soluciones, lo que permite reutilizarlos en diferentes diseños siempre y cuando se utilicen los protocolos de comunicación I2C implementados en cada uno de ellos y que se encuentran en el anexo E.1. Algunas pruebas se desarrollarán con un dispositivo I2C Máster diferente al dispositivo llamado “Entradas” debido

a que este no utiliza todas las funcionalidades que ofrecen los diferentes esclavos.

Las pruebas a llevarse a cabo en cada uno de los módulos son las siguientes:

- Modulo Display:
  - Inicialización del esclavo “Display”
  - Inicialización del Display conectado al modulo
  - Control de encendido /apagado de la luz de BackLight del Display
  - Respuesta a los comandos I2C enviados por el dispositivo Máster

- Modulo Teclado
  - Inicialización del esclavo “Teclado”
  - Barrido/Escaneo del teclado
  - Identificación de las teclas presionadas
  - Respuesta a los comandos I2C enviados por el dispositivo Máster

- Modulo Salidas
  - Inicialización del modulo “Salidas”
  - Inicialización del estado de la memoria Flash
  - Activación de las salidas
  - Manejo de la memoria Flash para almacenamiento de datos
  - Respuesta a los comandos I2C enviados por el dispositivo Máster

- Modulo Entradas
  - Comunicación RS-232 con el PC
  - Selección de entrada
  - Conversión Analógica / Digital
  - Espacio disponible en la memoria del dispositivo para la implementación del programa del usuario.
  - Manejo de la comunicación I2C

- Prototipo en general

- Envío de datos AL PC
- Envió de la memoria Flash al PC
- Recepción del estado de las salidas (Sobre escritura)
- Cambio de mensaje en el Display al existir enlace con el PC

3.1.1.2 Pruebas del programa diseñado para la interacción usuario - prototipo

Las pruebas que se realizaran al programa de interacción Usuario –Prototipo tienen como objetivo verificar el correcto funcionamiento de cada una de las opciones que el programa presenta al usuario, se verificará desde la creación de un nuevo documento que almacene la interacción, hasta ver si es posible salir de la aplicación sin guardar algún cambio en particular, lo que evita perder cambios que pueden resultar de vital importancia para el usuario.

Las pruebas verificarán que el funcionamiento del programa sea normal y que el usuario no tenga ningún inconveniente en utilizarlo, ya sea interactuando con el prototipo o simplemente accediendo al contenido de la memoria flash del mismo.

Las pruebas que se aplicarán a la interfaz del usuario para verificar que cualquier posibilidad de error estén contemplados son:

- Configuraciones del programa, mensajes de error y advertencias al usuario
- Comunicación con el prototipo, retardos y posibles desconexiones
- Interpretación de los datos recibidos y presentación de los mismos al usuario
- Funcionamiento de las opciones configuradas por parte del usuario para la interacción del programa con el prototipo.
- Almacenamiento en archivos de los datos enviados por el prototipo hacia el PC.

3.2 IMPLEMENTACION DEL PROYECTO

La implementación del proyecto se efectuará en 3 etapas, a saber:

- Desarrollo del prototipo en Protoboard
- Ensamblado y construcción del prototipo
- Instalación del software creado para la interacción Usuario – Prototipo.

En la etapa de desarrollo también se ejecutó el diseño de los programas requeridos para el funcionamiento del mismo, estos programas están explicados en el capítulo 2 y corresponde al código requerido por cada módulo del prototipo y también al software que permite la interacción entre el PC y el Prototipo.

3.2.1 DESARROLLO DEL PROTOTIPO EN PROTOBOARD

La implementación del prototipo en protoboard permite verificar posibles problemas y errores tanto en las conexiones como en el código de programa a instalarse en cada uno de los microcontroladores.

Siguiendo los diagramas esquemáticos presentados en el capítulo 1, se presenta en la figura 3.1 una imagen fotográfica del prototipo armado en protoboard y funcionando. La fuente de alimentación se encuentra en una protoboard más pequeña pero junto al proyecto principal.

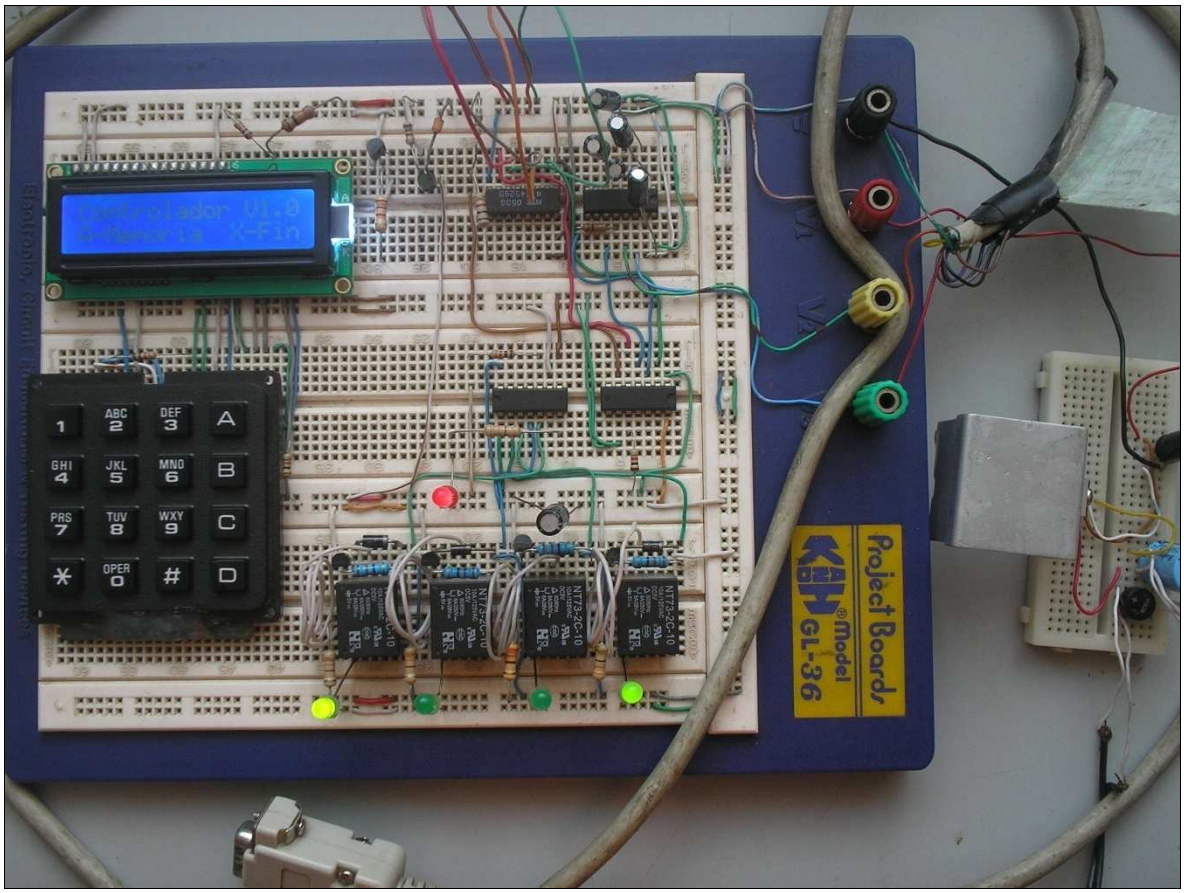


Figura 3.1: Imagen fotográfica del prototipo armado en protoboard.

Como se puede observar, no se ha utilizado el microcontrolador provisto en la herramienta EZ430-F2013, en su lugar se ha utilizado dispositivos de la misma familia (MSP430-F201X) cuyo encapsulado es de diferente presentación.

La programación de estos dispositivos se la realiza con la misma herramienta EZ430-F2013, lo que cambia es el conector con el que se los puede programar. En la figura 3.2 se muestra una fotografía con el programador desarrollado para este tipo de encapsulado incluido el programador original EZ430-F2013 y el microcontrolador del fabricante TI. En el anexo F.1 se incluye el diagrama esquemático que se utilizó para crear el programador.

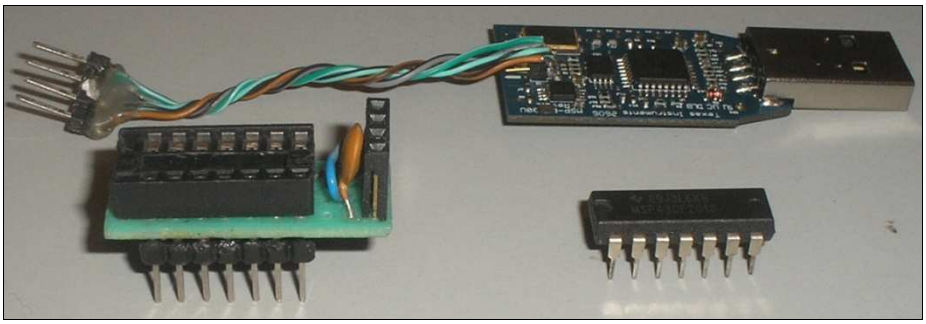


Figura 3.2: Fotografía en la que se muestra el programador desarrollado para este tipo de encapsulado incluido el programador EZ430-F2013 y el microcontrolador del fabricante TI.

3.2.2 ENSAMBLADO Y CONSTRUCCIÓN DEL PROTOTIPO

3.2.2.1 Diseño de los circuitos impresos

Para ejecutar la construcción del prototipo se requiere del diseño de los circuitos impresos, estos diseños fueron desarrollados con el programa Altium Designer versión Winter 09. El diseño del circuito impreso de cada modulo desarrollado anteriormente ha sido diseñado por separado y su interconexión se la realizara al finalizar la construcción. En las figuras 3.4, 3.5, 3.6 y 3.7 se muestran los gráficos con cada uno de los diseños que serán implementados para cada uno de los módulos que integran el prototipo. La fuente de alimentación será implementada en un circuito impreso con perforaciones previamente fabricadas. En el CD que complementa este proyecto se encuentra el directorio “PCB” que contiene todos los archivos creados dentro del programa Altium Designer y que permiten la impresión de todos los circuitos impresos aquí mencionados.

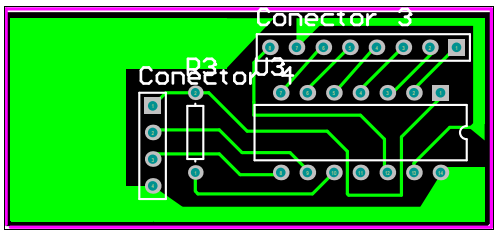


Figura 3.4: Diseño del circuito impreso para el modulo Teclado. (Elaborado por el autor).

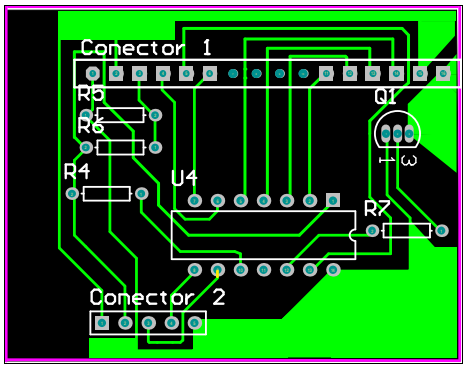


Figura 3.5: Diseño del circuito impreso para el modulo Display. (Elaborado por el autor).

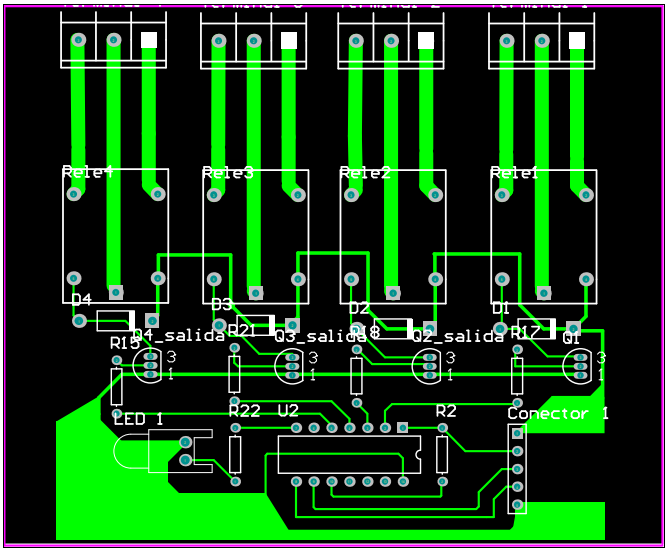


Figura 3.6: Diseño del circuito impreso para el modulo Salidas. (Elaborado por el autor).

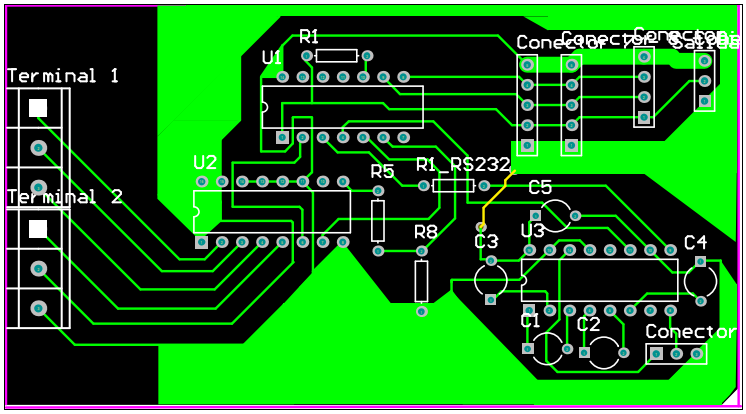


Figura 3.7: Diseño del circuito impreso para el modulo Entradas. (Elaborado por el autor).

3.2.2.2 Elaboración de los circuitos impresos

Para la elaboración de cada uno de los circuitos impresos se utilizó papel de transferencia térmica para plasmar el grafico del circuito, mientras que para el grabado químico de las plaquetas de circuito impreso se utilizó “cloruro de hierro (III)” o “cloruro férrico”. En la figura 3.8 se muestra una foto con los circuitos impresos listos para ser atacados por el cloruro de hierro (III). Estos diseños fueron implementados usando “Papel de transferencia térmica” y posteriormente corrigiendo las respectivas fallas manualmente con un marcador indeleble.

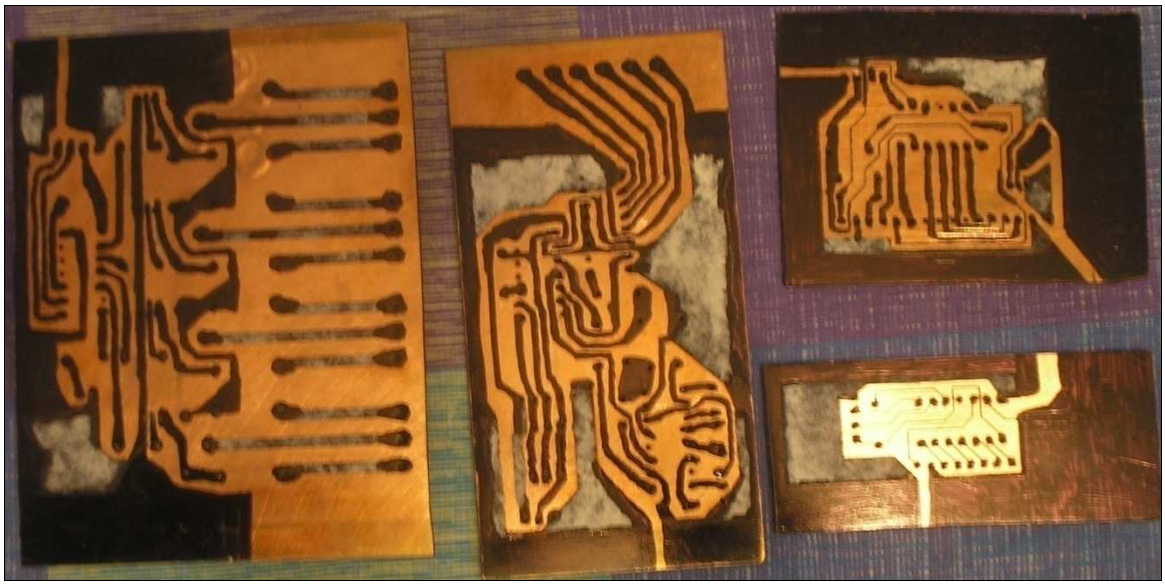


Figura 3.8: Foto con los circuitos impresos listos para ser atacados por el “cloruro de hierro (III)”.

Una vez finalizado el “ataque” a las plaquetas, se obtienen los circuitos impresos listos para la ubicación de los elementos correspondientes. En la figura 3.9 se muestra una fotografía con los circuitos impresos listos para su soldado.

Figura 3.9: Circuitos impresos listos para la ubicación de elementos.

3.2.2.3 Soldado de los elementos y conexión de los módulos

El soldado de los elementos se lo realiza siguiendo los diagramas esquemáticos utilizados para el diseño de los circuitos impresos. Para evitar que las pistas de cobre se oxiden, se las estaña completamente. En la figura 3.9 se muestran las soldaduras realizadas a los elementos de cada modulo además de las pistas totalmente estañadas, y en la figura 3.10 se muestran los elementos en sus respectivas ubicaciones.

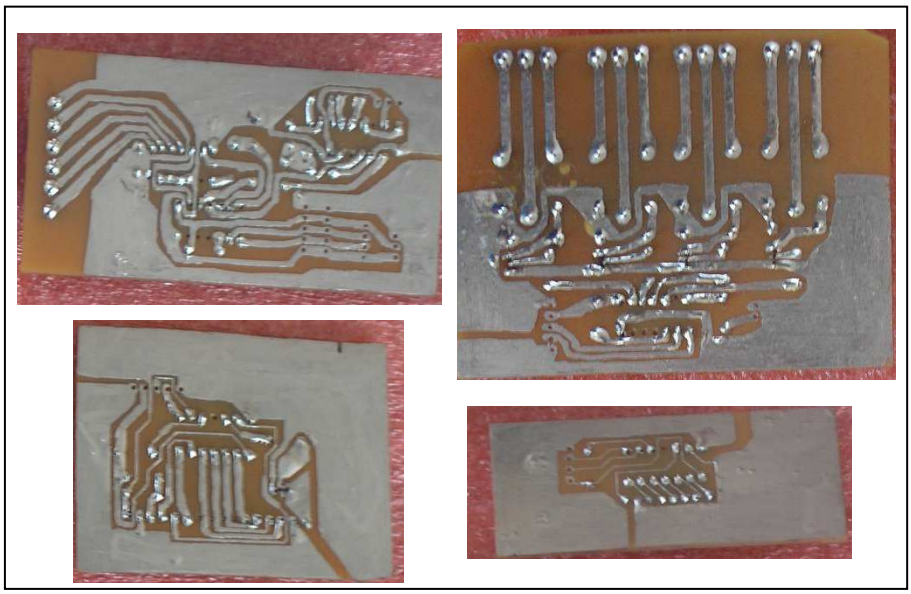


Figura 3.9: Soldaduras de los elementos.

