UNIVERSIDAD DE ZARAGOZA

ESCUELA UNIVERSITARIA DE INGENIERIA

TÉCNICA INDUSTRIAL DE ZARAGOZA

Escuela Universitaria Ingeniería Técnica Industrial ZARAGOZA

Memoria del proyecto fin de carrera:

Sistema de control para máquina cortacésped basado en microcontrolador PSoC

Dirigido por:

Antonio Bono Nuez

Realizado por:

Víctor Arnal Herguedas

INDICE

1. INTRODUCCIÓN

El proyecto fin de carrera *"Sistema de control de máquina cortacésped basado en PSoC"* tratará de mostrar al lector las características y posibilidades que ofrecen los "embedded systems" a través de una amplia explicación del PSoC CY8C29466-24PXI, tanto del software como del hardware, y mediante la creación de un sistema de control estándar para controlar ciertos aspectos de una máquina cortacésped. En este capítulo se va a contextualizar el proyecto, se explicarán los motivos del desarrollo del mismo y su alcance.

Los "embedded systems" o sistemas empotrados se basan en la tecnología de microcontroladores PSoC (Programmable System on Chip) de Cypress. El PSoC CY8C29466 de Cypress es un microcontrolador de 8 bits muy versátil ya que dispone de recursos analógicos (amplificadores operacionales, sensor de temperatura, conversor AD y DA, filtros,…) y digitales (temporizadores, contadores, moduladores de ancho de pulso (PWM), comunicación serie, generadores de números pseudo-aleatorios,…) completamente configurables en el mismo encapsulado. La arquitectura del microcontrolador, ofrece un máximo de 16 bloques digitales y un máximo de 12 bloques analógicos. Cada bloque tiene una serie de registros que determinan su funcionalidad y su conectividad con otros bloques, multiplexores, buses o puertos E/S.

El proyecto está confeccionado por 6 capítulos, que he intentado desarrollar de manera lineal para su fácil comprensión incluso para los lectores con bajos conocimientos en electrónica, que los cuales se indican a continuación:

1. Introducción: En este bloque se introduce al lector en los contenidos del proyecto final de carrera y se muestra la declaración de los objetivos a alcanzar en el desarrollo del mismo.

2. Revisión histórica de las máquinas cortacésped y situación actual: Se puede ver un pequeño seguimiento de la evolución de las máquinas cortacésped a lo largo de la historia, así como una reseña de la empresa en la que realice los estudios previos a la creación del sistema de control final.

3. PSoC: Programmable System on Chip: En este capítulo se muestran todas las características tanto hardware como software del microcontrolador utilizado para la realización del proyecto, el PSoC de Cypress, perteneciente a la familia de los "embedded systems".

4. Sistema de control estándar para máquinas cortacésped: Una vez conocido el microcontrolador utilizado, descrito en el capítulo 3, se muestran por separado cada una de las aplicaciones del mismo en una máquina cortacésped y posteriormente el diseño conjunto del sistema final.

5. Conclusiones: Como su nombre indica, en este capítulo se encuentan las conclusiones obtenidas a lo largo de la confección de esta proyecto final de carrera.

6. Bibliografía y agradecimientos: En este último capítulo, el lector podrá observar la bibliografía consultada.

1.1 Motivación y objetivos

La posibilidad de realizar unas prácticas en una empresa extranjera, para la que podía desarrollar un sistema de control que pudiera mejorar algunos aspectos de los productos que desarrollaban suponía un gran desafío. Tras hablar con los ingenieros de la empresa, llegamos a la conclusión que la creación de un sistema de control electrónico para suplir las carencias tanto de algunos actuadores existentes en las máquinas como la adquisición de nuevos datos de interés para el usuario final del producto, mejoraría enormemente la seguridad y la funcionalidad de sus máquinas cortacésped.

.

Por tanto, y debido a lo expuesto, los objetivos generales de este proyecto fin de carrera son:

1- Identificación del tipo de sensores y actuadores incorporados en las máquinas cortacésped y la posibilidad de mejorar los mismos mediante un sistema de control electrónico.

2- Conocimiento personal de una nueva herramienta de trabajo para el desarrollo de sistemas electrónicos, el microcontrolador PSoC, asi como la mejora en el ámbito de la programación en lenguaje C.

3- Desarrollo de un sistema de control estándar basado en el microcontrolador de Cypress para máquinas cortacésped

4- En el ámbito personal, conocer el funcionamiento de trabajo de una empresa en lo relacionado con el diseño y el desarrollo de nuevos sistemas de control, asi como el aprendizaje de una lengua extranjera nueva, el italiano.

1.2 Descripción y alcance del proyecto

El objetivo principal de este proyecto final de carrera aquí expuesto ha sido el desarrollo de un sistema de control estándar para máquinas cortacésped que permita el control de una determinada serie de sensores y, consecuentemente, de variables de interés para el usuario final de la máquina. Se ha pretendido realizar un sistema de control lo suficientemente versátil que pueda ser incorporado a cualquiera de los modelos producidos por la empresa.

Previo comienzo del diseño del dicho sistema, debía introducirme en el manejo del microcontrolador que iba a utilizar, el PSoC CY8C29466-24PXI, ya que no lo había utilizado previamente y requería de una preparación previa. A partir de un kit de evaluación que adquirí (sus contenidos y utilidades serán desarrollados en el punto 3.2 de esta memoria) los conocimientos necesarios para utilizar cada uno de los bloques tanto digitales como analógicos de este microcontrolador, asi como la inicialización de los mismos y las instrucciones en código C para su correcto funcionamiento. Una vez conocido el manejo del microcontrolador y desarrollado un sistema de control basado en el mismo, tuve que aprender a manejar las siguientes aplicaciones: Matlab para el desarrollo de una aplicación que representara gráficamente las señales obtenidas por el sistema creado con el PSoC y posteriormente, Altium Designer 2009 para la realización de un prototipo en una placa de circuito impreso.

Como ya he comentado, el objetivo final conseguido permitirá controlar, en tiempo real, una serie de variables de utilidad para el usuario de una máquina cortacésped y además, contar con la posibilidad de utilizar el sistema en cualquier máquina cortacésped.

2. REVISION HISTÓRICA DE MÁQUINAS CORTACÉSPED Y SITUACIÓN ACTUAL

2.1 Introducción

Como su nombre indica, un cortacésped o cortadora de césped es una máquina, manual o motorizada, usada para cortar el césped de forma que se obtenga una alfombra de hierba de altura uniforme. Estas máquinas evolucionaron de forma paralela a los motores de gasolina, ya que cuando estos últimos fueron los suficientemente pequeños y potentes supuso una revolución para ellas. En los siguientes apartados realizaré una pequeña revisión de la evolución en la historia de estas máquinas.

2.2 Historia de las máquinas cortacésped

La primera cortadora de césped fue inventada por Edwin Budding, en el año 1827, en Inglaterra, siendo patentada en 1830. Está segadora o cortacésped fue diseñada principalmente para cortar el césped en campos deportivos y grandes jardines, se inventó como una alternativa mejor a la guadaña. Aún así le costó diez años innovarla y conseguir que la segadora pudiera utilizarse mediante el uso de animales.

Fig. 2.1. Primera máquina cortacésped empujada por animales

En un acuerdo entre John Edwin Ferrabee y Edwin Budding el 18 de mayo de 1830, Ferrabee pagó los costes del desarrollo y obtuvo las cartas de las patentes y adquirió los derechos para fabricar, vender y conceder licencias a otros fabricantes en la producción de las cortadoras de césped. Thomas Green produjo la primera cortadora de césped en el año 1859, y en la década de 1860 comenzó la fabricación de las cortadoras de césped.

En 1862, la compañía de Farrabee fabricaba ocho modelos diferentes, cuya diferencia entre ellas era el tamaño de los rodillos que usaban. En 1870, Elwood McGuire de EE.UU., diseño una cortadora de césped que funcionaba empujada por ser humano, muy ligera lo que conllevo a un éxito comercial. Más tarde, en 1899, John Burr patentó una segadora con mejoras en el rotor y las paletas, con la colocación de un cambio en la rueda, para mejorar su rendimiento.

Fig. 2.2. Primera máquina cortacésped empujada por el hombre

JP Ingeniería de Leicester, se fundó después de la Primera Guerra Mundial, produjo una serie de cortacespedes muy populares. En este tiempo, el operador se montaba detrás de los animales que tiraban de las máquinas de gran tamaño, estas fueron las primeras segadoras de este estilo. El aumento de la popularidad de los deportes que se practicaban sobre césped ayudó a que la propagación de máquina corta césped.

Las cortadoras de césped se convirtieron en una alternativa más eficiente a la guadaña y al pastoreo de animales. En 1893, James Sumner, de Lancashire patentó la primera cortadora de césped a vapor, utilizaba como combustible gasolina o queroseno. Después de numerosos avances, las máquinas fueron vendidas por Stott Fertilizer e Insecticide Company de Manchester.

Los cortacespedes rotativos no se desarrollaron hasta que los motores fueron lo suficientemente pequeños y sobre todo potentes como para ejecutar las hojas a una alta velocidad, muchas personas experimentaron con las cuchillas rotativas a finales de 1920 y principios de 1930, y Power Specialties Ltd. presentó un cortacésped de gasolina.

Fig. 2.3. Primera máquina cortacésped propulsada por un motor de gasolina

A partir de entonces, las máquinas cortacésped fueron mejorando a la vez que lo hacían los motores de gasolina. Igualmente, la evolución en otros campos como en la automovilística, siderurgia, electrónica, etc. hizo que la máquina evolucionara hasta las máquinas cortacésped que existen en la actualidad.

Fig. 2.4. Máquina cortacésped actual

2.3 Grillo S.p.A.

Fig. 2.5. Logo de la empresa Grillo SpA

Grillo SpA es una empresa italiana nacida en 1953 dedicada a la producción y venta de máquinas agrícolas y de jardín. Está situada en la ciudad de Cesena, en la región de Forlì-Cesena (FC). Una pequeña reseña histórica de la empresa es la siguiente:

En 1953 nació el primer auténtico motocultor "GRILLO": era una máquina que podía hacer óptimamente muchos trabajos. La idea ganadora fue de separar el vehículo autopropulsado del accesorio. Con las azadas montadas se cavaba, con el arado se araba, con el remolque se transportaba, con la bomba se rociaba etc. En 1955 se inició la primera producción de serie: 15 máquinas en un año.

Hoy en día se venden decenas de millares de máquinas con estas características alrededor del mundo .Estos inicios, en los años siguientes, permitieron la creación de otros interesantes modelos como desbrozadoras, motoazadas y carretillas motorizadas.

GRILLO proyecta y fabrica máquinas para la agricultura profesional y el mantenimiento de grandes áreas verdes, así como de los jardines y huertos domésticos. Sus máquinas nacen de estudios de ingeniería aplicados a la utilización diaria, con el objetivo de crear nuevas soluciones técnicas capaces de facilitar el trabajo del hombre.

Para proyectar y producir, disponen de un eficiente equipo de técnicos y diseñadores. La producción industrial se realiza mediante la utilización de herramientas de alto nivel tecnológico tales como corte con láser, robots, línea de montaje parcialmente automatizada, etc. GRILLO produce 20.000 máquinas al año y en una gama de más de 50 modelos diferentes, con sus 200 empleados.

Fig. 2.6. Sede central de Grillo SpA en Cesena

Fue en esta empresa donde realice un stage entre noviembre de 2010 y agosto de 2011, donde diseñé los primeros prototipos sobre los que después basaría este proyecto final de carrera. Cuando llegué allí, pregunté a los ingenieros que tipo de sensores podían ser incorporados a las máquinas o que variables era interesante capturar y procesar para mejorar la seguridad o el uso de la máquina para el usuario. Mientras que alguno de los actuadores que incorporaban actualmente las máquinas no funcionaban como a los ingenieros les gustaría, otros sistemas estaban siendo estudiados pero no habían comenzado su desarrollo.

Entonces, me puse de acuerdo con ellos para conocer qué tipo de cambios querían realizar en los actuadores existentes y que sistemas nuevos querían incorporar. Ellos ya habían adquirido algunos sensores para incorporarlos a las máquinas, pero carecían de un sistema para controlarlos. Para los sistemas nuevos que consideraban interesantes, yo me encargué de buscar unos sensores adecuados.

Uno de los grandes problemas con los que me encontré, fue el medio en el que los sistemas debían funcionar. Al tratarse de máquinas cortacésped, debía escoger sensores robustos que resistieran situaciones adversas como suciedad, polvo e incluso agua. Además de idear un sistema que satisficiera las necesidades de los ingenieros, debía encontrar el sensor adecuado y en algunos casos, fabricar una carcasa o medio de protección de los sensores para evitar que sufrieran daños. En cualquier caso, los ingenieros de la empresa siempre me ofrecieron ideas y soporte frente a las dificultades que encontraba y en las que me ofrecer una solución.

2.4 Electrónica en máquinas cortacésped

En los últimos tiempos, la electrónica ha aparecido prácticamente en todos los campos. En el campo de las máquinas cortacésped, además de introducir sistemas de control de variables de interés para el usuario en los modelos actuales, se han creado sistemas autónomos para realizar estas tareas de forma programada y automática. Los robots cortadores son unos cortacéspedes automáticos guiados por un alambre que define el área a cortar.

El robot utiliza este cable para buscar el límite de la zona a recortar y en algunos casos para localizar a un muelle de carga. Este tipo de cortadoras de césped robóticas son capaces de mantener hasta 5 hectáreas (20.000 m2) de césped. Son cada vez más sofisticadas y contienen cada vez más sensores haciendo estas máquinas prácticamente autónomas.

A partir de un cable perimetral colocado sobre la zona de corte, este robot es capaz de recrear un mapa de la zona de trabajo y de forma programada por el propietario, realizar el corte del césped. Estás máquinas incorporar un microprocesador que guardará datos de la región de corte además de recoger los datos de los sensores montados sobre la máquina y actuar cuando las condiciones sean las propicias para realizar su trabajo. Estas máquinas totalmente eléctricas, cuentan con una estación de carga a la que son capaces de regresar cuando su batería es baja reduciendo al mínimo su interaccion con las personas.

Fig. 2.7. Robot cortacésped y detalle del display que incorpora

2.5 Conclusiones finales

Como en todos los campos, la electrónica cada vez está más presente en las máquinas cortacésped. La inclusión de sistemas electrónicos en las mismas requiere de un estudio previo que determine el entorno en el que la máquina va a trabajar, a que situaciones puede ser sometida y otros aspectos que llevarán al diseñador a utilizar unos sensores u otros. En mi caso, he concluido que el estudio del entorno en el que la máquina desarrollará su trabajo es realmente importante, ya que determinará el uso de un tipo determinado de sensor y las precauciones y medidas de seguridad que se deberán tomar para evitar daños en el sistema electrónico.

Igualmente, hay una gran diferencia entre el diseño creado en el laboratorio y el diseño final que se montará sobre la máquina porque siempre aparecen variables o situaciones como vibraciones, ruidos, etc. que en el diseño no tienes en cuenta y que posteriormente debes modificar para evitarlos.

3. SISTEMA DE CONTROL BASADO EN PSoC

Antes de comenzar el diseño de cualquier sistema electrónico es importante tener en cuenta los posibles componentes que serán utilizados en el mismo y las posibilidades que ofrece cada uno de ellos. Como elemento principal de un sistema electrónico, el microprocesador o microcontrolador que incorporará, deberá garantizar la capacidad de satisfacer todas las necesidades que pueda requerir dicho sistema.

Además, la interfaz de trabajo de este instrumento también ayudará al diseñador a decantarse por la elección de un microcontrolador u otro pero antes de decidirse entre uno u otro, el diseñador deberá conocer las necesidades de su sistema y la capacidad del microcontrolador para satisfacer todas ellas. Aspectos a tener en cuenta a la hora de realizar la decisión de la utilización de uno u otro microcontrolador pueden ser la capacidad de integración del mismo, la facilidad de programación, la interfaz de usuario que este ofrece o el coste del mismo.

Por tanto, tomando en cuenta los aspectos mencionados realizaré una comparación entre dos microcontroladores: el microcontrolador de Freescale MC9S08QG8 y el microcontrolador de Cypress PSoCCY8C9466-2PXI.

3.1 Propuesta de sistema de control

Como ya se ha comentado previamente, el objetivo de este proyecto final de carrera consiste en la creación de un sistema de control de algunas variables de interés para el usuario de una máquina cortacésped. En el punto 4 de esta memoria, se puede encontrar una minuciosa explicación de todos los aspectos concernientes al control de estas variables que a grandes rasgos consisten en el control de tres sensores y en la habilitación de una comunicación serie con el PC a través del puerto serie.

Para realizar el control de dichos sensores así como del puerto serie era necesaria la elección de un microcontrolador que tuviera la capacidad tanto de procesar los datos arrojados por estos así como de ofrecer una respuesta a los datos ofrecidos por estos. En la asignatura de microprocesadores e instrumentación electrónica tuve la oportunidad de conocer el funcionamiento de los microcontroladores.

El µC utilizado en la misma fue el MC9S08QG8 de Freescale que ofrecía una visión global del funcionamiento de los mismos y que me permitiría posteriormente la posibilidad de controlar otros µC diferentes.

Ya que el µC utilizado en el sistema final de control es el PSoC CY8C29466 de Cypress mostraré algunas de las características de los mismos así como sus interfaces de trabajo para justificar el uso del mismo. También tuve en cuenta la posibilidad de adquirir un STARTER KIT para poder conocer las capacidades de cada uno de estos µC antes de incorporarlos al sistema final de control.
Un STARTER KIT consiste en una pequeña placa que incorpora un µC el cual permite explotar todas las propiedades del mismo, permitiendo al usuario tener una idea global de las posibilidades que ofrece. Suele incorporar además un software y un hardware de control para poder programar al µC posibilitando así la creación de infinidad de pruebas.

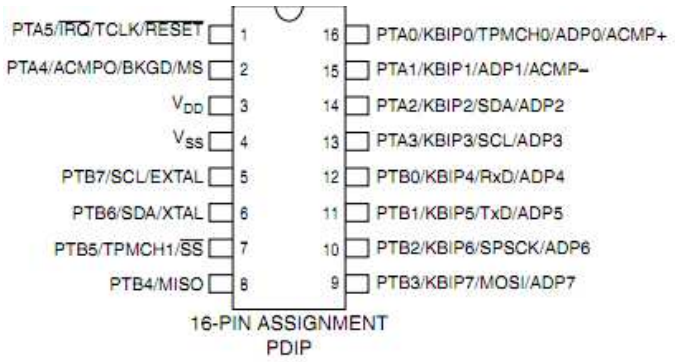
A continuación mostrare algunas de las propiedades de cada uno de los dos µC mencionados y las características que me llevaron a decantarme por uno de ellos.

3.1.1 Microcontroladores: MC9S08

El MC9S0QG8 es un microcontrolador de 8 bits de la familia FREESCALE que integra el CPU HCS08 de 20MHz. Este µC integra 8KB de memoria FLASH, 512B de RAM, un oscilador interno, hasta 14 pines de E/S y bloques de comunicación (SCI,SPI e I2C), temporización, conversión AD, etc. y se presenta en un encapsulado DIP de 16 bits.

El usuario dispondrá de hasta 14 GPIO´s (General Purpose Input/Output, E/S de propósito general) con las que podrá desarrollar infinidad de sistemas. Es uno de los aspectos importantes a tener en cuenta, ya que la cantidad de GPIO´s disponibles determinará la capacidad de integración del numero de variables externas controlables por el mismo. Cuenta con un interfaz de desarrollo de programaciónde los mismos, el Code Warrior, que incluye diversos módulos como un editor/compilador de lenguaje ensamblador y C, simulador y depurador. También dispone de un STARTER KIT que comentare a continuación.

En la siguiente figura se muestra el patillaje del S08 con un encapsulado DIP 16 y su mapa de memoria correspondiente:



0000-005F	Página de registros directos	96 bytes
0060-025F	Memoria RAM	512 bytes
0260-17FF	NO IMPLEMENTADO	5536 bytes
1800-184F	Página de registros altos	80 bytes
1850-DFFF	NO IMPLEMENTADO	51120 bytes
E000-FAFF	Memoria FLASH	
FFB0-FFBF	Registros no Volátiles	8192 bytes
FC00-FFFF	Vectores	

Fig. 3.1 Asignación de pines del S08QG8 y mapa de memoria

Como se puede observar, cada patilla puede adoptar diversas configuraciones. Cuenta con dos puertos, PTA y PTB, de propósito general que cuentan con 6 y 8 bits respectivamente mostrados en la asignación de pines como PTAx y PTBx, siendo x el número de bit en el puerto. También se puede observar que la parte baja del byte de estos puertos puede adoptar la configuración KBIPx, cuyas siglas significan *KeyBoard Interrupt Pin* o Pin de interrupción de teclado, que permitirá al usuario crear interrupciones a través de la configuración de estos.

Ya he comentado, que el S08 cuenta con un conversor AD de 10 bits y 8 canales integrado, al cual accederemos a través de la configuración tanto de los registros asociados asi como de los pines señalados con las siglas ADP (pines 9 a 16, ambos incluidos) y también, podemos utilizar un comparador analógico configurando los pines nombrados con ACMP+ y ACMP-.
Además de los conversor AD y del comparador analógico, el S08 cuenta con varios protocolos de comunicación serie. Configurando correctamente los registros y los pines asociados podremos realizar protocolos I2C a través de los pines SCL y SDA (pin 13 y pin 14) , SCI a través de los pines TxD y RxD (pin 11 y pin 12), SPI mediante los pines SS, MISO, MOSI y SPSCK (en los pines 7, 8, 9 y 10).

En los siguientes puntos mostrare algunas de las características tanto del starter kit como del software de Freescale, el Code Warrior.

- STARTER KIT DEMO9S08QG8

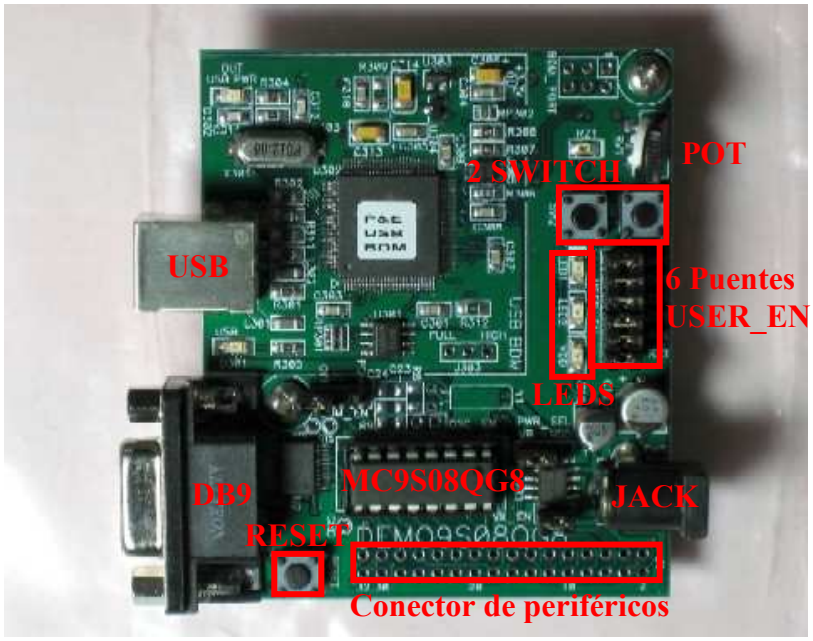


Fig. 3.2 Starter kit DEMO9S08QG8 de Freescale

La DEMO9S08QG8 es una tarjeta basada en el µC de Freescale. Esta tarjeta se utilizará a través del interfaz gráfico comentado, el Code Warrior, que permitirá al usuario editar y compilar programas así como su grabación en la memoria Flash del S08 que está integrado en la placa y además la opción de ejecutar dichos programas para comprobar su funcionamiento. Esta tarjeta incluirá, entre otras, las siguientes características:

1. El microcontrolador MC9S08QG8 con su encapsulado DIP16, alimentado a 3.3V y a 4 MHz.

2. Interfaz USB BDM de Motorola, que es la zona marcada por un cuadrado blanco en la placa Demo.

3. Un puerto USB desde el que se alimentará a la placa. También se puede alimentar con una fuente de 9V a través del regulador a 3.3V que incorpora.

4. Algunos periféricos: 3 interruptores (uno de ellos con la función de RESET), 3 diodos LED (uno de ellos conectado a Vcc), un potenciómetro, un fototransistor, un conector Jack y un conector DB9 para comunicación serie.

Además incluye un conector al que podremos conectar periféricos externos dependiendo de nuestras necesidades. Para poder utilizar esta placa en el ordenador, es necesario contar con el software Code Warrior de Freescale que describiré a continuación.

- CODE WARRIOR

Este entorno de desarrollo creado por Freescale permite realizar sistemas basados en la familia de µC HC(S)08. Este software incluye editor, ensamblador, compilador en C, simulador y depurador de hardware. Supone una ventaja la posibilidad de poder crear proyectos en varios lenguajes de programación. Para iniciar la creación de un proyecto se deben de seguir una serie de pautas. Una vez instalada la placa DEMO en el ordenador, se puede ejecutar el progama:

Fig. 3.3 Ventanas del programa Code Warrior de Freescale

Al crear un nuevo proyecto, deberemos elegir de la lista el µC con el que se va a trabajar y seguidamente el tipo de configuración que se desea. Normalmente, la configuración elegida será "Full Chip Simulation" que permitirá realizar simulaciones con todos los periféricos. Después se deberá elegir el código de programación que se utilizará en la confección del programa y una vez creado el proyecto la ventana de explotación donde se podrá confeccionar el código será la siguiente:

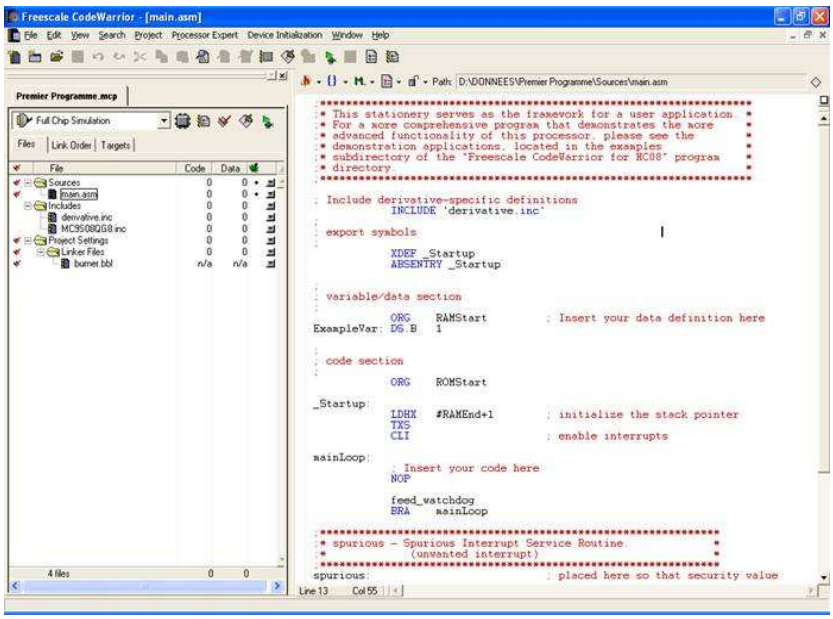


Fig. 3.4 Interfaz de trabajo del programa Code Warrior

En la derecha de la pantalla, se podrá confeccionar el código ensamblador o código C que posteriormente será grabado en el S08 y también se podrán modificar los registros de los diferentes periféricos de éste para realizar diversas tareas.