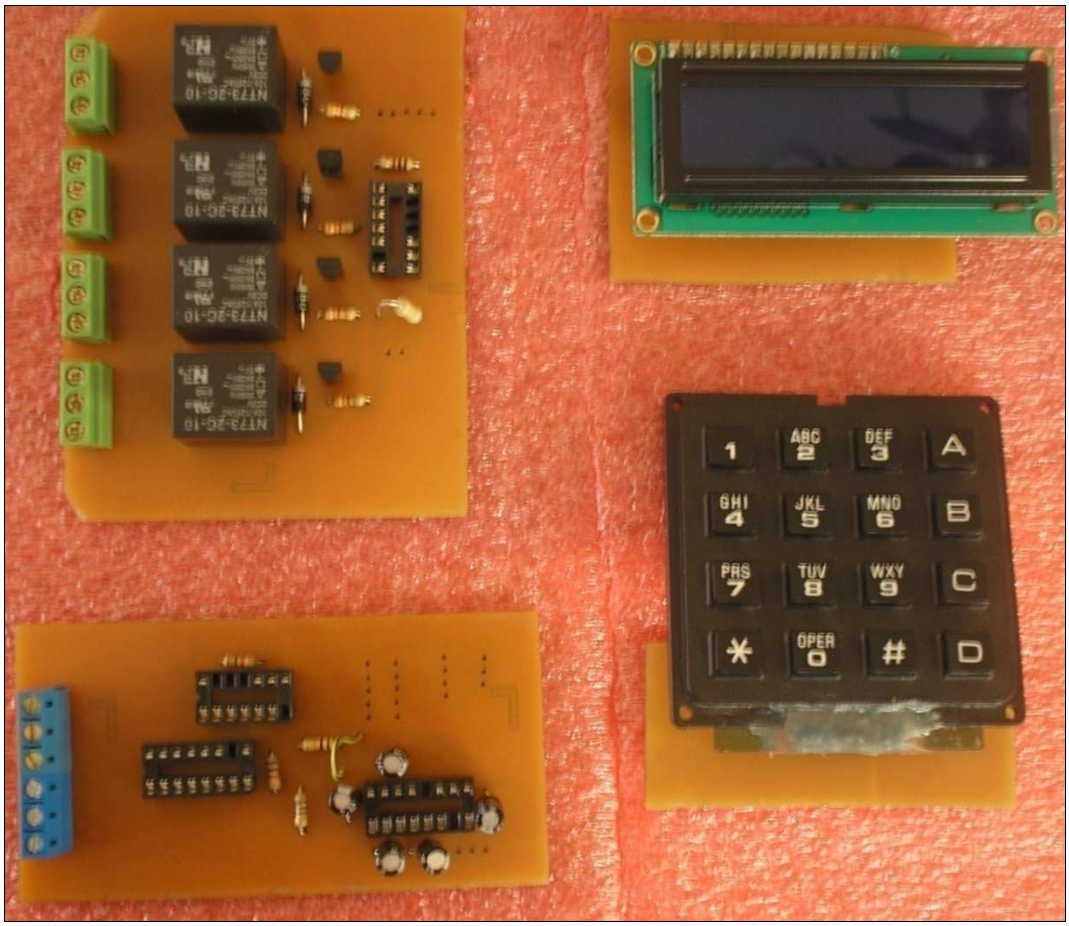


Figura 3.10: Ubicación de los elementos.

En los diseños realizados para cada uno de los circuitos impresos se anticipó la necesidad de interconectar los módulos entre sí, es por esto que el circuito impreso correspondiente al modulo “Entradas”, es el encargado de soportar todas las interconexiones necesarias de los módulos. En la figura 3.11 se muestra la interconexión entre todos los módulos del prototipo incluido la fuente de alimentación, mientras se está probando su funcionalidad.

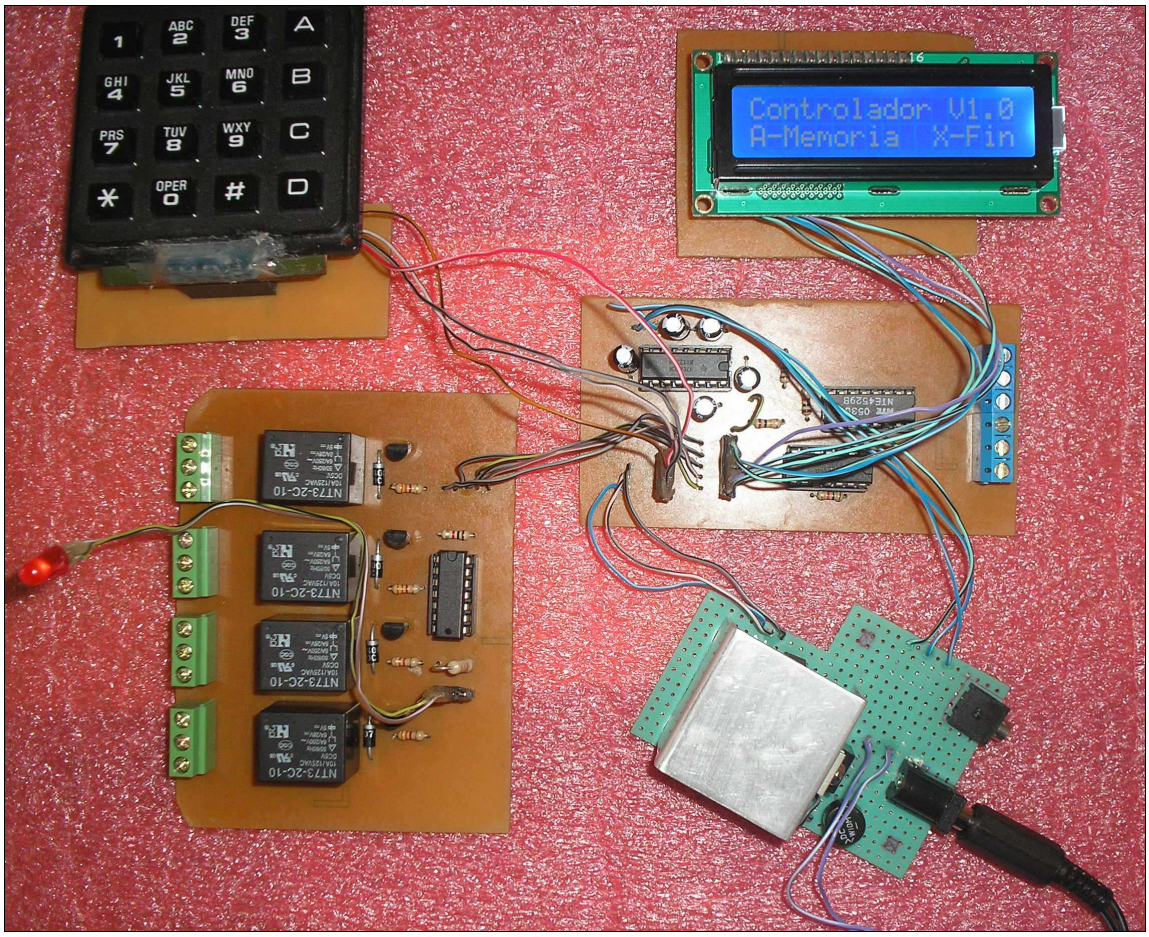


Figura 3.11: Interconexión de todos los módulos y pruebas de funcionamiento.

3.2.2.4 Diseño de la carcasa del prototipo

La carcasa o caja del prototipo será fabricada en madera, pero el diseño podrá ser replicado tanto en metal como en plástico. El diseño inicial será realizado sobre cartón, ya que este material permite su fácil manipulación y puede soportar los dispositivos construidos de manera temporal, permitiendo observar el resultado final del diseño con relativa facilidad. En la figura 3.12 se muestra una fotografía con el diseño en cartón de la carcasa a construirse, incluido los cables externos, y en la figura 3.13 se muestra una fotografía del prototipo finalizado.

Figura 3.12: Diseño en cartón de la carcasa

Figura 3.13: Prototipo finalizado.

3.2.3 INSTALACIÓN DEL SOFTWARE CREADO PARA LA INTERACCIÓN USUARIO – PROTOTIPO

A fin de lograr la instalación del software requerido para la interacción del usuario con el prototipo, en el CD que acompaña este proyecto se incluye la carpeta “Instaladores\Interfaz usuario”, dicha carpeta contiene el archivo “Interfaz.exe” que permitirá ejecutar la instalación del programa mencionado.

Para poder ejecutar este programa y permitir la comunicación con el prototipo, los siguientes requisitos deben ser cumplidos:

- Disponer de por lo menos 100 MB de espacio disponible en disco duro

- Tener un puerto serial disponible y correctamente configurado (Se puede utilizar un conversor USB – RS-232)

- Correr en el PC uno de los siguientes sistemas Operativos: Windows 2000 Service Pack 3; Windows 98; Windows 98 Second Edition; Windows ME; Windows Server 2003; Windows XP Service Pack 3, Windows Vista SP1

- Tener instalado Microsoft .NET Framework, versión 2.0. Este paquete de instalación se incluye en el CD anexo al proyecto en el directorio “Instaladores\extras” con el nombre “dotnetfx.exe”

Además, para permitir la programación del microcontrolador con el código de usuario especifico para cada aplicación, debe ejecutarse el programa “FET_R513.exe” que corresponde al Instalador del “IAR Embedded Workbench for MSP430 V4.11A Kickstart”, este archivo se encuentra en el CD adjunto al proyecto en la carpeta “Instaladores\Programador TI”. Los requisitos de esta herramienta están descritos en el capítulo 2. EL código fuente que permite realizar todos los cambios necesarios al programa del usuario y que debe editarse utilizando el programa antes mencionado se encuentra en la carpeta “Codigo Fuente\Tesis Micro TI V1.0”

Para mayor información, en el capítulo 2 se detalla la utilización del programa de interacción usuario – prototipo, al igual que el uso de la herramienta para la programación del microcontrolador.

3.3 RESULTADOS EXPERIMENTALES

Al igual que en el capítulo construcción y pruebas, los resultados experimentales se han divido en 2 etapas, cada una resume en su respectiva tabla el resultado obtenido por las pruebas realizadas.

La tabla 3.1 muestra la información respectiva a cada prueba y el resultado obtenido el experimentar sobre cada uno de los módulos y en la tabla 3.2 se indican los resultados al ejecutar el protocolo de pruebas del programa de interacción Usuario - Prototipo.

Nótese que existen comprobaciones que pueden calificarse únicamente como Satisfactoria o insatisfactoria mientras que otras se presentaran como una escala de aprobación, en aquellas que se precise entregar un valor concreto se incluirá la unidad de medida correspondiente.

Modulo	Tipo de prueba	Resultado	Observaciones
Display	Inicialización del esclavo “Display”	Correcto	Ninguna
Display	Inicialización del Display conectado al modulo	Correcto	Ninguna
Display	Control de encendido /apagado de la luz de BackLight del Display	Correcto	Comando no utilizado por el prototipo
Display	Respuesta a los comandos I2C enviados por el dispositivo Máster	Correcto	Algunos comandos no son utilizados por el prototipo
Teclado	Inicialización del esclavo “Teclado”	Correcto	Ninguna
Teclado	Barrido/Escaneo del teclado	Correcto	Soporta detección de varias teclas, esta característica no es utilizada por el prototipo
Teclado	Identificación de las teclas presionadas	Correcto	Ninguna
Teclado	Respuesta a los comandos I2C enviados por el dispositivo Máster	Correcto	Ninguna
Salidas	Inicialización del modulo “Salidas”	Correcto	Ninguna
Salidas	Inicialización del estado de la memoria Flash	Correcto	Ninguna
Salidas	Activación de las salidas	Correcto	Ninguna

Salidas	Manejo de la memoria Flash para almacenamiento de datos	Correcto	Ninguna
Salidas	Respuesta a los comandos I2C enviados por el dispositivo Máster	Correcto	Ninguna
Entradas	Comunicación RS-232 con el PC	Satisfactoria	Debido al procesamiento interno el prototipo puede no detectar el envió de información
Entradas	Selección de entrada	Correcto	Ninguna
Entradas	Conversión Analógica / Digital	Satisfactoria	Si la entrada es conectada directamente a GND (Tierra), se detecta una pequeña fluctuación en el valor convertido y no se entrega un valor "cero" como es lo esperado
Prototipo	Espacio disponible en la memoria del dispositivo para la implementación del programa del usuario.	Satisfactoria	La memoria disponible en cada microcontrolador es de aproximadamente 2KB, el código de programa principal debe incluirse dentro del dispositivo Master llamado Entradas, este dispositivo incluye la librería para la comunicación I2C Master entre otras secuencias de programa, lo que deja un espacio de memoria "reducido" para el código de programa de usuario
Prototipo	Manejo de la comunicación I2C	Correcto	Ninguna
Prototipo	Envío de datos AL PC	Correcto	Ninguna
Prototipo	Envió de la memoria Flash al PC	Correcto	Ninguna
Prototipo	Recepción del estado de las salidas (Sobre escritura)	Satisfactoria	Para asegura la recepción del comando de sobre escritura por parte del prototipo, el PC debe enviarlo por duplicado.
Prototipo	Cambio de mensaje en el Display al existir enlace con el PC	Correcto	Ninguna
Prototipo	Consumo Máximo de la fuente de 3.4V DC	10mA	Corriente consumida por los microcontroladores
Prototipo	Consumo total de corriente DC del prototipo (Máxima carga)	380mA	Ninguna
Prototipo	Consumo de corriente DC de los relés (activados todos al mismo tiempo)	240mA	Voltaje de alimentación de los Relés: 5V
Prototipo	Voltaje de alimentación	5V, 3.4V	Se requiere de 2 voltajes debido a que los microrontroladores pueden alimentarse con un voltaje máximo de 3.6 V

Prototipo	Consumo total de corriente AC del prototipo (Máxima carga)	616mA	Ninguna
Prototipo	Voltaje de entrada AC medido (Secundario Transformador)	12.99V	Ninguna

Tabla 3.1: Resultados obtenidos sobre el Prototipo. (Elaborada por el autor).

Tipo de prueba	Resultado	Observaciones
Configuraciones del programa, mensajes de error y advertencias al usuario	Satisfactorio	Los errores encontrados en el desarrollo del programa han sido controlados aunque no se descarta la posibilidad de encontrarse nuevos. Las advertencias al usuario que le indican el cierre de la aplicación sin guardar cambios, configuraciones equivocadas o incorrectas, entre otras, están desarrolladas correctamente.
Comunicación con el prototipo, retardos y posibles desconexiones	Satisfactorio	La comunicación entre el PC y el prototipo se realiza sin ningún problema, una vez que el prototipo logra recibir el código respectivo. El usuario al utilizar el modulo de manejo de memoria Flash debe a veces repetir la petición de "envío de memoria" o de "borrado de memoria" debido a que el microcontrolador no detecta el comando correspondiente.
Interpretación de los datos recibidos y presentación de los mismos al usuario	Correcto	Ninguna
Funcionamiento de las opciones configuradas por parte del usuario para la interacción del programa con el prototipo.	Satisfactorio	Las funciones descritas en el capítulo 2 están implementadas y son fácilmente accesibles al usuario, pero, el programa podría incluir mayores funcionalidades que le permitan controlar de mejor manera las salidas del prototipo, por ejemplo, mediante un programa de usuario específico que interprete el valor de cada entrada.
Almacenamiento en archivos de los datos enviados por el prototipo hacia el PC	Correcto	Ninguna

Tabla 3.2: Resultados obtenidos sobre el programa de interacción Usuario – Prototipo. (Elaborada por el autor).

CAPITULO 4

CAPITULO 4. CONCLUSIONES Y RECOMENDACIONES

La realización de este proyecto ha dejado diferentes conclusiones y recomendaciones:

4.1 CONCLUSIONES

- EL microcontrolador MSP430F2013 de acuerdo con el fabricante fue diseñado para ser usado en aplicaciones del tipo instrumental, es decir, para realizar mediciones y conversiones de señales, es por este motivo que tiene un bajo número de pines y que su consumo de energía es bajo. Este microcontrolador se ha adaptado perfectamente a este proyecto ya que se ha aprovechado la comunicación I2C disponible en el mismo.

- Al utilizar la conversión Análogo digital Sigma/Delta se puede aprovechar los 16 bits que se entregan como resultado de la conversión, esto permite tener un rango más amplio de valores y por tanto conseguir mayor precisión en el uso de sensores.

- El seleccionar las entradas analógicas utilizando el multiplexer ayuda a mantener el uso de un número bajo de pines, que de otra manera necesitaría utilizar las entradas análogas integradas en el microcontrolador, lo que limitaría el número de entradas disponibles.

- La ayuda prestada por la herramienta “Flash Emulation Tools” es muy importante porque permite verificar cualquier problema que se presente en el prototipo durante su desarrollo, facilitando su construcción e implementación.

- El prototipo trabaja de manera muy estable sin necesidad de estar conectado con el PC, lo que lo convierte en un dispositivo muy versátil al momento de aplicarlo en un proceso específico.

- El trabajo del prototipo en conjunto con el PC permite tener disponibles muchas más opciones al usuario para controlar un proceso. El programa desarrollado incluye herramientas para manipular los datos enviados desde el prototipo y así extender la funcionalidad del mismo.

- El manejo de la comunicación I2C puede resultar complejo, pero aplicando las librerías provistas por la empresa Texas Instruments y comprendiendo el modo de trabajo del protocolo en su aspecto general puede facilitar la construcción e implementación de cualquier diseño en forma modular.

- Los menús integrados en el prototipo son reducidos debido principalmente al espacio disponible en la memoria Flash de cada microcontrolador. Si se desea aumentar las funcionalidades debe utilizarse un dispositivo de la familia MSP430 que posea mayor memoria.

- La respuesta que el PC espera a continuación de que se haya enviado un comando al prototipo puede tener problemas debido a los retardos que sufre al procesar todas las subrutinas requeridas para ejecutar el código de usuario incluido las funciones propias del mismo. Es por este motivo que el PC debe enviar más de una petición para asegurar la recepción del comando por parte del prototipo.

4.2 RECOMENDACIONES

- Al implementar un diseño es recomendable informarse bien utilizando toda la información del fabricante, especialmente las hojas de errata, lo que permitirá aplicar de manera conveniente el microcontrolador y así evitar problemas en la implementación.

- Las entradas del conversor A/D sigma/delta del F2013, al usarse divisores de voltaje o similares, debe procurarse el uso de resistencias relativamente bajas (menores a 1000 ohmios), el uso de resistencias demasiado altas impide la circulación de la corriente mínima necesaria para el conversor,

esto provoca una lectura errónea del voltaje, es decir, a bajos voltajes y altas resistencias el conversor asume la escala máxima.

- El uso del conversor Sigma / Delta incorporado en el microcontrolador MSP430F2013 proporciona una mayor precisión de la señal de entrada debido a que utiliza 16 bits, es así que es recomendable usar dicho conversor que utilizar el conversor A/D incluido en el microcontrolador MSP430F2012 que es de 12 bits.

- Si se utiliza la herramienta de depuración mientras el diseño a verificarse tenga conectada su propia alimentación, debe evitarse que ambos voltajes sean suministrados al diseño al mismo tiempo. Es preferible desconectar la alimentación provista por la herramienta “Flash Emulation Tool” hacia el microcontrolador que se está depurando, de esta manera se evita posibles daños en la misma..