El acceso a dichos registros se puede realizar a través del menú desplegable que aparece en la izquierda de la pantalla, donde están todos los archivos relacionados con el proyecto que se está realizando. Una vez confeccionado el código, se puede compilar en busca de errores y posteriormente grabarlo en el S08 para comprobar su funcionamiento.

La necesidad de un mayor número de GPIO´s para controlar todos los sensores que adquirieran variables de interés y su posterior procesamiento acompañado de una interfaz gráfica mucho más novedosa y configurable, me llevo a tomar la decisión de utilizar el PSoC CY8C29466 que al ser una herramienta mucho más moderna y potente permite desarrollar sistemas totalmente configurados por el usuario, con una capacidad de hasta 24 GPIO´s configurables.
En los siguientes apartados mostraré todas las características y propiedades del microcontrolador de Cypress así como de su interfaz de usuario, el PSoC Designer IDE.

3.2 PSoC CY8C29466-24PXI

El PSoC modelo CY8C29466-24PXI es el microcontrolador que he escogido para realizar mi sistema de control por su gran facilidad de configuración, tanto del software como del hardware. Incluido en la familia de los "embedded systems" (del inglés, sistemas empotrados), permiten reducir sustancialmente el tamaño de cualquier diseño electrónico gracias a su alto número de bloques configurables (Conversores A/D, D/A, amplificadores de ganancia programables, PWM, UART, etc.) permitiendo obtener un producto final con un alto nivel de prestaciones en un espacio realmente reducido.

Fig. 3.5 Detalle del PSoC CY8C29466-24PXI

La programación del mismo se realiza a través de un software creado por Cypress, el PSoC Designer, que permite realizar la configuración tanto del SW como del HW del PSoC con una interfaz gráfica sencilla e intuitiva que ofrece un amplio abanico de opciones, dando la oportunidad al usuario de realizar un sistema de acuerdo a sus necesidades gracias a la alta personalización de cada uno de los bloques configurables mencionados, y de los que posteriormente, realizaré un análisis a fondo.

Además, permite al usuario realizar su proyecto en 2 lenguajes de programación (C o ensamblador) facilitando aun más su tarea.

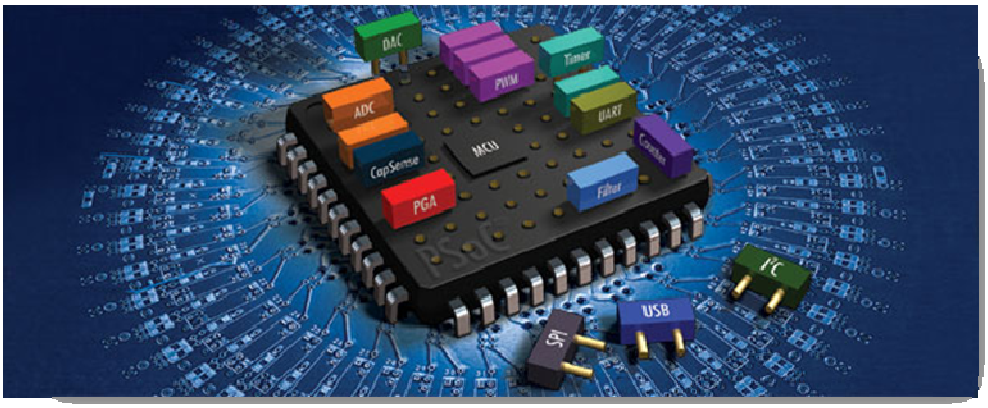


Fig. 3.6 Imagen ilustrativa de los innumerables bloques que incluye el PSoC

A continuación comentaré todos los aspectos reseñables de este microcontrolador, valores típicos, características eléctricas, configuraciones posibles, software de programación, así como un detallado análisis de la configuración de algunos de los bloques más importantes y de igual manera, los utilizados por mí para la realización del proyecto.

3.2.1 Introducción/Generalidades

El PSoC CY8C29466-24PXI de Cypress es un microcontrolador de 8 bits con una alta capacidad de configuración, tanto de software como de hardware. Dentro del mismo podemos encontrar infinidad de bloques de usuario, analógicos y digitales, incluso de comunicación. Estos son:

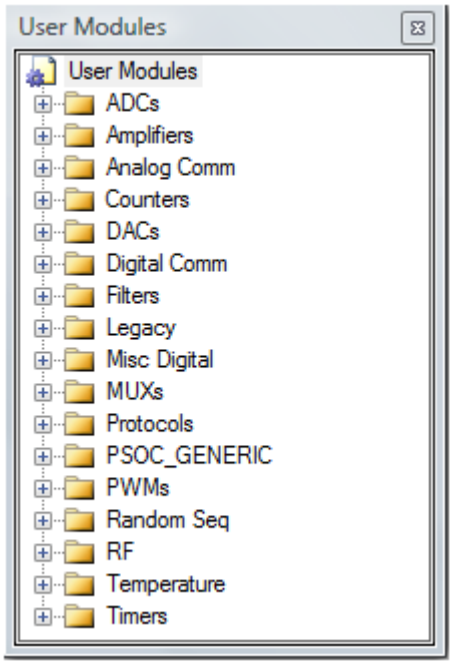
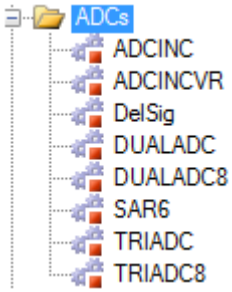


Fig. 3.7 Muestra de todos los Módulos de usuario con los que cuenta el PSoC

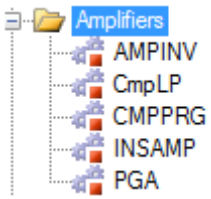
A. Bloques analógicos: Cada uno de los mencionados a continuación, son los bloques genéricos de carácter analógico, incluyendo cada uno diversidad de configuraciones posibles. La enumeración de los bloques de usuario se rige por la aparición en el software de Cypress.

- Conversores A/D



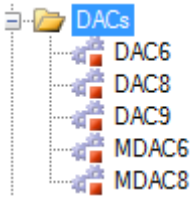
Dentro de los Conversores A/D, podemos encontrarlos incrementales, incrementales con número de bits configurables, Delta Sigma, de doble y triple entrada de 8 bits, de doble y triple entrada con numero de bits configurables y por último, uno de aproximaciones sucesivas.

- Amplificadores



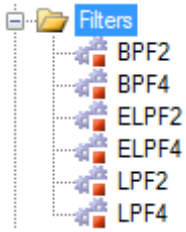
Los más comunes que podemos encontrar dentro de los amplificadores son los de ganancia programable, los inversores y los de instrumentación, los cuales posibilitan una amplia gama de ganancias. También es posible encontrar amplificadores configurados para funcionar como comparadores de baja potencia y como comparadores programables.

- Conversores D/A



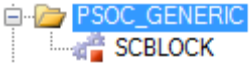
Los Conversores D/A genéricos disponibles son de 6, 8 y 9 bits. También es posible utilizar DACS que multiplican la salida analógica mediante transformaciones de los bits de entrada digitales.

- Filtros



En el bloque de filtros, encontramos entre otros filtros de paso de banda y filtros paso bajo. Ambos tipos con la posibilidad de elegirlos de 2 o 4 polos.

- Bloques genéricos del PSoC

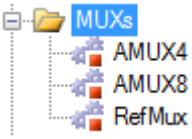


Dentro de este apartado podemos encontrar un bloque con una alta capacidad de configuración. Se trata de un condensador controlado por interruptores (Switched Capacitor BLOCK)

- Sensor de Temperatura

Por último, tenemos un bloque para la medición de la temperatura de la memoria Flash.

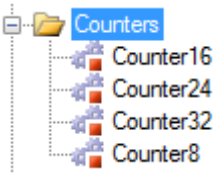
- Multiplexores



Estos multiplexores analógicos permiten seleccionar al usuario la entrada analógica que entrará en uno de los bloques CT (Continous Time), normalmente amplificadores. Es posible elegir entre 4 y 8 entradas.

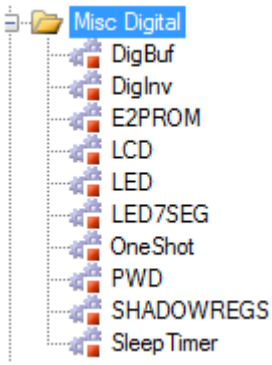
B. Bloques Digitales: Los bloques digitales que podemos encontrar incluyen contadores, moduladores de ancho de pulso (PWM), multiplexores, etc. A continuación se muestra una enumeración de los mismos:

- Contadores



Los contadores que podemos encontrar son de 8, 16, 24 o 32 bits y tienen una fácil configuración tanto software (interrupciones para lectura, reset de contaje, etc.) como hardware (ck de contaje, nº de bits...).

- Miscelánea Digital

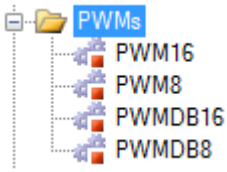


En este bloque encontramos una gran diversidad de funcionalidades muy útiles que facilitan mucho la labor del programador. Un claro ejemplo es el módulo del LCD que permite una sencilla escritura en el mismo con un simple par de instrucciones evitando así la típica rutina para inicializarlo.

También podemos encontrar módulos como el E2PROM que permite leer/escribir datos en memoria no volátil, el módulo de control de pantallas de 7segmentos de hasta 8 dígitos, un Sleep timer(para dejar el micro en suspensión), un módulo para configurar los GPIO´s en los que conectaremos un diodo LED,

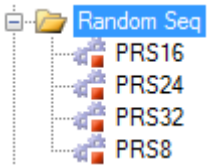
Así como otros bloques interesantes: Un buffer digital(un amplificador en modo seguidor), un inversor lógico, un generador de 1 pulso en respuesta a una señal entrada especificada (One Shot), un discriminador de ancho de pulso (PWD) y los SHADOW registers (que permiten utilizar de manera simultánea un puerto para varias tareas sin que se vean afectadas cada una de ellas por separado).

- Moduladores de ancho de pulso (PWM)



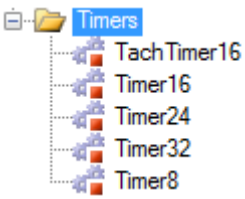
Este bloque permite integrar en el PSoC un PWM y su configuración dependiendo de las necesidades del programador. Así, este puede ser de 8 o 16 bits y puede tener 8 bits de generador de banda muerta (Dead Band Generator). Es posible configurar el ck del PWM permitiendo obtener cualquier tipo de señal

- Generador de secuencias pseudo aleatorias (PSR)



Este bloque proporciona una secuencia de números aleatorios que te permite establecer el algoritmo que regirá esta secuencia de números. Un bloque bastante útil para realizar, por ejemplo, juegos de azar (un dado, una carta, etc.)

- Temporizadores (Timers)



Los timers, parecidos a los contadores, pero conceptualmente diferentes pueden escogerse con diferente número de bits, siendo estos 8, 16, 24 o 32 bits y al igual que los contadores se pueden configurar totalmente (Sw y Hw).

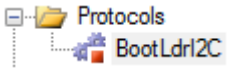
C. Bloques de comunicación: Es necesario realizar un apartado para enumerar las posibilidades que tiene el PSoC para comunicarse con otros dispositivos. Estos protocolos de comunicación son tanto digitales como analógicos y son los siguientes:

- Comunicación analógica

El DTMF (Dual Tone Multiple Frequency) es un generador de señales configurado como un conversor D/A de 6 bits, que envía información en forma de tonos analógicos a una frecuencia determinada por el usuario.

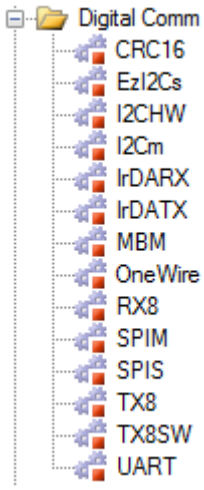
- Comunicación digital

  - Protocolo BootLoader I²C



Este bloque de usuario permite realizar una configuración inicial al PSoC mediante el protocolo de comunicación I²C

  - Módulos



Como se puede observar, existe una gran variedad de módulos de usuario de comunicación digital. Entre los protocolos más conocidos podemos destacar:

UART (Universal Asynchronous Receiver Transmitter), I²C (Inter-Integrated Circuits) , SPI (Serial Peripherial Interface) y OneWire(un cable) los cuales permiten realizar la comunicación entre el PSoC(en el papel de maestro) y un número n de dispositivos (esclavos) con solo dos hilos, que variará su función dependiendo del protocolo.

Otros bloques de comunicación digital que podemos utilizar son los protocolos de CRC (Cyclical Redundancy Check), recepción IR, MBM (Mail Box Manager) y receptor serie de 8 bits(para RS-232, UART separada en módulos TX y RX independientes).

A la vista del alto número de bloques que incluye el PSoC, es fácil tener una visión global de todas las posibilidades que ofrece este microcontrolador, el cual permite reducir el número de componentes electrónicos periféricos externos de cualquier diseño, el tiempo de diseño y, en consecuencia, el coste del proyecto.

En los próximos apartados, procederé a ampliar toda la información sobre el PSoC citada hasta ahora.

3.2.2 Arquitectura

La imagen que se muestra a continuación, disponible en el TRM (Technical Reference Manual) de Cypress, dispone la situación de todas las partes (numeradas) que componen la arquitectura interna del PSoC CY8C29x66:

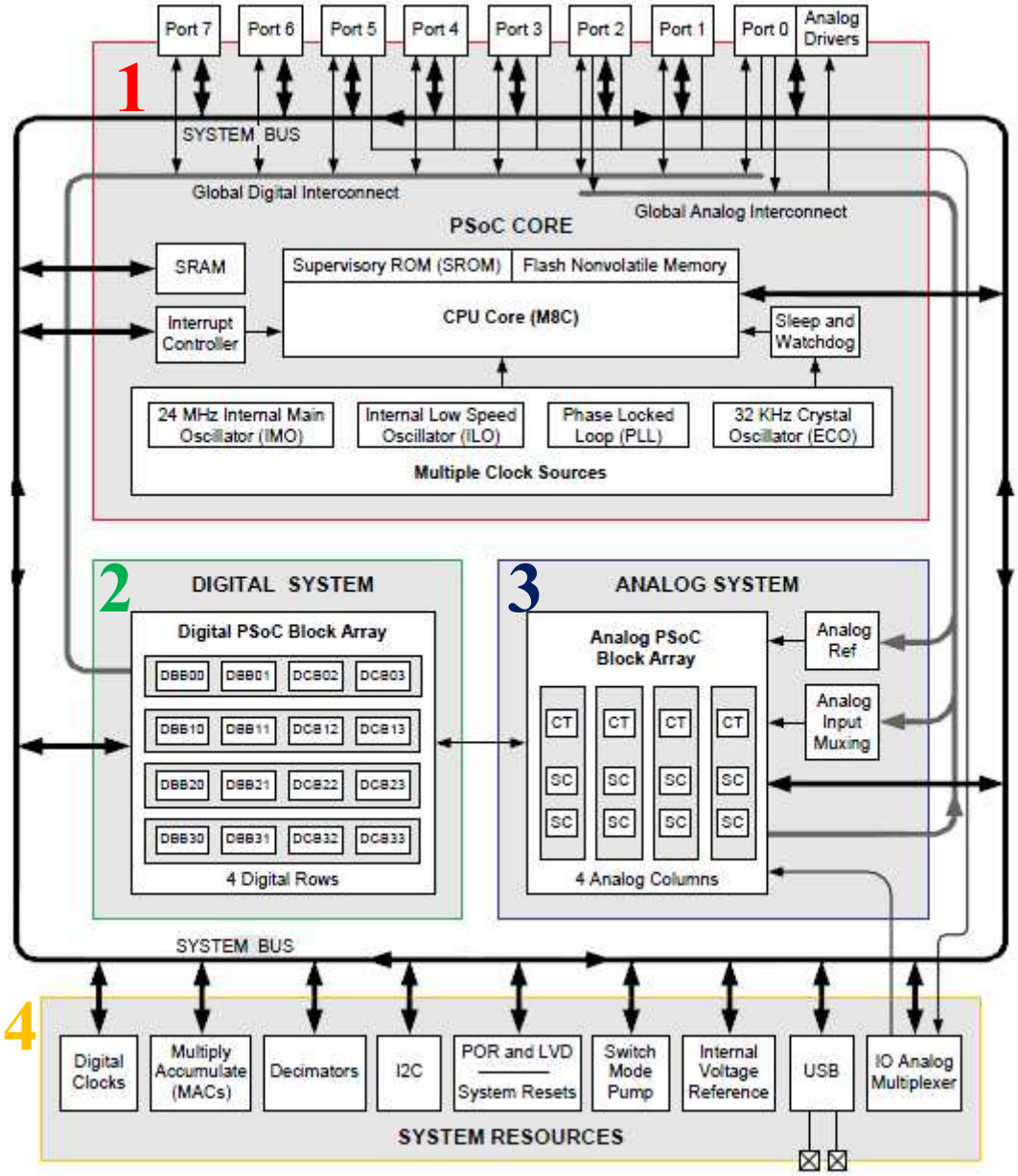


Fig. 3.8 Arquitectura interna del PSoC CY8C29x66

Las 4 partes en que se puede descomponer la estructura interna del PSoC son:

1. PSoC Core (Núcleo del PSoC)
2. Digital System (Sistema Digital)
3. Analog System (Sistema Analógico)
4. System Resources (Recursos del sistema)

3.2.2.1 PSoC Core (Núcleo del PSoC)

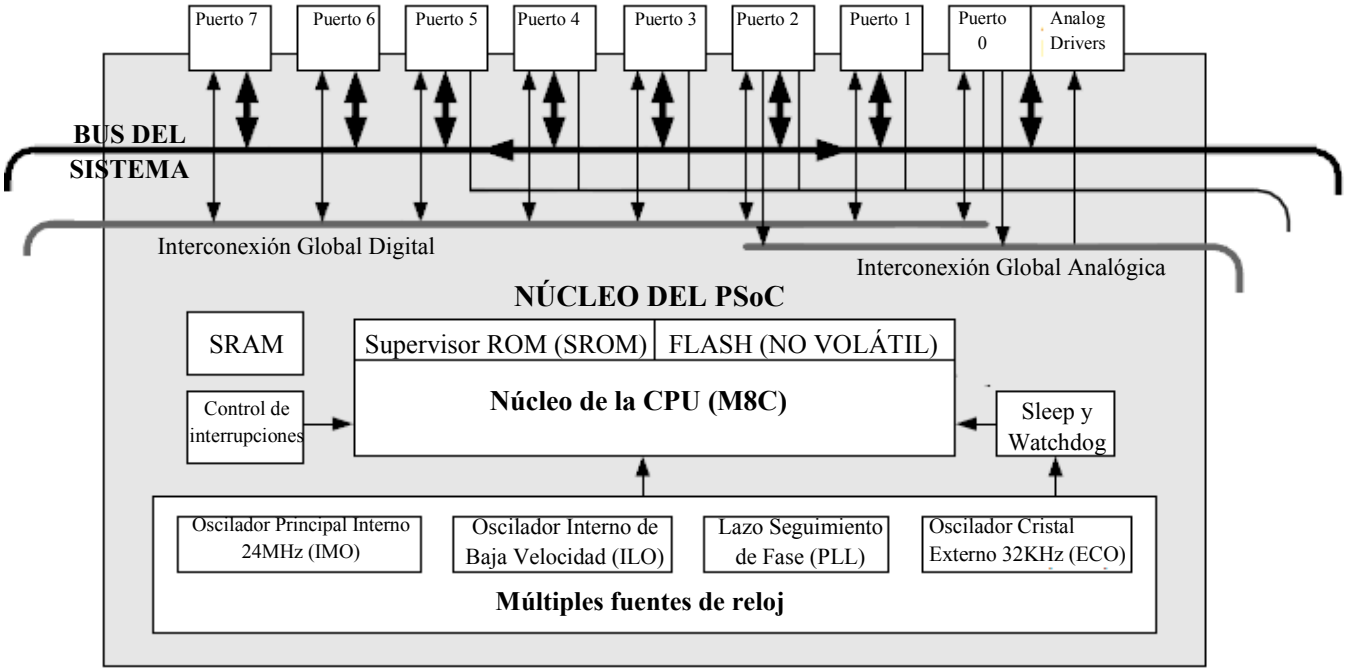


Fig. 3.9 Núcleo del PSoC

El componente principal del núcleo del PSoC es un potente microcontrolador de 8 bits que ofrece 4 MIPS (millones de instrucciones por segundo) a 24 MHz, la velocidad de su oscilador interno principal, que mediante flexibles generadores (en PSoC Designer, a través de VC1, VC2 y VC3) permiten obtener múltiples fuentes de reloj para casi cualquier requerimiento.

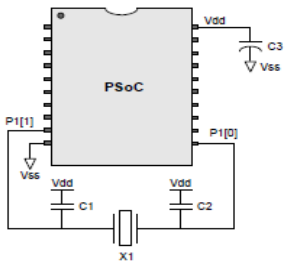


Fig. 3.10 Detalle de conexión del cristal externo de 32 KHz (ECO)

Además de este oscilador de 24 MHz, llamado IMO (Internal Main Oscillator), dispone de otras fuentes de reloj como el ILO (Internal Low Speed Oscillator), con una frecuencia de 32KHz, utilizado para despertar al sistema tras un estado en modo Sleep o para realizar resets con el Watchdog, aunque también puede utilizarse como fuente para bloques digitales.

Otra fuente de tiempo de la que puede disponer el núcleo es la llamada ECO (External Crystal Oscillator), que permite añadir un cristal de 32KHz externo, que generaría las demás fuentes de tiempo con una pequeña configuración del software (el cristal proporcionaría la referencia al IMO en el modo PLL).

La otra fuente que dispone se conoce como PLL (Phase-Locked Loop) que se encarga de proporcionar una frecuencia de oscilación de 24 MHz cuando se utiliza un cristal de 32KHz externo.

En lo referente a memoria, el núcleo posee 32 KB de memoria FLASH, no volátil, para el almacenamiento del programa y 2 KB de SRAM para almacenamiento de datos. Como se puede ver en el esquema, el llamado supervisor ROM (SROM), que contiene el código usado para realizar el boot (configuración inicial) del PSoC y las operaciones de la Flash.

Al igual que la mayoría de microcontroladores, el PSoC dispone de 5 registros que utiliza durante su funcionamiento. Dichos registros tienen 8 bits (el PC tiene 16) y son los siguientes: Acumulador (Accumulator), Índice (indeX), Contador de programa (ProgramCounter), Puntero de pila (StackPointer) y el indicador de estados (flags).

A través del bus del sistema, el núcleo se comunica con los otros bloques del integrado y con el exterior. Para captar las señales externas, el núcleo cuenta con un número variable de puertos E/S dependiendo del modelo del PSoC. En este caso, el 29466 cuenta con 3 puertos de 8 entradas/salidas de propósito general llamadas GPIOs (General Purpose Input Output). Estos puertos interconectan el bloque analógico y el bloque digital con el núcleo, lo que hace posible la interpretación de cualquier naturaleza de señal recibida en cualquiera de sus 24 entradas.

Cada una de estas 24 entradas/salidas, GPIO's, son totalmente configurables por software dependiendo de la función que desempeñe y de la naturaleza de la señal, analógica o digital. Los modos de configuración son los siguientes: high Z, high Z analog, open drain high, open drain low, pull up, pull down, strong y strong slow.

Cada uno de los modos descritos será descrito más a fondo en el apartado 3.2.6.
Por defecto, las GPIO están configuradas con alta impedancia analógica y por norma general se configurarán, como strong si tiene función de salida y como high z si su función es de entrada. Además, es de gran utilidad utilizar las resistencias de "pull up" y "pull down" internas de cada uno de las GPIO, ya que estos circuitos son cuando se escribe un '1' o un '0', respectivamente. En el apartado de entradas/salidas se describe cada uno de estos modos de una forma más concreta.

3.2.2.2 Sistema Analógico

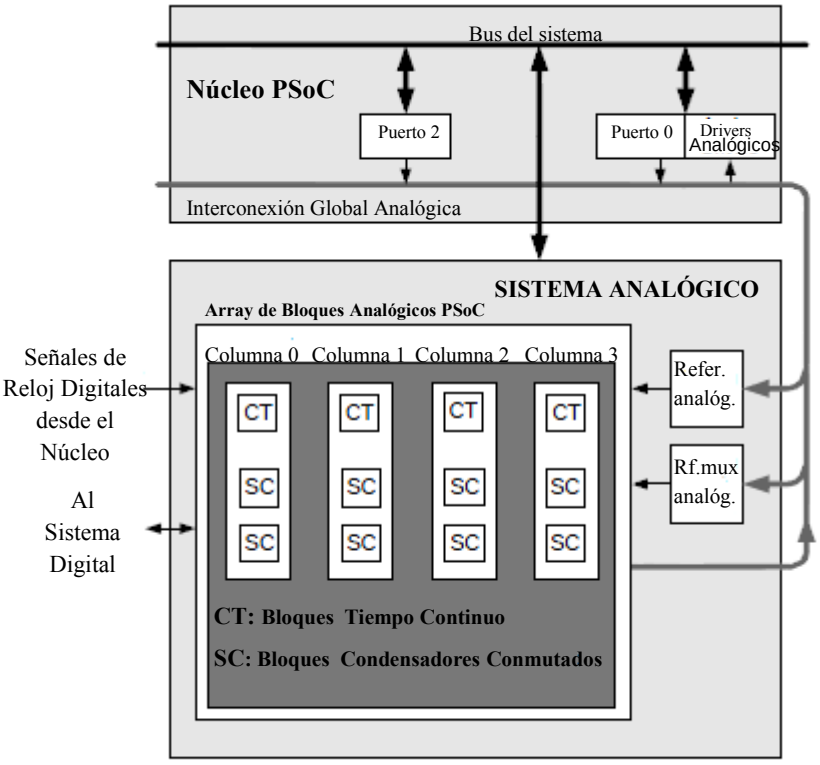


Fig. 3.11 Esquema de bloques del sistema analógico del PSoC

El sistema analógico es una de las piezas claves del PSoC, ya que se encarga de la recepción así como del procesamiento y posterior manipulación de las señales analógicas.
Dependiendo del modelo de PSoC, este puede contener de 1 a 4 columnas de bloques analógicos. En el caso que nos concierne, el 29466 está formado por un array de cuatro columnas con tres bloques analógicos cada una. Estos tres bloques, están montados sobre amplificadores operacionales de bajo offset y bajo ruido y mediante un registro el PSoC determina la tipología de señal analógica dentro de los mismos. Estos bloques se llaman, CT (de tiempo continuo) y SC (condensadores conmutados, del inglés, Switched Capacitors). Dependiendo de la aplicación o de la naturaleza de la señal, se utilizara uno u otro.
Dada dicha configuración, el usuario tiene un gran elenco de posibilidades a la hora de crear su proyecto embebido, ya que dependiendo de sus necesidades de hardware, este utilizara una clase u otra de recursos.