- Debido al tamaño de memoria Flash disponible en cada microcontrolador (2KB) y ya que se debe integrar la librería necesaria para la comunicación I2C entre otras sub-funciones, la memoria total disponible para el almacenamiento de datos en el prototipo es de 992 bytes. Si, en la memoria se almacenan los datos enviados en grupo de 2 bytes cada 10 minutos, el tiempo total (sin interrupciones) que puede almacenar datos el prototipo será de 3 días, 10 horas y 40 minutos, por lo que es recomendable extraer estos datos antes de que la memoria se llene.

- Dado que las características que poseen los microcontroladores de la familia MSP430F2012 y MSP430F2013 en lo que respecta al número de pines es limitada, es recomendable utilizar el protocolo I2C para intercomunicar los diferentes módulos que pueden constituir un diseño en especial y de esta manera permitir un mejor uso de cada uno de los pines.

- Para mayor detalle sobre cualquier aspecto del microcontrolador MSP430F2012 y MSP430F2013, se recomienda consultar los manuales provistos por la empresa Texas Instruments. Este siempre será un excelente punto de partida para cualquier diseño que se desee implementar

- Al utilizar el módulo de administración de memoria Flash, en caso de recibir una notificación desfavorable (Sin respuesta del prototipo), es recomendable realizar un nuevo intento hasta conseguir una respuesta favorable desde el prototipo.

- EL uso de los microcontroladores y de herramientas programáticas como Visual Studio se ha vuelto muy necesario en aplicaciones prácticas en el área técnica, por lo que es recomendable que los graduandos investiguen sobre estas tecnologías y las apliquen en proyectos similares.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

- BALENA FRANCESCO, Programación avanzada con Visual Basic 2005, México, McGraw-Hill, 2008

- JOYANES AGUILAR LUIS, Microsoft Visual C++ 1.5/2: Iniciación y referencia, Madrid, Mcgraw-Hill, 1996

- TEXAS INSTRUMENTS, MSP430x20x1, MSP430x20x2, MSP430x20x3 Mixed Signal Microcontroller
  SLAS491A.pdf

- TEXAS INSTRUMENTS, eZ430-F2013 Development Tool User's Guide
  SLAU176.pdf

- TEXAS INSTRUMENTS, MSP430x2xx Family User's Guide (Rev E)
  SLAU144e.pdf

- MAXIM, +5V-Powered, Multichannel RS-232 Drivers/Receivers
  MAX220-MAX249.pdf

- TEXAS INSTRUMENTS, MSP430x3xx User’s Guide
  slau012a.pdf

- TEXAS INSTRUMENTS, MSP430F20xx Device Erratasheet
  slaz026k.pdf

- TEXAS INSTRUMENTS, MSP-FET430 FLASH Emulation Tool (FET) (For use with IAR Workbench Version 3+) User's Guide (Rev. K)
  slau138k.pdf

- NXP, I2C-bus specification and user manual - Rev. 03 - 19 June 2007
  UM10204_3.pdf

SITIOS WEB

- WORLD INTELLECTUAL PROPERTY ORGANIZATION, Brown-out detector
  http://www.wipo.int/pctdb/en/wo.jsp?IA=IE2004000074&wo=2004107577&DISPLAY=STATUS

- TEXAS INSTRUMENTS, IAR Embedded Workbench Kickstart - Free IDE - IAR-KICKSTART - TI Tool Folder
  http://focus.ti.com/docs/toolsw/folders/print/iar-kickstart.html

- CURSO DE VISUAL BASIC, Guía del Estudiante Capitulo 1
  http://www.geocities.com/udea2000/visual.htm

- COLABORA.NET, Historia de VisualBasic .NET
  http://www.elguille.info/colabora/NET2005/Percynet_Historia_Visual_Basic_NET.htm

- WIKIPEDIA, LA ENCICLOPEDIA LIBRE, Microsoft Visual Studio
  http://es.wikipedia.org/wiki/Microsoft_Visual_Studio

- UCONTROL, Usando LCDs
  http://www.ucontrol.com.ar/wiki/index.php/Usando_LCDs

- TEXAS INSTRUMENTS, MSP430 Ultra-Low Power Microcontrollers User's Guides
  http://focus.ti.com/mcu/docs/mcuprodtechdoc.tsp?sectionId=95&tabId=1201&familyId=342&techDoc=6&documentCategoryId=6

GLOSARIO

GLOSARIO

ACLK: Auxiliary Clock (Reloj Auxiliar). ACLK puede ser escogido como LFXT1CLK o VLOCLK mediante software. ACLK está dividido en factores de 1, 2, 4, o 8 según se requiera. ACLK es escogido por software en módulos individuales periféricos.

Byte: Conjunto de 8 bits que representan un valor decimal de 0 a 255 o un carácter en general

DCOCLK: Digitally Controlled Oscillator Clock. Señal de reloj proveniente del oscilador controlado digitalmente que se encuentra integrado en el modulo básico de reloj

Display: Periférico de visualización de datos que permite mostrar información al usuario. Este término se usa para denominar a los visores de caracteres de pequeño tamaño

Deshabilitado: Estado en el cual un dispositivo esta desconectado o apagado. Si se trata de un bit, este tendrá el valor binario “0” o el nivel lógico necesario para “desactivar” la función asociada.

Habilitado: Estado en el cual un dispositivo está conectado o encendido. Si se trata de un bit, este tendrá el valor binario “1” o el nivel lógico necesario para “activar” la función asociada.

JEDEC: Asociación de Tecnología de Estado Sólido (Solid State Technology Association), antiguamente **J**oint **E**lectron **D**evice **E**ngineering **C**ouncil, es la rama de la **EIA** (Electronic Industries Alliance) para la estandarización de la ingeniería de semiconductores.

LFXT1CLK: Low / High Frecuency External Clock (Reloj externo de Baja/Alta frecuencia). Oscilador de baja/alta frecuencia que puede ser usado con cristales de reloj de vigilancia de frecuencia baja o puede utilizar fuentes de

reloj externas de 32,768 Hz, o puede utilizar cristales estándar, resonadores, o fuentes de reloj externas entre los 400 KHz hasta los 16 MHz

MCLK: Master Clock (Reloj Maestro). MCLK es escogido mediante software de una de estas fuentes: LFXT1CLK, VLOCLK, o DCOCLK. MCLK es dividido para 1, 2, 4, o 8 según sea necesario. MCLK es usado por la CPU y el sistema.

Microcontrolador: Dispositivo electrónico que contiene una CPU, memoria, módulos periféricos y permite ser programado para realizar tareas especificas.

Operando: Dato binario de 8 o 16 bits que se utiliza como parte de un comando dentro del código ensamblador en la programación de cada microcontrolador. Este dato puede servir para un operando de origen como para un operando de destino y depende de la instrucción utilizada

Oscilador: Dispositivo electrónico que genera un tren de pulsos a una frecuencia especifica

Pin: Patilla metálica presente en el microcontrolador. Por lo general establecen conexión con dispositivos o funciones internas del mismo

Periférico: Dispositivo que realiza la tarea específica para la cual fue diseñado. Por lo general debe interactuar con señales físicas provenientes del exterior o en otros casos debe generarlas.

Reset, reseteo, resetear: Acción de “poner en estado inicial” un dispositivo, o en el caso de un dato (bit, byte, word) configurarlo con el valor predeterminado, que por lo general es “0”.

Reinicialización: Conjunto de acciones que permiten al dispositivo volver a iniciar su funcionamiento. En el caso de un dato (bit, byte, word) funciona igual que realizar un Reset

RISC: Reduced Instruction Set Computer. Es una filosofía de diseño de CPU para computadoras y microcontroladores que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse.

Set, seteo, setear: Acción de “activar” un dispositivo, o en el caso de un dato (bit, byte, word) configurarlo con el valor que indique su estado de “activación”, que por lo general es “1” en el caso del bit.

SMCLK: Sub-main clock (Reloj subprincipal). SMCLK es configurable igual que MCLK, pero es usado para módulos individuales periféricos.

Spy-Bi-Wire: Protocolo de depuración que utiliza únicamente 2 “hilos” para la comunicación con el microcontrolador. Este protocolo fue desarrollado por la empresa Texas Instruments.

Tstg: Temperature Storage. Temperatura a la cual se puede almacenar el dispositivo.

VCC: Voltaje en corriente directa o voltaje de alimentación.

Vector de reset: Locación de memoria de 16 bits que almacena la dirección de memoria donde se encuentran las instrucciones a ejecutarse en caso de que el dispositivo reciba la orden de realizar un reset.

VFB: Voltaje Fuse Blow. Voltaje necesario para quemar el fusible de protección de la interfaz JTAG, también llamada Spy-bi-wire

VLOCLK: Very Low power Clock (Reloj de bajo consumo). Reloj de bajo consumo con frecuencia típica de 12 KHz.

VSS: Voltaje negativo de alimentación.

Word, palabra: Término utilizado para referirse a un dato de 16 bits de longitud.

ANEXO A

CIRCUITO DE BROWN-OUT

ANEXO A

A.1 CIRCUITO DE BROWN-OUT

(WO/2004/107577) BROWN-OUT DETECTOR

- Biblio. Data
- Description
- Claims
- National Phase
- Notices
- Documents

Latest bibliographic data on file with the International Bureau

-
  - Permanent Link
  - Bookmark this page

Pub. No.:	WO/2004/107577	**International Application No.:**	PCT/IE2004/000074
Publication Date:	09.12.2004	**International Filing Date:**	19.05.2004

IPC: ***H03K 17/14*** (2006.01), ***H03K 17/22*** (2006.01)

Applicants: **ANALOG DEVICES, INC.** [US/US]; Three Technology Way, P.O. Box 9106, Norwood, MA 02062-9106 (US) *(All Except US)*. **GUBBINS, David, P.** [IE/IE]; Mountshannon Road, Lisnagry, County Limerick (IE) *(US Only)*.

Inventor: **GUBBINS, David, P.**; Mountshannon Road, Lisnagry, County Limerick (IE).

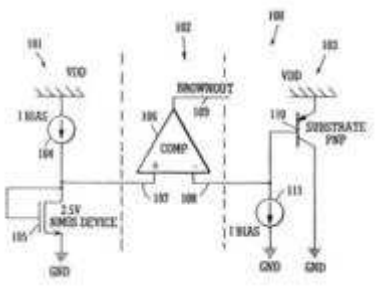
Agent: **SHORTT, Peter, Bernard**; Tomkins & Co., 5 Dartmouth Road, Dublin 6 (IE).

Priority Data: 10/452,856 02.06.2003 US

Title: BROWN-OUT DETECTOR

Abstract:

A brown-out detector that continuously monitors power supply voltage and provides an output signal that transitions to a logic HIGH state when the monitored power supply voltage exceeds a predetermined threshold



value. One embodiment of the present invention comprises a first voltage reference with respect to ground that varies in direct proportion to absolute temperature, a second voltage reference with respect to the supply voltage that varies inversely with absolute temperature, and a comparator having the first voltage reference coupled to one input, and the second voltage reference coupled to the other input, such that the comparator output changes state when the power supply voltage exceeds a predetermined threshold voltage that is relatively independent of absolute temperature. The circuit may also be configured such that the first voltage reference varies inversely with absolute temperature, while the second voltage reference varies in direct proportion to absolute temperature.

Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
African Regional Intellectual Property Org. (ARIPO) (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW)
Eurasian Patent Organization (EAPO) (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM)
European Patent Office (EPO) (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR)
African Intellectual Property Organization (OAPI) (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publication Language: English (EN)

Filing Language: English (EN)

ANEXO B

HOJAS DE DATOS, DESCRIPCIÓN Y DETALLES TÉCNICOS DEL MICROCONTROLADOR MSP430F2013

ANEXO B

HOJAS DE DATOS, DESCRIPCIÓN Y DETALLES TÉCNICOS DEL MICROCONTROLADOR MSP430F2013

B.1 FUNCIÓN DE LOS TERMINALES DEL MSP430X20X3

Terminal Functions, MSP430x20x2 (Continued)

TERMINAL				DESCRIPTION
NAME	PW, or N NO.	RSA NO.	I/O	
DV_{CC}	NA	16		Digital supply voltage
AV_{CC}	NA	15		Analog supply voltage
DV_{SS}	NA	14		Digital ground reference
AV_{SS}	NA	13		Analog ground reference
QFN Pad	NA	Package Pad	NA	QFN package pad connection to V_{SS} recommended.

† TDO or TDI is selected via JTAG instruction.

NOTE: If XOUT/P2.7 is used as an input, excess current will flow until P2SEL.7 is cleared. This is due to the oscillator output driver connection to this pad after reset.

Terminal Functions, MSP430x20x3

TERMINAL				DESCRIPTION
NAME	PW, or N NO.	RSA NO.	I/O	
P1.0/TACLK/ACLK/A0+	2	1	I/O	General-purpose digital I/O pin Timer_A, clock signal TACLK input ACLK signal ouput SD16_A positive analog input A0
P1.1/TA0/A0−/A4+	3	2	I/O	General-purpose digital I/O pin Timer_A, capture: CCI0A input, compare: Out0 output SD16_A negative analog input A0 SD16_A positive analog input A4
P1.2/TA1/A1+/A4−	4	3	I/O	General-purpose digital I/O pin Timer_A, capture: CCI1A input, compare: Out1 output SD16_A positive analog input A1 SD16_A negative analog input A4
P1.3/VREF/A1−	5	4	I/O	General-purpose digital I/O pin Input for an external reference voltage/internal reference voltage output (can be used as mid-voltage) SD16_A negative analog input A1
P1.4/SMCLK/A2+/TCK	6	5	I/O	General-purpose digital I/O pin SMCLK signal output SD16_A positive analog input A2 JTAG test clock, input terminal for device programming and test
P1.5/TA0/A2−/SCLK/TMS	7	6	I/O	General-purpose digital I/O pin Timer_A, compare: Out0 output SD16_A negative analog input A2 USI: external clock input in SPI or I2C mode; clock output in SPI mode JTAG test mode select, input terminal for device programming and test
P1.6/TA1/A3+/SDO/SCL/TDI/TCLK	8	7	I/O	General-purpose digital I/O pin Timer_A, capture: CCI1B input, compare: Out1 output SD16_A positive analog input A3 USI: Data output in SPI mode; I2C clock in I2C mode JTAG test data input or test clock input during programming and test
P1.7/A3−/SDI/SDA/TDO/TDI†	9	8	I/O	General-purpose digital I/O pin SD16_A negative analog input A3 USI: Data input in SPI mode; I2C data in I2C mode JTAG test data output terminal or test data input during programming and test

Terminal Functions, MSP430x20x3 (Continued)

TERMINAL				DESCRIPTION
NAME	PW, or N NO.	RSA NO.	I/O	
XIN/P2.6/TA1	13	12	I/O	Input terminal of crystal oscillator General-purpose digital I/O pin Timer_A, compare: Out1 output
XOUT/P2.7	12	11	I/O	Output terminal of crystal oscillator General-purpose digital I/O pin
$\overline{\text{RST}}$/NMI/SBWTDIO	10	9	I	Reset or nonmaskable interrupt input Spy-Bi-Wire test data input/output during programming and test
TEST/SBWTCK	11	10	I	Selects test mode for JTAG pins on Port1. The device protection fuse is connected to TEST. Spy-Bi-Wire test clock input during programming and test
V_{CC}	1	NA		Supply voltage
V_{SS}	14	NA		Ground reference
DV_{CC}	NA	16		Digital supply voltage
AV_{CC}	NA	15		Analog supply voltage
DV_{SS}	NA	14		Digital ground reference
AV_{SS}	NA	13		Analog ground reference
QFN Pad	NA	Package Pad	NA	QFN package pad connection to V_{SS} recommended.

† TDO or TDI is selected via JTAG instruction.

NOTE: If XOUT/P2.7 is used as an input, excess current will flow until P2SEL.7 is cleared. This is due to the oscillator output driver connection to this pad after reset.

B.2 INFORMACION DE LAS APLICACIONES (MODULOS) DEL MSP430X20X3

MSP430x20x1, MSP430x20x2, MSP430x20x3 MIXED SIGNAL MICROCONTROLLER

SLAS491A – AUGUST 2005 – REVISED OCTOBER 2005

APPLICATION INFORMATION, MSP430x20x3

Port P1 (P1.0 to P1.3) pin functions, MSP430x20x3

PIN NAME (P1.X)	X	FUNCTION	CONTROL BITS / SIGNALS			
			P1DIR.x	P1SEL.x	SD16AE.x	INCHx
P1.0/TACLK/ACLK/A0+	0	P1.0† Input/Output	0/1	0	0	N/A
		Timer_A2.TACLK/$\overline{\text{INCLK}}$	0	1	0	N/A
		ACLK	1	1	0	N/A
		A0+ (see Note 3)	X	X	1	0
P1.1/TA0/A0−/A4+	1	P1.1† Input/Output	0/1	0	0	N/A
		Timer_A2.CCI0A	0	1	0	N/A
		Timer_A2.TA0	1	1	0	N/A
		A0− (see Notes 3, 4)	X	X	1	0
		A4+ (see Note 3)	X	X	1	4
P1.2/TA1/A1+/A4−	2	P1.2† Input/Output	0/1	0	0	N/A
		Timer_A2.CCI1A	0	1	0	N/A
		Timer_A2.TA1	1	1	0	N/A
		A1+ (see Note 3)	X	X	1	1
		A4− (see Notes 3, 4)	X	X	1	4
P1.3/VREF/A1−	3	P1.3† Input/Output	0/1	0	0	N/A
		VREF	X	1	0	N/A
		A1− (see Notes 3, 4)	X	X	1	1

† Default after reset (PUC/POR)

NOTES:
1. N/A: Not available or not applicable.
2. X: Don't care.
3. Setting the SD16AE.x bit disables the output driver as well as the input schmitt trigger to prevent parasitic cross currents when applying analog signals.
4. With SD16AE.x = 0 the negative inputs are connected to VSS if the corresponding input is selected.

Port P1 (P1.4 to P1.7) pin functions, MSP430x20x3

PIN NAME (P1.X)	X	FUNCTION	CONTROL BITS / SIGNALS					
			P1DIR.x	P1SEL.x	USIP.x	SD16AE.x	INCHx	JTAG Mode
P1.4/SMCLK/A2+/ TCK	4	P1.4† Input/Output	0/1	0	N/A	0	N/A	0
		N/A	0	1	N/A	0	N/A	0
		SMCLK	1	1	N/A	0	N/A	0
		A2+ (see Note 3)	X	X	N/A	1	2	0
		TCK (see Note 5)	X	X	N/A	X	X	1
P1.5/TA0/SCLK/A2–/ TMS	5	P1.5† Input/Output	0/1	0	X	0	N/A	0
		N/A	0	1	X	0	N/A	0
		Timer_A2.TA0	1	1	X	0	N/A	0
		SCLK	X	X	1	0	N/A	0
		A2– (see Notes 3, 4)	X	X	X	1	2	0
		TMS (see Note 5)	X	X	X	X	X	1
P1.6/TA1/SDO/SCL/A3+/ TDI	6	P1.6† Input/Output	0/1	0	X	0	N/A	0
		Timer_A2.CCI1B	0	1	X	0	N/A	0
		Timer_A2.TA1	1	1	X	0	N/A	0
		SDO (SPI) / SCL (I2C)	X	X	1	0	N/A	0
		A3+ (see Note 3)	X	X	X	1	3	0
		TDI (see Note 5)	X	X	X	X	X	1
P1.7/SDI/SDA/A3–/ TDO/TDI	7	P1.7† Input/Output	0/1	0	X	0	N/A	0
		N/A	0	1	X	0	N/A	0
		DVSS	1	1	X	0	N/A	0
		SDI (SPI) / SDA (I2C)	X	X	1	0	N/A	0
		A3– (see Notes 3, 4)	X	X	X	1	3	0
		TDO/TDI (see Notes 5, 6)	X	X	X	X	X	1

† Default after reset (PUC/POR)

NOTES: 1. N/A: Not available or not applicable.
2. X: Don't care.
3. Setting the SD16AE.x bit disables the output driver as well as the input schmitt trigger to prevent parasitic cross currents when applying analog signals.
4. With SD16AE.x = 0 the negative inputs are connected to VSS if the corresponding input is selected.
5. In JTAG mode the internal pull-up/down resistors are disabled.
6. Function controlled by JTAG

Port P2 (P2.6) pin functions, MSP430x20x3

PIN NAME (P2.X)	X	FUNCTION	CONTROL BITS / SIGNALS	
			P2DIR.x	P2SEL.x
P2.6/XIN/TA1	6	P2.6 Input/Output	0/1	0
		XIN† (see Note 3)	0	1
		Timer_A2.TA1	1	1

† Default after reset (PUC/POR)

NOTES: 1. N/A: Not available or not applicable.
2. X: Don't care.
3. XIN is used as digital clock input if the bits LFXT1Sx in register BCSCTL3 are set to 11.