Como se puede ver en el esquema, en lo referente a entradas y salidas, este bloque tiene una comunicación bidireccional tanto con el bus del sistema (Núcleo del PSoC, del que recibe las señales de reloj digitales), así como con el sistema digital. La comunicación de estos bloques con los puertos de E/S (GPIO´s) se realiza a través de una red de multiplexores que se detallará a continuación.

3.2.2.2.1 Interfaz, array y comunicación del sistema analógico

Como ya se ha comentado, el CY8C29466 cuenta con un sistema analógico en el que los bloques analógicos están organizados en columnas formando un array de cuatro columnas. Cada una de estas columnas contiene un bus comparador y 3 bloques analógicos asociados: un bloque de tiempo continuo (CT) y dos bloques de condensadores conmutados (SC) tipo C y tipo D. En la siguiente imagen se muestra el diagrama de bloques de una de ellas:

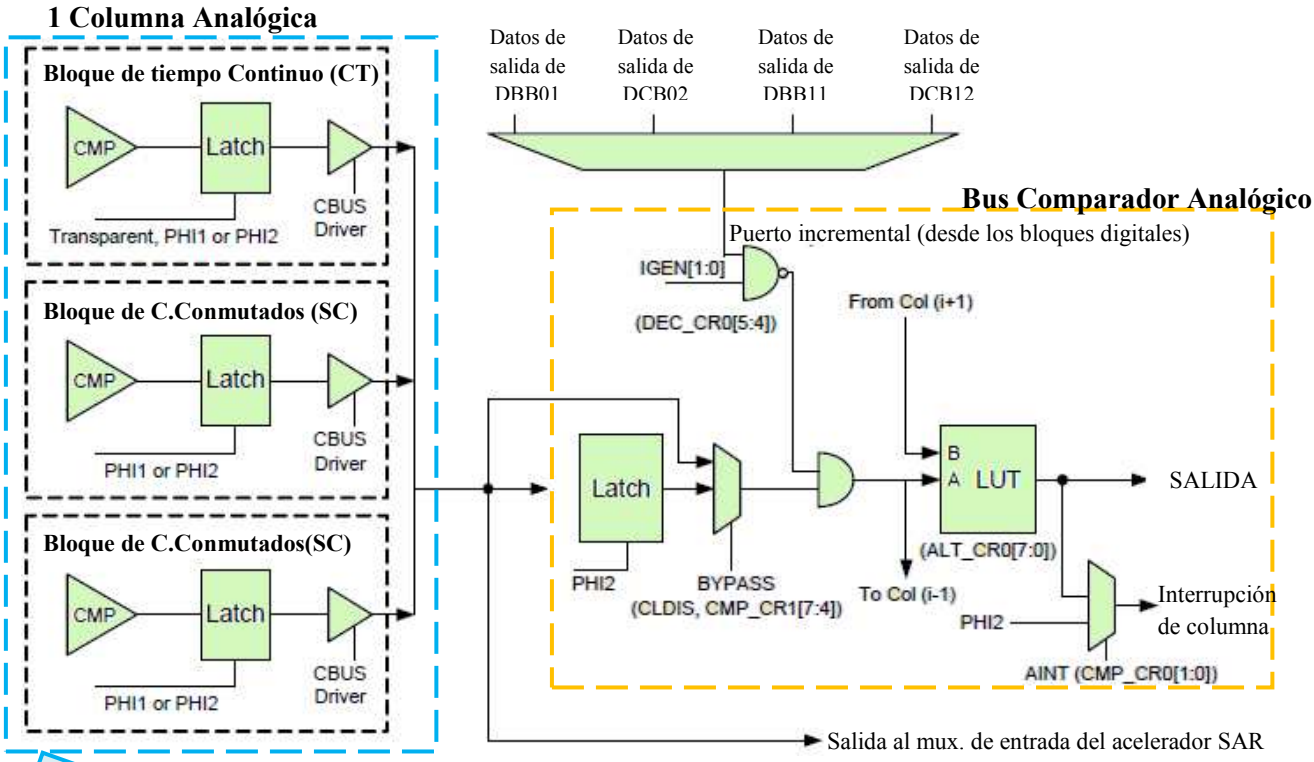


Fig.3.12 Diagrama de bloques de una columna analógica con su bus comparador asociado

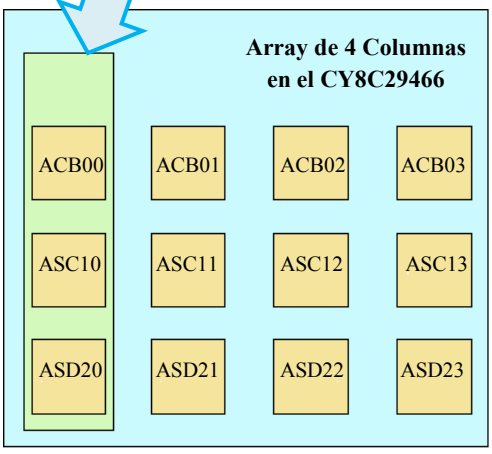


Fig.3.13 Configuración del array del CY8C29466

Como se muestra en la imagen, cada uno de los bloques de la columna analógica contiene un registro de retención (latch) que mantendrá el dato hasta que esté disponible, para después enviarlo al bus comparador analógico. Este bloque comparador, que también cuenta con un registro de retención con la misma función que el anterior, conducirá las señales a los bloques digitales como datos de entrada. También servirá como entrada al decimator y como entrada a interrupciones. A través del bloque LUT, *look up table,* se podrán modificar o combinar las salidas del bus comparador analógico con otros buses a partir de 2 entradas A y B proporcionando 16 posibilidades de funciones lógicas para dichas entradas. En la siguiente imagen podemos ver la configuración del CY8C29466, que cuenta con 4 columnas analógicas. El detalle del cuadro verde muestra la columna que ha sido expandida en la imagen anterior incluyendo el comparador de bus. A continuación mostraré la red de configuración de entrada analógica.

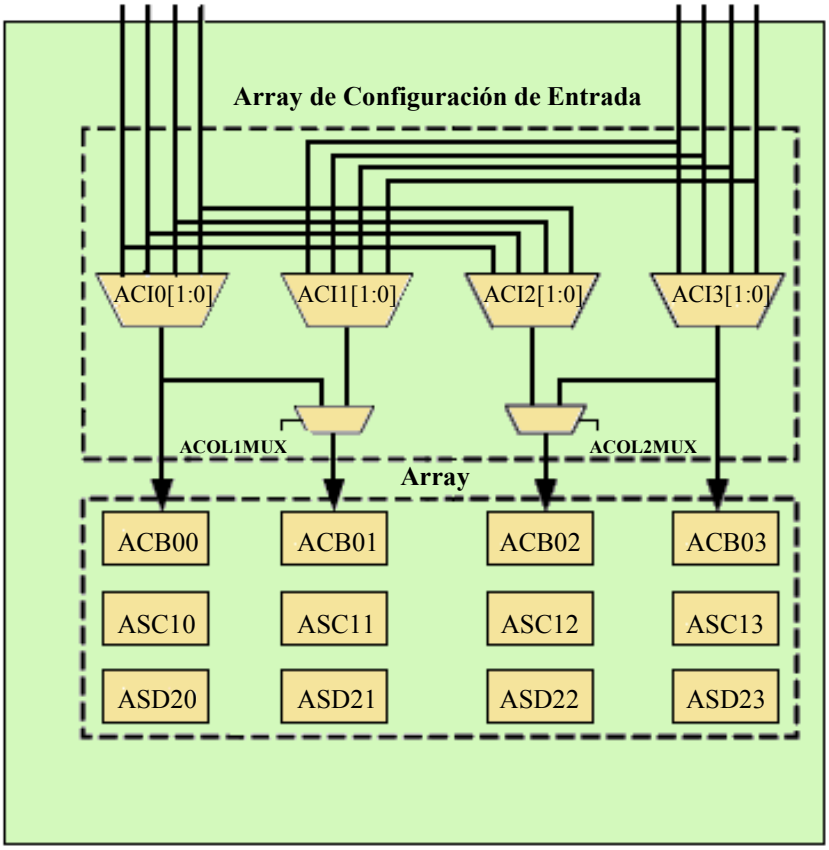


Fig. 3.14 Configuración del array del CY8C29466

El array de 4 columnas del PSoC a través de una red de multiplexores permite la unión entre los bloques analógicos y los pines exteriores del encapsulado de 28 pines. Como se puede ver, las columnas de las esquinas son alimentados por un multiplexor de 4 entradas mientras que las columnas centrales pueden ser alimentados por uno o dos multiplexores.

A través de los dos multiplexores centrales de 2 entradas, se podrá seleccionar a través de una patilla de control, la señal que recibirán las columnas centrales. Las patillas de control de dichos multiplexores reciben las señales llamadas ACOL1MUX y ACOL2MUX. Estos multiplexores son interruptores CMOS con una resistencia típica del rango de 2KΩ. Para realizar la conexión de los multiplexores con los pines existen cuatro drivers analógicos para llevar a la salida valores analógicos para llevar a la salida valores analógicos en los pines P0[2], P0[3], P0[4] y P0[5]. Cabe decir, que en otros modelos que cuentan con una o dos columnas, las patillas y pines de salida son diferentes. A continuación se muestra un detalle de la conexión de los arrays analógicos con los pines asociados a la salida de señales analógicas.

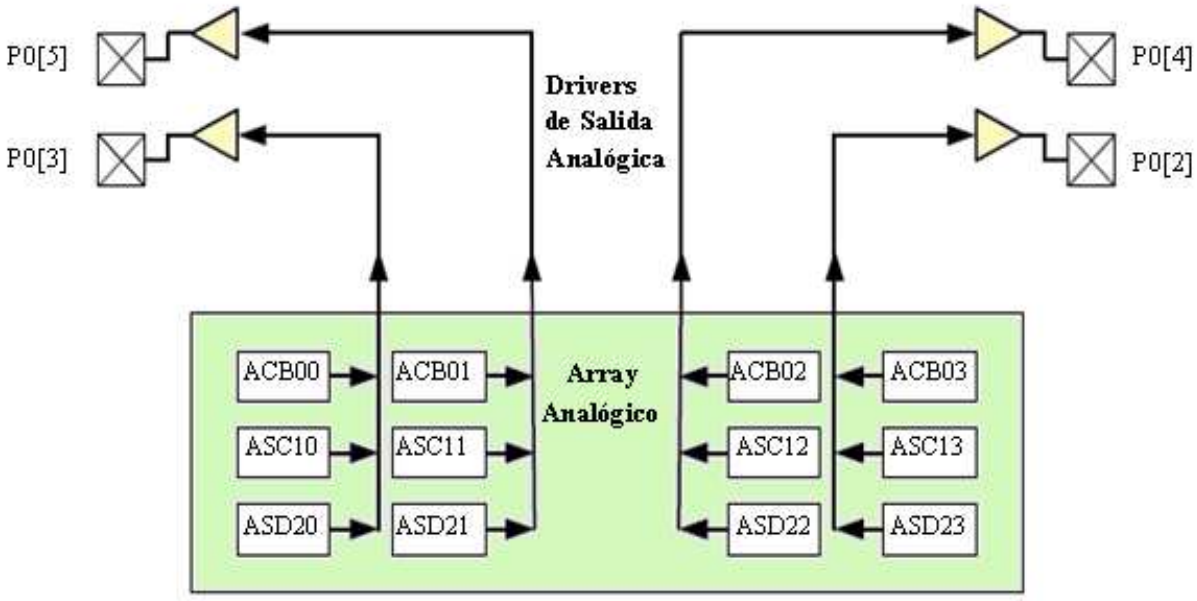


Fig.3.15 Conexión del array analógico con los pines del CY8C29466

3.2.2.3 Sistema Digital

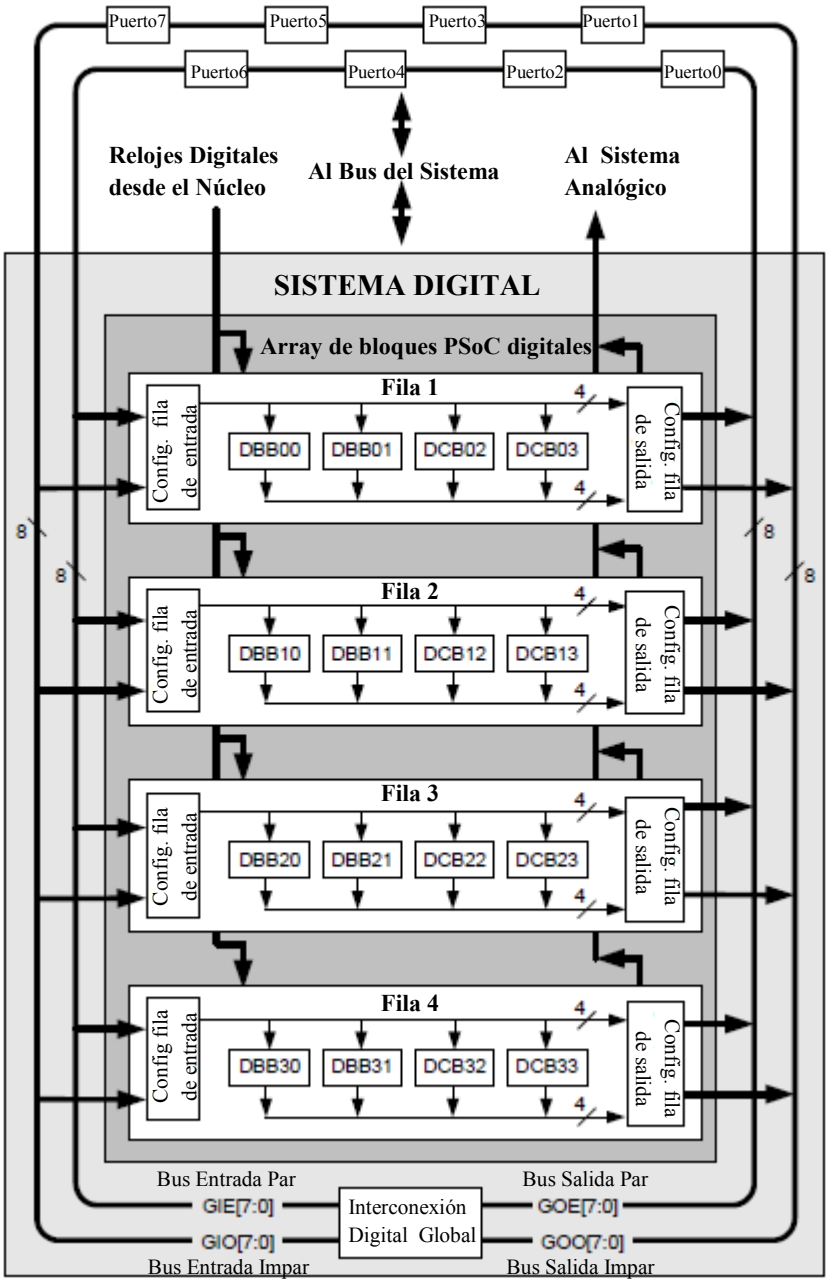


Fig. 3.16 Esquema de bloques del sistema digital del PSoC

El sistema digital se encarga de realizar todas las operaciones de manipulación y tratamiento de datos digitales (contadores, PWM´s, temporizadores, etc.) así como de protocolos de comunicación (SPI, I2C, etc.).

Al igual que el sistema analógico, dependiendo del modelo de PSoC el número de filas en este caso, es variable. En el caso del 29466, el sistema digital cuenta con 4 filas y 4 bloques digitales por fila, con un número total de 16 bloques digitales totalmente configurables. Cada uno de estos bloques es de 8 bits y tienen una alta capacidad de integración, gracias a una excelente comunicación bidireccional entre todos ellos, teniendo la posibilidad de unir bloques para alcanzar un bloque con hasta un máximo de 32bits.

Tanto la entrada como la salida a cada una de las filas de bloques que se han descrito, se pueden configurar: el usuario puede elegir el bus por el que quedarán conectados las entradas y las salidas a los bloques con los pines pares o impares de los puertos. Esta conexión se realiza mediante la Interconexión Digital Global (GDI), que consiste en 4 buses de 8 bits, 2 de entrada (Global Input Odd (impar) y Global Input Even (par)) que permiten la llegada de señales desde los pines del puerto hasta en núcleo del PSoC y 2 de salida (Global Output Odd y Global Output Even) que permiten la salida de señales desde el núcleo del PSoC a los pines de los puertos.

Dichos bloques digitales pueden conectarse con cualquier GPIO a través de los puertos descritos y éstos tienen la capacidad de conectar cualquier señal con cualquier pin dependiendo de las necesidades del usuario. Por otro lado, existe la posibilidad de utilizar la capacidad de los buses para multiplexar las señales, asi como de realizar operaciones lógicas sencillamente.

En la siguiente página se puede observar las conexiones internas que comunican el array de bloques digitales con sus respectivos pines, la cual recibe el nombre de interconexión digital global. Como se puede observar los pines y los puertos están separados dependiendo de si son pares e impares (even y odd). También se pueden observar las conexiones de los relojes digitales.

3.2.2.3.1 Interfaz, array y comunicación del sistema digital

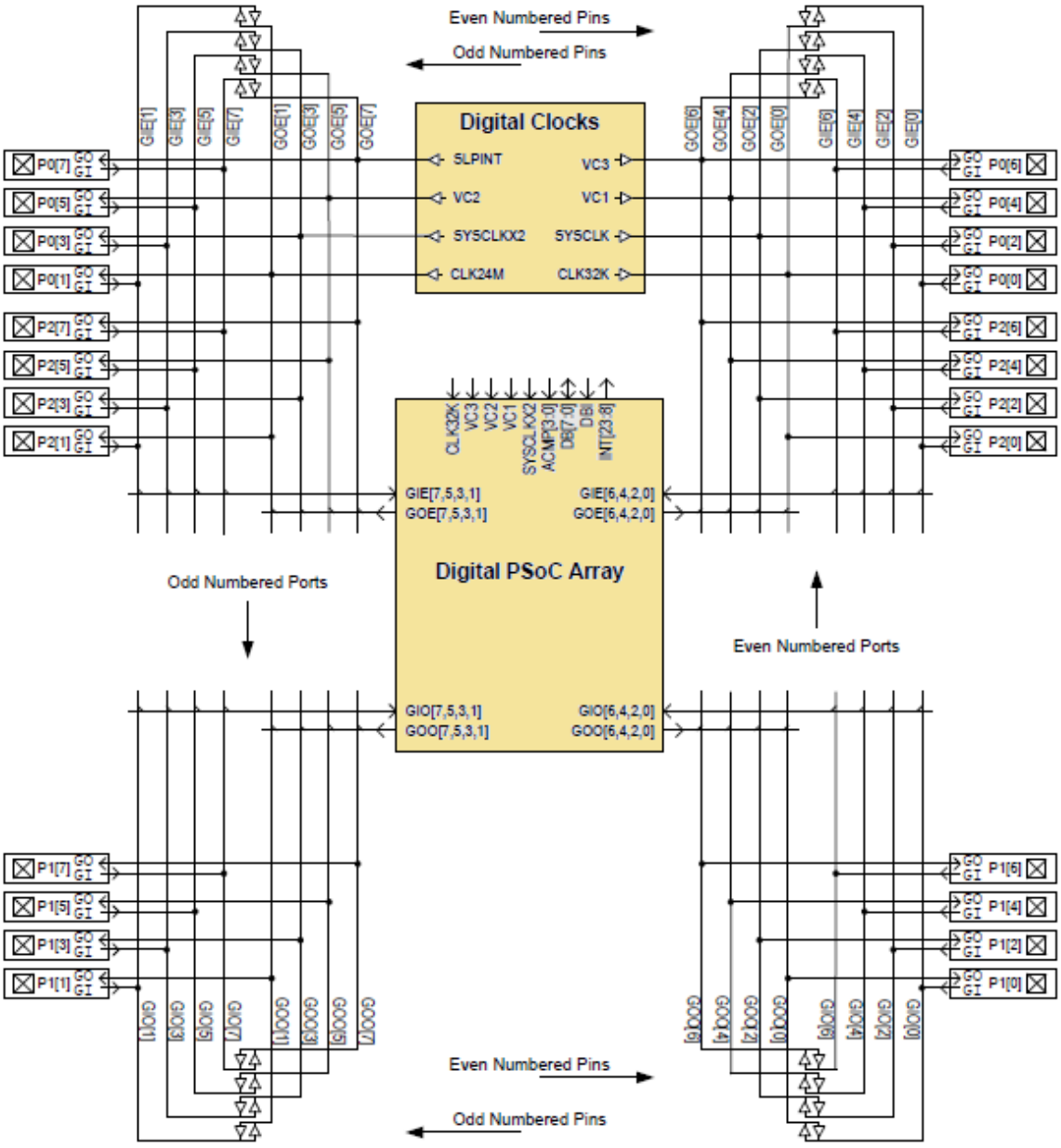


Fig. 3.17 Interconexión Digital Global del CY8C29466

Como se puede ver en el diagrama de conexiones, el PSoC que he utilizado cuenta con 3 puertos de 8 bits de los cuales, dos están conectados a buses globales pares y el restante a un bus global impar. Ahora nos centraremos en el array digital del PSoC y sus componentes internos.

El array está formado por cuatro bloques organizados en filas, estando estas totalmente comunicadas entre sí, permitiendo una configuración total de las funciones desempeñadas por las mismas y siempre a través del usuario.

La interconexión del array digital realiza la conexión entre la interconexión digital global y la interconexión digital de filas que variará según el modelo de PSoC. La imagen que se muestra en la siguiente página muestra el interior del array digital en la que se muestran todas las conexiones entre las cuatro filas de bloques digitales.

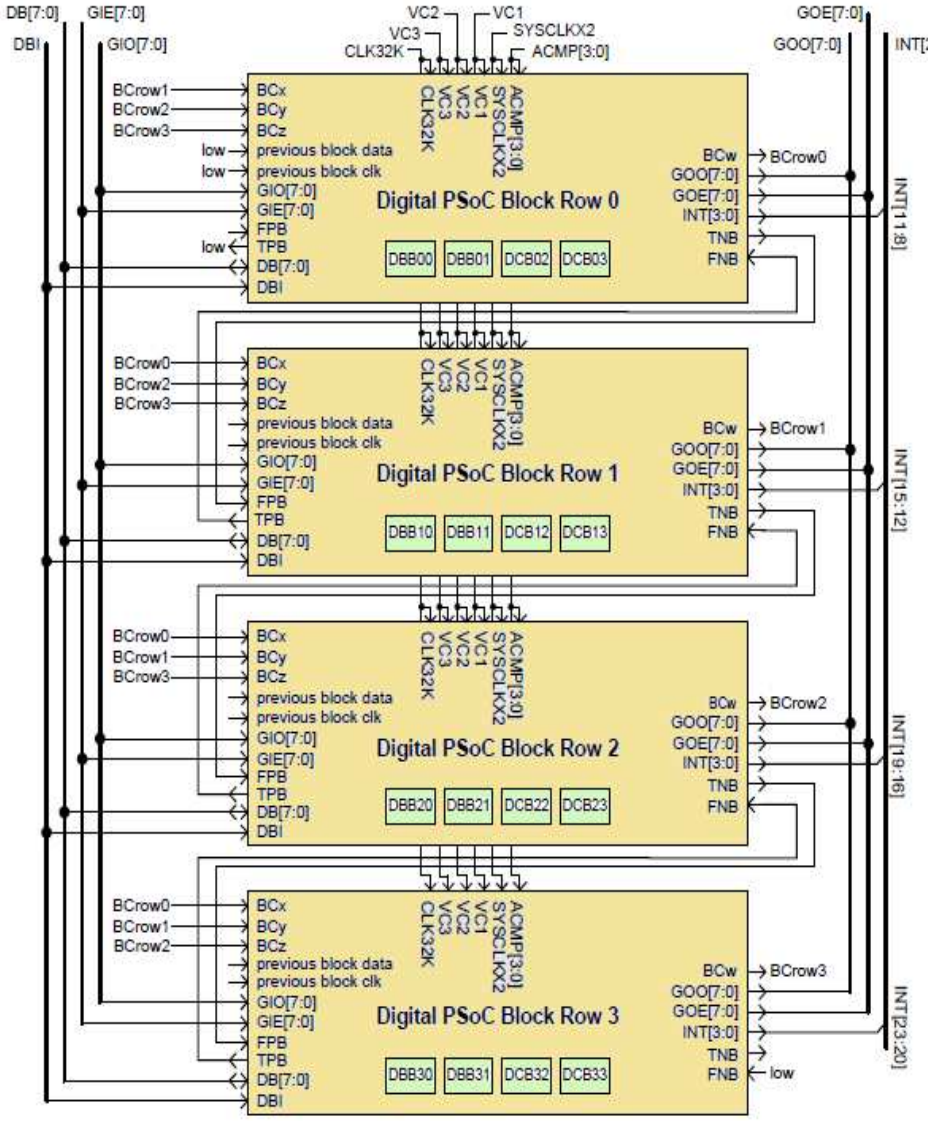


Fig. 3.18 Array de Interconexión Digital del CY8C29466

Cada una de las cuatro filas de bloques digitales está totalmente interconectada con las demás y con el exterior, recibiendo además las señales de reloj provenientes del núcleo. Es importante destacar que todas las filas tienen idénticas conexiones a las entradas y salidas globales. Para diferenciarlas, y darle a cada fila una dirección única las conexiones se realizan de la siguiente forma:

- Cada fila y los bloques que contienen tienen una **dirección única**.
- Cada uno de dichos bloques tienen un **vector único de interrupción**, teniendo siempre mayor prioridad los bloques que se encuentran en las filas superiores con respecto a las inferiores.
- Cada fila tiene una **red de emisión** particular que puede ser utilizada internamente por un bloque digital o de manera externa. Si es utilizada de manera externa, será utilizada por otra de las filas del array.

- Las filas de los array forman **cadenas** de bloques de una longitud igual al número de filas mutliplicadas por cuatro. Como se puede ver en la imagen, y a excepción del primer bloque de la primera fila y el último de la última fila que no están conectados, todas las filas están interconectadas entre sí.

Profundizando más en el sistema digital y su sistema de interconexión, nos centraremos ahora en uno de los bloques de filas del array digital mostrando su contenido y su correspondiente conexión entre sus componentes internos. La imagen de la siguiente página muestra uno de estos bloques.