Port P2 (P2.7) pin functions, MSP430x20x3

PIN NAME (P2.X)	X	FUNCTION	CONTROL BITS / SIGNALS	
			P2DIR.x	P2SEL.x
P2.7/XOUT	7	P2.7 Input/Output	0/1	0
		DVSS	0	1
		XOUT† (see Note 3)	1	1

† Default after reset (PUC/POR)

NOTES: 1. N/A: Not available or not applicable.
2. X: Don't care.
3. If the pin P2.7/XOUT is used as an input a current can flow until P2SEL.7 is cleared due to the oscillator output driver connection to this pin after reset.

B.3 FUENTES DE INTERRUPCION

INTERRUPT SOURCE	INTERRUPT FLAG	SYSTEM INTERRUPT	WORD ADDRESS	PRIORITY
Power-up External reset Watchdog Timer+ Flash key violation PC out-of-range (see Note 1)	PORIFG RSTIFG WDTIFG KEYV (see Note 2)	Reset	0FFFEh	31, highest
NMI Oscillator fault Flash memory access violation	NMIIFG OFIFG ACCVIFG (see Notes 2 & 4)	(non)-maskable, (non)-maskable, (non)-maskable	0FFFCh	30
			0FFFAh	29
			0FFF8h	28
Comparator_A+ (MSP430x20x1 only)	CAIFG (see Note 3)	maskable	0FFF6h	27
Watchdog Timer+	WDTIFG	maskable	0FFF4h	26
Timer_A2	TACCR0 CCIFG (see Note 3)	maskable	0FFF2h	25
Timer_A2	TACCR1 CCIFG. TAIFG (see Notes 2 & 3)	maskable	0FFF0h	24
			0FFEEh	23
			0FFECh	22
ADC10 (MSP430x20x2 only)	ADC10IFG (see Note 3)	maskable	0FFEAh	21
SD16_A (MSP430x20x3 only)	SD16CCTL0 SD16OVIFG, SD16CCTL0 SD16IFG (see Notes 2 & 3)	maskable		
USI (MSP430x20x2, MSP430x20x3 only)	USIIFG, USISTTIFG (see Notes 2 & 3)	maskable	0FFE8h	20
I/O Port P2 (two flags)	P2IFG.6 to P2IFG.7 (see Notes 2 & 3)	maskable	0FFE6h	19
I/O Port P1 (eight flags)	P1IFG.0 to P1IFG.7 (see Notes 2 & 3)	maskable	0FFE4h	18
			0FFE2h	17
			0FFE0h	16
(see Note 5)			0FFDEh ... 0FFC0h	15 ... 0, lowest

NOTES: 1. A reset is generated if the CPU tries to fetch instructions from within the module register memory address range (0h–01FFh).
2. Multiple source flags
3. Interrupt flags are located in the module
4. (non)-maskable: the individual interrupt-enable bit can disable an interrupt event, but the general interrupt enable cannot.
5. The interrupt vectors at addresses 0FFDEh to 0FFC0h are not used in this device and can be used for regular program code if necessary.

B.4 MAPA DE LAS DIRECCIONES DE MEMORIA DE LOS PERIFERICOS

PERIPHERALS WITH WORD ACCESS			
ADC10 (MSP430x20x2 only)	ADC control 0 ADC control 1 ADC memory	ADC10CTL0 ADC10CTL0 ADC10MEM	01B0h 01B2h 01B4h
SD16_A (MSP430x20x3 only)	General Control Channel 0 Control Interrupt vector word register Channel 0 conversion memory	SD16CTL SD16CCTL0 SD16IV SD16MEM0	0100h 0102h 0110h 0112h
Timer_A	Capture/compare register Capture/compare register Timer_A register Capture/compare control Capture/compare control Timer_A control Timer_A interrupt vector	TACCR1 TACCR0 TAR TACCTL1 TACCTL0 TACTL TAIV	0174h 0172h 0170h 0164h 0162h 0160h 012Eh
Flash Memory	Flash control 3 Flash control 2 Flash control 1	FCTL3 FCTL2 FCTL1	012Ch 012Ah 0128h
Watchdog Timer+	Watchdog/timer control	WDTCTL	0120h
PERIPHERALS WITH BYTE ACCESS			
ADC10 (MSP430x20x2 only)	Analog enable	ADC10AE	04Ah
SD16_A (MSP430x20x3 only)	Channel 0 Input Control Analog Enable	SD16INCTL0 SD16AE	0B0h 0B7h
USI (MSP430x20x2 and MSP430x20x3 only)	USI control 0 USI control 1 USI clock control USI bit counter USI shift register	USICTL0 USICTL1 USICKCTL USICNT USISR	078h 079h 07Ah 07Bh 07Ch
Comparator_A+ (MSP430x20x1 only)	Comparator_A+ port disable Comparator_A+ control 2 Comparator_A+ control 1	CAPD CACTL2 CACTL1	05Bh 05Ah 059h
Basic Clock System+	Basic clock system control 3 Basic clock system control 2 Basic clock system control 1 DCO clock frequency control	BCSCTL3 BCSCTL2 BCSCTL1 DCOCTL	053h 058h 057h 056h
Port P2	Port P2 resistor enable Port P2 selection Port P2 interrupt enable Port P2 interrupt edge select Port P2 interrupt flag Port P2 direction Port P2 output Port P2 input	P2REN P2SEL P2IE P2IES P2IFG P2DIR P2OUT P2IN	02Fh 02Eh 02Dh 02Ch 02Bh 02Ah 029h 028h
Port P1	Port P1 resistor enable Port P1 selection Port P1 interrupt enable Port P1 interrupt edge select Port P1 interrupt flag Port P1 direction Port P1 output Port P1 input	P1REN P1SEL P1IE P1IES P1IFG P1DIR P1OUT P1IN	027h 026h 025h 024h 023h 022h 021h 020h
Special Function	SFR interrupt flag 2 SFR interrupt flag 1 SFR interrupt enable 2 SFR interrupt enable 1	IFG2 IFG1 IE2 IE1	003h 002h 001h 000h

B.5 CARACTERÍSTICAS ELÉCTRICAS SOBRE LOS VALORES RECOMENDADOS DEL MSP430X20X3

MSP430x20x1, MSP430x20x2, MSP430x20x3
MIXED SIGNAL MICROCONTROLLER

SLAS491A – AUGUST 2005 – REVISED OCTOBER 2005

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

low power mode supply currents (into V_{CC}) excluding external current (see Notes 1 and 2)

PARAMETER		TEST CONDITIONS		VCC	MIN	TYP	MAX	UNIT
$I_{LPM0, 1MHz}$	Low-power mode 0 (LPM0) current, see Note 3	f_{MCLK} = 0MHz, $f_{SMCLK} = f_{DCO}$ = 1MHz, f_{ACLK} = 32,768Hz, BCSCTL1 = CALBC1_1MHz, DCOCTL = CALDCO_1MHz, CPUOFF = 1, SCG0 = 0, SCG1 = 0, OSCOFF = 0		2.2 V		65	80	µA
				3 V		85	100	
I_{LPM0}100kHz	Low-power mode 0 (LPM0) current, see Note 3	f_{MCLK} = 0MHz, $f_{SMCLK} = f_{DCO(0, 0)} \approx$ 100kHz, f_{ACLK} = 0Hz, RSELx = 0, DCOx = 0, CPUOFF = 1, SCG0 = 0, SCG1 = 0, OSCOFF = 1		2.2 V		37	48	µA
				3 V		41	52	
I_{LPM2}	Low-power mode 1 (LPM2) current, see Note 4	$f_{MCLK} = f_{SMCLK}$ = 0MHz, f_{DCO} = 1MHz, f_{ACLK} = 32,768Hz, BCSCTL1 = CALBC1_1MHz, DCOCTL = CALDCO_1MHz, CPUOFF = 1, SCG0 = 0, SCG1 = 1, OSCOFF = 0		2.2 V		22	29	µA
				3 V		25	32	
$I_{LPM3,LFXT1}$	Low-power mode 3 (LPM3) current, see Note 4	$f_{DCO} = f_{MCLK} = f_{SMCLK}$ = 0MHz, f_{ACLK} = 32,768Hz, CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 0	T_A = –40°C	2.2 V		0.7	1.2	µA
			T_A = 25°C			0.7	1.0	
			T_A = 85°C			1.4	2.3	
			T_A = –40°C	3 V		0.9	1.2	
			T_A = 25°C			0.9	1.2	
			T_A = 85°C			1.6	2.8	
$I_{LPM3,VLO}$	Low-power mode 3 current, (LPM3) see Note 4	$f_{DCO} = f_{MCLK} = f_{SMCLK}$ = 0MHz, f_{ACLK} from internal LF oscillator (VLO), CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 0	T_A = –40°C	2.2 V		0.4	0.7	µA
			T_A = 25°C			0.5	0.7	
			T_A = 85°C			1.0	1.6	
			T_A = –40°C	3 V		0.5	0.9	
			T_A = 25°C			0.6	0.9	
			T_A = 85°C			1.3	1.8	
I_{LPM4}	Low-power mode 4 (LPM4) current, see Note 5	$f_{DCO} = f_{MCLK} = f_{SMCLK}$ = 0MHz, f_{ACLK} = 32,768Hz, CPUOFF = 1, SCG0 = 1, SCG1 = 1, OSCOFF = 1	T_A = –40°C	2.2 V/3 V		0.1	0.5	µA
			T_A = 25°C			0.1	0.5	
			T_A = 85°C			0.8	1.5	

NOTES:
1. All inputs are tied to 0 V or V_{CC}. Outputs do not source or sink any current.
2. The currents are characterized with a Micro Crystal CC4V−T1A SMD crystal with a load capacitance of 9 pF. The internal and external load capacitance is chosen to closely match the required 9pF.
3. Current for brownout and WDT clocked by SMCLK included.
4. Current for brownout and WDT clocked by ACLK included.
5. Current for brownout included.

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

Schmitt-trigger inputs – Ports P1 and P2

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
V_{IT+}	Positive-going input threshold voltage			0.45		0.75	V_{CC}
			2.2 V	1.00		1.65	V
			3 V	1.35		2.25	
V_{IT-}	Negative-going input threshold voltage			0.25		0.55	V_{CC}
			2.2 V	0.55		1.20	V
			3 V	0.75		1.65	
V_{hys}	Input voltage hysteresis ($V_{IT+} - V_{IT-}$)		2.2 V	0.2		1.0	V
			3 V	0.3		1.0	
R_{Pull}	Pull-up/pull-down resistor	For pull-up: $V_{IN} = V_{SS}$; For pull-down: $V_{IN} = V_{CC}$		20	35	50	kΩ
C_I	Input Capacitance	$V_{IN} = V_{SS}$ or V_{CC}			5		pF

inputs – Ports P1 and P2

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$t_{(int)}$	External interrupt timing	Port P1, P2: P1.x to P2.x, External trigger puls width to set interrupt flag, (see Note 1)	2.2 V/3 V	20			ns

NOTES: 1. An external signal sets the interrupt flag every time the minimum interrupt puls width $t_{(int)}$ is met. It may be set even with trigger signals shorter than $t_{(int)}$.

leakage current – Ports P1 and P2

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$I_{lkg(Px.x)}$	High-impedance leakage current	see Notes 1 and 2	2.2 V/3 V			±50	nA

NOTES: 1. The leakage current is measured with V_{SS} or V_{CC} applied to the corresponding pin(s), unless otherwise noted.
2. The leakage of the digital port pins is measured individually. The port pin is selected for input and the pull-up/pull-down resistor is disabled.

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

outputs – Ports P1 and P2

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage	$I_{(OHmax)}$ = –1.5 mA (see Notes 1)	2.2 V	V_{CC}–0.25		V_{CC}	V
		$I_{(OHmax)}$ = –6 mA (see Notes 2)	2.2 V	V_{CC}–0.6		V_{CC}	
		$I_{(OHmax)}$ = –1.5 mA (see Notes 1)	3 V	V_{CC}–0.25		V_{CC}	
		$I_{(OHmax)}$ = –6 mA (see Notes 2)	3 V	V_{CC}–0.6		V_{CC}	
V_{OL}	Low-level output voltage	$I_{(OLmax)}$ = 1.5 mA (see Notes 1)	2.2 V	V_{SS}		V_{SS}+0.25	V
		$I_{(OLmax)}$ = 6 mA (see Notes 2)	2.2 V	V_{SS}		V_{SS}+0.6	
		$I_{(OLmax)}$ = 1.5 mA (see Notes 1)	3 V	V_{SS}		V_{SS}+0.25	
		$I_{(OLmax)}$ = 6 mA (see Notes 2)	3 V	V_{SS}		V_{SS}+0.6	

NOTES: 1. The maximum total current, I_{OHmax} and I_{OLmax}, for all outputs combined, should not exceed ±12 mA to hold the maximum voltage drop specified.
2. The maximum total current, I_{OHmax} and I_{OLmax}, for all outputs combined, should not exceed ±48 mA to hold the maximum voltage drop specified.

output frequency – Ports P1 and P2

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$f_{Px.y}$	Port output frequency (with load)	P1.4/SMCLK, C_L = 20 pF, R_L = 1 kOhm (see Note 1 and 2)	2.2 V			10	MHz
			3 V			12	MHz
f_{Port_CLK}	Clock output frequency	P2.0/ACLK, P1.4/SMCLK, C_L = 20 pF (see Note 2)	2.2 V			12	MHz
			3 V			16	MHz

NOTES: 1. A resistive divider with 2 times 0.5 kΩ between V_{CC} and V_{SS} is used as load. The output is connected to the center tap of the divider.
2. The output voltage reaches at least 10% and 90% V_{CC} at the specified toggle frequency.

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

POR/brownout reset (BOR) (see Notes 1 and 2)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$V_{CC(start)}$	(see Figure 8)	$dV_{CC}/dt \le 3$ V/s			$0.7 \times V_{(B\_IT-)}$		V
$V_{(B\_IT-)}$	(see Figure 8 through Figure 10)	$dV_{CC}/dt \le 3$ V/s				1.71	V
$V_{hys(B\_IT-)}$	(see Figure 8)	$dV_{CC}/dt \le 3$ V/s		70	130	180	mV
$t_{d(BOR)}$	(see Figure 8)					2000	µs
$t_{(reset)}$	Pulse length needed at RST/NMI pin to accepted reset internally		2.2 V/3 V	2			µs

NOTES: 1. The current consumption of the brownout module is already included in the I_{CC} current consumption data. The voltage level $V_{(B\_IT-)} + V_{hys(B\_IT-)}$ is ≤ 1.8V.

2. During power up, the CPU begins code execution following a period of $t_{d(BOR)}$ after $V_{CC} = V_{(B\_IT-)} + V_{hys(B\_IT-)}$. The default DCO settings must not be changed until $V_{CC} \ge V_{CC(min)}$, where $V_{CC(min)}$ is the minimum supply voltage for the desired operating frequency.

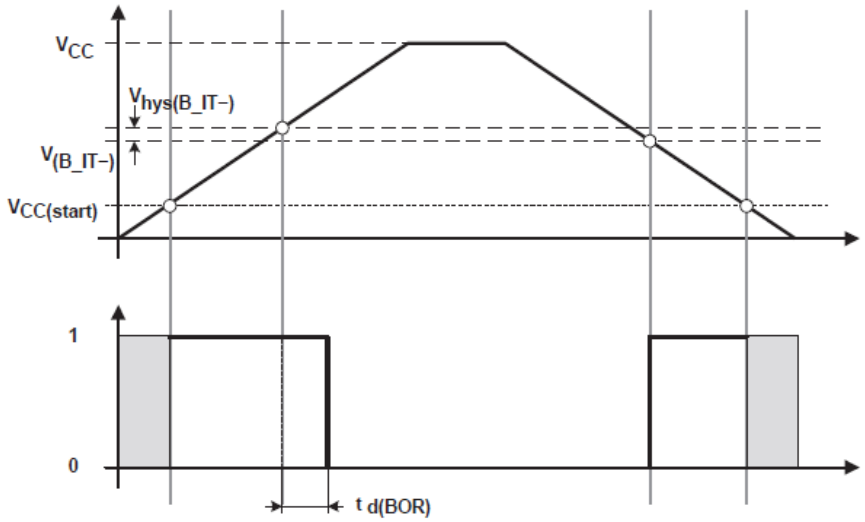


Figure 8. POR/Brownout Reset (BOR) vs Supply Voltage

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

Timer_A

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
f_{TA}	Timer_A clock frequency	Internal: SMCLK, ACLK; External: TACLK, INCLK; Duty Cycle = 50% ±10%	2.2 V			10	MHz
			3 V			16	
$t_{TA,cap}$	Timer_A, capture timing	TA0, TA1	2.2 V/3 V	20			ns

USI, Universal Serial Interface (MSP430x20x2, MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
f_{USI}	USI clock frequency	External: SCLK; Duty Cycle = 50% ±10%; SPI Slave Mode	2.2 V			10	MHz
			3 V			16	
$V_{OL,I2C}$	Low-level output voltage on SDA and SCL	USI module in I2C mode $I_{(OLmax)}$ = 1.5 mA	2.2 V/3 V	V_{SS}		V_{SS}+0.4	V

typical characteristics – USI low-level output voltage on SDA and SCL (MSP430x20x2, MSP430x20x3 only)

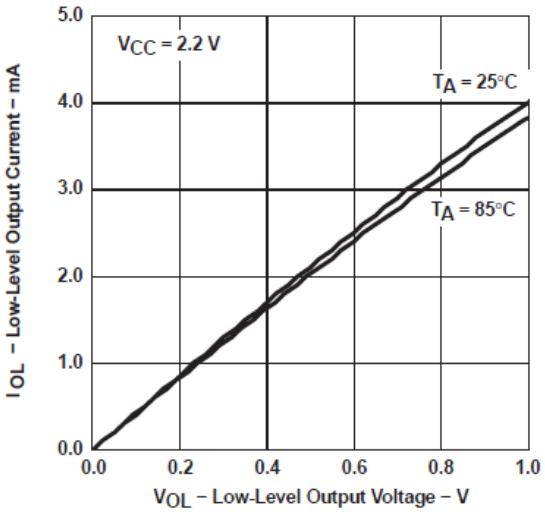


Figure 14. USI Low-Level Output Voltage vs. Output Current

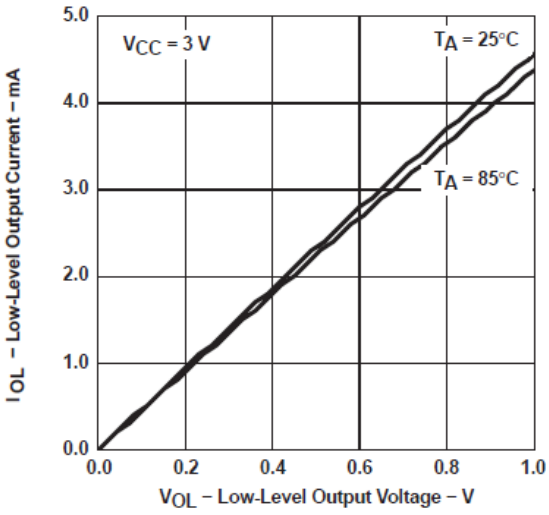


Figure 15. USI Low-Level Output Voltage vs. Output Current

MSP430x20x3 electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

SD16_A, power supply and recommended operating conditions (MSP430x20x3 only)

PARAMETER		TEST CONDITIONS		VCC	MIN	TYP	MAX	UNIT
AV_{CC}	Analog supply voltage range	$AV_{CC} = DV_{CC} = V_{CC}$ $AV_{SS} = DV_{SS} = V_{SS} = 0V$			2.5		3.6	V
I_{SD16}	Analog supply current including internal reference	SD16LP = 0, f_{SD16} = 1 MHz, SD16OSR = 256	GAIN: 1,2	3 V		730	1050	µA
			GAIN: 4,8,16	3 V		810	1150	
			GAIN: 32	3 V		1160	1700	
		SD16LP = 1, f_{SD16} = 0.5 MHz, SD16OSR = 256	GAIN: 1	3 V		720	1030	
			GAIN: 32	3 V		810	1150	
f_{SD16}	SD16 input clock frequency	SD16LP = 0 (Low power mode disabled)		3 V	0.03	1	1.1	MHz
		SD16LP = 1 (Low power mode enabled)		3 V	0.03	0.5		

SD16_A, input range (MSP430x20x3 only)

PARAMETER		TEST CONDITIONS		VCC	MIN	TYP	MAX	UNIT
$V_{ID,FSR}$	Differential full scale input voltage range (see Note 1)	Bipolar Mode, SD16UNI = 0			$-(V_{REF}/2)/GAIN$		$+(V_{REF}/2)/GAIN$	mV
		Unipolar Mode, SD16UNI = 1			0		$+(V_{REF}/2)/GAIN$	mV
V_{ID}	Differential input voltage range for specified performance (see Note 1)	SD16REFON=1	SD16GAINx=1			±500		mV
			SD16GAINx=2			±250		
			SD16GAINx=4			±125		
			SD16GAINx=8			±62		
			SD16GAINx=16			±31		
			SD16GAINx=32			±15		
Z_I	Input impedance (one input pin to AV_{SS})	f_{SD16} = 1MHz	SD16GAINx=1	3 V		200		kΩ
			SD16GAINx=32	3 V		75		
Z_{ID}	Differential Input impedance (IN+ to IN–)	f_{SD16} = 1MHz	SD16GAINx=1	3 V	300	400		kΩ
			SD16GAINx=32	3 V	100	150		
V_I	Absolute input voltage range				AV_{SS} -0.1V		AV_{CC}	V
V_{IC}	Common-mode input voltage range				AV_{SS} -0.1V		AV_{CC}	V

NOTES: 1. The analog input range depends on the reference voltage applied to V_{REF}. If V_{REF} is sourced externally, the full-scale range is defined by $V_{FSR+} = +(V_{REF}/2)/GAIN$ and $V_{FSR-} = -(V_{REF}/2)/GAIN$. The analog input range should not exceed 80% of V_{FSR+} or V_{FSR-}.

MSP430x20x3 electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

SD16_A, SINAD performance (f_{SD16} = 1MHz, SD16OSRx = 1024, SD16REFON = 1, MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	PW, or N		RSA		UNIT
				MIN	TYP	MIN	TYP	
$SINAD_{1024}$	Signal-to-Noise + Distortion Ratio (OSR = 1024)	SD16GAINx = 1, Signal Amplitude: V_{IN} = 500mV, Signal Frequency: f_{IN} = 100Hz	3 V	84	85	TBD	TBD	dB
		SD16GAINx = 2, Signal Amplitude: V_{IN} = 250mV, Signal Frequency: f_{IN} = 100Hz	3 V	82	83	TBD	TBD	
		SD16GAINx = 4, Signal Amplitude: V_{IN} = 125mV, Signal Frequency: f_{IN} = 100Hz	3 V	78	79	TBD	TBD	
		SD16GAINx = 8, Signal Amplitude: V_{IN} = 62mV, Signal Frequency: f_{IN} = 100Hz	3 V	73	74	TBD	TBD	
		SD16GAINx = 16, Signal Amplitude: V_{IN} = 31mV, Signal Frequency: f_{IN} = 100Hz	3 V	68	69	TBD	TBD	
		SD16GAINx = 32, Signal Amplitude: V_{IN} = 15mV, Signal Frequency: f_{IN} = 100Hz	3 V	62	63	TBD	TBD	

SD16_A, SINAD performance (f_{SD16} = 1MHz, SD16OSRx = 256, SD16REFON = 1, MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	PW, or N		RSA		UNIT
				MIN	TYP	MIN	TYP	
$SINAD_{256}$	Signal-to-Noise + Distortion Ratio (OSR = 256)	SD16GAINx = 1, Signal Amplitude: V_{IN} = 500mV, Signal Frequency: f_{IN} = 100Hz	3 V	80	81	TBD	TBD	dB
		SD16GAINx = 2, Signal Amplitude: V_{IN} = 250mV, Signal Frequency: f_{IN} = 100Hz	3 V	74	75	TBD	TBD	
		SD16GAINx = 4, Signal Amplitude: V_{IN} = 125mV, Signal Frequency: f_{IN} = 100Hz	3 V	69	70	TBD	TBD	
		SD16GAINx = 8, Signal Amplitude: V_{IN} = 62mV, Signal Frequency: f_{IN} = 100Hz	3 V	63	64	TBD	TBD	
		SD16GAINx = 16, Signal Amplitude: V_{IN} = 31mV, Signal Frequency: f_{IN} = 100Hz	3 V	58	59	TBD	TBD	
		SD16GAINx = 32, Signal Amplitude: V_{IN} = 15mV, Signal Frequency: f_{IN} = 100Hz	3 V	52	53	TBD	TBD	

MSP430x20x3 electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

typical characteristics – SD16_A SINAD performance over OSR (MSP430x20x3 only)

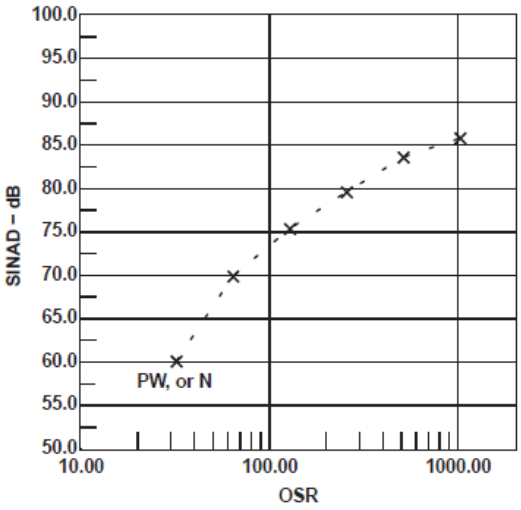


Figure 22. SINAD performance over OSR, f_{SD16} = 1MHz, SD16REFON = 1, SD16GAINx = 1

SD16_A, performance (f_{SD16} = 1MHz, SD16OSRx = 256, SD16REFON = 1, MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
G	Nominal Gain (see Note 1)	SD16GAINx = 1	3 V	0.97	1.00	1.02	
		SD16GAINx = 2	3 V	1.90	1.96	2.02	
		SD16GAINx = 4	3 V	3.76	3.86	3.96	
		SD16GAINx = 8	3 V	7.36	7.62	7.84	
		SD16GAINx = 16	3 V	14.56	15.04	15.52	
		SD16GAINx = 32	3 V	27.20	28.35	29.76	
dG/dT	Gain Temperature Drift	SD16GAINx = 1 (see Note 2)	3 V		15		ppm/°C
dG/dV_{CC}	Gain Supply Voltage Drift	SD16GAINx = 1; V_{CC} = 2.5V - 3.6V (see Note 3)	2.5V-3.6V		0.35		%/V
E_{OS}	Offset Error (see Note 1)	SD16GAINx = 1	3 V			±0.2	%FSR
		SD16GAINx = 32	3 V			±1.5	
dE_{OS}/dT	Offset Error Temperature Coefficient (see Note 1)	SD16GAINx = 1	3 V		±4	±20	ppm FSR/°C
		SD16GAINx = 32	3 V		±20	±100	
CMRR	Common-Mode Rejection Ratio	SD16GAINx = 1, Common-mode input signal: V_{ID} = 500 mV, f_{IN} = 50 Hz, 100 Hz	3 V		>90		dB
		SD16GAINx = 32, Common-mode input signal: V_{ID} = 16 mV, f_{IN} = 50 Hz, 100 Hz	3 V		>75		
PSRR	Power Supply Rejection Ratio	SD16GAINx = 1	3 V		>80		dB

NOTES: 1. Not production tested, limits characterized.
2. Calculated using the box method: (MAX(−40...85°C) − MIN(−40...85°C))/MIN(−40...85°C)/(85°C − (−40°C))
3. Calculated using the box method: (MAX(2.5...3.6V) − MIN(2.5...3.6V))/MIN(2.5...3.6V)/(3.6V − 2.5V)

MSP430x20x3 electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

SD16_A, temperature sensor (MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
TC_{Sensor}	Sensor temperature coefficient	See Note 1		1.18	1.32	1.46	mV/K
$V_{Offset,Sensor}$	Sensor offset voltage	See Note 1		−100		100	mV
V_{Sensor}	Sensor output voltage (see Note 3)	Temperature sensor voltage at T_A = 85°C	3 V	435	475	515	mV
		Temperature sensor voltage at T_A = 25°C	3 V	355	395	435	
		Temperature sensor voltage at T_A = 0°C (see Note 1)	3 V	320	360	400	

NOTES: 1. Not production tested, limits characterized.
2. The following formula can be used to calculate the temperature sensor output voltage:
$V_{Sensor,typ} = TC_{Sensor}$ (273 + T [°C]) + $V_{Offset,sensor}$ [mV] or
$V_{Sensor,typ} = TC_{Sensor}$ T [°C] + $V_{Sensor}(T_A = 0°C)$ [mV]
3. Results based on characterization and/or production test, not TC_{Sensor} or $V_{Offset,sensor}$.

SD16_A, built-in voltage reference (MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
V_{REF}	Internal reference voltage	SD16REFON = 1, SD16VMIDON = 0	3 V	1.14	1.20	1.26	V
I_{REF}	Reference supply current	SD16REFON = 1, SD16VMIDON = 0	3 V		190	280	µA
TC	Temperature coefficient	SD16REFON = 1, SD16VMIDON = 0	3 V		18	50	ppm/K
C_{REF}	V_{REF} load capacitance	SD16REFON = 1, SD16VMIDON = 0 (see Note 1)			100		nF
I_{LOAD}	$V_{REF(I)}$ maximum load current	SD16REFON = 1; SD16VMIDON = 0	3 V			±200	nA
t_{ON}	Turn on time	SD16REFON = 0 → 1; SD16VMIDON = 0; C_{REF} = 100nF	3 V		5		ms
PSRR	Line regulation	SD16REFON = 1; SD16VMIDON = 0	3 V		10		uV/V

NOTES: 1. There is no capacitance required on V_{REF}. However, a capacitance of at least 100nF is recommended to reduce any reference voltage noise.

SD16_A, reference output buffer (MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$V_{REF,BUF}$	Reference buffer output voltage	SD16REFON = 1, SD16VMIDON = 1	3 V		1.2		V
$I_{REF,BUF}$	Reference Supply + Reference output buffer quiescent current	SD16REFON = 1, SD16VMIDON = 1	3 V		385	600	µA
$C_{REF(O)}$	Required load capacitance on V_{REF}	SD16REFON = 1, SD16VMIDON = 1		470			nF
$I_{LOAD,Max}$	Maximum load current on V_{REF}	SD16REFON = 1, SD16VMIDON = 1	3 V			±1	mA
Maximum voltage variation vs. load current		$\|I_{LOAD}\|$ = 0 to 1mA	3 V	−15		+15	mV
t_{ON}	Turn on time	SD16REFON = 0 → 1; SD16VMIDON = 1; C_{REF} = 470nF	3 V		100		µs

SD16_A, external reference input (MSP430x20x3 only)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$V_{REF(I)}$	Input voltage range	SD16REFON = 0	3 V	1.0	1.25	1.5	V
$I_{REF(I)}$	Input current	SD16REFON = 0	3 V			50	nA

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

Flash Memory

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$V_{CC(PGM/ERASE)}$	Program and Erase supply voltage			2.2		3.6	V
f_{FTG}	Flash Timing Generator frequency			257		476	kHz
I_{PGM}	Supply current from V_{CC} during program		2.2 V/3.6 V		1	5	mA
I_{ERASE}	Supply current from V_{CC} during erase		2.2 V/3.6 V		1	7	mA
t_{CPT}	Cumulative program time (see Note 1)		2.2 V/3.6 V			10	ms
$t_{CMErase}$	Cumulative mass erase time		2.2 V/3.6 V	20			ms
	Program/Erase endurance			10^4	10^5		cycles
$t_{Retention}$	Data retention duration	T_J = 25°C		100			years
t_{Word}	Word or byte program time	see Note 2			30		t_{FTG}
$t_{Block, 0}$	Block program time for 1st byte or word				25		
$t_{Block, 1-63}$	Block program time for each additional byte or word				18		
$t_{Block, End}$	Block program end-sequence wait time				6		
$t_{Mass Erase}$	Mass erase time				10593		
$t_{Seg Erase}$	Segment erase time				4819		

NOTES: 1. The cumulative program time must not be exceeded when writing to a 64-byte flash block. This parameter applies to all programming methods: individual word/byte write and block write modes.
2. These values are hardwired into the Flash Controller's state machine ($t_{FTG} = 1/f_{FTG}$).

RAM

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$V_{(RAMh)}$	RAM retention supply voltage (see Note 1)	CPU halted	1.6			V

NOTE 1: This parameter defines the minimum supply voltage V_{CC} when the data in RAM remains unchanged. No program execution should happen during this supply voltage condition.

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (continued)

JTAG and Spy-Bi-Wire Interface

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
f_{SBW}	Spy-Bi-Wire input frequency		2.2 V / 3 V	0		20	MHz
$t_{SBW,Low}$	Spy-Bi-Wire low clock pulse length		2.2 V / 3 V	0.025		15	us
$t_{SBW,En}$	Spy-Bi-Wire enable time (TEST high to acceptance of first clock edge, see Note 1)		2.2 V/ 3 V			1	us
$t_{SBW,Ret}$	Spy-Bi-Wire return to normal operation time		2.2 V/ 3 V	15		100	us
f_{TCK}	TCK input frequency – 4-wire JTAG (see Note 2)		2.2 V	0		5	MHz
			3 V	0		10	MHz
$R_{Internal}$	Internal pull-down resistance on TEST		2.2 V/ 3 V	25	60	90	kΩ

NOTES: 1. Tools accessing the Spy-Bi-Wire interface need to wait for the maximum $t_{SBW,En}$ time after pulling the TEST/SBWCLK pin high before appling the first SBWCLK clock edge.
2. f_{TCK} may be restricted to meet the timing requirements of the module selected.

JTAG Fuse (see Note 1)

PARAMETER		TEST CONDITIONS	VCC	MIN	TYP	MAX	UNIT
$V_{CC(FB)}$	Supply voltage during fuse-blow condition	T_A = 25°C		2.5			V
V_{FB}	Voltage level on TEST for fuse-blow			6		7	V
I_{FB}	Supply current into TEST during fuse blow					100	mA
t_{FB}	Time to blow fuse					1	ms

NOTES: 1. Once the fuse is blown, no further access to the JTAG/Test, Spy-Bi-Wire, and emulation feature is possible and JTAG is switched to bypass mode.

B.6 SET DE INSTRUCCIONES

Instruction Set Overview

B.1 Instruction Set Overview

The following list gives an overview of the instruction set.

				Status Bits			
				V	N	Z	C
*	ADC[.W];ADC.B	dst	dst + C –> dst	*	*	*	*
	ADD[.W];ADD.B	src,dst	src + dst –> dst	*	*	*	*
	ADDC[.W];ADDC.B	src,dst	src + dst + C –> dst	*	*	*	*
	AND[.W];AND.B	src,dst	src .and. dst –> dst	0	*	*	*
	BIC[.W];BIC.B	src,dst	.not.src .and. dst –> dst	–	–	–	–
	BIS[.W];BIS.B	src,dst	src .or. dst –> dst	–	–	–	–
	BIT[.W];BIT.B	src,dst	src .and. dst	0	*	*	*
*	BR	dst	Branch to	–	–	–	–
	CALL	dst	PC+2 –> stack, dst –> PC	–	–	–	–
*	CLR[.W];CLR.B	dst	Clear destination	–	–	–	–
*	CLRC	Clear carry bit		–	–	–	0
*	CLRN	Clear negative bit		–	0	–	–
*	CLRZ	Clear zero bit		–	–	0	–
	CMP[.W];CMP.B	src,dst	dst – src	*	*	*	*
*	DADC[.W];DADC.B	dst	dst + C –> dst (decimal)	*	*	*	*
	DADD[.W];DADD.B	src,dst	src + dst + C –> dst (decimal)	*	*	*	*
*	DEC[.W];DEC.B	dst	dst – 1 –> dst	*	*	*	*
*	DECD[.W];DECD.B	dst	dst – 2 –> dst	*	*	*	*
*	DINT	Disable interrupt		–	–	–	–
*	EINT	Enable interrupt		–	–	–	–
*	INC[.W];INC.B	dst	Increment destination, dst +1 –> dst	*	*	*	*
*	INCD[.W];INCD.B	dst	Double-Increment destination, dst+2–>dst	*	*	*	*
*	INV[.W];INV.B	dst	Invert destination	*	*	*	*
	JC/JHS	Label	Jump to Label if Carry-bit is set	–	–	–	–
	JEQ/JZ	Label	Jump to Label if Zero-bit is set	–	–	–	–
	JGE	Label	Jump to Label if (N .XOR. V) = 0	–	–	–	–
	JL	Label	Jump to Label if (N .XOR. V) = 1	–	–	–	–
	JMP	Label	Jump to Label unconditionally	–	–	–	–
	JN	Label	Jump to Label if Negative-bit is set	–	–	–	–
	JNC/JLO	Label	Jump to Label if Carry-bit is reset	–	–	–	–
	JNE/JNZ	Label	Jump to Label if Zero-bit is reset	–	–	–	–

				Status Bits			
				V	N	Z	C
	MOV[.W];MOV.B	src,dst	src –> dst	–	–	–	–
*	NOP		No operation	–	–	–	–
*	POP[.W];POP.B	dst	Item from stack, SP+2 → SP	–	–	–	–
	PUSH[.W];PUSH.B	src	SP – 2 → SP, src → @SP	–	–	–	–
	RETI		Return from interrupt TOS → SR, SP + 2 → SP TOS → PC, SP + 2 → SZP	*	*	*	*
*	RET		Return from subroutine TOS → PC, SP + 2 → SP	–	–	–	–
*	RLA[.W];RLA.B	dst	Rotate left arithmetically	*	*	*	*
*	RLC[.W];RLC.B	dst	Rotate left through carry	*	*	*	*
	RRA[.W];RRA.B	dst	MSB → MSB → ....LSB → C	0	*	*	*
	RRC[.W];RRC.B	dst	C → MSB → .....LSB → C	*	*	*	*
*	SBC[.W];SBC.B	dst	Subtract carry from destination	*	*	*	*
*	SETC		Set carry bit	–	–	–	1
*	SETN		Set negative bit	–	1	–	–
*	SETZ		Set zero bit	–	–	1	–
	SUB[.W];SUB.B	src,dst	dst + .not.src + 1 → dst	*	*	*	*
	SUBC[.W];SUBC.B	src,dst	dst + .not.src + C → dst	*	*	*	*
	SWPB	dst	swap bytes	–	–	–	–
	SXT	dst	Bit7 → Bit8 ..... Bit15	0	*	*	*
*	TST[.W];TST.B	dst	Test destination	0	*	*	1
	XOR[.W];XOR.B	src,dst	src .xor. dst → dst	*	*	*	*

Note: Asterisked Instructions

Asterisked (*) instructions are emulated. They are replaced with core instructions by the assembler.