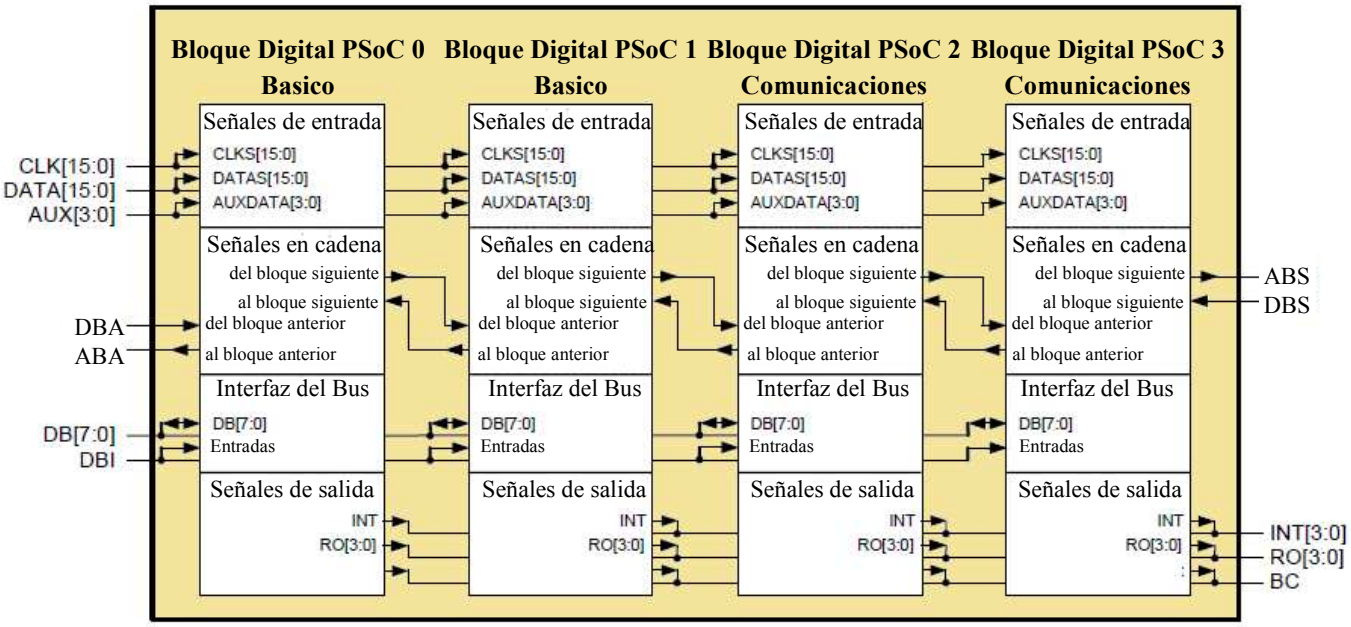


Fig. 3.19 Interconexión Digital de una fila del array del CY8C29466

Como se puede ver, cada uno de las filas del array digital, contiene cuatro bloques digitales. Dos de estos bloques serán básicos o DBB (Digital Basic Block) y los otros dos serán de comunicación o DCB (Digital Communication Block). Cada uno de estos cuatro bloques se compone de cuatro "subapartados" ya que cada uno cumple una función determinada. Por un lado, tenemos los bloques que reciben/envían las señales a procesar/procesadas.

Si bien el "bloque" de señales de entrada recibe la señal de reloj, la señal del dato a procesar y un dato auxiliar, el "bloque" de señales de salida enviara al exterior las señales procesadas por el bloque correspondiente. También podemos encontrar otros dos "subapartados" que se encargan de interconectar la fila con las demás y con el bus del sistema. Uno de ellos, se encargará de unir los cuatro bloques digitales de la fila entre si y con los bloques digitales de las demás filas. El otro, se encargará de unir cada uno de los bloques de la fila con el bus del sistema. Para finalizar, mostraré el interior de un bloque digital. Cada uno de los bloques digitales simples del PSoC podrá ser configurado para desempeñar una de las siguientes cinco funciones: temporizador, contador, modulador de anchura de pulso(PWM), creador de secuencias pseudo aleatorias (PRS) o comprobador de redundancia cíclica(CRC). Los bloque digitales de comunicación incluirán además dos funciones adicionales: maestro o esclavo de comunicación SPI y comunicación UART que será explicada posteriormente.

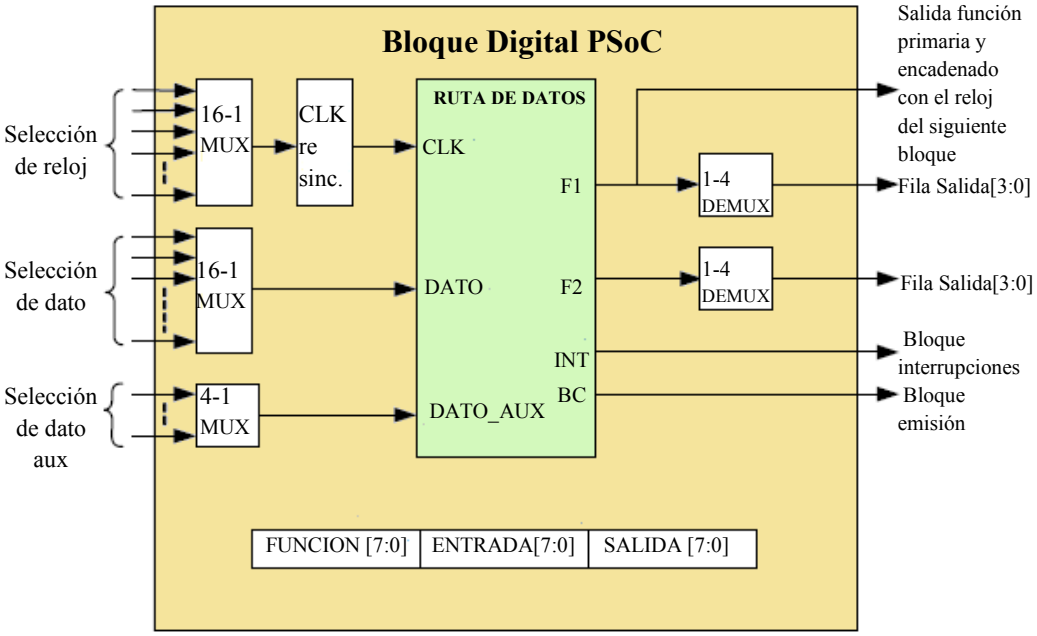


Fig. 3.20 Bloque digital del CY8C29466

3.2.2.4 System Resources (Recursos del sistema)

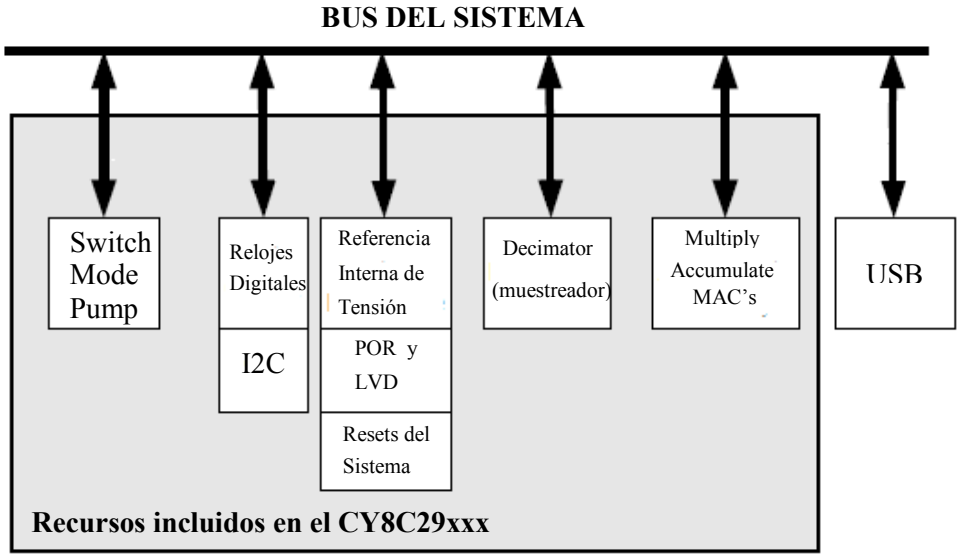


Fig. 3.21 Recursos del sistema del CY8C29xxx

El PSoC CY8C29466-24PXI cuenta en su arquitectura una serie de funcionalidades, hardware y software, que amplían aun más las capacidades de éste. Esto en la mayoría de los casos, mejora sustancialmente la experiencia del usuario ya que, por ejemplo, le permite reducir todavía más las dimensiones del hardware o simplificar decisiones de diseño mediante software. Dichos recursos (y sus principales utilidades) son los siguientes:

- **Switch Mode Pump (SMP)**: Mediante una simple acomodación de hardware externa al PSoC, este circuito integrado permite obtener tensiones de alimentación constantes a partir de una batería.

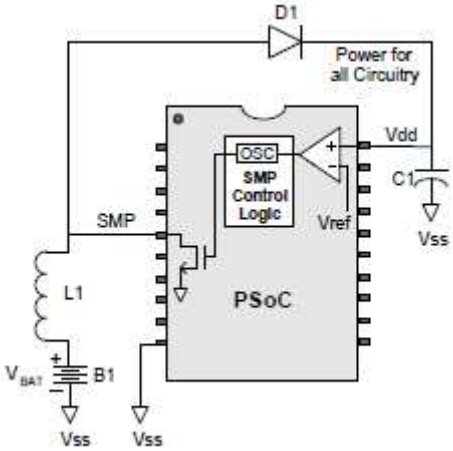


Fig. 3.22 Hardware necesario para la utilización del SMP

En la fig. 3.22, podemos ver la configuración típica para su correcto funcionamiento: se coloca una bobina (la cual se elegirá con un valor adecuado correspondiente al valor que queremos que suministre a la carga en serie) entre la batería, y la patilla correspondiente al SMP incluyendo un diodo, que se recomienda Schottky (gracias a su rápida velocidad de conmutación y su baja caída de tensión inversa), hacia la patilla de alimentación. Un condensador se debe colocar entre la alimentación y la masa, el cual determinara el rizado y el tiempo de mantenimiento de la tensión de salida.

Este recurso, que es bastante útil cuando se quiere conseguir una alimentación constante del sistema, está relacionado con el Power on Reset (Energia en el re-arranque) y con el Sleep Mode (Modo "dormido").

- **I²C (Inter Integrated Circuit)**: Sus siglas provenientes del inglés significan Circuitos Inter-Integrados, y es un bus de comunicación muy utilizado y extendido en la industria por su simplicidad de operación. Requiere de tres hilos para operar correctamente, uno de datos, otro de reloj y otro de referencia(GND) y el PSoC permite una total configuración como maestro / esclavo o receptor / transmisor , utilizando únicamente dos GPIO´s. El PSoC interactúa con el bloque mediante los registros de E/S y el usuario realizara mediante software la configuración y utilización del mismo (interrupciones, rutinas, etc.).

- **Relojes Digitales**: A partir de dos relojes internos, el IMO y el ILO (siglas en inglés de Internal Main Oscillator (Oscilador interno principal) y Internal Low Speed Oscillator (Oscilador interno de baja velocidad), respectivamente) el PSoC obtiene todas las fuentes de reloj.

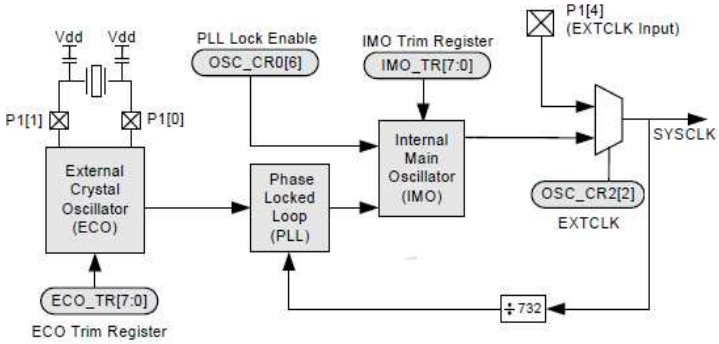


Fig.3.23 Diagrama de operación del IMO

El IMO es un oscilador, que por defecto funcionará a 24MHz y que a partir del cual, se basan la mayoría de las señales de reloj de los circuitos del PSoC. También puede funcionar en modo PLL(del inglés, Phase Locked Loop, Bucle de seguimiento de fase) llevándolo a una referencia de 32KHz de precisión aproximadamente(que se consigue realizando una división de los 24MHz por 732, como se puede ver en el diagrama). Esta operación se puede realizar fácilmente actuando mediante software sobre los registros de configuración del IMO. Existe además la posibilidad de incorporar un oscilador de cristal externo ECO (del inglés, External Crystal Oscillator) que vendría a remplazar al oscilador interno y que realizaría la misma función del mismo, esto es, servir de base de tiempos para todos los circuitos del PSoC.

A partir del IMO, o del cristal externo, se obtendrá la red interna de bases de tiempo, la cual recibe el nombre de SYSCLK o reloj del sistema. Esta señal se diversificara en numerosas señales, que describiré brevemente, como se puede ver en el siguiente diagrama de flujo:

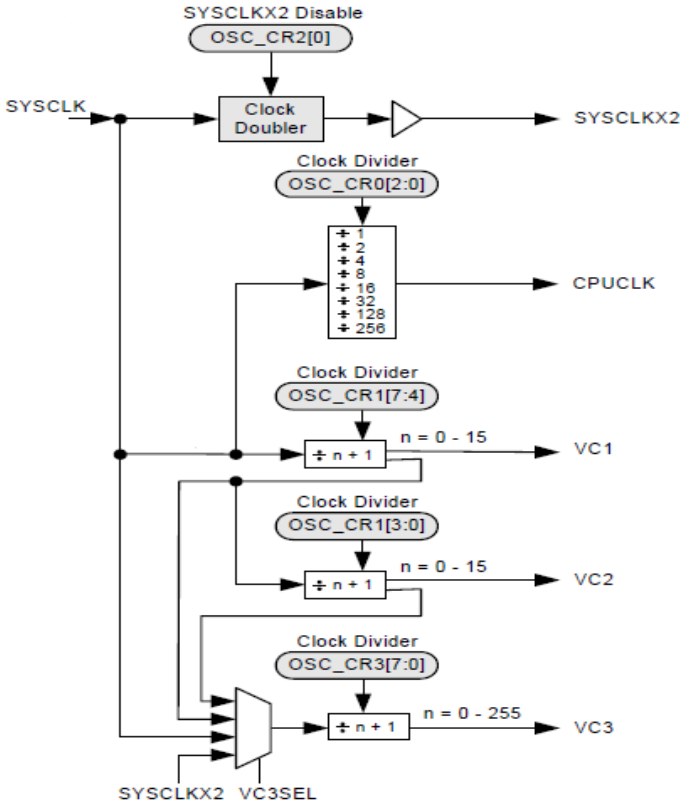


Fig.3.24 Diagrama de la red interna de bases de tiempos SYSCLK

A partir de la red de tiempos SYSCLK obtendremos las siguientes señales:

- SYSCLKx2: El reloj del sistema doblará su frecuecia de trabajo mediante un doblador de reloj.
- CPUCLK: La velocidad de trabajo de la CPU también vendrá derivada de SYSCLK donde el usuario elegirá el factor de división a aplicar entre 1, 2, 4, 8, 16, 32, 64, 128 y 256. Este valor es de frecuencia es facilmente configurable mediante el software de Cypress (PSoC Designer) eligiendo el valor de un menú desplegable que posteriormente explicaré ampliamente en el apartado 3.3 Software de programación.
- VC1: Las siglas en inglés significan Variable Clock 1, y es el primero de los 3 relojes variables con los que cuenta el usuario para obtener una gran diversidad de frecuencias de trabajo. Al igual que con CPUCLK, el usuario puede elegir fácilmente el factor de division mediante el software. En este caso el factor de división seleccionable varía entre 1 y 16.

- VC2: Deriva de la señal VC1. Igualmente su factor de división es seleccionable por el usuarion entre 1 y 16.
- VC3: De igual manera que VC1 y VC2 es una frecuencia determinada por el usuario, con la diferencia de que este puede elegir la fuente de tiempos que recibirá el factor divisorio, pudiendo ser SYSCLK, VC1, VC2 o SYSCLKx2. En este caso el factor divisorio seleccionable oscilara entre 1 y 256

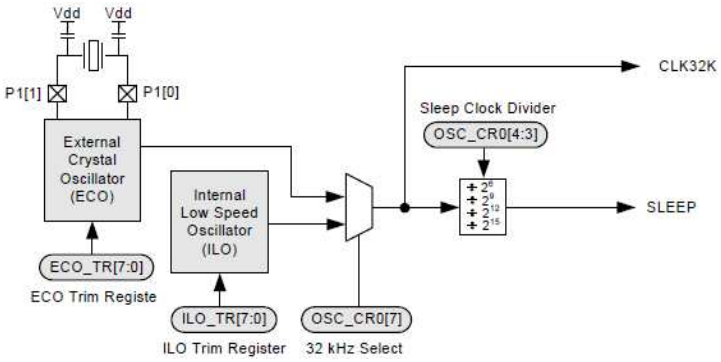


Fig.3.25 Diagrama de operación del ILO

El ILO es un oscilador con una frecuencia nominal de 32 KHz que el PSoC utiliza para las interrupciones de modo Sleep y Wake up y para las operaciones con el perro guardian o Watchdog, siendo esta su función principal. Aunque el usuario también puede utilizar la señal como una fuente de reloj. El ILO, en ausencia de un cristal externo, funcionará constantemente y también puede funcionar como oscilador general del sistema. Entonces, el ILO generará otras 2 señales:

- CLK32K: Esta señal, dependiendo de la inclusión de cristal externo, la proporcionará el ILO o el cristal. Puede funcionar como fuente general de tiempos en detrimento del IMO, modificando un registro.
- SLEEP: A partir de la señal de 32 KHz (independientemente de su origen) , el usuario puede elegir el intervalo de tiempo que adquirirá el sistema en caso de caer en suspensión, o modo SLEEP, pudiendo ajustarlo a 4 valores oscilando de 1.95ms a 1s.

La gran variedad de bases de tiempos digitales a obtenibles a partir de dos relojes internos, permiten al usuario realizar un proyecto embebido personalizando totalmente las fuentes de reloj de todos sus componentes, sin la necesidad de incluir cristales externos o hardware adicional.

Esto supone un gran ahorro tanto de componentes asi como de tiempo y tamaño de diseño, facilitando enormemente la labor del programador/usuario.

- **Referencia interna de tensión**: El PSoC cuenta con una referencia interna de que resulta de dos componentes, un generador de tensión de banda prohibida (bandgap) que genera 1.20V y del siguiente circuito: un amplificador que regula la ganancia de la tensión del generador a 1.30V. La conexión AO y C se realiza mediante un interruptor CMOS lo que permite al sistema utilizar la tensión de referencia Vref mientras el circuito de referencia es apagado.
  La tension temperatura ambiente es de unos 1.30V. Además, de este bloque se obtiene una tensión proporcional a la temperatura para la medición de la misma.

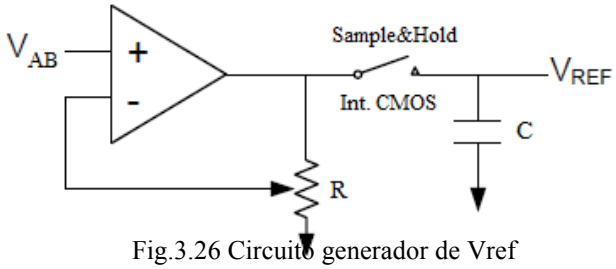


Fig.3.26 Circuito generador de Vref

- **Power On Reset y Low Voltage Detect (POR y LVD)**: Estos dos recursos del sistema sirven como protección del sistema en casos de bajo voltaje.

  El POR realiza mediciones la tensión de alimentación Vdd y a la vez mantiene el sistema en reset hasta que la misma adquiere valores aceptables de operación, evitando de esta manera que el sistema arranque de forma inesperada o con valores insuficientes de tensión.

  De igual manera, el LVD (del inglés Detección de Bajo Voltaje) realiza mediciones de Vdd pero a diferencia del POR, este proporciona una interrupción cuando dicha tensión baja por debajo de determinado valor. Puede resultar muy útil a la hora de realizar operaciones previas a la pérdida total de energía o para la ejecución de subrutinas en determinados valores críticos de Vdd.

- **Resets del sistema**: Existen diferentes formas de que un reset se produzca en el PSoC dependiendo de que lo desencadena. Estos son principalmente los siguientes:

  1. *Reset en el arranque (Power on Reset)*: Este reset suele darse cuando hay bajo suministro de tensión y puede ser ocasionado por múltiples fuentes.
  2. *Reset externo*: Este reset lo producirá el usuario introduciendo un nivel alto en el pin XRES del PSoC, en el caso del CY82C9466 es el pin 19.
  3. *Reset del Watchdog o perro guardián*: Es un reset opcional configurable por el usuario, para evitar normalmente bucles infinitos.
  4. *Reset interno*: Puede darse durante la secuencia de arranque si el código SROM determina que la lectura de la Flash no es valida

En el caso de que se produzca un reset en el arranque o un reset externo producirá la conmutación de los pines de prop. gral. (GPIO's), P1 [0] y P1 [1], de la siguiente forma:

En el caso del POR, el reset interno ocasiona, como podemos ver en el gráfico, un nivel alto a P1 [0] durante 256 ciclos de reloj en modo sleep (T1) y un nivel bajo a P1 [1]. Tras ese número de ciclos, P1 [0] también recibirá un nivel bajo y ambos puertos permanecerán así durante otros 256 ciclos. Tras ese periodo (T2) se producirá una transición mediante la cual ambos pines pasan a comportarse como alta impedancia y comenzará la operación normal de la CPU.

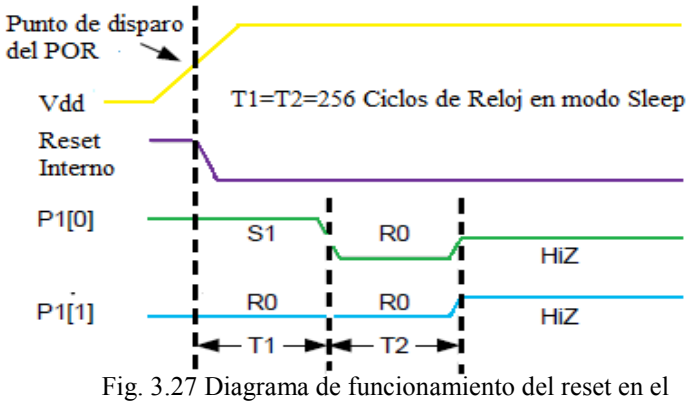


Fig. 3.27 Diagrama de funcionamiento del reset en el arranque y sus consecuencias sobre los GPIO´s P1 [0] y P1

Si en cambio el reset es producido por un reset externo, esto es, cuando el usuario introduce un nivel alto en el pin XRES (19), ambos GPIO´s toman un valor bajo y continuaran asi durante 8 ciclos de reloj en modo Sleep. Tras ese periodo y tras una transición ambos pines tomaran un estado de alta impedancia y comenzara la operación normal de la CPU, como en el caso anteriormente descrito.

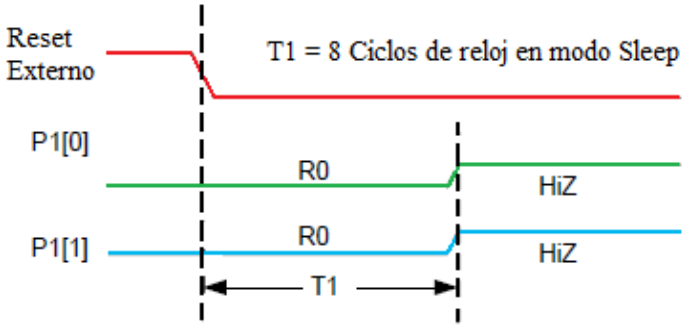


Fig. 3.28 Diagrama de funcionamiento del reset en el arranque y sus consecuencias sobre los GPIO´s P1 [0] y

- **Decimator (Muestreador):** El PSoC cuenta con un pequeño hardware que asistirá en procesos de señales digitales. Puede ser de tipo 1 o tipo 2, pero en el caso que nos concierne, el CY8C29466, solo cuenta con el tipo 2. Este recurso del sistema puede ser usado principalmente para conversiones AD como por ejemplo la conversión delta sigma o la conversión AD incremental. En la siguiente imagen se puede ver la arquitectura del bloque:

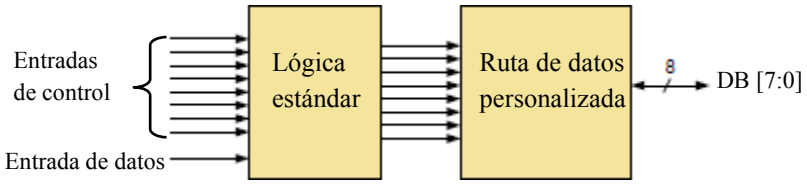


Fig. 3.29 Arquitectura del muestreador tipo 2 incluido en el CY8C29466

- **Multiply accumulate (MAC):** Este bloque del sistema corresponde con un multiplicador rápido de 8 bits con una acumulación de 32 bits. Como única interfaz, este recurso basado en registros es el bus del sistema, por lo que no se requieren fuentes de reloj especiales para que sea utilizado por bloques analógicos o digitales del PSoC. En el caso que nos concierne, el CY8C29466 cuenta con dos bloques MAC independientes entre sí. A continuación, detallo su diagrama de bloques y su funcionamiento:

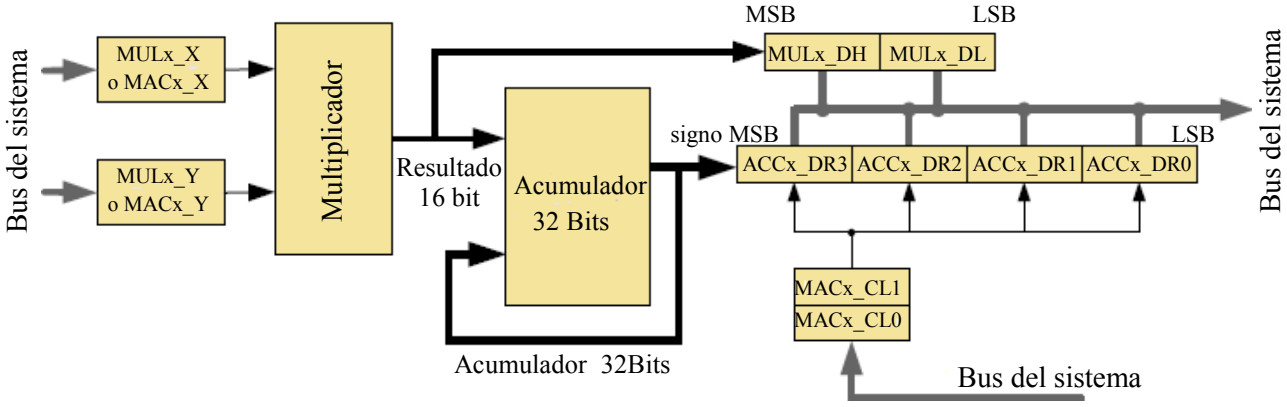


Fig. 3.30 Diagrama de bloques del multiplicador rápido con acumulador de 32 bits del CY8C29466

El usuario podrá elegir entre una multiplicación simple o de una multiplicación con acumulación dependiendo de los registros utilizados en dicha operación. Para una multiplicación simple, como se puede observar en el diagrama, el MAC acepta dos valores de 8 bits(con o sin signo) llamados MULx_X y MULx_Y. El resultado de dicha multiplicación aparecerá instantáneamente en los registros MULx_DH y MULx_DL cuyo valor permanecerá tras un re-arranque o tras un reset ocasionado por el usuario.