* The status bit is affected

– The status bit is not affected

0 The status bit is cleared

1 The status bit is set

B.7 FAQ: PREGUNTAS MÁS FRECUENTES

A.1 Hardware

1. **The state of the device** (CPU registers, RAM memory, etc.) **is undefined following a reset**. Exceptions to the above statement are that the PC is loaded with the word at 0xFFFE (i.e., the reset vector), the status register is cleared, and the peripheral registers (SFRs) are initialized as documented in the device-family user's guides. C-SPY resets the device after programming it.
2. **MSP430F22xx Target Socket Module (MSP-TS430DA38) – Important Information**
   Due to the large capacitive coupling introduced by the device socket between the adjacent signals XIN/P2.6 (socket pin 6) and $\overline{\text{RST}}$/SBWTDIO (socket pin 7), in-system debugging can disturb the LFXT1 low-frequency crystal oscillator operation (ACLK). This behavior applies only to the Spy-Bi-Wire (2-wire) JTAG configuration and only to the period while a debug session is active.
   Workarounds:
   - Use the 4-wire JTAG mode debug configuration instead of the Spy-Bi-Wire (2-wire) JTAG configuration. This can be achieved by placing jumpers JP4 through JP9 accordingly.
   - Use the debugger option “Release JTAG On Go” that can be selected from the IDE drop down menu. This prevents the debugger from accessing the MSP430 while the application is running. Note that, in this mode, a manual halt is required to see if a breakpoint was hit. See the IDE documentation for more information on this feature.
   - Use an external clock source to drive XIN directly.
3. With current interface hardware and software, there is a weakness when adapting target boards that are powered externally. This leads to an accidental fuse check in the MSP430. This is valid for PIF and UIF but is mainly seen on UIF. A solution is being developed.
   Workarounds:
   - Connect $\overline{\text{RST}}$/NMI pin to JTAG header (pin 11), LPT/USB tools are able to pull the RST line, which also resets the device internal fuse logic.
   - Use the debugger option "Release JTAG On Go" that can be selected from the IDE dropdown menu. This prevents the debugger from accessing the MSP430 while the application is running. Note that in this mode, a manual halt is required to see if a breakpoint was hit. See the IDE documentation for more information on this feature.
   - Use an external clock source to drive XIN directly.
4. When the MSP-FET430X110 is used as an interface to an MSP430 on the user's circuit (i.e., there is no MSP430 device in the FET socket), **the XOUT and XIN signals from the FET should not be connected to the corresponding pins of the in-circuit MSP430**. Similarly, when using the interface module, do not connect the XOUT and XIN signals from the interface module to the corresponding pins of the in-circuit MSP430.
5. The 14-conductor **cable** connecting the FET interface module and the target socket module **must not exceed 8 inches (20 centimeters) in length**.
6. The signal assignment on the **14-conductor cable is identical** for the **parallel port interface** and the **USB FET**.
7. **To utilize the on-chip ADC voltage references, C6** (10 µF, 6.3 V, low leakage) **must be installed** on the target socket module.
8. **Crystals/resonators Q1 and Q2** (if applicable) **are not provided** on the target socket module. For MSP430 devices that contain user-selectable loading capacitors, the effective capacitance is the selected capacitance plus 3 pF (pad capacitance) divided by two.
9. **Crystals/resonators have no effect on the operation of the tool and C-SPY** (as any required clocking/timing is derived from the internal DCO/FLL).
10. **On 20-pin and 28-pin devices** with multiplexed port/JTAG pins (P1.4 to P1.7), **it is required that "Release JTAG On Go" be selected to use these pins in their port capacity**. See Section C.1.2 for additional information regarding this mechanism.
11. **As an alternative to sharing the JTAG and port pins** (on 20 and 28 pin devices), **consider using an MSP430 device that is a "superset" of the smaller device**. A very powerful feature of the MSP430 is that the family members are code and architecturally compatible, so code developed on one device (for example, one without shared JTAG and port pins) ports effortlessly to another (assuming an equivalent set of peripherals).

12. **Information memory may not be blank** (erased to 0xFF) when the device is delivered from TI. Customers should erase the information memory before its first usage. Main memory of packaged devices is blank when the device is delivered from TI.
13. **The device current increases by approximately 10 µA when a device in low-power mode is stopped** (using ESC), **and then the low-power mode is restored** (using Go). This behavior appears to happen on all devices except the MSP430F12x.
14. The following **ZIF sockets** are used in the FET tools and target socket modules:
    - 14-pin device (PW package): Enplas OTS-14-065-01
    - 28-pin device (PW package): Enplas OTS-28-0.65-01
    - 28-pin device (DW package): Wells-CTI 652 D028
    - 38-pin device (DA package): Yamaichi IC189-0382-037
    - 40-pin device (RHA package): Enplas QFN-40B-0.5-01
    - 48-pin device (DL package): Yamaichi IC51-0482-1163
    - 64-pin device (PM package): Yamaichi IC51-0644-807
    - 80-pin device (PN package): Yamaichi IC201-0804-014
    - 100-pin device (PZ package): Yamaichi IC201-1004-008

    ENPLAS: www.enplas.com
    Wells-CTI: www.wellscti.com/
    Yamaichi: www.yamaichi.us/
15. **Supply current measurement on target socket modules**. On each module a jumper connects V_{CC} with VCC430. If this jumper is removed and an ampere meter is connected to the jumper pins, the supply current of the module can be measured. As the pullup resistor (47 kΩ) on the reset line is connected to V_{CC}, the MSP430 device sees a marginal voltage at pin $\overline{RST}$/NMI if V_{CC} is present and the jumper is open. Therefore, V_{CC} should be applied after the ampere meter has been connected.

A.2 Program Development (Assembler, C-Compiler, Linker)

1. **The files supplied in the 430\tutor folder work only with the simulator**. Do not use the files with the FET (see FAQ Program Development #11).
2. **A common MSP430 "mistake" is to fail to disable the Watchdog mechanism**; the Watchdog is enabled by default, and it resets the device if not disabled or properly handled by the application (see FAQ Program Development #14).
3. **When adding source files to a project, do not add files that are included by source files that have already been added to the project** (for example, an .h file within a .c or .s43 file). These files are added to the project file hierarchy automatically.
4. **In assembler, enclosing a string in double quotes** ("string") **automatically appends a zero byte** to the string (as an end-of-string marker). Enclosing a string in single-quotes ('string') does not.
5. When using the compiler or the assembler, **if the last character of a source line is backslash (\), the subsequent carriage return/line feed is ignored** (i.e., it is as if the current line and the next line are a single line). When used in this way, the backslash character is a "line continuation" character.
6. **The linker output format must be "Debug information for C-SPY" (.d43) for use with C-SPY**. C-SPY does not start otherwise, and an error message is output. C-SPY cannot input a .TXT file.
7. **Position-independent code can be generated** using Project → Options → General Options → Target → Position-Independent Code.
8. **Within the C libraries, GIE (Global Interrupt Enable) is disabled before** (and restored after) **the hardware multiplier is used**. To disable this behavior, contact TI for the source code for these libraries.
9. **It is possible to mix assembler and C programs within the Workbench**. See the Assembler Language Interface chapter of the C/C++ Compiler Reference Guide from IAR.
10. The Workbench can produce an object file in TI .TXT format. **C-SPY cannot input an object file in TI .TXT format**. An error message is output in this case.

B.8

11. **The example programs given in the KickStart documentation (i.e., Demo, Tutor, etc.) are not correct**. The programs work only in the simulator. The programs do not function correctly on an actual device, because the Watchdog mechanism is active. The programs need to be modified to disable the Watchdog mechanism. Disable the Watchdog mechanism with this C-statement:
`WDTCTL = WDTPW + WDTHOLD;`
or with this assembler statement:
`mov.w # WDTPW+WDTHOLD,&WDTCTL`
12. **Access to MPY using an 8-bit operation is flagged as an error**. Within the .h files, 16-bit registers are defined in such a way that 8-bit operations upon them are flagged as an error. This feature is normally beneficial and can catch register access violations. However, in the case of MPY, it is also valid to access this register using 8-bit operators. If 8-bit operators are used to access MPY, the access violation check mechanism can be defeated by using "MPY_" to reference the register. Similarly, 16-bit operations on 8-bit registers are flagged.
13. **Constant definitions (#define) used within the .h files are effectively reserved** and include, for example, C, Z, N, and V. Do not create program variables with these names.
14. **The CSTARTUP that is implicitly linked with all C applications does not disable the Watchdog timer**. Use `WDT = WDTPW + WDTHOLD;` to explicitly disable the Watchdog. This statement is best placed in the __low_level_init() function that gets executed before main().
    If the Watchdog timer is not disabled, and the Watchdog triggers and resets the device during CSTARTUP, **the source screen goes blank**, as C-SPY is not able to locate the source code for CSTARTUP. Be aware that CSTARTUP can take a significant amount of time to execute if a large number of initialized global variables are used.

```
int __low_level_init(void)
{
    /* Insert your low-level initializations here */

    WDTCTL = WDTPW + WDTHOLD; // Stop Watchdog timer

    /*==================================*/
    /* Choose if segment initialization */
    /* should be done or not.    */
    /* Return: 0 to omit seg_init  */
    /*         1 to run seg_init  */
    /*==================================*/
    return (1);
}
```

15. **Compiler optimization can remove unused variables and/or statements that have no effect** and can effect debugging. Optimization: NONE is supported within Project → Options → C/C++ Compiler → Code → Optimizations. Alternatively, variables can be declared volatile.
16. **The IAR tutorial assumes a Full or Baseline version of the Workbench**. Within a KickStart system, it is not possible to configure the C compiler to output assembler mnemonics.
17. Existing **projects from an IAR 1.x system can be used within the new IAR 2.x/3.x system**; refer to the IAR document migration guide for EW430 x.x. This document is located in <Installation Root>\Embedded Workbench x.x\430\doc\migration.htm
18. **Assembler projects must reference the code segment** (RSEG CODE) **to use the Linker → Processing → Fill Unused Code Memory** mechanism. No special steps are required to use Linker → Processing → Fill Unused Code Memory with C projects.
19. **Ensure that the proper C runtime library is selected for C-only and mixed C/assembly language projects** (Project → General Options → Library Configuration → Library). For assembly-only projects, the runtime library must not get linked in, otherwise the build fails and a linker error is output (e.g., that the RESET vector is allocated twice).

20. **Numerous C and C++ runtime libraries are provided with the Workbench**:
    cl430d: C, 64-bit doubles
    cl430dp: C, 64-bit doubles, position independent
    cl430f: C, 32-bit doubles
    cl430fp: C, 32-bit doubles, position independent
    dl430d: C++, 64-bit doubles
    dl430dp: C++, 64-bit doubles, position independent
    dl430f: C++, 32-bit doubles
    dl430fp: C++, 32-bit doubles, position independent
    See the IAR MSP430 C/C++ compiler reference guide for more information on which library to use.

A.3 Debugging (C-SPY)

1. **Debugging with C-SPY does not seem to affect an externally connected MSP430 device**. Should this be the case, check whether the main debugger menu bar contains a menu item called Simulator. If so, an actual C-SPY MSP430 core simulator session is running, and no actual communication with the target device is established. **Solution: Ensure that the C-SPY driver is set to FET Debugger** (Project → Options → Debugger → Setup → Driver).
2. **C-SPY reports that it cannot communicate with the device**. Possible solutions to this problem include:
   - Ensure that the correct debug interface is selected; use Project → Options → FET Debugger → Connection.
   - Ensure that the correct parallel port (LPT1, 2, or 3) is being specified in the C-SPY configuration in the case a parallel port MSP-FET430PIF interface is used; use Project → Options → FET Debugger → Connection → Parallel Port → LPT1 (default) or LPT2 or LPT3. Check the PC BIOS for the parallel port address (0x378, 0x278, 0x3bc), and the parallel port configuration (ECP, Compatible, Bidirectional, or Normal) (see FAQ Debugging #8). For users of IBM ThinkPad™ computers, try port specifications LPT2 and LPT3, even if the operating system reports the parallel port is located at LPT1.
   - Ensure that no other software application has reserved/taken control of the parallel port (for example, printer drivers, ZIP drive drivers, etc.) if a parallel port MSP-FET430PIF interface is used. Such software can prevent the C-SPY/FET driver from accessing the parallel port and, hence, communicating with the device.
   - It may be necessary to reboot the computer to complete the installation of the required port drivers.
   - Ensure that the MSP430 device is securely seated in the socket (so that the "fingers" of the socket completely engage the pins of the device), and that its pin 1 (indicated with a circular indentation on the top surface) aligns with the "1" mark on the PCB.

> **CAUTION**
>
> **Possible Damage to Device**
>
> Always handle MSP430 devices using a vacuum pick-up tool only; do not use your fingers, as they can easily bend the device pins and render the device useless. Also, always observe and follow proper ESD precautions.

3. **C-SPY reports that the device JTAG security fuse is blown**. With current MSP-FET430PIF and MSP430-FET430UIF JTAG interface tools there is a weakness when adapting target boards that are powered externally. This leads to an accidental fuse check in the MSP430 and results in the JTAG security fuse being recognized as blown although it is not. This is valid for MSP-FET430PIF and MSP-FET430UIF but is mainly seen on MSP-FET430UIF.
   Workarounds:
   - Connect the device $\overline{\text{RST}}$/NMI pin to JTAG header (pin 11), MSP-FET430PIF/MSP-FET430UIF interface tools are able to pull the $\overline{\text{RST}}$ line, this also resets the device internal fuse logic.
   - Do NOT connect both V_{CC} Tool (pin 2) and V_{CC} Target (pin 4) of the JTAG header and also specify a value for V_{CC} in the debugger that is equal to the external supply voltage.

4. **C-SPY can download data into RAM, information, and flash main memories**. A warning message is output if an attempt is made to download data outside of the device memory spaces.
5. **C-SPY can debug applications that utilize interrupts and low power modes** (see FAQ Debugging #26).
6. **C-SPY cannot access the device registers and memory while the device is running**. C-SPY displays "-" to indicate that a register/memory field is invalid. The user must stop the device to access device registers and memory. Any displayed register/memory fields are then updated.
7. **When C-SPY is started, the flash memory is erased and the opened file is programmed** in accordance with the download options as set in Project → Options → FET Debugger → Download Control. This initial erase and program operations can be disabled selecting Project → Options → FET Debugger → Download Control → Suppress Download. Programming of the flash can be initiated manually with Emulator → Init New Device.
8. **The parallel port designators (LPTx) have the following physical addresses: LPT1: 378h, LPT2: 278h, LPT3: 3BCh**. The configuration of the parallel port (ECP, Compatible, Bidirectional, Normal) is not significant; ECP seems to work well (see FAQ Debugging #1 for additional hints on solving communication problems between C-SPY and the device).
9. **C-SPY may assert $\overline{\text{RST}}$/NMI to reset the device** when C-SPY is started and when the device is programmed. The device is also reset by the C-SPY RESET button, and when the device is manually reprogrammed (Emulator → Init New Device), and when the JTAG is resynchronized (Emulator → Resynchronize JTAG). When $\overline{\text{RST}}$/NMI is not asserted (low), C-SPY sets the logic driving $\overline{\text{RST}}$/NMI to high-impedance, and $\overline{\text{RST}}$/NMI is pulled high via a resistor on the PCB.

   $\overline{\text{RST}}$/NMI may get asserted and negated after power is applied when C-SPY is started. $\overline{\text{RST}}$/NMI may then get asserted and negated a second time after device initialization is complete.

   Within C-SPY, Emulator → "Power on" Reset cycles the power to the target to generate a power-on reset.
10. **C-SPY can debug a device whose program reconfigures the function of the $\overline{\text{RST}}$/NMI pin to NMI**.
11. **The level of the XOUT/TCLK pin is undefined when C-SPY resets the device**. The logic driving XOUT/TCLK is set to high-impedance at all other times.
12. **When making current measurements of the device, ensure that the JTAG control signals are released** (Emulator → Release JTAG on Go), otherwise the device will be powered by the signals on the JTAG pins and the measurements will be erroneous (see FAQ Debugging #14 and Hardware #13).
13. **Most C-SPY settings** (breakpoints, etc.) **are preserved between sessions**.
14. **When C-SPY has control of the device, the CPU is ON** (i.e., it is not in low-power mode) regardless of the settings of the low-power mode bits in the status register. Any low-power mode conditions are restored prior to Step or Go. Consequently, do not measure the power consumed by the device while C-SPY has control of the device. Instead, run your application using Go with JTAG released (see FAQ Debugging #12 and Hardware #13).
15. The View → Memory → Memory Fill dialog of C-SPY requires **hexadecimal values** for Starting Address, Length, and Value to be **preceded with "0x"**. Otherwise the values are interpreted as decimal.
16. The Memory debug view of C-SPY (View → Memory) can be used to view the RAM, the information memory, and the flash main memory. The Memory utility of C-SPY can be used to modify the RAM; **the information memory and flash main memory cannot be modified using the Memory utility**. The information memory and flash main memory can be programmed only when a project is opened and the data is downloaded to the device, or when Emulator → Init New Device is selected.
17. **C-SPY does not permit the individual segments of the information memory and the flash main memory to be manipulated separately**; consider the information memory to be one contiguous memory, and the flash main memory to be a second contiguous memory.
18. The Memory window correctly displays the contents of memory where it is present. However, **the Memory window incorrectly displays the contents of memory where there is none present**. Memory should be used only in the address ranges specified by the device data sheet.