En el caso de una multiplicación con acumulación, la operación comenzará en el momento en el que se escriba un valor en los registros MACx_X y MACx_Y, guardándose el resultado en los registros ACCx_DR3, ACCx_DR2, ACCx_DR1 y ACCx_DR0. En este caso, el valor de la multiplicación también se guardara en un acumulador como un valor de 32 bits con signo. En caso de querer borrar los registros que almancenan el resultado de multiplicaciones acumuladas, bastará con escribir en cualquiera de los registros MACx_CL 1 o MACx_CL 0.

3.2.3 Entradas/Salidas - General Purpose Input Output(GPIO)

En lo concerniente a la comunicación, el PSoC cuenta con los GPIO, siglas en inglés de General Purpose Input Output (entradas salidas de propósito general), que son los responsables de unir el núcleo del microprocesador con el exterior. Estos ofrecen una gran variedad de configuraciones para trabajar tanto con señales digitales como analógicas.

Cada puerto de entradas/salidas (nombrados como IO (input/output) a partir de ahora para mayor comodidad) contara con 8 bits y cada uno de estos puertos contara con 8 GPIO´s idénticos teniendo cada uno de estos una identificación única. En nuestro caso, el CY8C29466 contará con 3 puertos llamados P[0], P[1] y P[2]. El diagrama de bloques de un GPIO es como el que se muestra a continuación:

Dependiendo de la elección de una de las configuraciones arriba mencionadas, el GPIO hará uso de una u otra parte del circuito que contiene. Por un lado, si el pin es configurado como entrada, la región utilizada será la marcada con un cuadro verde, la cual dirigirá la señal dirigirá la señal recibida al núcleo del sistema donde será procesada dependiendo de la naturaleza de la misma, pudiendo ser una señal analógica(AIN), una señal de interrupción dirigida al bloque I2C, etc. De igual manera, si el pin es configurado como salida, la región utilizada será la marcada por el cuadro naranja, en la que la señal recibida por el núcleo como por ejemplo una señal analógica(AOUT), una señal del bus global,etc. se dirigirán hacia el pin de salida especificado.

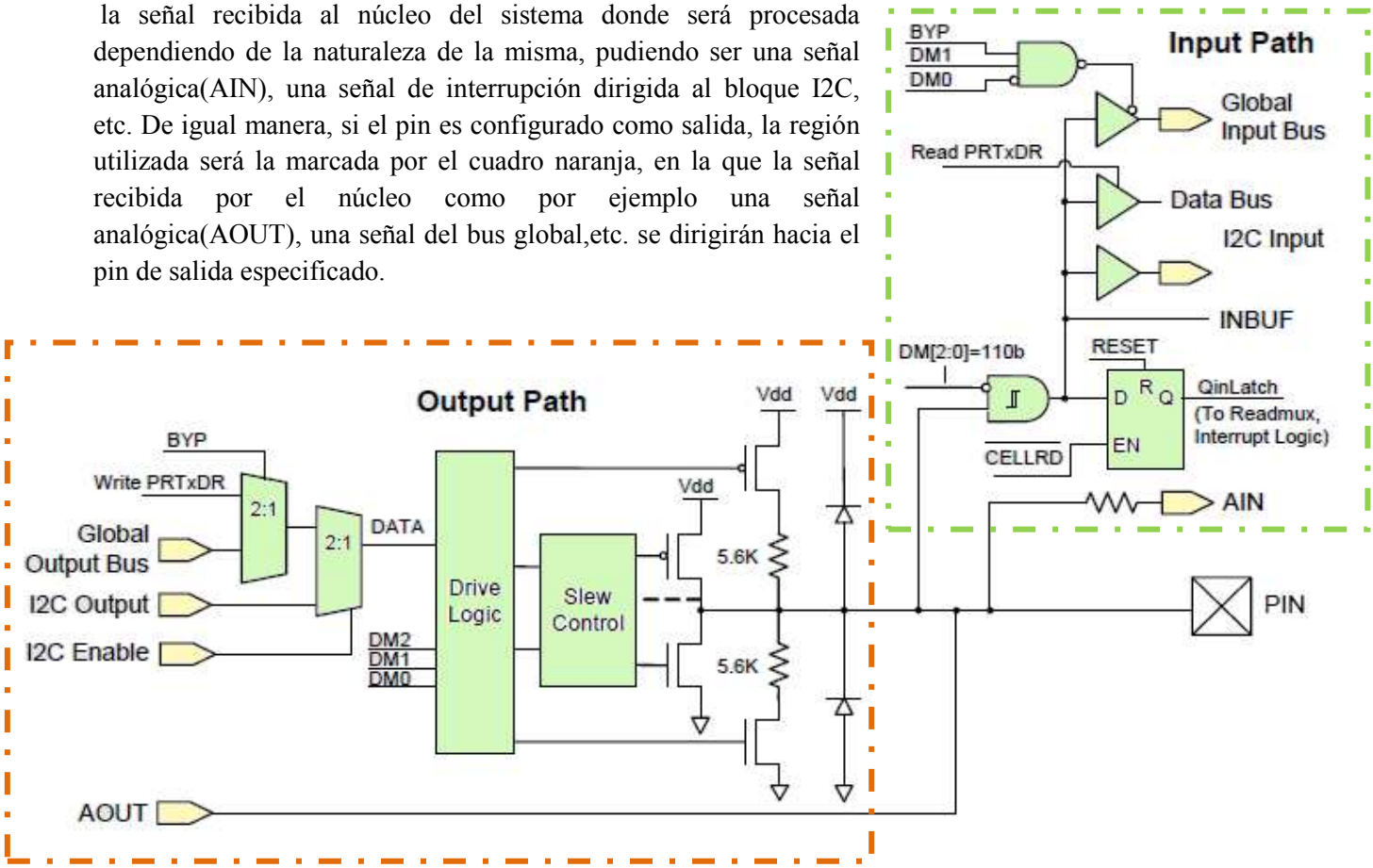
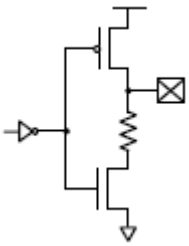


Fig. 3.31 Diagrama de bloques de un GPIO

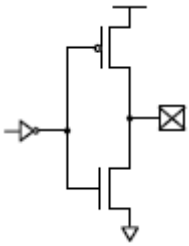
Cada uno de estos GPIO´s podrá utilizarse de una forma diversa dependiendo de la función que vaya a desempeñar:

- IO Globales: entradas/salidas para bloques digitales del PSoC
- IO Analógicas: entradas/salidas para bloques analógicos del PSoC
- IO Digitales: entradas/salidas controladas por software

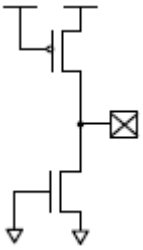
Además, cada GPIO adoptará mediante SW un modo de conducción de cada pin dependiendo de la naturaleza de la señal que vaya a pasar por el mismo. Esta operación se realiza fácilmente mediante el PSoC Designer y básicamente se basa en la modificación de un registro llamado PRTxDMx que se realizará de forma automática a través de su interfaz gráfica. Dichos modos de conducción, la configuración del registro PRTxDMx y la situación del pin en situación de '0' y de '1' se muestran a continuación:



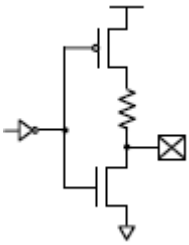
Modo: Pull down Resistivo
PRTxDMx: [000]
Situación en '1': Strong
Situación en '0': Resistivo



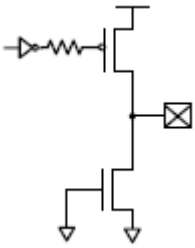
Modo: Conductor Strong
PRTxDMx: [001]
Situación en '1': Strong
Situación en '0': Strong



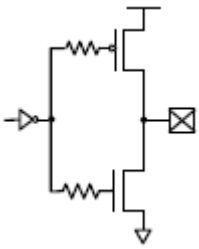
Modo: Alta Impedancia (High Z)
PRTxDMx: [010]
Situación en '1': Alta Z
Situación en '0': Alta Z



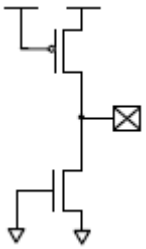
Modo: Pull up Resistivo
PRTxDMx: [011]
Situación en '1': Resistivo
Situación en '0': Strong



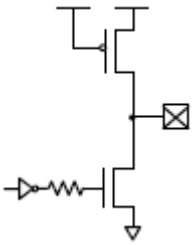
Modo: Drenaje abierto/ conductor alto
PRTxDMx: [100]
Situación en '1': Strong lento
Situación en '0': Alta Z



Modo: Conductor Strong lento
PRTxDMx: [101]
Situación en '1': Strong lento
Situación en '0': Strong lento



Modo: Alta Impedancia Analógica
PRTxDMx: [110]
Situación en '1': Alta Z
Situación en '0': Alta Z



Modo: Drenaje abierto/ conductor bajo
PRTxDMx: [111]
Situación en '1': Alta Z
Situación en '0': Strong lento

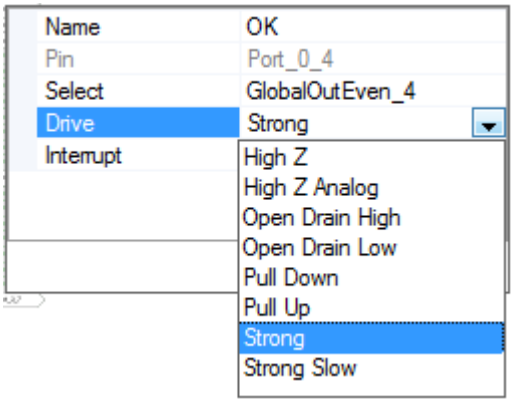


Fig. 3.32 Menú desplegable de un GPIO en PSoC Designer IDE

A la izquierda se muestra una captura de pantalla del PSoC Designer, el software de Cypress para la configuración del mismo. Como se puede ver, es un menú desplegable de configuración de un GPIO, concretamente el puerto P0 [4]. En el tema que nos concierne, para elegir el modo de conducción no hace falta recurrir al registro PRTxDRx para modificar dicha modalidad, sino que se puede realizar fácilmente a través de este menú. En el apartado concerniente al SW de Cypress realizaré una explicación exhaustiva de todas las configuraciones posibles de cada GPIO.

3.2.3.1 E/S Globales

Los puertos GPIO se pueden configurar para ser usadas para interconectar señales a/desde los bloques digitales como entradas/salidas globales. Desactivada por defecto, requiere de la modificación de dos parámetros: el bit de selección de puerto global a través de PRTxGS del puerto que deseemos debe recibir un nivel alto.

Dependiendo de si queremos que sea una entrada o una salida deberemos modificar el modo de conducción. Para el caso de salida usaremos el de alta impedancia, y para el caso de salida cualquiera de los otros tipos que no sean de alta impedancia.

3.2.3.2 E/S Digital

Como ya he explicado en el apartado 3.2.4 Sistema digital , una de las principales operaciónes de los GPIO´s digitales es de enviar y recibir información desde el núcleo del PSoC hacia el exterior. Esto se lleva a cabo a través del registro de datos del puerto: PRTxDR.

De esta forma, el microprocesador del núcleo puede conocer el estado de un pin leyendo el estado del registro PRTxDR, transformando después el voltaje actual del pin en un valor lógico y devolviéndolo al microprocesador. Algunos ejemplos simples de instrucciones de lectura, escritura y modificación de este puerto son las operaciones lógicas NOT, OR, AND, y XOR.

Considerando la x de PRTxDR el numero de puerto que será sometido a modificación, es fácil cambiar el valor de un bit de un puerto. Por ejemplo, si queremos modificar cualquiera de los valores de un bit del puerto 1, bastará con lo siguiente: PRT1DR|=0x10. De esta manera, accederemos al byte del puerto 1, pudiendo modificar cualquiera de sus 8 bits (en hexadecimal). En este caso, se habrá modificado a '1' el bit 4 del puerto 1 mediante la operación OR.

3.2.3.3 E/S analógicas

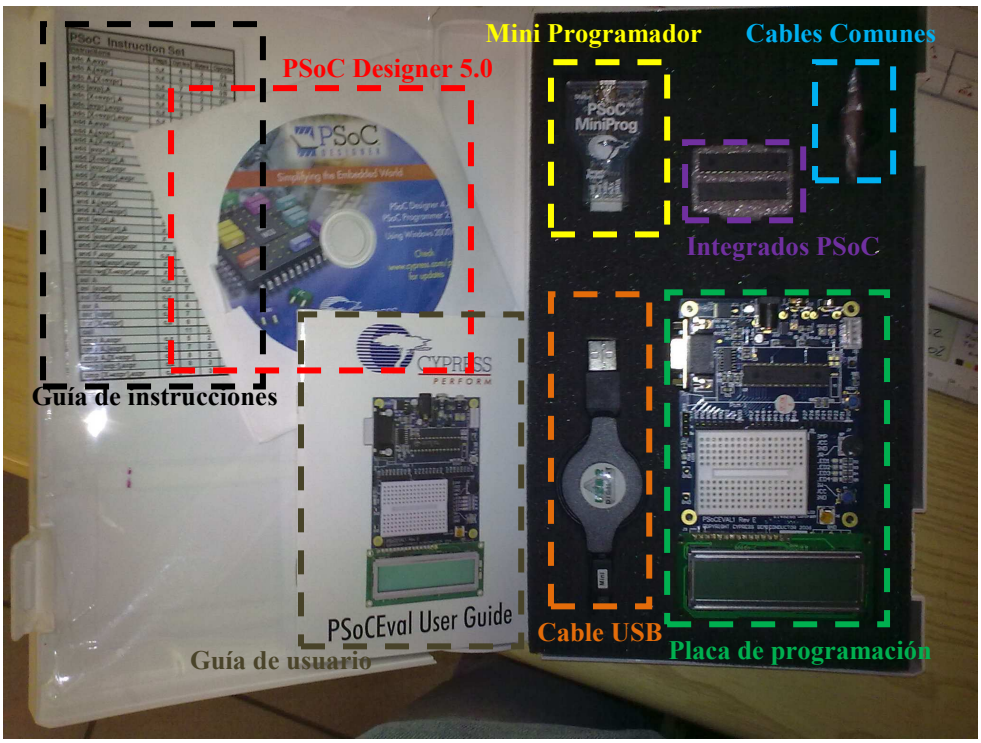
Las señales de naturaleza analógica entrarán en el núcleo del PSoC a través de los pines del integrado por la patilla AOUT del bloque del GPIO como puede verse en la figura x.x . Normalmente para la recepción de señales analógicas, el modo elegido será de Alta Impedancia Analógica ya que este modo desactiva el inversor Schmitt. En lo referente a las salidas analógicas, están explicadas detalladamente en el apartado 3.2.3.

3.3 Hardware y Software de diseño/programación PSoC

Para realizar la programación tanto del software como del hardware del PSoC, Cypress cuenta con una herramienta útil y sencilla para el desarrollo de aplicaciones basadas en sistemas embebidos. Por un lado, para conocer las capacidades del PSoC conviene adquirir en un principio la herramienta de evaluación CY3210, hardware que permite conocer y testear las capacidades del PSoC y que a la vez ayuda al usuario a familiarizarse con el software de programación, el PSoC Designer. Ambas herramientas de desarrollo serán explicadas ampliamente en los siguientes apartados.

3.3.1 CY3210-PSoC EVAL 1

El CY3210-PSoC EVAL 1 es una placa de evaluación que permite conocer las características y capacidades de los productos de Cypress. Antes de comenzar con la realización de mi proyecto, he debido conocer y aprender a utilizar la herramienta en la que se basaría mi sistema de control. El contenido de este kit de evaluación, que cuenta con todo lo necesario para empezar a desarrollar desde cero proyectos basados en PSoC, se enumera a continuación:



- Placa de evaluación PSoC
- Mini programador PSoC
- 2 integrados PSoC: CY8C29466-24PXI y CY8C27443-24PXI
- Cable USB A-miniB
- Cable común de cobre
- Software de programación PSoC Designer 5.0
- Guía de usuario de la placa de evaluación
- Guía de instrucciones en ensamblador para el PSoC

Fig.3.33 Contenido del kit CY3210-PSoC EVAL 1 de Cypress

Lo fundamental de este kit es la placa de programación, el software PSoC Designer, los integrados PSoC y el mini programador los cuales explicaré más detalladamente. La placa de programación cuenta con todo tipo de utilidades para experimentar con el PSoC mientras que el mini programador nos será útil para, como su nombre indica, programar el integrado PSoC con el código desarrollado en el PSoC Designer, que será explicado en profundidad posteriormente.

Los otros componentes incluidos en el kit, como la guía de usuario y la guía de instrucciones, nos facilitaran los primeros pasos con esta herramienta. El cable USB servirá para conectar el programador al PC y los cables de cobre serán útiles para realizar conexiones simples en la placa de programación. A continuación se detalla el contenido de la placa de programación.

3.3.1.1 Placa de programación y mini programador

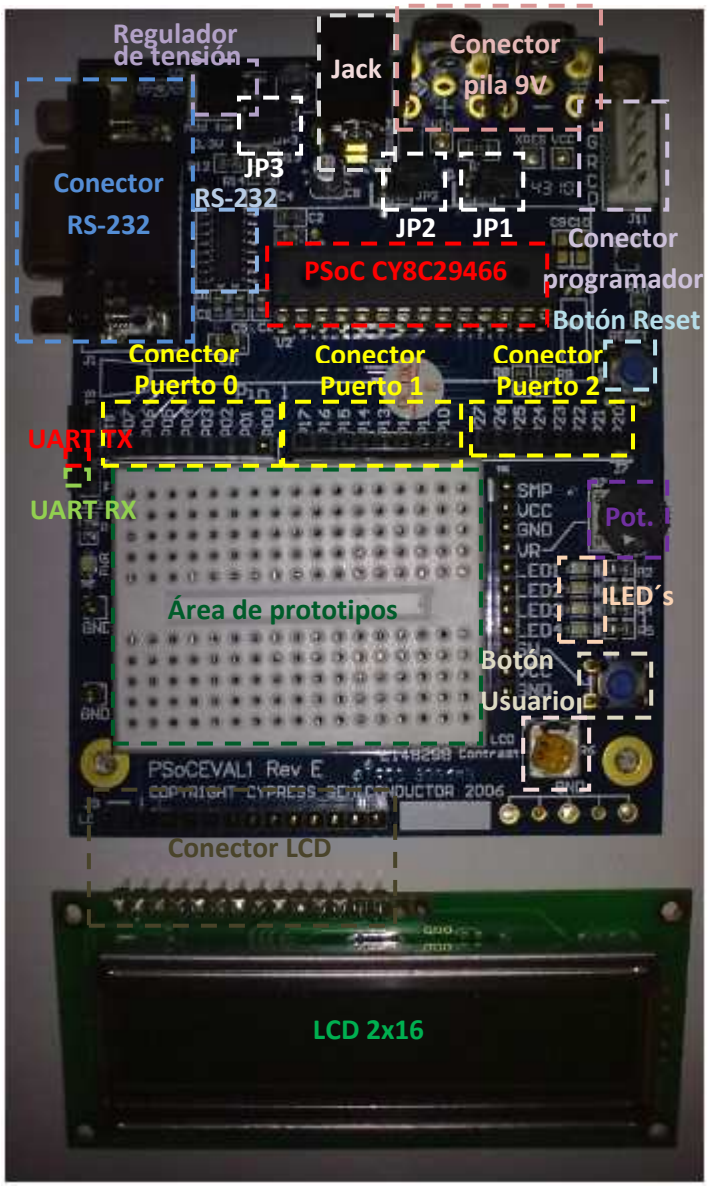


Fig. 3.34 Placa de programación del kit CY-3210

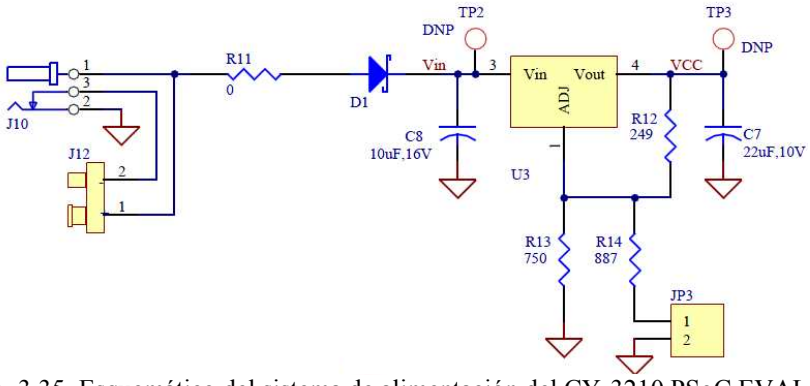


Fig. 3.35. Esquemático del sistema de alimentación del CY-3210 PSoC EVAL1

La placa de programación permite experimentar con el dispositivo PSoC como parte del aprendizaje previo a la creación de un proyecto basado en el mismo. Esta se puede dividir en cinco subpartes dependiendo de la función que representa cada una de ellas en el conjunto de la misma. Estas partes son las siguientes: Sistema de alimentación, interfaz de programación, interfaz del RS-232, área de prototipos y área de visualización. Cada una de estas partes realiza una función fundamental para el desarrollo de aplicaciones en la placa de programación y cada una de estas partes están formadas por los siguientes componentes:

- Sistema de alimentación (zona superior central): Cuenta con un sistema de alimentación realmente versátil ya que cuenta con varias posibilidades. Lo más común es alimentar la placa con los 5V que le proporciona el mini programador mediante USB o bien mediante una pila de 9V gracias al zócalo que incorpora.

  Además existe la posibilidad de alimentarla mediante un transformador de 9V a 12V DC conectado al Jack que incorpora. Mediante el regulador de tensión incorporado, se convertirá la tensión de alimentación (si procede) en 3.3V /5V dependiendo de las necesidades del proyecto a realizar.

  Por otro lado, no es posible usar más de una fuente de alimentación al mismo tiempo o si es menor a 7V o superior a 12V.

  También cuenta con 3 jumpers para habilitar/deshabilitar funciones: el jumper 1 y 2 tienen relación con el interfaz del RS-232 y se explicara su función a continuación mientras que el jumper 3, como se puede ver en el esquemático de la derecha, determinará el funcionamiento del regulador de tensión a 3.3V (conectado) o 5V (desconectado).

- Interfaz de programación (zona superior derecha): Para evitar aumentar el tamaño de la PCB y permitir reprogramar al usuario el PSoC tantas veces como necesite, la placa de evaluación CY-3210 incluye un conector ISSP donde se conectará el mini programador, que además de permitir reprogramar el PSoC podrá servir de alimentación de la placa de programación.

  En la siguiente imagen podemos ver el mini programador PSoC, donde además del alzado se muestran sendas imágenes de la planta y de la planta inferior, donde se encuentran por un lado, el conector ISSP de 5 pines que se conectará a la placa de programación y, por el otro, un conector mini USB tipo B para su conexión a un PC permitiendo la programación o alimentación de la placa CY3210. También se puede ver el esquemático del conector de programación conectado al PSoC donde queda resaltado por un círculo rojo el conector ISSP que se encuentra en la placa de programación.

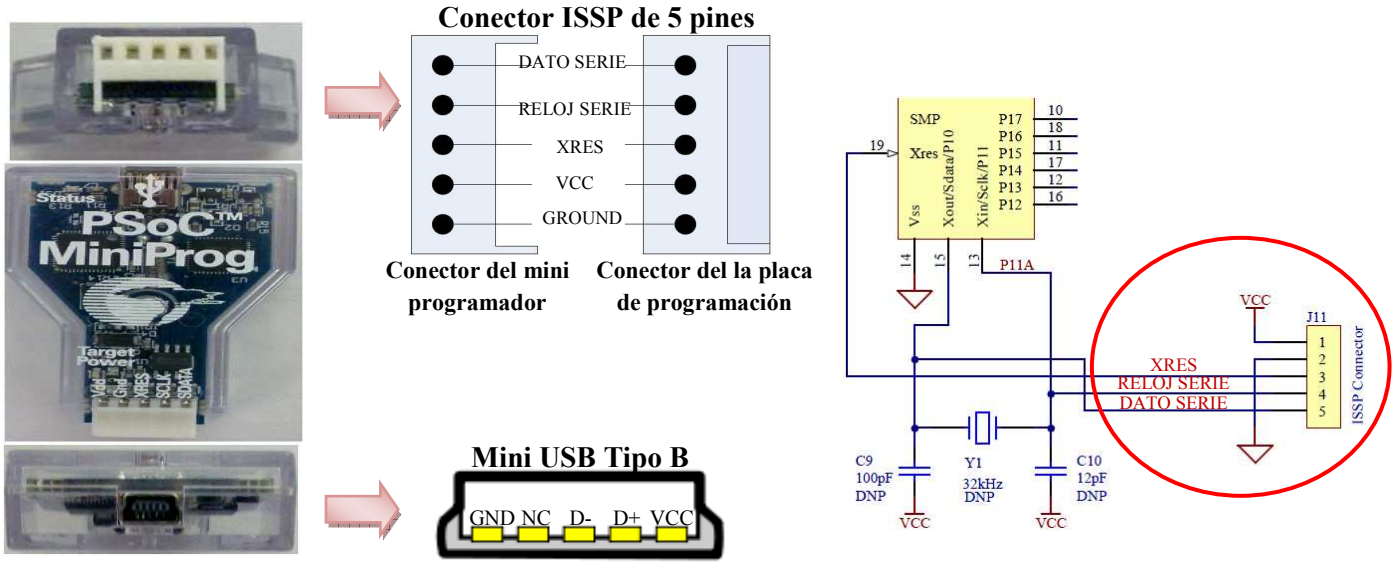


Fig. 3.36 Detalle de los conectores del Mini Programador PSoC para CY-3210 PSoC EVAL1

- Interfaz del RS-232 (Zona superior izquierda): La placa de programación cuenta con un transceptor RS-232, con una configuración Rx y Tx, para realizar pruebas de comunicación serie con envio/recepción de un bit cada vez. Esta aplicación se utilizará a través de la API UART (Universal Asynchronous Receiver-Transmitter) en el PSoC Designer.

  La tensión de salida será de Vcc-0.6V(para nivel alto) y de 0.4V(para el nivel bajo). Los jumpers J1 y J2 deberán ser modificados para poder hacer uso de la comunicación UART a través del RS-232: conectando J1, uniremos el puerto P[1]6 con el pin Rx y conectando J2, uniremos el puerto P[2]7 con el pin Tx. En la siguiente imagen se muestra la conexión esquemática de los jumpers , el transceptor RS-232 y el conector DB9:

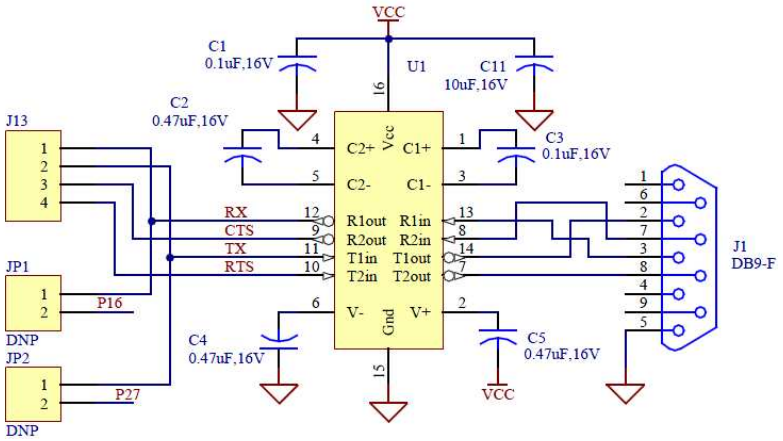


Fig. 3.37. Esquemático del transceptor RS-232, con el conector DB9 y la configuración de Jumpers

- Área de prototipos(zona media e inferior): En esta región de la placa es donde se encuentran un gran número de funcionalidades para realizar pruebas con el PSoC. Podemos encontrar cinco conectores, tres de los cuales son extensores de los puertos del PSoC, mientras que los dos restantes, uno es para la conexión del LCD y el otro reúne cuatro diodos LED, un pulsador, un potenciómetro, dos patillas de alimentación y dos de masa. En esta región podemos encontrar un alto número de puntos a alimentación y a masa para facilitar las conexiones en las pruebas. Los esquemáticos son los siguientes:

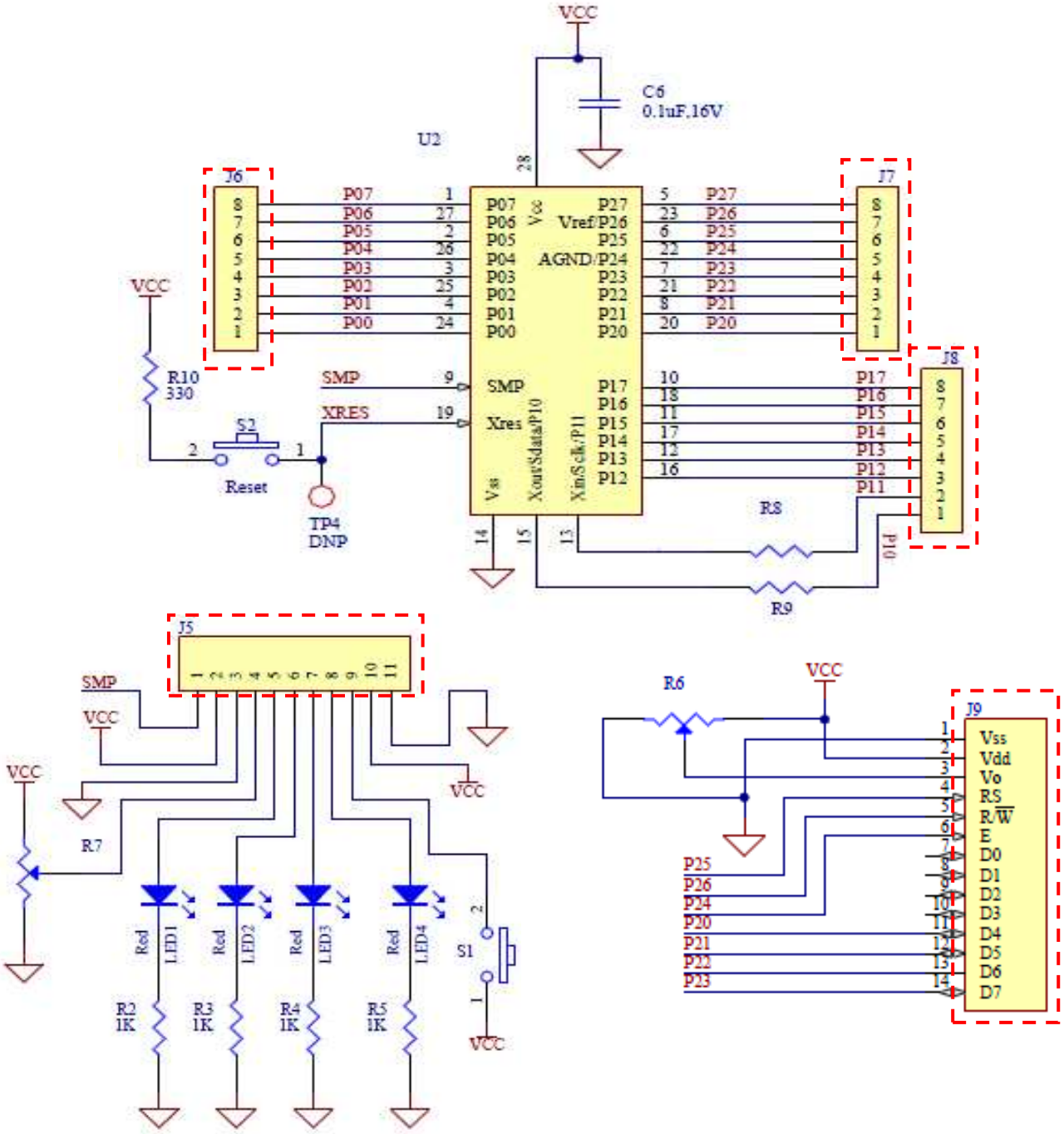


Fig. 3.38 Esquemáticos que conforman el área de prototipos. Detallados en rojo se muestran los conectores en la placa.

3.3.2 PSoC Designer

PSoC Designer IDE (entorno de desarrollo integrado) es el software de Cypress que permite desarrollar aplicaciones en los PSoC. Mediante un intuitivo interfaz gráfico es posible configurar todas las opciones tanto hardware como software del PSoC. Este software permite al usuario utilizar dos lenguajes de programación (lenguaje C o lenguaje ensamblador) incluidos en el mismo para el desarrollo de proyectos basados en el PSoC.

Fig. 3.39 Logo de PSoC Designer

La versión sobre la que he trabajado es la 5.0, si bien existen actualizaciones en la página web de Cypress para descargar actualizaciones tanto del PSoC Designer como del PSoC programmer. Una vez instalado el software con el CD de instalación o bien, descargando el ejecutable desde la web, ejecutaremos el programa desde el icono que se generará automáticamente en el escritorio:

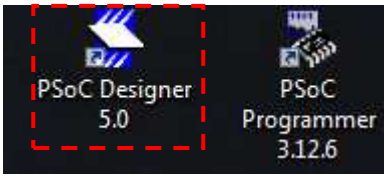


Fig. 3.40 Iconos generados en el escritorio tras la instalación del PSoC Designer

Tras haber ejecutado el programa, aparecerá la pantalla de inicio que vemos en la imagen inferior, donde podremos iniciar un nuevo proyecto o abrir un proyecto usado recientemente. En caso de iniciar un nuevo proyecto, aparecerá una ventana en la que elegiremos el modelo de PSoC y el lenguaje de programación que se usara en el proyecto. En las siguientes se explicaran en profundidad todas las pantallas de explotación del programa asi como las opciones de personalización a la hora de realizar un proyecto.

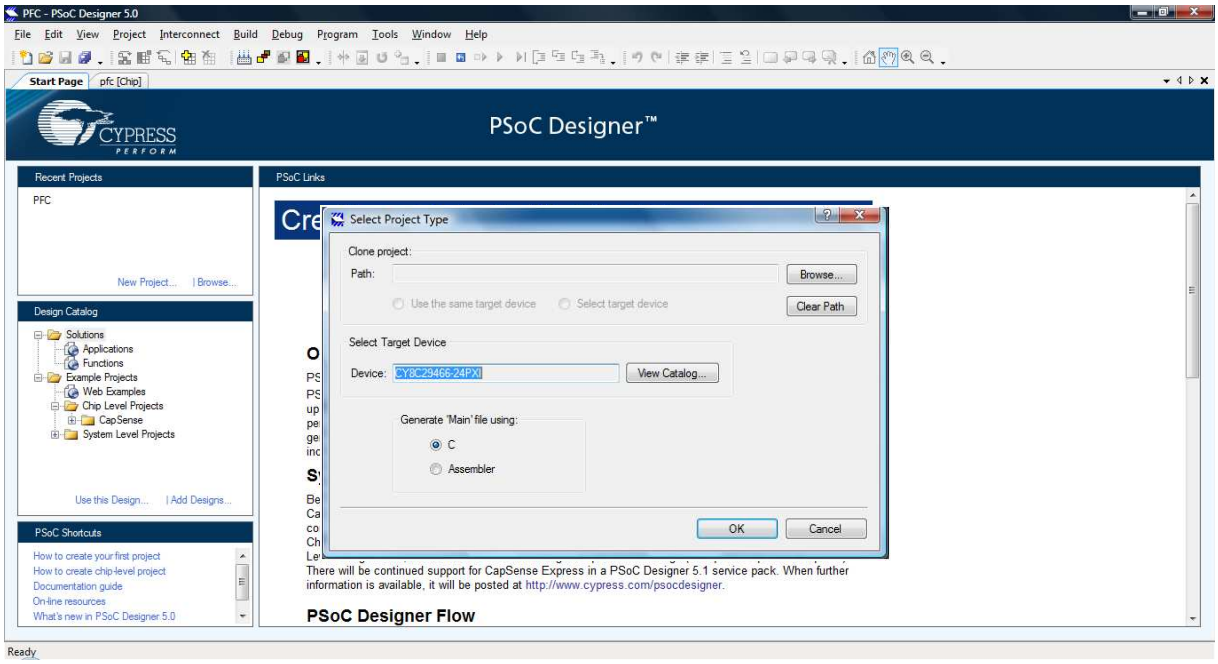


Fig. 3.41 Pantalla de inicio del PSoC Designer

3.3.2.1 Editor de hardware (Device Editor)

Una vez creado el proyecto o seleccionado el proyecto sobre el que trabajar, nos encontraremos con la siguiente pantalla de explotación. Cada una de las sub ventanas han sido nombradas de la A a la H y cada una de ellas aporta la siguiente información:

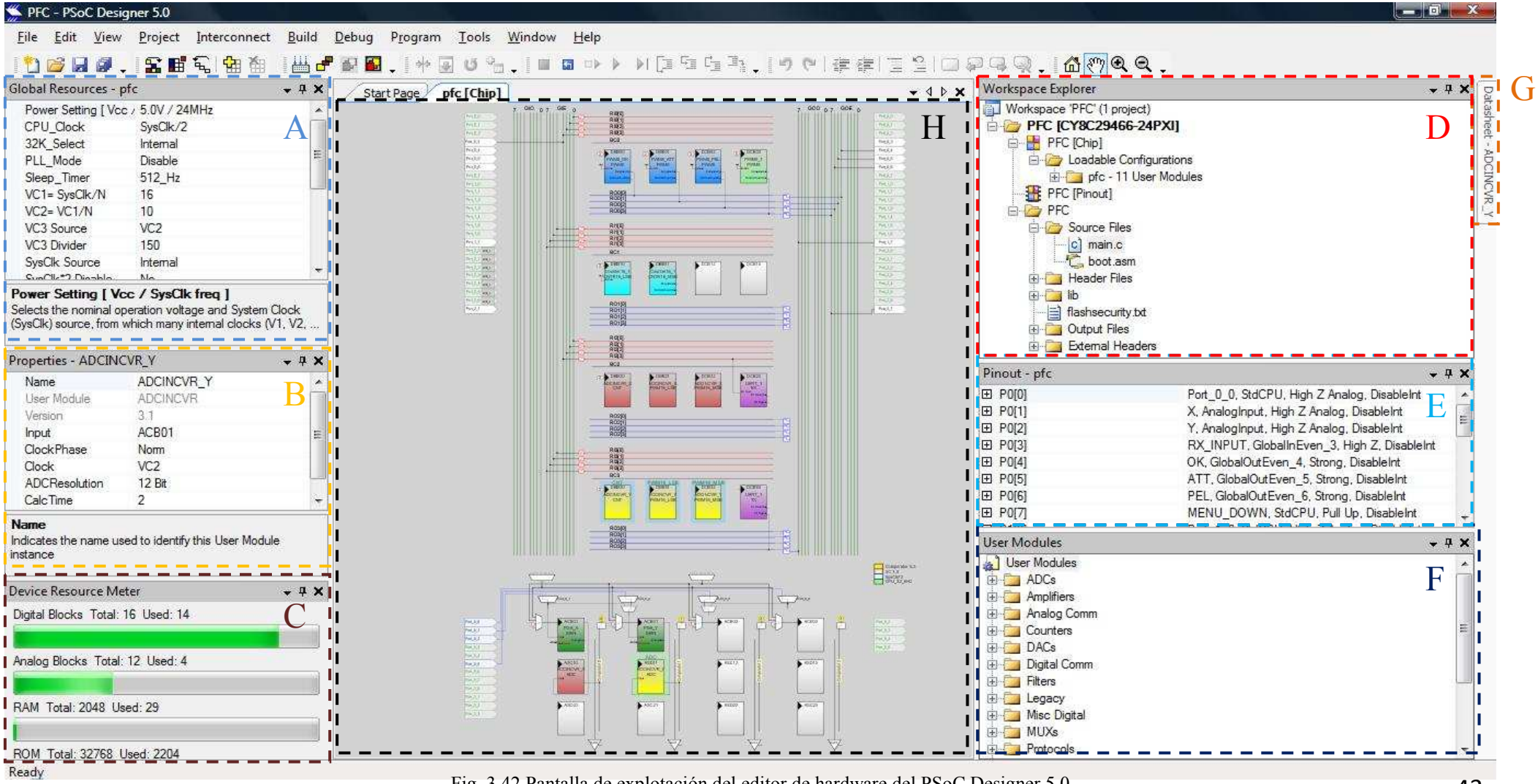


Fig. 3.42 Pantalla de explotación del editor de hardware del PSoC Designer 5.0

A. Recursos Globales (Global Resources)

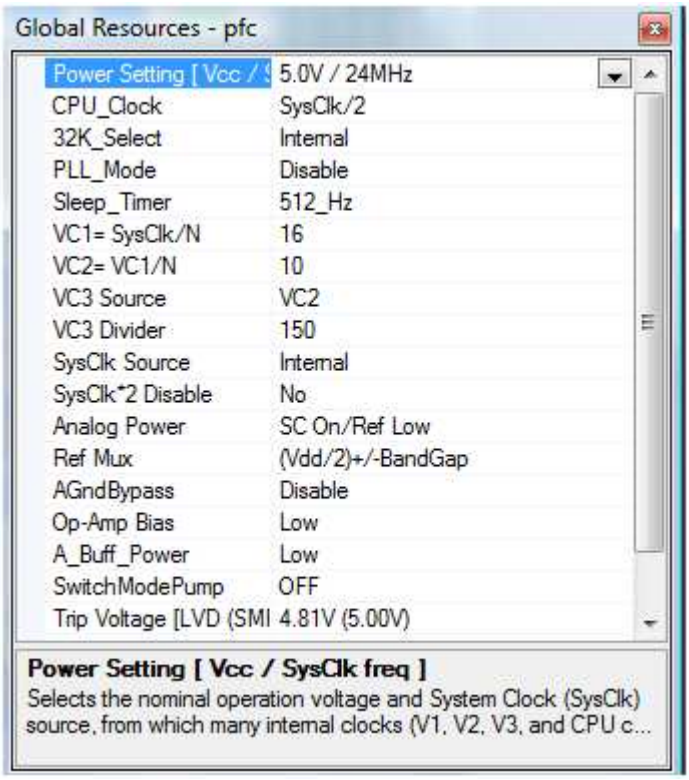


Fig. 3.43 Recursos Globales

En esta ventana se puede configurar los parámetros globales del proyecto. Se puede observar como cuenta con una pequeña ventana donde se muestra una explicación de cada una de las opciones. Las opciones más destacables de configuración son las siguientes:

- Power Setting (Configuración de Energía): Se puede seleccionar la tensión de alimentación y el reloj del sistema(SysClk)
- CPU_Clock (Reloj de la CPU): Sirve para elegir el la velocidad del reloj de la CPU a partir de SysClk que se puede dividir desde valores entre 2 y 256. De esta manera, se podrá seleccionar un reloj de la CPU entre 93.75KHz y 24MHz.
- Sleep_Timer(Temporizador de modo Sleep): Permite seleccionar el valor del temporizador del modo Sleep entre 1 y 512Hz
- VC1, VC2, VC3 Source (fuente de VC1, VC2 y VC3) y VC3 Divider (Divisor de VC3): Estas opciones permiten configurar las señales de reloj internas que se pueden utilizar para los bloques tanto analógicos como digitales. VC1 será un múltiplo entre 1 y 16 de SysClk, VC2 un múltiplo entre 1 y 16 de VC1 mientras que para VC3 elegiremos la fuente que será dividida para obtener el valor de la frecuencia del mismo, entre 1 y 255.
- SysClk Source (Fuente de SysClk) y SysClk*2 Disable (Deshabilitación de SysClk*2): Mientras la primera opción determina si la fuente de SysClk es proporcionada internamente por el PSoC o bien por un cristal externo, la segunda permite escoger la posibilidad de utilizar el reloj del sistema a doble velocidad.

B. Propiedades del bloque

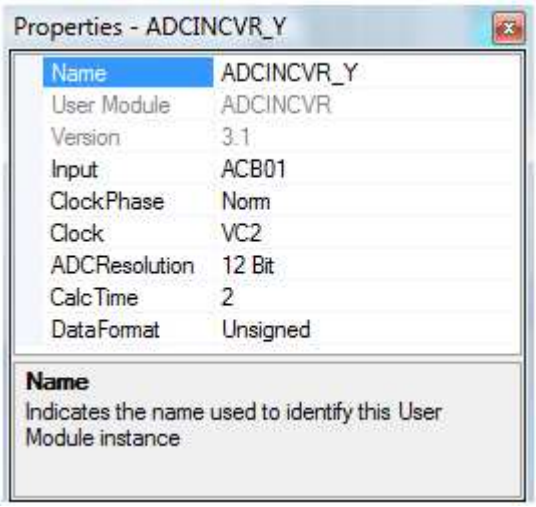


Fig. 3.44 Propiedades del bloque

Cada uno de los bloques, ya sea analógico o digital, puede ser totalmente configurable pudiendo elegir desde el nombre o el puerto de entrada/salida hasta la fuente de reloj.

En el caso de la imagen, el cuadro corresponde con las propiedades de un bloque conversor analógico digital en la que se puede elegir la entrada al bloque, la señal de reloj o el numero de bits del mismo.

Como ya se ha comentado, cada bloque tiene unas propiedades diferentes y, por lo tanto esta ventana variará dependiendo del bloque utilizado.

C. Medidor de recursos del dispositivo (Device resource meter)

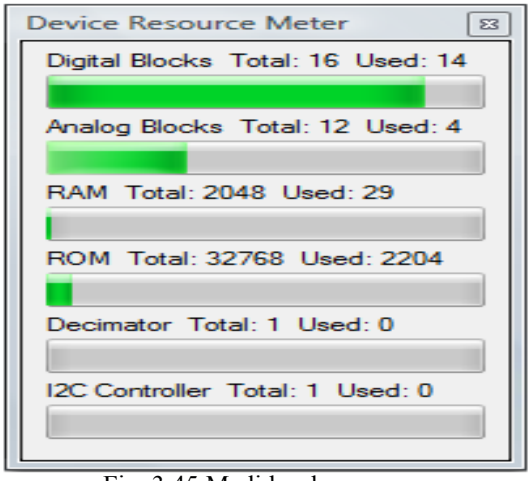


Fig. 3.45 Medidor de recursos

Aunque esta ventana no tiene ninguna capacidad de configuración del PSoC, es muy útil para el usuario ya que permite conocer la cantidad de espacio que utiliza nuestro proyecto en todo momento.

Como se puede ver en la imagen, podemos ver el que el medidor tiene las lecturas del total de bloques digitales, bloques analógicos, memoria RAM y ROM, el muestreador, el controlador I2C y la cantidad de bloques utilizados en el proyecto. De esta forma, el usuario podrá saber la disponibilidad de espacio disponible mientras realiza su proyecto.

D. Explorador del proyecto (Workspace explorer)

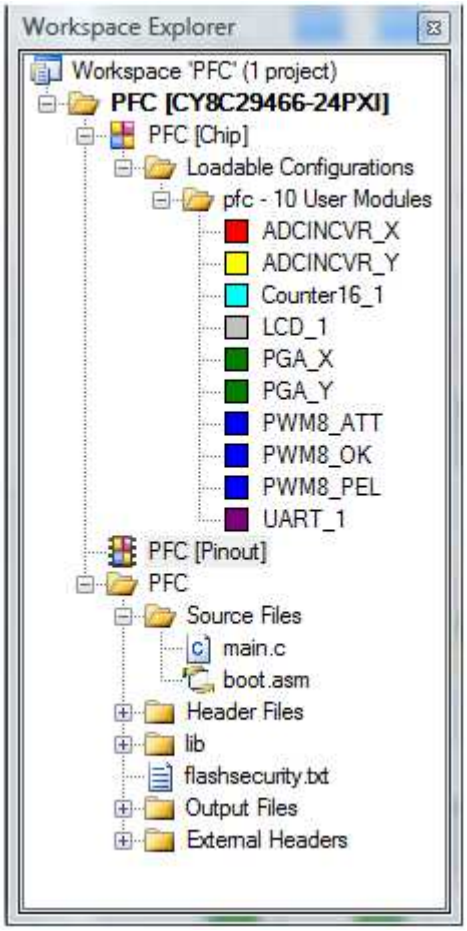
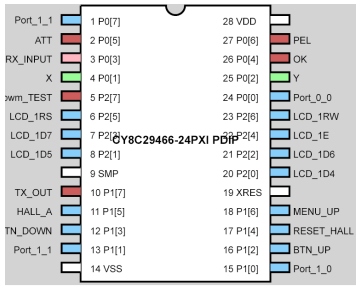


Fig. 3.46 Explorador de proyecto

Todo proyecto cuenta con una serie de ficheros y archivos asociados. Para poder acceder a ellos en todo momento contamos con esta ventana que permite al usuario ver los archivos que conforman un proyecto. Como se puede ver en la imagen, el proyecto PFC basado en el PSoC CY8C29466-24PXI, cuenta con carpetas principales que contienen otras subcarpetas. Estas 2 carpetas principales son:

- Chip: Esta carpeta se corresponde con la configuración del hardware del proyecto en la que se incluyen los bloques de usuario utilizados y la disposición de los pines utilizados en el PSoC. Por ejemplo, en este proyecto se han utilizado varios Conversores AD, PWM´s, etc. Para tener una idea de la disposición de las patillas utilizadas se muestra una imagen reducida:



- PFC: Esta carpeta incorpora todos los archivos relacionados con el software del proyecto, desde el código en C o en ensamblador, hasta la configuración interna del hardware, como por ejemplo opciones de la memoria flash, librerías, etc.

E. Configuración de los pines (Pinout)

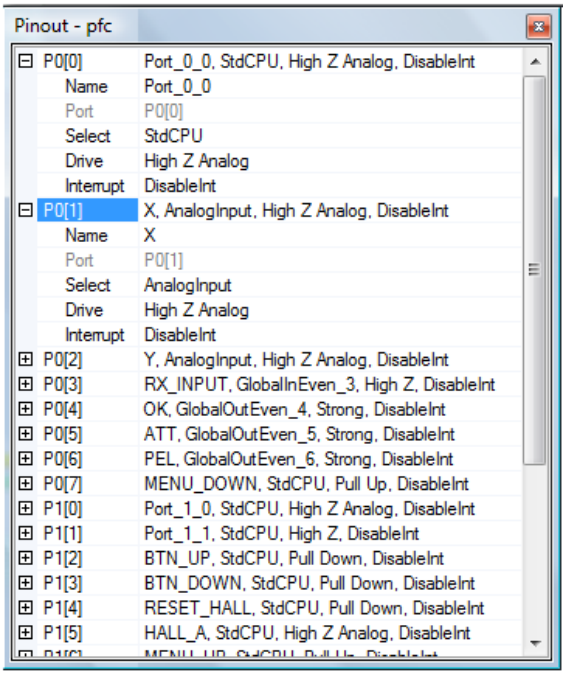


Fig.3.47 Ventana de situación de los pines del proyecto

En esta ventana el usuario puede visualizar la configuración de cada uno de los pines del PSoC. Si bien, cada uno de los pines puede ser configurado en la interfaz gráfica pulsando simplemente sobre el pin a modificar, esta pantalla permite visualizar en todo momento la situación de estos. Los modos de configuración de los pines han sido explicados en el apartado 3.2.3 y por lo tanto no serán comentados nuevamente.

Si queremos ver todas las opciones de configuración de un pin en concreto, bastará con pulsar sobre el botón con el signo + y se desplegaran todas las opciones disponibles como se puede ver en la imagen. En este caso, el pin P0[0] no ha sido modificado y tiene las opciones por defecto donde se puede destacar que el conductor es alta ganancia analógica y que indica que tiene configuración estándar de la CPU mediante StdCPU. Sin embargo, el pin P0[1] utilizado en el proyecto como entrada analógica del sensor de aceleración, podemos ver como incluye el nombre elegido (X) y que recibirá una entrada analógica.

F. Módulos de usuario

Desde la ventana de módulos de usuario, podremos elegir los bloques digitales, analógicos o de comunicación que deseemos para realizar nuestro proyecto. En el apartado 3.3.3 explicaré detalladamente los módulos de usuario que he utilizado en mi proyecto. Todos los módulos de usuario se han mostrado en el apartado 3.2.1 por lo que no repetiré su explicación.

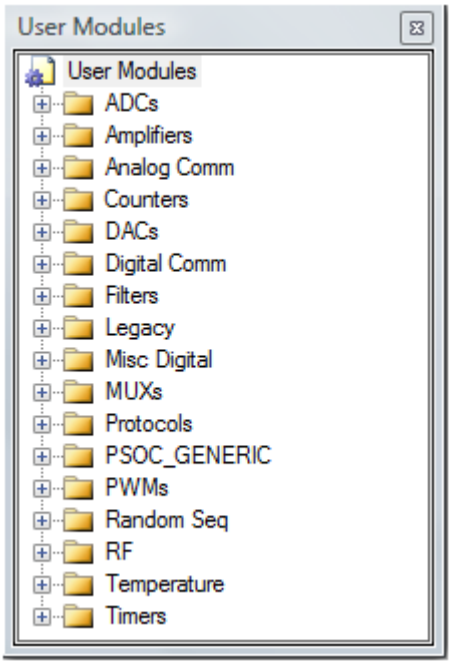


Fig. 3.48 Muestra de todos los Módulos de usuario con los que cuenta el PSoC

G. Ventana de datasheet

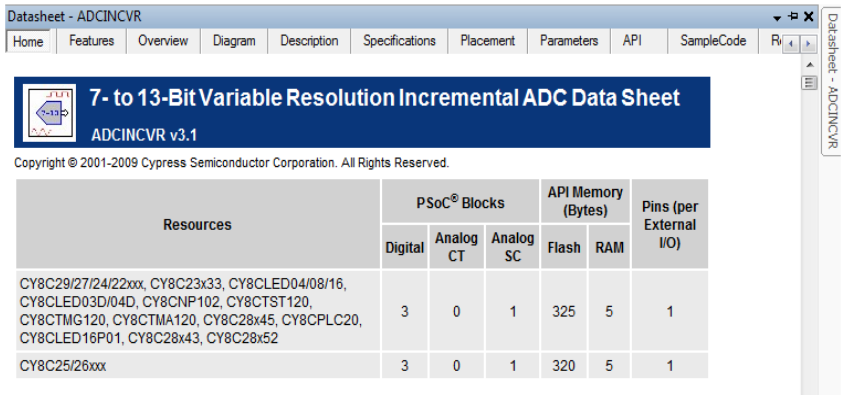


Fig. 3.49 Datasheet incluido en PSoC Designer

Todos los bloques de usuario cuentan con un datasheet donde figuran todos sus datos tanto de hardware como de software. El usuario puede conocer cuantos bloques ocupa en el PSoC, los pines que abarcará del integrado,etc. Además incluye las instrucciones generales de utilización y un ejemplo de código tanto C como ensamblador para poder realizar correctamente su inclusión en el programa.