19. C-SPY utilizes the system clock to control the device during debugging. Therefore, **device counters, etc., that are clocked by the Main System Clock (MCLK) are affected when C-SPY has control of the device**. Special precautions are taken to minimize the effect upon the Watchdog Timer. The CPU core registers are preserved. All other clock sources (SMCLK, ACLK) and peripherals continue to operate normally during emulation. In other words, **the Flash Emulation Tool is a partially intrusive tool**.

    Devices that support clock control (Emulator → Advanced → Clock Control) can further minimize these effects by selecting to stop the clock(s) during debugging (see FAQ Debugging #24).
20. **There is a time after C-SPY performs a reset of the device** [when the C-SPY session is first started, when the flash is reprogrammed (via Init New Device), and when JTAG is resynchronized (Resynchronize JTAG)] and before C-SPY has regained control of the device **that the device executes code normally**. This behavior may have side effects. Once C-SPY has regained control of the device, it performs a reset of the device and retains control.
21. When programming the flash, **do not set a breakpoint on the instruction immediately following the write to flash operation**. A simple workaround to this limitation is to follow the write to flash operation with a NOP, and set a breakpoint on the instruction following the NOP (see FAQ Debugging #23).
22. The **Dump Memory length specifier is restricted to four hexadecimal digits (0 to FFFF)**. This limits the number of bytes that can be written from 0 to 65535. Consequently, it is not possible to write memory from 0 to 0xFFFF inclusive, as this would require a length specifier of 65536 (or 10000h).
23. Multiple internal machine cycles are required to clear and program the flash memory. **When single stepping over instructions that manipulate the flash**, control is given back to C-SPY before these operations are complete. Consequently, **C-SPY updates its memory window with erroneous information**. A workaround to this behavior is to follow the flash access instruction with a NOP, and then step past the NOP before reviewing the effects of the flash access instruction (see FAQ Debugging #21).
24. **Peripheral bits that are cleared when read during normal program execution** (i.e., interrupt flags) **are cleared when read while being debugged** (i.e., memory dump, peripheral registers).

    When using certain MSP430 devices (such as MSP430F15x/16x and MSP430F43x/44x devices), bits do not behave this way (i.e., the bits are not cleared by C-SPY read operations).
25. **C-SPY cannot be used to debug programs that execute in the RAM of 'F12x and 'F41x devices**. A workaround to this limitation is to debug programs in flash.
26. **While single stepping with active and enabled interrupts, it can appear that only the interrupt service routine (ISR) is active** (i.e., the non-ISR code never appears to execute, and the single step operation always stops on the first line of the ISR). However, this behavior is correct because the device always processes an active and enabled interrupt before processing non-ISR (i.e., mainline) code. A workaround for this behavior is, while within the ISR, to disable the GIE bit on the stack so that interrupts are disabled after exiting the ISR. This permits the non-ISR code to be debugged (but without interrupts). Interrupts can later be reenabled by setting GIE in the status register in the Register window.

    On devices with the clock control emulation feature, it may be possible to suspend a clock between single steps and delay an interrupt request (Emulator → Advanced → Clock Control).
27. **The base** (decimal, hexadecimal, etc.) **property of Watch Window variables is not preserved between C-SPY sessions**; the base reverts to Default Format.
28. On devices equipped with a Data Transfer Controller (DTC), **the completion of a data transfer cycle preempts a single step of a low-power mode instruction**. The device advances beyond the low-power mode instruction only after an interrupt is processed. Until an interrupt is processed, it appears that the single step has no effect. A workaround to this situation is to set a breakpoint on the instruction following the low-power mode instruction, and then execute (Go) to this breakpoint.
29. **The transfer of data by the Data Transfer Controller (DTC) may not stop precisely when the DTC is stopped in response to a single step or a breakpoint**. When the DTC is enabled and a single step is performed, one or more bytes of data can be transferred. When the DTC is enabled and configured for two-block transfer mode, the DTC may not stop precisely on a block boundary when stopped in response to a single step or a breakpoint.

30. The C-SPY **Register window supports instruction cycle length counters**. The cycle counter is active only while single stepping. The count is reset when the device is reset, or the device is run (Go). The count can be edited (normally set to zero) at any time.
31. **It is possible to use C-SPY to get control of a running device whose state is unknown**. Simply use C-SPY to program a dummy device, and then start the application with Release JTAG on Go selected. Remove the JTAG connector from the dummy device and connect to the unknown device. Select Debug → Break (or the Stop hand) to stop the unknown device. The state of the device can then be interrogated.
32. Reseting a program temporarily requires a breakpoint if Project → Options → Debugger → Setup → Run To is enabled. If N or more breakpoints are set, Reset sets a virtual breakpoint and runs to the Run To function. Consequently, **it may require a significant amount of time before the program resets** (i.e., stops at the Run To function). During this time the C-SPY indicates that the program is running, and C-SPY windows may be blank (or may not be correctly updated).
33. Run To Cursor temporarily requires a breakpoint. If N breakpoints are set and virtual breakpoints are disabled, **Run To Cursor incorrectly uses a virtual breakpoint**. This results in very slow program execution.
34. **The simulator is a CPU core simulator only**; peripherals are not simulated, and interrupts are statistical events.
35. On devices without data breakpoint capabilities, it is possible to associate with an instruction breakpoint an (arbitrarily complex) expression that C-SPY evaluates when the breakpoint is hit. **This mechanism can be used to synthesize a data breakpoint**. See the C-SPY documentation for a description of this complex breakpoint mechanism.
36. **The ROM Monitor** referenced by the C-SPY documentation applies only to older MSP430Exxx (EPROM) based devices; it **can be ignored** when using the FET and the flash-based MSP430F devices.
37. **Special function registers** (SFRs) and the peripheral registers **are displayed in View → Register**.
38. **The putchar()/getchar() breakpoints are set only if these functions are present** (and the mechanism is enabled). Note that putchar()/getchar() could be indirectly referenced by a library function.
39. **The flash program/download progress bar does not update gradually**. This behavior is to be expected. The progress bar updates whenever a "chunk" of memory is written to flash. The development tools attempt to minimize the number of program chunks to maximize programming efficiency. Consequently, it is possible, for example, for a 60K-byte program to be reduced to a single chunk, and the progress bar is updated until the entire write operation is complete.

B.9 MENUS DE COMANDOS DEL EMULADOR C-SPY

C.1 Menus

C.1.1 Emulator → Device Information

Opens a window with information about the target device being used. Also, this window allows adjusting the target voltage in the case an MSP-FET430UIF interface is used to supply power to the target by performing a right-click inside this window. The supply voltage can be adjusted between 1.8 V and 5 V. This voltage is available on pin 2 of the 14-pin target connector to supply the target from the MSP-FET430UIF. If the target is supplied externally, the external supply voltage should be connected to pin 4 of the target connector, so the MSP-FET430UIF can set the level of the output signals accordingly.

C.1.2 Emulator → Release JTAG on Go

C-SPY uses the device JTAG signals to debug the device. On some MSP430 devices, these JTAG signals are shared with the device port pins. Normally, C-SPY maintains the pins in JTAG mode so that the device can be debugged. During this time the port functionality of the shared pins is not available.

However, when Release JTAG On Go is selected, the JTAG drivers are set to three-state, and the device is released from JTAG control (TEST pin is set to GND) when Go is activated. Any active on-chip breakpoints are retained, and the shared JTAG port pins revert to their port functions.

At this time, C-SPY has no access to the device and cannot determine if an active breakpoint (if any) has been reached. C-SPY must be manually commanded to stop the device, at which time the state of the device is determined (i.e., was a breakpoint reached?) (see FAQ Debugging #12).

C.1.3 Emulator → Resynchronize JTAG

Regain control of the device.

It is not possible to Resynchronize JTAG while the device is operating.

C.1.4 Emulator → Init New Device

Initialize the device according to the settings in the Download Options. Basically, the current program file is downloaded to the device memory. The device is then reset. This option can be used to program multiple devices with the same program from within the same C-SPY session.

It is not possible to select Init New Device while the device is operating.

C.1.5 Emulator → Secure - Blow JTAG Fuse

Blows the fuse on the target device. After the fuse is blown, no communication with the device is possible.

C.1.6 Emulator → Breakpoint Usage

List all used hardware and virtual breakpoints, as well as all currently defined EEM breakpoints.

C.1.7 Emulator → Advanced → Clock Control

Disable the specified system clock while C-SPY has control of the device (following a Stop or breakpoint). All system clocks are enabled following a Go or a single step (Step/Step Into) (see FAQ Debugging #19).

C.1.8 Emulator → Advanced → Emulation Mode

Specify the device to be emulated. The device must be reset (or reinitialized through Init New Device) following a change to the emulation mode. See Appendix D for more information.

C.1.9 Emulator → Advanced → Memory Dump

Write the specified device memory contents to a specified file. A conventional dialog is displayed that permits the user to specify a file name, a memory starting address, and a length. The addressed memory is then written in a text format to the named file. Options permit the user to select word or byte text format, and address information and register contents also can be appended to the file.

C.1.10 Emulator → Advanced → Breakpoint Combiner

Open the Breakpoint Combiner dialog box. The Breakpoint Combiner dialog box permits one to specify breakpoint dependencies. A breakpoint is triggered when the breakpoints are encountered in the specified order.

C.1.11 Emulator → State Storage Control

Open the State Storage dialog box. The State Storage dialog box permits the user to use the state storage module. The State Storage Module is not present on all MSP430 derivatives. Refer to Table 2-1 for implementation details

See the IAR C-SPY FET Debugger section in the MSP430 IAR Embedded Workbench IDE User Guide.

C.1.12 Emulator → State Storage Window

Open the State Storage window, and display the stored state information as configured by the State Storage dialog.

See the IAR C-SPY FET Debugger section in the MSP430 IAR Embedded Workbench IDE User Guide.

C.1.13 Emulator → Sequencer Control

Open the Sequencer dialog box. The Sequencer dialog box permits the user to configure the sequencer state machine.

See the IAR C-SPY FET Debugger section in the MSP430 IAR Embedded Workbench IDE User Guide.

C.1.14 Emulator → "Power on" Reset

Cycle power to the device to effect a reset.

C.1.15 Emulator → GIE on/off

Enables or disables all interrupts. Needs to be restored manually before Go.

C.1.16 Emulator → Leave Target Running

If C-SPY is closed, the target keeps running the user program.

C.1.17 Emulator → Force Single Stepping

On Go the program is executed by single steps. The cycle counter works correctly only in this mode.

Note: Availability of Emulator → Advanced menus

Not all Emulator → Advanced menus are supported by all MSP430 devices. These menus are grayed out.

ANEXO C

HOJA DE DATOS DEL INTEGRADO MAX-232

ANEXO C

C.1 HOJA DE DATOS DEL INTEGRADO MAX-232

+5V-Powered, Multichannel RS-232 Drivers/Receivers

ABSOLUTE MAXIMUM RATINGS—MAX220/222/232A/233A/242/243

Supply Voltage (V_{CC}) ..... -0.3V to +6V
Input Voltages
- T_{IN} ..... -0.3V to (V_{CC} - 0.3V)
- R_{IN} (Except MAX220) ..... ±30V
- R_{IN} (MAX220) ..... ±25V
- T_{OUT} (Except MAX220) (Note 1) ..... ±15V
- T_{OUT} (MAX220) ..... ±13.2V

Output Voltages
- T_{OUT} ..... ±15V
- R_{OUT} ..... -0.3V to (V_{CC} + 0.3V)

Driver/Receiver Output Short Circuited to GND ..... Continuous
Continuous Power Dissipation (T_A = +70°C)
- 16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
- 18-Pin Plastic DIP (derate 11.11mW/°C above +70°C) 889mW
- 20-Pin Plastic DIP (derate 8.00mW/°C above +70°C) 440mW
- 16-Pin Narrow SO (derate 8.70mW/°C above +70°C) 696mW
- 16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
- 18-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
- 20-Pin Wide SO (derate 10.00mW/°C above +70°C) 800mW
- 20-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
- 16-Pin CERDIP (derate 10.00mW/°C above +70°C) 800mW
- 18-Pin CERDIP (derate 10.53mW/°C above +70°C) 842mW

Operating Temperature Ranges
- MAX2_ _AC_ _, MAX2_ _C_ _ ..... 0°C to +70°C
- MAX2_ _AE_ _, MAX2_ _E_ _ ..... -40°C to +85°C
- MAX2_ _AM_ _, MAX2_ _M_ _ ..... -55°C to +125°C

Storage Temperature Range ..... -65°C to +160°C
Lead Temperature (soldering, 10sec) ..... +300°C

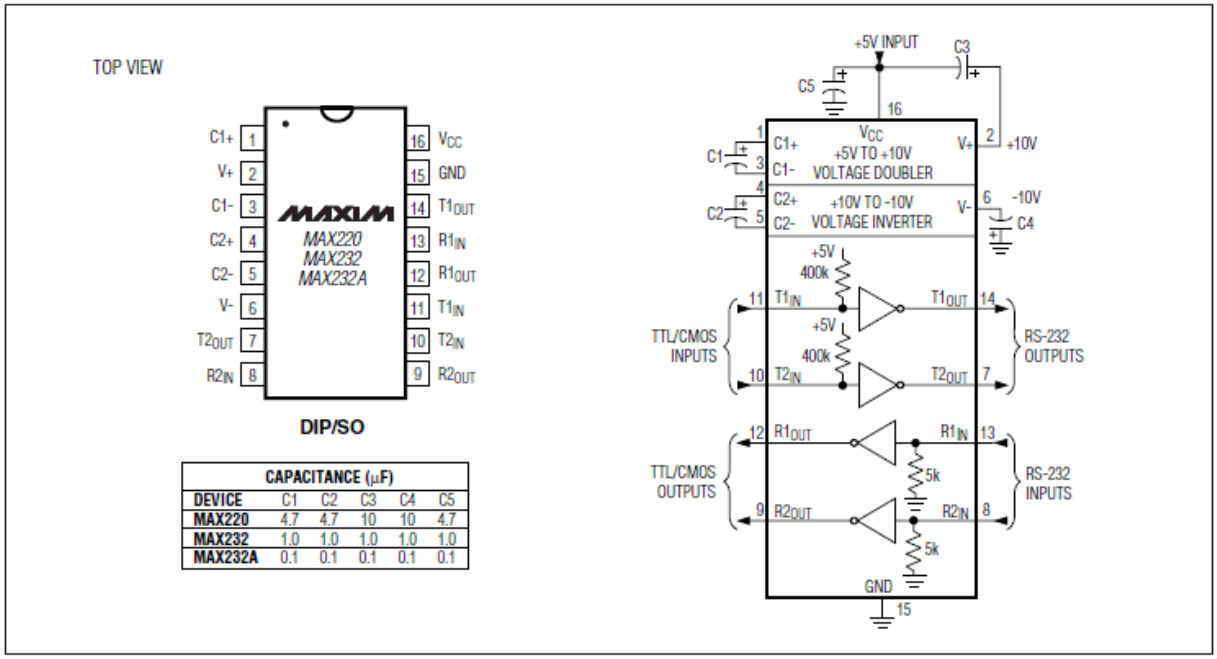
Note 1: Input voltage measured with T_{OUT} in high-impedance state, $\overline{SHDN}$ or V_{CC} = 0V.
Note 2: For the MAX220, V+ and V- can have a maximum magnitude of 7V, but their absolute difference cannot exceed 13V.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS—MAX220/222/232A/233A/242/243

(V_{CC} = +5V ±10%, C1–C4 = 0.1µF, MAX220, C1 = 0.047µF, C2–C4 = 0.33µF, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
RS-232 TRANSMITTERS						
Output Voltage Swing	All transmitter outputs loaded with 3kΩ to GND		±5	±8		V
Input Logic Threshold Low				1.4	0.8	V
Input Logic Threshold High	All devices except MAX220		2	1.4		V
	MAX220: V_{CC} = 5.0V		2.4			
Logic Pull-Up/Input Current	All except MAX220, normal operation			5	40	µA
	$\overline{SHDN}$ = 0V, MAX222/242, shutdown, MAX220			±0.01	±1	
Output Leakage Current	V_{CC} = 5.5V, $\overline{SHDN}$ = 0V, V_{OUT} = ±15V, MAX222/242			±0.01	±10	µA
	V_{CC} = $\overline{SHDN}$ = 0V, V_{OUT} = ±15V			±0.01	±10	
Data Rate				200	116	kb/s
Transmitter Output Resistance	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300	10M		Ω
Output Short-Circuit Current	V_{OUT} = 0V		±7	±22		mA
RS-232 RECEIVERS						
RS-232 Input Voltage Operating Range					±30	V
RS-232 Input Threshold Low	V_{CC} = 5V	All except MAX243 $R2_{IN}$	0.8	1.3		V
		MAX243 $R2_{IN}$ (Note 2)	-3			
RS-232 Input Threshold High	V_{CC} = 5V	All except MAX243 $R2_{IN}$		1.8	2.4	V
		MAX243 $R2_{IN}$ (Note 2)		-0.5	-0.1	
RS-232 Input Hysteresis	All except MAX243, V_{CC} = 5V, no hysteresis in shdn.		0.2	0.5	1	V
	MAX243			1		
RS-232 Input Resistance			3	5	7	kΩ
TTL/CMOS Output Voltage Low	I_{OUT} = 3.2mA			0.2	0.4	V
TTL/CMOS Output Voltage High	I_{OUT} = -1.0mA		3.5	V_{CC} - 0.2		V
TTL/CMOS Output Short-Circuit Current	Sourcing V_{OUT} = GND		-2	-10		mA
	Shrinking V_{OUT} = V_{CC}		10	30		



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 5. MAX220/MAX232/MAX232A Pin Configuration and Typical Operating Circuit