H. Interfaz gráfica del usuario

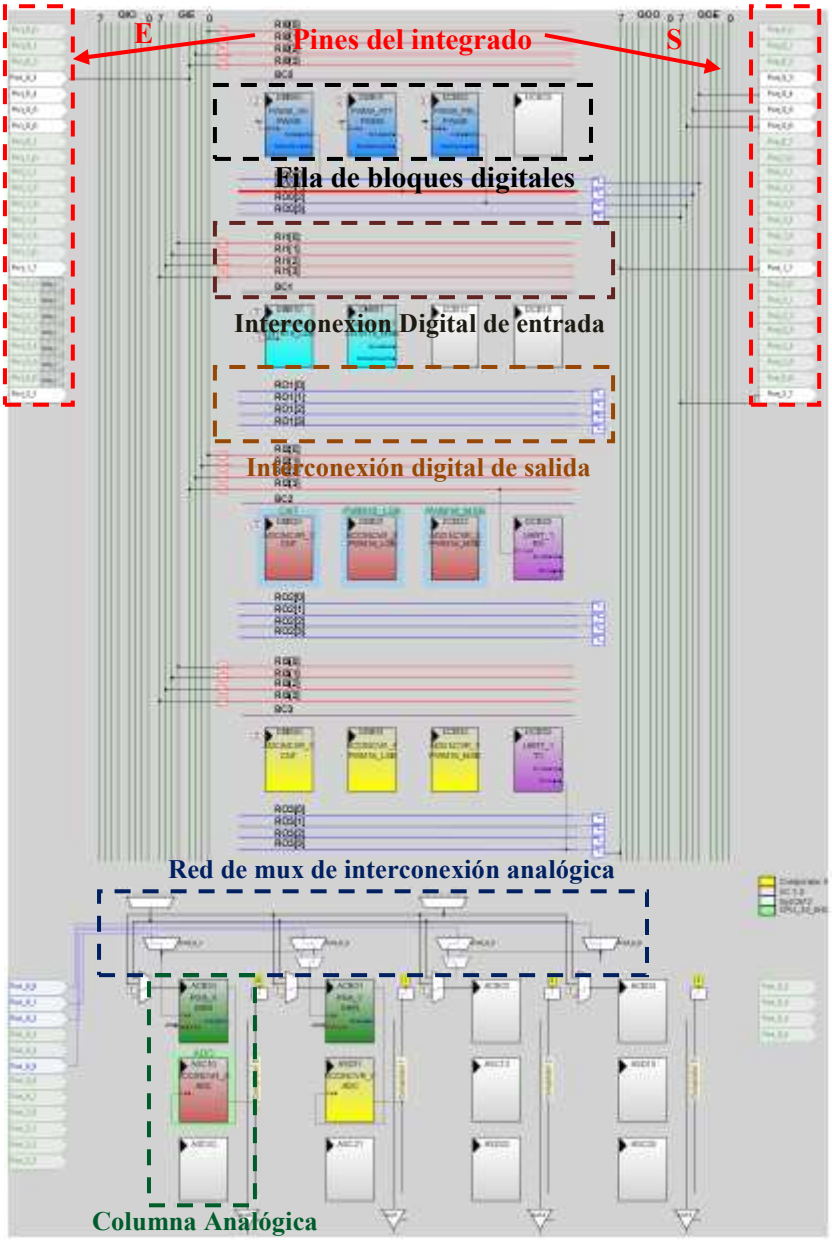


Fig. 3.50. Interfaz gráfica de usuario

Como ya se ha visto en los apartados 3.2.2.2 y 3.2.2.3, figura la disposición de los sistemas analógico y digital del PSoC. Por ello, el interfaz gráfico de usuario cuenta con un aspecto similar. En la parte superior encontramos las cuatro filas de cuatro bloques digitales cada una totalmente interconectada entre ella y a los pines del integrado, asi como cada uno de los bloques que incorpora, lo que facilita una configuración totalmente personalizada ya que facilita al usuario elegir desde la posición del bloque en la fila hasta la línea de interconexión entre el bloque y el pin, ya sea de entrada o de salida.

En la parte inferior, podemos ver las cuatro columnas analógicas totalmente interconectadas por la red de multiplexores. Como en el caso de las filas digitales, el usuario puede configurar totalmente los bloques analógicos, desde su disposición, pines de entrada y de salida, etc.Considero el interfaz gráfico de configuración una de las razones mas importantes para crear un proyecto basado en PSoC gracias a su sencillo y altísimo nivel de configuración de todas las opciones ya que le aporta una gran versatilidad.

3.3.2.2 Editor de software:

Una vez realizada la configuración del hardware y de los bloques utilizados en el proyecto, se debe realizar la confección del código, bien en ensamblador bien en C, dependiendo de la opción elegida al iniciarlo, como ya se ha explicado. Como con el editor de hardware, se han marcado las ventanas más destacables de la letra A a la D que serán explicadas posteriormente. La ventana de creación de código es la siguiente:

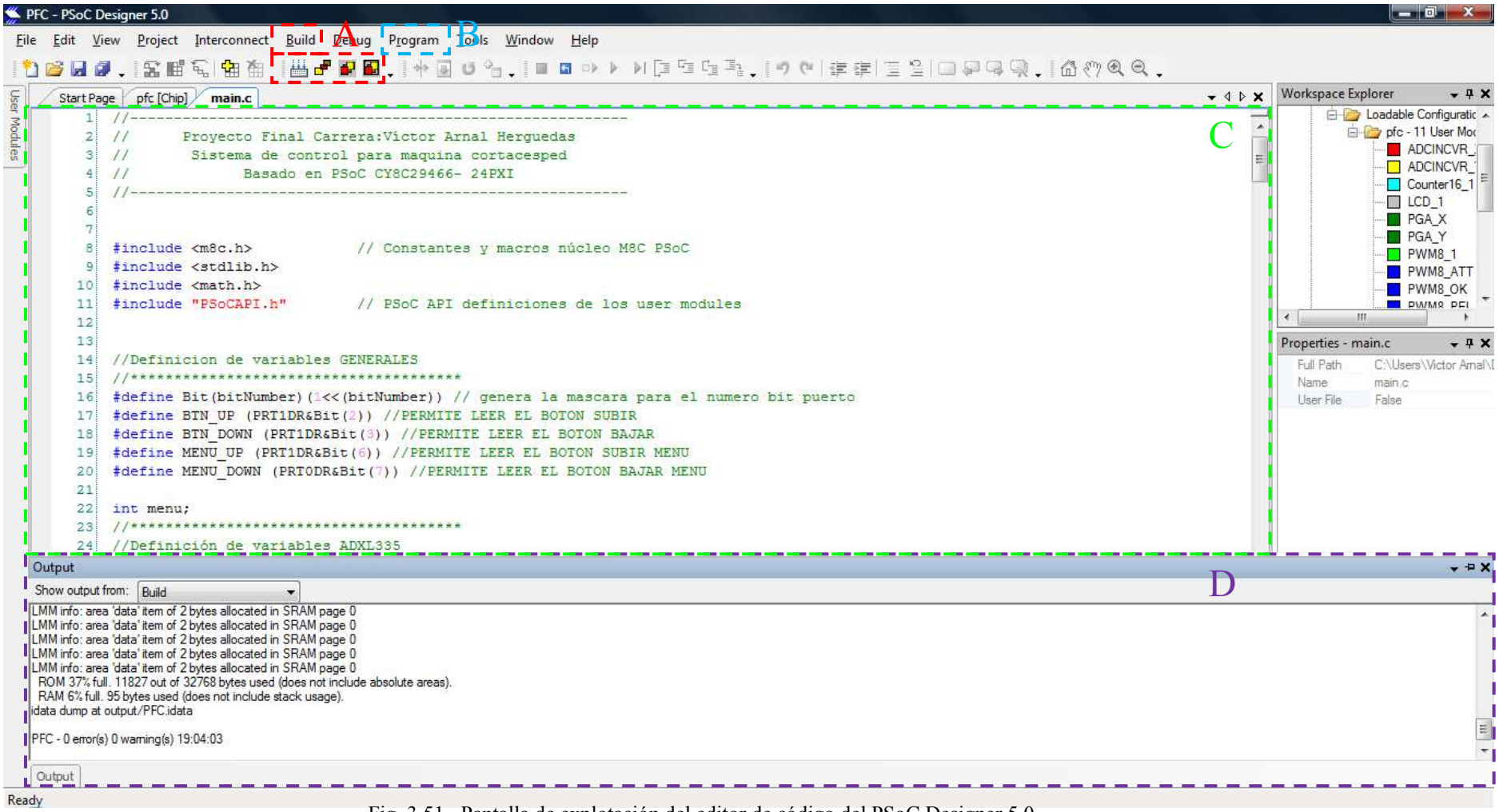


Fig. 3.51. Pantalla de explotación del editor de código del PSoC Designer 5.0

A. Pestaña de compilación de proyecto (Build)

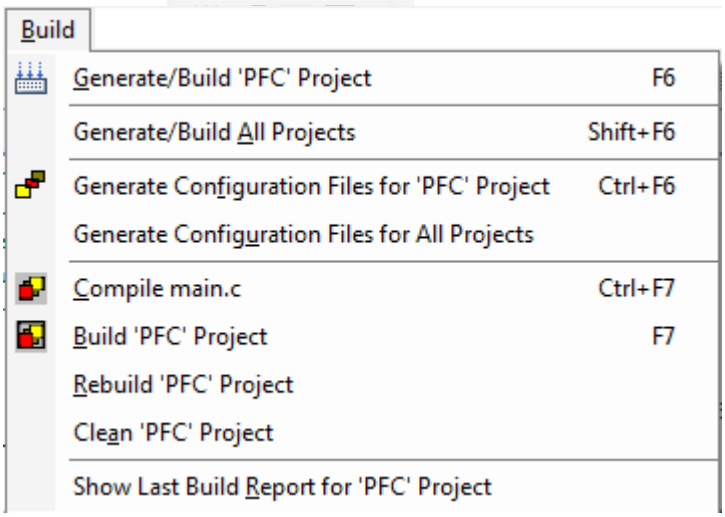


Fig. 3.52 Pestaña con las opciones de compilación/generación de proyecto

La pestaña Build, asi como como el menú de selección rápida, permite generar los archivos de configuración asi como compilar el código creado para un proyecto en concreto. Dependiendo de las necesidades del usuario, podrá realizar desde la simple compliación del código para comprobar la ausencia de errores o bien, la generación de todos los archivos necesarios para programar el PSoC y ejecutar el programa a través del mini programador. Una vez realizada una compilación o una generación de proyecto, aparecerá la ventana Output en la parte inferior de la ventana donde mostrará información acerca de los datos procesados.

B. Pestaña de Programación (Program)

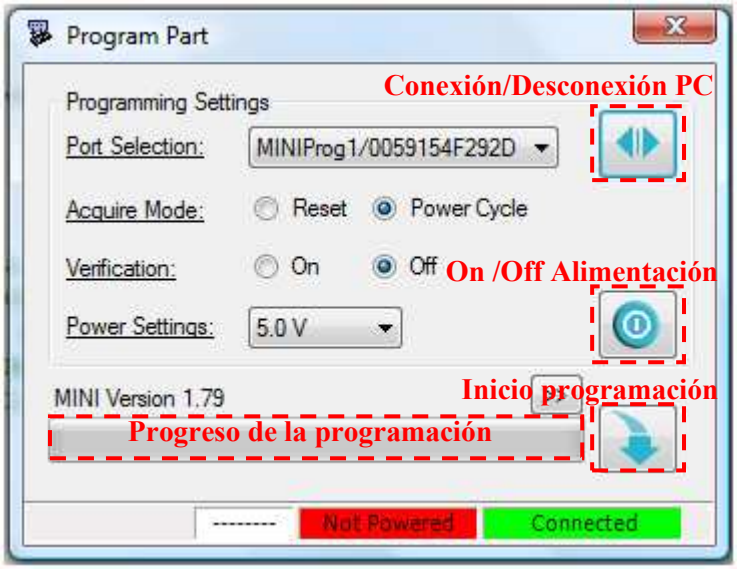


Fig. 3.53 Ventana de programación del PSoC

Una vez que la compilación está libre de errores se podrá programar el PSoC para comprobar el satisfactorio funcionamiento del código y de la configuración del hardware realizado. Pulsando sobre la pestaña Program Part, accederemos a esta ventana donde se especifican los datos del programador y donde se ofrece al usuario las opciones de adquisición asi como de energía. Una vez elegidas las opciones deseadas se deberá pulsar el botón señalado para iniciar la programación. Para alimentar la placa una vez finalizada la programación bastara con pulsar el botón señalado en la imagen.

C. Ventana de Código

Esta es la ventana dedicada a la confección del código, ya sea lenguaje C o ensamblador.

D. Ventana de salida (Output)

Cada vez que se compila o se genera un proyecto, esta ventana indica si existen fallos o advertencias además de los archivos generados en caso de no exisitir ningún error.

3.3.3 Módulos de usuario

Como ya he explicado en el apartado 3.3.2.1, explicare detalladamente los módulos de usuario que he utilizado en mi proyecto mostrando sus características, conexiones típicas y ejemplos extraídos del código (en mi caso C) que posibilitan su funcionamiento.

3.3.3.1 LCD

Este módulo de usuario permite la escribir cadenas de caracteres en un LCD, basándose en el protocolo del driver estándar de LCD (Hitachi HD44780), que será el usado para el desarrollo del proyecto. Este LCD estándar cuenta con dos líneas de 16 caracteres cada una y requiere de 7 pines de E/S para su funcionamiento. En la siguiente imagen, podemos ver la disposición del módulo LCD en la pantalla de interfaz gráfica, asi como el diagrama de bloques del mismo:

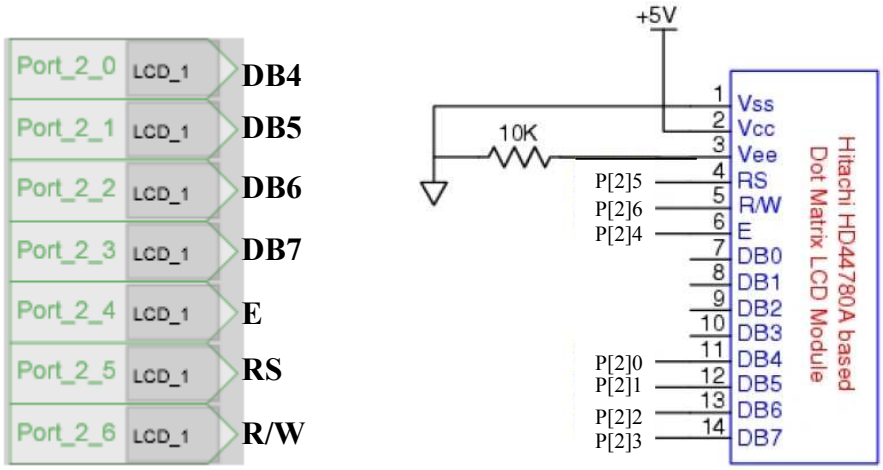


Fig. 3.54 Disposición del módulo LCD en el interfaz de PSoC Designer y su diagrama de bloques

3.3.3.1.1 Descripción / Configuración

Como ya se ha comentado, este bloque de usuario se basa en el driver estándar de Hitachi HD44780 que cuenta con un 8 pines destinados a datos (DB0-DB7), uno a lectura/escritura(R/W), otro a selección de registro(RS) y otro para la habilitación(E). Para evitar un alto uso de pines E/S, se utiliza el modo de 4 bits. En la figura x.x, se puede ver la

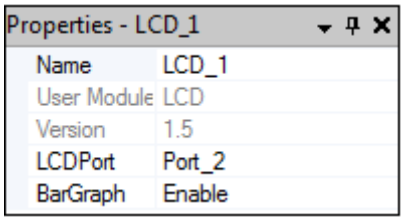


Fig. 3.55 Propiedades del LCD

Para configurar el módulo LCD, basta con acceder a la ventana de propiedades, donde nos encontraremos las opciones que se ven en la imagen ss.: Podremos elegir un nombre (que luego utilizaremos para la programación del software), el puerto en el que se colocará, y la habilitación / deshabilitación del modo gráfico que se comentara a continuación. Si el modo gráfico está activo, es posible crear barras tanto horizontales como verticales en el LCD.

Para la creación de gráficas en el LCD deberemos tener en cuenta cada uno de los 32 caracteres con los que cuenta. Todos han sido numerados para facilitar su uso como se puede ver en la figura x.x Cada uno de estos caracteres tiene 5 pixeles horizontales por 8 verticales con un total de 40 pixeles por carácter. Dependiendo de la necesidad de realizar gráficos horizontales o verticales se deberá tener en cuenta la unidad mínima que se puede dibujar en cada uno de ellos.

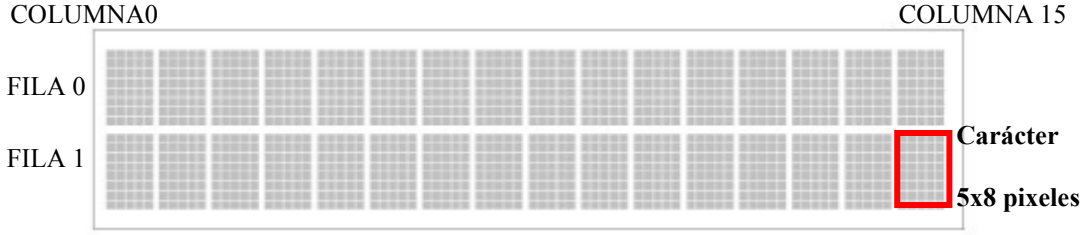


Fig. 3.56 LCD de 2x16 carácteres

3.3.3.1.2 Instrucciones C por defecto (API´s)

El acrónimo API, del inglés Application Programming Interface, hace referencia a las instrucciones con las que el usuario puede utilizar y configurar los bloques de usuario. En el caso del LCD, nos encontramos con instrucciones que van desde la colocación del cursor en una posición determinada hasta la inclusión de gráficos dinámicos. Todas ellas se encuentran recogidas en el datasheet localizado en el PSoC Designer y serán explicadas a continuación. Conviene aclarar que están divididas por el tipo de funcionalidad que ofrece cada una, además de un ejemplo de algunas de ellas:

Funciones básicas:

- *LCD_Start*: Inicializa el LCD en modo 4 bits. Se deberá utilizar antes que las demás API´s del LCD. En el caso de mi proyecto en código C, esta instrucción es como se muestra a continuación: `LCD_1_Start();`

- *LCD_Init:* Realiza un reset de la pantalla LCD.

- *LCD_Position:* Posiciona el cursor al carácter indicado. Se debe tener en cuenta que el carácter de arriba a la izquierda corresponde con la fila 0 y columna 0, mientra que la de abajo a la derecha será la fila 1 columna 15. En mi proyecto, dicha instrucción se usa como se muestra a continuación (en este caso corresponde a la situación del cursor en la fila0, columna3): `LCD_1_Position(0,3);`

- *LCD_Control(LCD_DISP_CLEAR_HOME):* El API LCD_Control permite escribir un byte al registro de control del LCD. En este caso, la función mostrada permite borrar el LCD.

Funciones de Impresión de cadenas:

- *LCD_PrString:* Realiza la impresión en pantalla de una variable almacenada en la memoria RAM en la posición actual del cursor. Esta API será útil para sacar por pantalla datos que serán variables como por ejemplo datos de un sensor. El sensor de aceleración se muestra en pantalla a través de esta instrucción de la siguiente forma: `LCD_1_PrString(mm_char);`

- *LCD_PrCString:* Realiza la impresión en panatalla de una variable almacenada en la memoria ROM en la posición actual del cursor. Esta API, a diferencia de la anterior, permite imprimir en el LCD un dato que será constante, es decir, una cadena de caracteres que permanecerá constante. La instrucción será de la siguiente manera: `LCD_1_PrCString("Grillo SpA");`

Ejemplo de aplicación de las funciones básicas e impresión de cadenas:

```
char *mm_char;
LCD_1_Start();
LCD_1_Position(0,3);
LCD_1_PrCString("Grillo SpA");
LCD_1_Position(1,6);
LCD_1_PrString(mm_char);
```

Como resultado de este código, en pantalla aparecerá en la primera línea del LCD la cadena Grillo Spa mientras que en la segunda aparecerá el valor de la variable mm_char, que variará en función del valor que el sensor le envía al PSoC.

Funciones de impresión de números:

- *LCD_PrHexByte*: Imprimirá en pantalla una cadena de 8 bits como dos números hexadecimales en la posición actual del cursor. Modelo: `LCD_1_PrHexByte( bValue);`

- *LCD_PrHexInt:* Similar al caso anterior, imprimirá en este caso una variable int (corresponde con dos bytes) como cuatro números hexadecimales en la posición del cursor en ese momento.
  Modelo: `LCD_1_PrHexInt(INT iValue);`

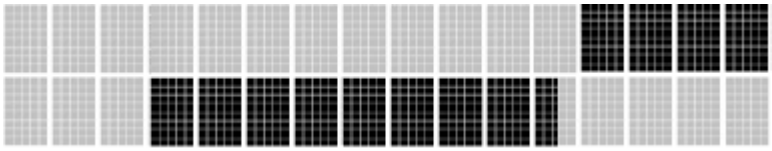
Funciones de gráficos de barras horizontales

- *LCD_InitBG:* Habilita la impresión de barras horizontales en el LCD. Debe usarse antes que cualquier instrucción de dibujo de las mismas. Se debe aclarar, que esta instrucción sólo habilita el espacio para los datos en RAM del grafico especificado y por lo tanto no imprime gráfico alguno. Por otro lado, también se deberá elegir el tipo de barra horizontal que se quiere imprimir: Si se desea una barra sólida con una longitud especificada se usará el comando LCD_SOLID_BG, mientras que si se quiere dibujar una línea simple se usará el comando LCD_LINE_BG. Modelo: *`LCD_1_InitBG(bBGType);`*

- *LCD_DrawBG:* Imprime un grafico/barra horizontal a partir de cuatro variables introducidas por el usuario de la siguiente manera. La API es la siguiente:

```
void LCD_DrawBG(BYTE bRow, BYTE bCol, BYTE bLen, BYTE
                bPixelColEnd);
```

  donde bRow y bCol son la fila y columna de inicio, bLen la longitud máxima de caracteres completos de la barra y bPixelColEnd el pixel columna de finalización de impresión. En la siguiente imagen se muestra un ejemplo:



```
LCD_1_Start();
LCD_1_InitBG(LCD_1_SOLID_BG);
LCD_1_DrawBG(0,11,4,80);
LCD_1_DrawBG(1,3,10,43);
```

Fig. 3.57 Ejemplo de impresión de barras horizontales en el LCD

Funciones de gráficos de barras verticales

- *LCD_InitVBG:* API análoga a LCD_InitBG pero para la impresión de barras verticales. En este caso se podrán dibujar gráficos verticales que abarquen las 2 filas del LCD pero no se podrá elegir entre barra sólida o línea simple.Por tanto, el comando es mucho más sencillo. Modelo: *`LCD_1_InitVBG();`*

- *LCD_DrawVBG:* Imprimirá en el LCD una barra vertical dependiendo como con la instrucción para las barras horizontales de cuatro variables. En este caso la API será:

```
void LCD_DrawBG(BYTE bRow, BYTE bCol, BYTE bHeight, BYTE bPixelRowEnd);
```

donde bRow y bcol son la fila y columna de inicio, bHeigh la altura máxima de caracteres completos de la barra y bPixelRowEnd el pixel fila de finalización de impresión. En la siguiente imagen se muestra un ejemplo:

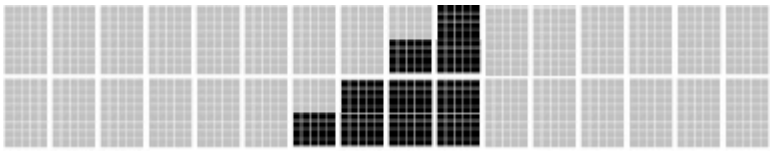


Fig. 3.58. Ejemplo de impresión de barras verticales en el LCD

```
LCD_1_Start();
LCD_1_InitVBG();
LCD_1_DrawVBG(1,6,1,4);
LCD_1_DrawVBG(1,7,1,8);
LCD_1_DrawVBG(1,8,2,12);
LCD_1_DrawVBG(1,9,2,16);
```

3.3.3.2 Conversores A/D

De los Conversores analógico a digital con los que cuenta el PSoC Designer, he escogido el A/D incremental con resolución variable de 7 a 13 bits (ADCINCVR) ya que cuenta con un mayor nivel de configuración y unas características más completas. Ofrece una salida en complemento a 2 posibilitando una entrada entre –Vcc y +Vcc centrada en GND. El conversor ADCINCVR ocupara 3 bloques digitales y uno analógico (de tipo SC, revisar el punto 3.2.2.2.1), además de un bloque analógico adicional para un amplificador de ganancia programable, como se muestra en la figura x.x que corresponde con la disposición de los bloques en el interfaz gráfico del PSoC Designer.

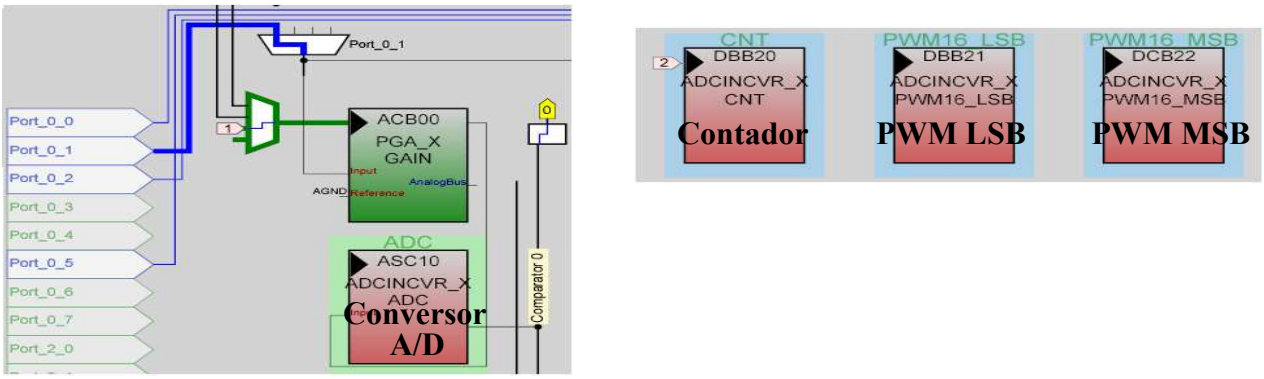


Fig. 3.59 Bloques digitales y analógicos que componen el ADCINCVR en el interfaz gráfica del PSoC Designer

El bloque digital se compone principalmente de un operacional configurado como integrador que cuenta con la posibilidad de reset. Como se puede ver en el siguiente diagrama de bloques, el conversor A/D está formado por dicho integrador, un contador de 8 bits y un PWM de 16 bits. Dependiendo de las variables de configuración que comentaré en el siguiente apartado, el A/D podrá alcanzar un rango de muestreo entre 4 y 10Ksps.