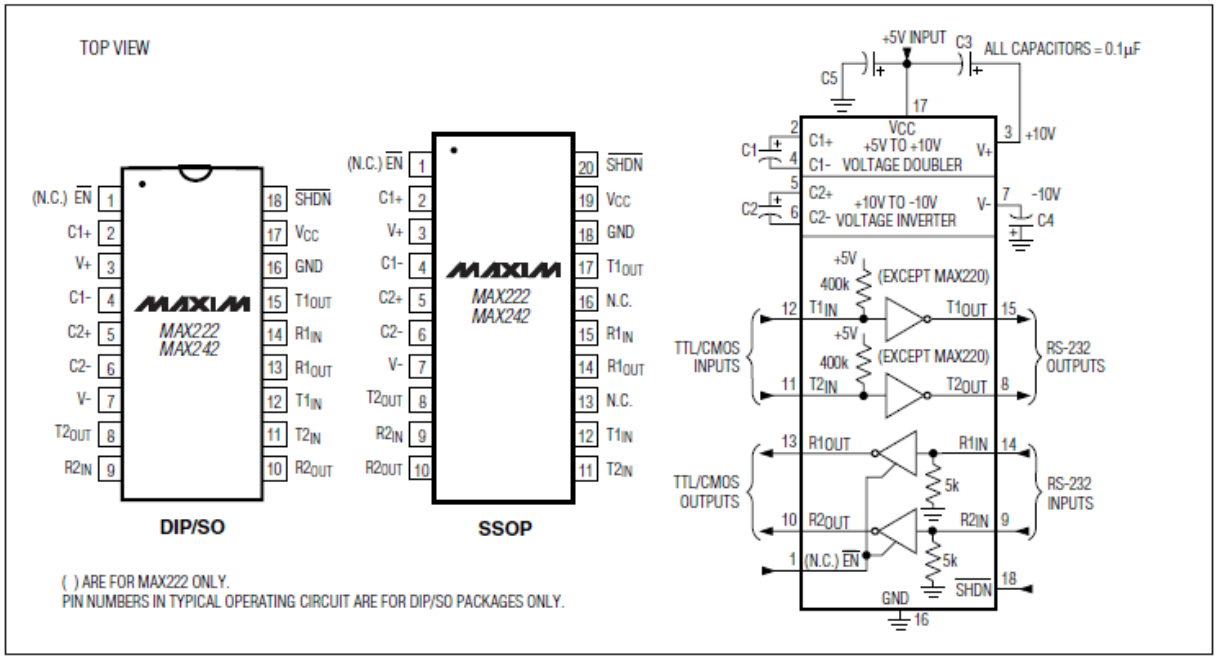


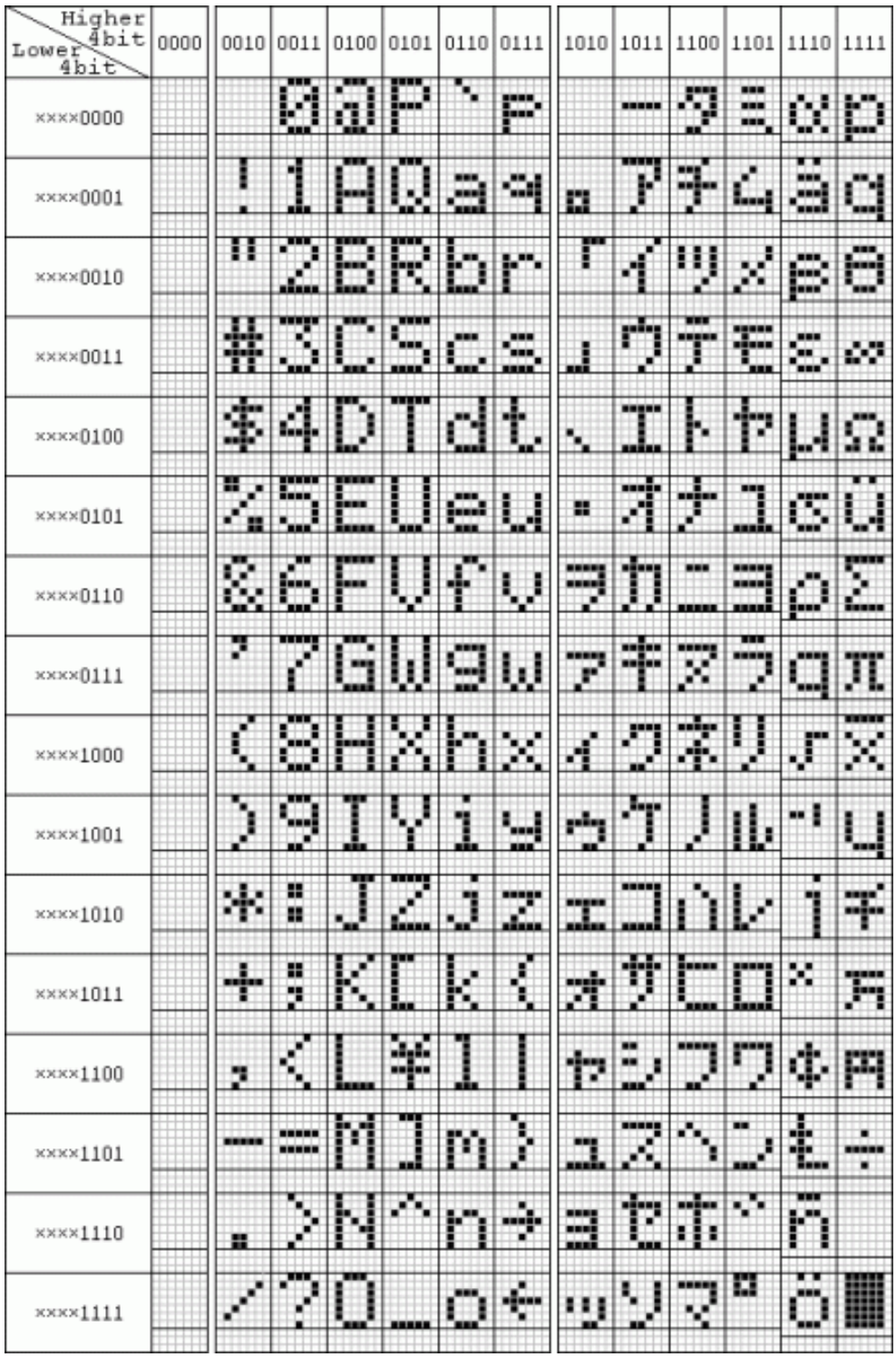
Figure 6. MAX222/MAX242 Pin Configurations and Typical Operating Circuit

ANEXO D

TABLA DE CARACTERES DISPONIBLES EN LA MEMORIA ROM DEL DISPLAY LCD

ANEXO D

D.1 TABLA DE CARACTERES DISPONIBLES EN LA MEMORIA ROM DEL DISPLAY LCD



ANEXO E

COMUNICACIÓN CON LOS PERIFÉRICOS

ANEXO E

COMUNICACIÓN CON LOS PERIFÉRICOS

E.1 MANEJO DEL MÓDULO "DISPLAY"

```
;  Codigos a usar:
;  Para la comunicacion I2C se requiere usar los siguientes codigos:
;
;  Recepcion de datos (comando, escritura o direccion de memoria) COD + NUMBYTES
;
;  1) envio de comando para el LCD (0001-0001): luego de enviado este codigo se
;  debe enviar uno de los siguientes comandos:
;
;  Borrar la pantalla del LCD (borrar la memoria)  0000-0001
;  Mover el cursor/memoria al inicio              0000-0010
;  Seteo del modo de entrada (bits opcionales)    0000-01DC
;  C = Sentido del cursor   0=Decrementar/1=Incrementar
;  D = Desplazamiento       0=No desplazo/1=Desplazo
;  Estado del LCD (tiene bits opcionales):        0000-1LCP
;  L = Estado del LCD       0=Apagado / 1=Encendido
;  C = Cursor del LCD       0=Apagado / 1=Encendido
;  P = Parpadeo del cursor  0=No parpadea / 1=Parpadea
;  Desplazamiento Cursor/Pantalla(bits opcionales) 0001-DM00
;  M = Moviento             0=Desplazo cursor/1=Desplazo display
;  D = Direccion            0=Izquierda/1=Derecha
;
;  2) Escritura en LCD (caracteres): se enviara (0010-0001) y a continuacion el
;  caracter a escribir, tomando en cuenta los codigos del LCD
;
;  3) Seteo de locacion de memoria LCD (0011-0001): luego de enviado este codigo se
;  debe enviar la direccion de memoria respectiva, disponiedo de 80 bytes en total
;  (de 0 a 79), internamente de convertira al valor requerido:
;  de  0-39: primera linea => de 00H a 27H
;  de 40-79: segunda linea => de 40H a 67H
```

```
;
;   4) Estado de la BackLight (con bit de opcion) - (010B-0000): Permite encender
;      o apagar la luz de BackLight del display (solo se envia 1 byte)
;      B = Estado Backlight   0=Apagado / 1=Encendido
;
;   5) Apagado Automatico (con bit de opcion) - (011T-0000): Permite encender o
;      apagar el timer interno de apagado automatico del display (1 minuto)
;      T = Estado del Timer  0=Apagado/Encendido
;
;      NOTA: apagara TODO el display, asi que debe usarse la opcion siguiente
;      para activar correctamente el Display
;
;      6) Encender o apagar Todo el LCD - (100E-0000):Permite apagar o encender
completamente el LCD
;      ademas al encender se regresa al estado anterior en que quedo el mismo
;      E = Estado del LCD    0=Apagado/Encendido
;
;   7) Presentar mensajes predeterminados en pantalla: (1100-XXXX) a continuacion
;      debe enviarse el numero de mensaje a presentar, siendo posible 256 textos
;      diferentes. La lista de texto esta presentada el la parte de constantes.
;      Cada texto esta predeterminado su lugar de ubicacion. Para los textos que
;      requieren seteo de valores extras, debe enviarse el numero de bytes extras
;      para presentar esos valores, el valor predeterminado de XXXX es 0001 (1 byte),
;      debe sumarse los bytes que seran enviados como valores extras para dicho mensaje.
;      El mensaje ubicara cada valor en pantalla de acuerdo a su programacion.
;
;   Envio de Datos: (lectura de memoria)
;
;   8) Simplemente se enviara el caracter que la memoria tenga disponible ese
;      momento, es decir, si se requiere un caracter de una ubicacion de memoria
;      especifica, DEBE enviarse previemante la direccion de memoria. EL Master
;      simplemente envia la peticion de lectura y recibe el byte.

;OJO: los textos estan justificas en un espacio de 16 caracteres
;                '*-*-*-*-*-*-*-*-'
;Valores a guardar en la memoria de informacion
        RSEG    INFO
```

```
; Menu general
Texto_1_1   DB     'Controlador V1.0'
Texto_1_2   DB     'A-Memoria  X-Fin'
;Texto_1_2   DB     'A-Memoria B-User'
;Menu Memoria 1
Texto_2_1   DB     'Total:    992 B '
Texto_2_2   DB     'Usada:       B '
;Menu Memoria 2
Texto_3_1   DB     ' Memoria Llena! '
Texto_3_2   DB     ' Total:    992 B'
;Menu salidas
Texto_4_1   DB     'Controlador y PC'
Texto_4_2   DB     ' ! CONECTADOS ! '
```

E.2 MANEJO DEL MÓDULO “TECLADO”

```
;   Codificacion para la comunicacion I2C
;
;   1) El teclado al detectar una tecla detendra su funcionamiento en espera del
;      polling, si no se detecta ninguna tecla y se le hace polling, se envia
;      (0000-0000) que significa: numero de bytes a enviar=0
;
;      Cuando se detecta una o mas teclas se envia el numero de teclas detectadas
;      y a continuacion las teclas detectadas, estas teclas se detectan solamente
;      si se presionan simultáneamente
```

E.3 MANEJO DEL MÓDULO “SALIDAS”

```
;   Rev:1.0 27-04-2009: se establece codificacion de envio y recepcion:
;
;   Recepcion: siempre se reciben 1 byte, este indicara la funcion y el numero de
;   bytes consecutivos enviados junto a este, tendra la siguiente estructura:
;   (CODF-NUMB), CODF= codigo de la funcion. NUMB= numero de bytes (0-15 bytes)
;
;   despues de analizar el primer byte, almacenara en el buffer los bytes
```

```
;   correspondientes y llamara a la funcion requerida.
;
;   las funciones posibles son (recibido desde MASTER):
;   Modulo de RX
;   1) recepcion del valor a guardar(1000-0010) CODF=1000 NUMB=2(0010)
;   2) recepcion del estado de las salidas(1010-0001) CODF=1010 NUMB=1(0001)
;   3) Borrado de memoria (1110-0000) CODF=1110 NUMB=0
;      si se recibe este byte debe borrarse la memoria Flash donde se encuentran
;      los datos capturados, comenzando desde cero con la captura
;   4 y 5) si se recibe (1010-0000) significa que se desactiva el control
;      de salida externo, Si se recibe (1011-0001) significa que se activa el control
;      de salida externo ademas del byte de salida actual, recepcion del estado
;      de las salidas externo(1011-0001) CODF=1010 NUMB=1(0001)
;
;   Modulo de TX (Primero Master envia peticion!)
;   TX1) peticion de envio del estado de las salidas(0010-0000) CODF=0010 NUMB=0
;      Master espera la recepcion de 1 byte con el estado de las salidas
;      despues de enviar este byte (se deja indicando para TX este byte)
;   TX2) peticion de envio del total de memoria capturada(0100-0000) CODF=0100 NUMB=0
;      Master esperara 2 bytes con el tamaño total de bytes a recibir
;      16 bits = 0 a 65535 bytes (0 en caso de no haber nada que enviar)
;       Nota: el total de bytes disponibles es de 992.
;      OJO: se complementa con la peticion siguiente:
;    TX3) Peticion de envio de los datos de la memoria capturada (0110-0000) CODF=0110
NUMB=0
;      indica que ahora debe enviarse la memoria capturada.
;
;      al finalizar, si se borra o no la memoria depende del envio del comando.
```

E.4 COMUNICACIÓN PROTOTIPO - PC

```
'Version 0.2
'
'1)Conexion con el PC: se envia desde el PC el siguiente byte:
'byte = (1000|0000) Codigo PC
'Se debe contestar con el siguiente byte
'byte = (1000|0001) Codigo Micro
'si el PC no recibe esta contestacion, dara por terminada la
comunicacion
```

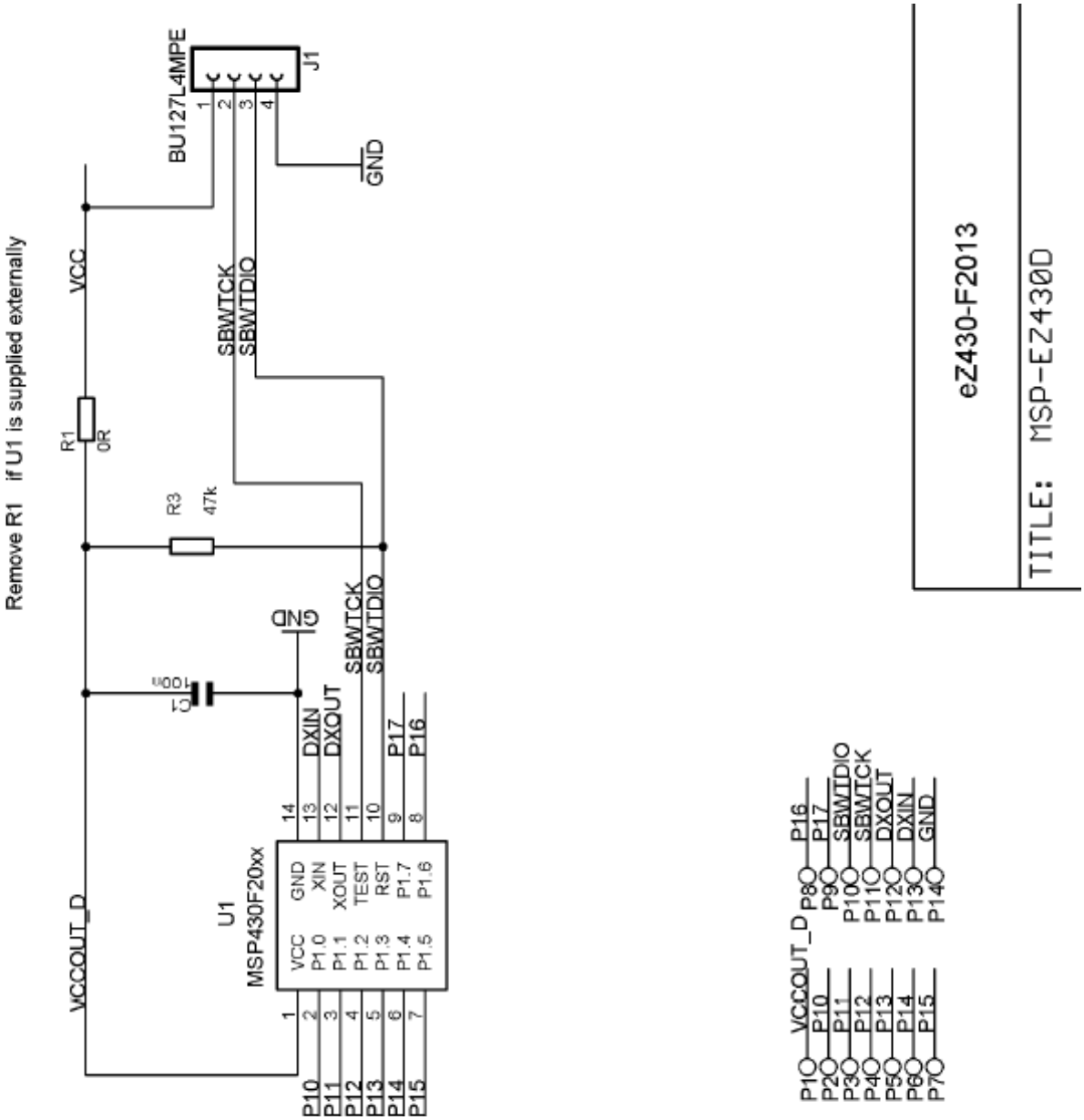
```
'
'Una vez establecida la comunicacion, el Micro enviara los datos
despues de recibir el codigo
'de PC, para iniciar una nueva transmision debe el PC realizar una
nueva peticion.
'Si el micro o el pc no reciben una respuesta, terminara la
comunicacion
'
'
'OJO: la temporizacion la llevara el PC, transmitira cada 250ms, pero
el micro estara pendiente
'cada 2 segundos, es asi que en caso de desconexion, el micro pasara a
estar desconectado y el PC tambien
'
'
'2)El valor de las salidas y entradas se enviara como un unico
paquete, se recibira
'primero un byte de anuncio y luego 9 bytes de datos
'Los 9 bytes estan determinados si no se ha implementado el envio de
otros datos!
'Byte de anuncio: (0001|1001) (el numero 9 esta indicado)
'
'Nota: las Salidas utilizaran la siguiente codificacion de 1 byte:
'Byte (0010|ABCD): A,B,C,D estado de la salida 1,2,3,4 respectivamente
'
'
'Al finalizar la recepcion, el PC debe enviar de nueva cuenta el
codigo de PC,
'para volver a pedir un envio de datos. El byte de salidas es el
mismo.
'
'
'3) Se permitira Sobre-escribir el valor de las salidas, sin importar
el estado
'actual seteado por el prototipo, se enviara un byte que indique la
activacion
'o desactivacion del mismo:
'(0011|000X) X=Activado/Desactivado= 1/0
'
'
'4) la obtencion de datos de la memoria Flash del micro se realiza
primero enviando
'del PC el byte (0100|0000), se espera el codigo de envio de memoria
(0100|0010),
'a continuacion se envia el total de bytes capturado LSB y MSB en ese
orden (el numero 2 esta indicado)
'Seguido el microcontrolador enviara los bytes correspondientes sin
esperar ninguna señal del PC
'
'
'Si se reciben todos los bytes sin ningun problema, el Pc puede o no
enviar el
'codigo de borrado de memoria (0101|0000), para indicar la
finalizacion el microcontrolador enviara
'el codigo (0101|0001). El tiempo que tarda en borrar la memoria es
minimo.
'
```

ANEXO F

PROGRAMADOR PARA LOS DISPOSITIVOS DE LA FAMILIA MSP430-F20XX

ANEXO F

F.1 PROGRAMADOR PARA LOS DISPOSITIVOS DE LA FAMILIA MSP430-F20XX



Fuente: Texas Instruments; SLAU176A.pdf.