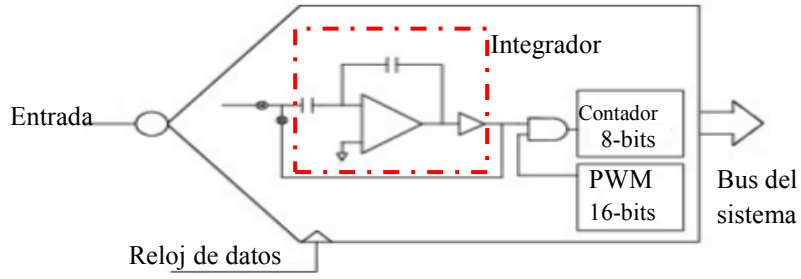


Fig. 3.60 Diagrama de bloques del ADCINCVR

3.3.3.2.1 Descripción / Configuración

El bloque ADCINCVR permite la configuración de los parámetros más relevantes del conversor como la señal de reloj que recibirá, la resolución del mismo o el tiempo que necesita para obtener el resultado (Calc Time). Una descripción de cada una de ellas se muestra a continuación:

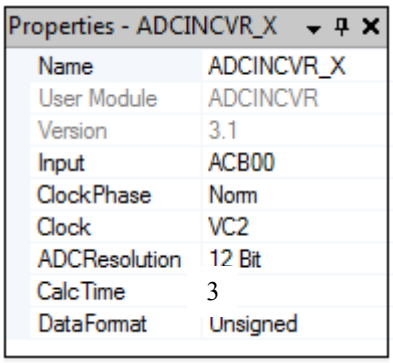


Fig. 3.61 Propiedades del ADCINCVR

- Nombre: Permite introducir el nombre con el que el bloque será nombrado posteriormente en la confección del código.

- Input (Entrada): Se puede seleccionar la señal de entrada al A/D. Debido a la imposibilidad de introducir la señal directamente desde el pin que recibe la señal del sensor(en este caso el sensor de aceleración) me vi obligado a hacer uso de un amplificador de ganancia programable (con ganancia 1 para no adulterar la señal). Como se puede observar en la fig ss., la señal de entrada al A/D es la salida del bloque de tiempo continuo ACB00, o lo que es lo mismo, el PGA

- Clock Phase(Fase del reloj): La elección de la fase del reloj se usa para sincronizar la salida de un condensador conmutado de un bloque analógico del PSoC con la entrada de otro. En este caso utilizaremos el modo normal.

- Clock (Reloj): El usuario cuenta con la posibilidad de escoger la señal de reloj, que será determinante para determinar la velocidad del conversor (reloj de datos) ya que forma parte del cálculo de la variable Calc Time, y este a su vez en el cálculo de la velocidad de muestreo (Sample Rate). En este caso, el reloj escogido es VC2=VC1/10 el cual tiene una frecuencia de 150KHz, siendo VC1=24MHz/16.

- Resolución del conversor A/D: Es posible elegir la resolución del A/D. También forma parte del cálculo de Calc Time. En este caso, la resolución elegida son 12 bits. Si bien la resolución se puede determinar directamente desde la ventana de propiedades, es posible utilizar una API para realizarlo, aunque de esa manera el ADC se para y debe ser reiniciado.

- CalcTime(Tiempo de cálculo): Es la cantidad de tiempo que necesita la CPU para calcular una integración antes de realizar la siguiente. Para el cálculo del tiempo de conversión del A/D contamos con la siguiente fórmula, en la que intervienen las variables DataClock:

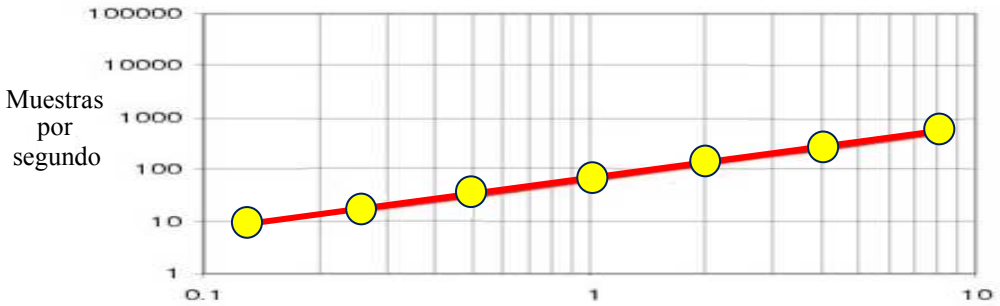
$$CalcTime = \frac{DataClock * 180}{CPU\ Clock} = \frac{150KHz * 180}{12MHz} = 2.25$$

El resultado lo debemos aproximar al siguiente número entero, por lo tanto, CalcTime será 3. A partir de este valor, podremos calcular las muestras por segundo con la siguiente operación:

$$SampleRate = \frac{DataClock}{2^{bits+2} + CalcTime} = \frac{150KHz}{2^{12+2} + 3} = 9.15\ Muestras/segundo$$

- Formato de datos: También se puede escoger el formato en el que el conversor sacará los datos. En esta opción, he escogido datos sin signo.

De esta manera, el conversor A/D quedará configurado con una resolución de 12 bits y con una velocidad de más o menos 9 muestras por segundo, con datos sin signo. Ahora describiré las API´s más importantes o reseñables de este bloque. En el siguiente gráfico se muestra la relación de muestras por segundo con la velocidad del reloj de un conversor AD de 12 bits con una velocidad de la CPU de 12MHz.



3.3.3.2.2 Instrucciones C por defecto (API´s)

Las instrucciones para utilizar el ADCINCVR están recogidas en el datasheet adjunto en el PSoC Designer. Las más destacables son las siguientes:

- *ADCINCVR_Start*: Realiza toda la inicialización necesaria de este módulo de usuario y permite al usuario establecer el nivel de energía. Un byte establece el nivel de potencia entre los siguientes: OFF – LOWPOWER – MEDPOWER – HIGHPOWER. Tras la configuración y el reset, el bloque analógico asignado al bloque A/D se desconecta, por lo que es necesaria su activación previa a su uso. Ejemplo: `ADCINCVR_X_Start(ADCINCVR_X_HIGHPOWER);`

- *ADCINCVR_SetPower:* Permite al usuario cambiar el nivel de energía del conversor A/D mientras opera con el. Similar al API anterior. Un ejemplo de uso seria el siguiente: `ADCINCVR_X_SetPower(ADCINCVR_X_HIGHPOWER);`

- *ADCINCVR_SetResolution:* Con esta API se puede establecer la resolución del conversor si no se ha realizado en la pantalla de explotación del PSoC Designer. En el caso de que el A/D estuviera realizando una conversión, este se detendría y realizaría la el cambio de la nueva resolución. Ejemplo: `ADCINCVR_X_SetResolution(ADCINCVR_X_12);`

- *ADCINCVR_Stop:* Como su nombre indica, detiene el bloque SC del conversor A/D y deshabilita los bloques digitales asociados a este. Resulta recomendable utilizar esta instrucción si el conversor queda en desuso. Ejemplo: `ADCINCVR_X_Stop();`

- *ADCINCVR_GetSamples:* Inicializa y arranca el conversor A/D recogiendo el número de muestras que especifica el usuario. Es necesario activar las interrupciones globales usando el macro `M8C_EnableGInt` (). Si a la hora de determinar el número de muestras que debe recoger el conversor, el usuario usa el valor '0', el conversor recogerá muestras indefinidamente. Ejemplo: `ADCINCVR_X_GetSamples(0);`

- *ADCINCVR_StopAD:* Esta instrucción detendrá el conversor A/D instantáneamente. Modelo: `ADCINCVR_X_StopAD();`

- *char ADCINCVR_fIsData, char ADCINCVR_fIsDataAvailable*: Cualquiera de estas dos instrucciones devolverá un valor distinto a cero cuando el dato convertido por el A/D esté disponible para su lectura. Modelo: `ADCINCVR_X_fIsDataAvailable() != 0)`

- *int ADCINCVR_iGetData():* Esta API devolverá el último dato convertido por el conversor. El usuario debe asegurarse que la conversión ha finalizado y que el dato obtenido ha sido valido mediante la API anterior (ADCINCVR_fIsDataAvailable) y además debe realizarse antes de la siguiente conversión para evitar que el dato quede sobreescrito. Modelo: `ADCINCVR_X_iGetData();`

- *ADCINCVR_ClearFlag():* Borra el flag de dato disponible. Deberá ser llamada tras haber completado una lectura de dato. Modelo: `ADCINCVR_X_ClearFlag();`

- *int ADCINCVR_iGetDataClearFlag():*Esta instrucción resulta de la combinación de las dos anteriores. Por lo tanto, devolverá el último dato convertido y borrará el flag de dato disponible simultáneamente. Modelo: `ADCINCVR_X_iGetDataClearFlag();`

Una vez comentadas todas las API´s involucradas con el funcionamiento del conversor A/D, se muestra un ejemplo del funcionamiento de algunas de ellas(extracto del código del proyecto):

```
#include <m8c.h>              // Constantes y macros núcleo M8C PSoC
#include "PSoCAPI.h"          // PSoC API definiciones de los user
modules

int valorX

void main(void)
{
      ADCINCVR_X_Start(ADCINCVR_X_HIGHPOWER); //Inicializas el ADC

      ADCINCVR_X_SetResolution(12); //Establece resolución 12Bits

      ADCINCVR_X_GetSamples(0);     //ADC en modo continuo

      M8C_EnableGInt;   // Interrupciones globales activadas


while(1){
            if (ADCINCVR_X_fIsDataAvailable() != 0)   //fin de
      conversion

            {
            valorX = ADCINCVR_X_iGetData();     //obtencion de la
conversión

            ADCINCVR_X_ClearFlag();             //limpiamos tras leer
            }}
```

3.3.3.3 Amplificadores de ganancia programable (PGA)

El PGA consiste en un amplificador no inversor cuya ganancia puede ser modificada por el usuario y como características principales cuenta con una alta impedancia de entrada y un ancho de banda dilatado. Como se puede ver en el diagrama de bloques de la figura x.x el usuario puede escoger entre varias referencias (Vss, GND analógico, un bloque CT o un bloque de SC). Dispone de un número limitado de ganancias programables(33 en total) que oscilan entre 0.062(mín.) y 48(máx.). La ecuación por la cual se rige este amplificador variará en función del valor de la ganancia. En el caso de ser una ganancia mayor o igual a 1 será la siguiente:

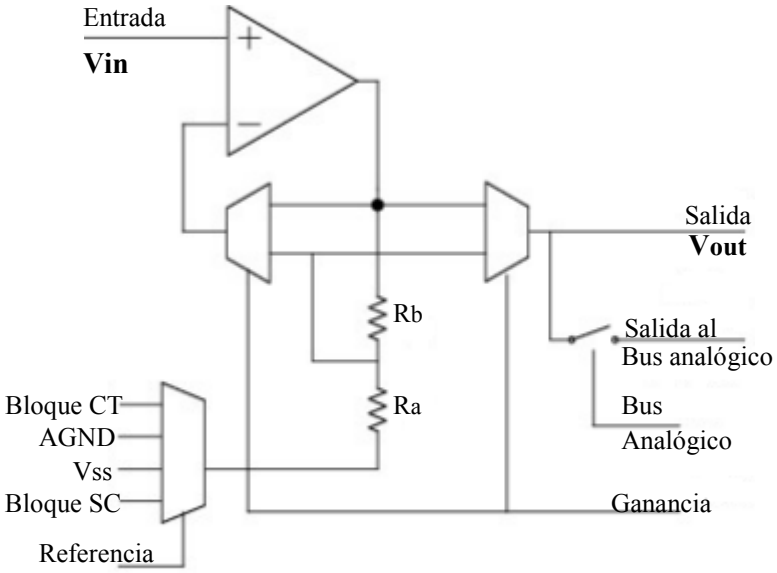


Fig. 3.62. Diagrama de bloques del Amplificador de ganancia programable

$$Vo = (V_{in} - V_{GND}) \cdot \left(1 + \frac{R_b}{R_a}\right) + V_{GND}$$

Sin embargo, para ganancias menores a 1 la ecuación será esta:

$$Vo = (V_{in} - V_{GND}) \cdot \left(\frac{R_a}{R_a + R_b}\right) + V_{GND}$$

Dependiendo del valor de ganancia elegido por el usuario, el editor del PSoC asignará los valores de las resistencias Ra y Rb automáticamente para adecuarse a dicho valor de ganancia. En la figura ss. podemos ver el bloque PGA en el editor de dispositivo en PSoC Designer donde ocupa un bloque analógico y como se puede apreciar, el usuario puede configurar todas las entradas y salidas a este bloque, además de la referencia del mismo y la señal de reloj. La razón de la inserción de este amplificador en mi proyecto es servir como nexo de unión entre el GPIO y el A/D debido a la imposibilidad de conectar directamente la entrada del bloque A/D a los pines del encapsulado destinados a E/S analógicas.

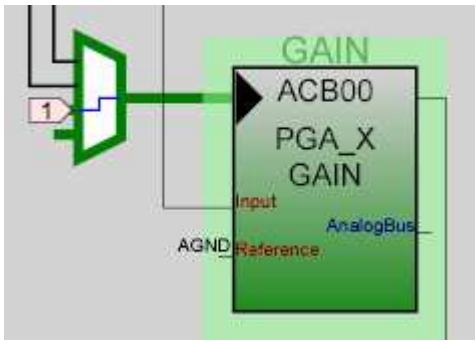


Fig. 3.63. Bloque analógico del PGA en PSoC Designer

3.3.3.3.1 Descripción/Configuración

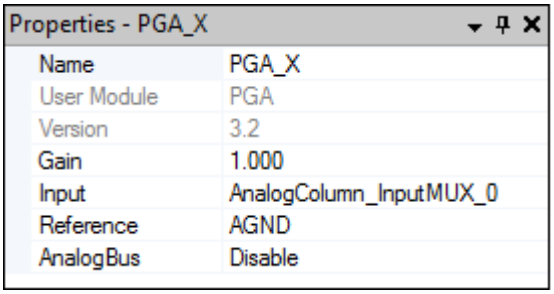


Fig. 3.64 propiedades de un PGA en PSoC Designer

La ventana de propiedades del PGA permite modificar seleccionar las siguientes características:

- Nombre: Permite renombrar el bloque PGA. Será el nombre con el que referirse posteriormente al utilizar las API´s del bloque.

- Ganancia (Gain): El usuario tiene la posibilidad de elegir el valor fundamental del bloque PGA. Al hacer click sobre el valor de la ganancia, se desplegará un menú donde aparecerán todos los valores de ganancia posibles y estos son: 0.062, 0.125, 0.188, 0.25, 0.312, 0.375, 0.438, 0.5, 0.563, 0.625, 0.688, 0.75, 0.813, 0.875, 0.937, 1, 1.062, 1.143, 1.231, 1.333, 1.455, 1.6, 1.777, 2, 2.286, 2.667, 3.2, 4, 5.333, 8, 16, 24 y 48.

- Entrada (Input): Se puede escoger el bloque PSoC adyacente o la señal externa que entrará en el amplificador.

- Referencia: Como hemos podido ver en el diagrama de bloques del amplificador el usuario podía elegir la referencia a masa del mismo. Mediante este menú desplegable se podrá elegir la referencia entre las siguientes: Vss, GND analógico, un bloque CT o un bloque de SC.

- Bus analógico: Permite habilitar/deshabilitar la salida al bus analógico

Una vez elegidas todas las opciones, el PGA quedará totalmente configurado según las exigencias del usuario.

3.3.3.3.2 Instrucciones C por defecto (API´s)

Las API´s del amplificador de ganancia programable junto con un modelo de utilización en código C son las siguientes:

- *PGA_Start:* Realiza la inicialización de todo lo necesario para iniciar el amplificador y le asignara un nivel de energía entre los siguientes (PGA_OFF, PGA_LOWPOWER, PGA_MEDPOWER y PGA_HIGHPOWER). Una vez establecido el nivel de energía, obtendremos valores en la salida. Modelo: `PGA_X_Start(PGA_X_HIGHPOWER);`

- *PGA_SetPower:* Permite asignar un nivel de energía a los bloques CT del PSoC. Además podría utilizarse para apagar y encender el PGA. Modelo: `PGA_X_SetPower(PGA_X_OFF);`

- *PGA_SetGain:* El usuario podrá establecer la ganancia del amplificador a través de esta API. Para escribir la ganancia se deberá conocer el valor de amplificación y escribirlo de la siguiente manera: **PGA_G[numero entero]_[numero decimal]** con un máximo de 3 números. Una vez utilizada y, hasta que el amplificador adopte el valor seleccionado, el PGA tomará ganancia 1. Modelo: `PGA_X_SetGain(PGA_X_G1_78);`

- *PGA_Stop:* Como su nombre indica, esta instrucción detendrá el PGA. Modelo: `PGA_X_Stop();`

Un pequeño ejemplo de utilización de las instrucciones comentadas en código C se muestra a continuación:

```
#include <m8c.h>                    // Constantes y macros núcleo M8C
PSoC
#include "PSoCAPI.h"          // PSoC API definiciones de los user
modules

void main()
{
    PGA_X_SetGain(PGA_X_G1_00);    //Establece la ganancia del PGA
    PGA_X_Start(PGA_X_HIGHPOWER);  //Enciende Amplificador
}
```

3.3.3.4 PWM´s

El bloque PWM , siglas inglesas de Pulse Wave Modulator, corresponde con un modulador de anchura de pulso con un periodo y ancho de banda programables en el que el usuario podrá elegir entre uno de 8 bits, o bien, uno de 16 bits según convenga a su proyecto. En la siguiente figura podemos ver el diagrama de bloques correspondiente a ambos moduladores, ya que la única diferencia entre ellos son el número de bits del registro de periodo, del registro de ancho de pulso y del contador.

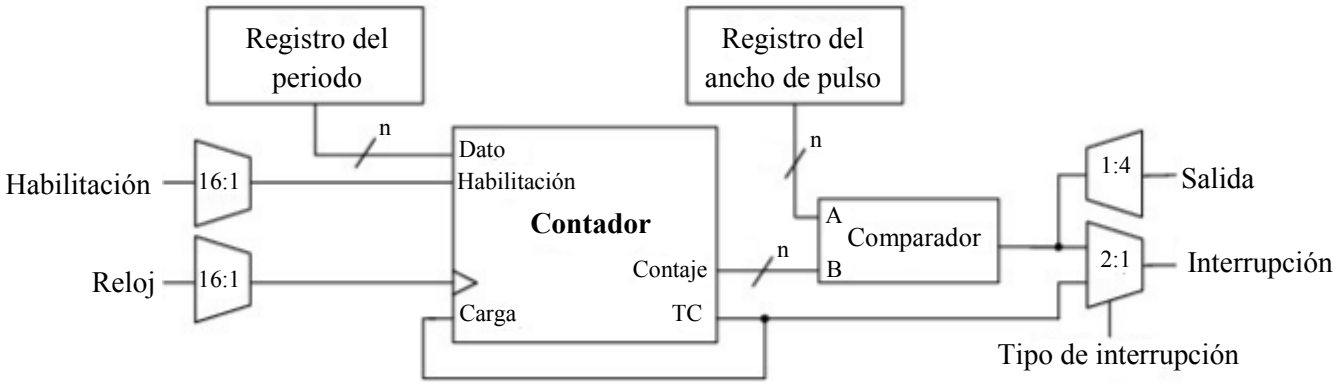


Fig. 3.65 Diagrama de bloques de un PWM

En el diagrama podemos ver como el contador recibe en su entrada "dato" las señales del registro de periodo, de 8 o 16 bits, al que el usuario introducirá el valor del periodo deseado de contaje. Además, las señales de reloj y habilitación pueden provenir de diversas fuentes también configurables. Como salida, el contador proporcionará el contaje y una señal de final de contaje (TC). Se puede apreciar que la salida TC se dirige tanto al mux. de interrupción así como a la entrada "carga" del contador. El contaje será comparado en cada ciclo del reloj con el valor contenido en el registro de ancho de pulso, que será el elegido también por el usuario. El resultado de comparación, será dirigido tanto a la salida como al mux. de interrupción. Como se puede ver en la imagen de la izquierda, el bloque PWM ocupará uno o dos bloques digitales, dependiendo del número de bits. En el caso del de 16 bits, se usarán dos bloques digitales con las conexiones internas sincronizadas para conseguir que funcione como uno solo. Las salida que ofrece el bloque PWM en función de la señal de reloj y del valor del periodo en su registro correspondiente es la siguiente:

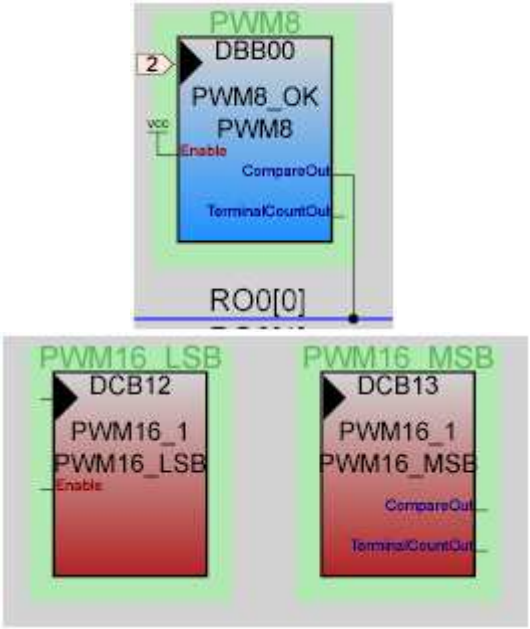


Fig. 3.66 Bloques digitales de los PWM de 8 y 16 bits en PSoC Designer

$$T_{OUT} = (ValorPeriodo + 1)/F_{CLOCK}$$

$$F_{OUT} = F_{CLOCK}/(ValorPeriodo + 1)$$

La relación entre el valor del ancho de pulso y el valor del periodo establecen el ciclo de trabajo de la onda de salida del PWM. Usando las siguientes ecuaciones, dependiendo de si el valor del ancho de pulso es menor, igual o mayor (en el caso de que el valor del ancho de pulso sea mayor que el valor del periodo, el ciclo de trabajo será del 100%) que el valor del periodo, se puede obtener dicho ciclo de trabajo:

$$DutyCycle_{menor\,que} = \frac{ValorAncho\,de\,Pulso}{ValorPeriodo + 1} \qquad DutyCycle_{menor\,o\,igual\,que} = \frac{ValorAncho\,de\,Pulso + 1}{ValorPeriodo + 1}$$

3.3.3.4.1 Descripción/Configuración

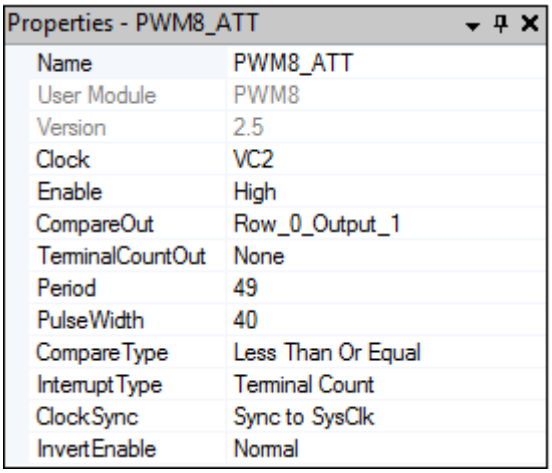


Fig. 3.67 Ventana de propiedades del bloque PWM de 8 bits

Para definir las propiedades características del bloque PWM, tomaré como ejemplo el cuadro de propiedades de un PWM de 8 bits, donde se podrá escoger entre las siguientes:

- Nombre: El usuario determinará el nombre del bloque que luego usará para su definición en la confección de código.
- Reloj (Clock): Se puede escoger entre una gran variedad de fuentes de reloj (hasta un total de 16) para el PWM. En este caso está seleccionado el reloj VC2.
- Habilitación (Enable): Al igual que la fuente de reloj, se pueden escoger hasta 16 fuentes como habilitación del PWM. Como es lógico, un nivel alto en esta entrada resultará en un contaje continúo mientras que un nivel bajo detendrá el contaje sin resetearlo.
- Salida de comparación (Compare Output): Esta señal puede ser deshabilitada o conectada a los bloques digitales colindantes además de poder conectarse a los buses filas de salida.
- Salida del Contaje Terminal (Terminal Count Output): Como se ha explicado anteriormente, corresponde con la configuración de la señal auxilar que proporciona el contador, TC. De igual manera que la anterior, puede deshabilitarse o conectarse al bus de filas de salida.
- Periodo: El usuario puede elegir el periodo del bloque PWM con esta opción. En el caso del PWM8, el valor oscilará entre 0 y 255, mientras que en el PWM16 se podrán escoger valores entre 0 y 65535. El valor será guardado en el registro de periodo.
- Ancho de pulso(Pulse Widht): Al igual que con el periodo, el usuario establecerá el valor de esta característica, que variará entre 0 y el valor del periodo. También se podrá modificar a través de la API específica al efecto.
- Tipo de comparación (Compare Type): Se puede establecer si la comparación será "menor que" o "menor o igual que".
- Tipo de interrupción (Interrupt Type): Esta característica determinará la señal que desencadenará las interrupciones del bloque PWM. Se puede escoger entre el flanco de subida de la señal de salida, o bien, la señal TC del contador.
- Sincronización del reloj(ClockSync): Este parámetro se usa para controlar una desviación de la señal de reloj debido a las adiciones de contajes que se podrían ocasionar por parte de los bloques digitales. El usuario deberá escoger entre tres opciones:

    1. Sincronizar con el reloj del sistema(Sync to SysClk): Se usará para cualquier reloj de entrada derivado del reloj de 24 MHz(VC1, VC2, VC3, 32KHz y bloques PSoC con señales de reloj basadas en SySClk) asi como señales externas.

2. Sincronizar con el reloj del sistema por 2(Sync to SysClk*2): Esta opción deberá usarse en caso de utilizar cualquier señal de reloj de entrada basada en la de 48MHz y menor a esta.

3. Sincronizar directamente al reloj del sistema (Use SysClk Direct): Si el usuario va a utilizar como señal de reloj SysClk deberá utilizar esta opción aunque no sirva para sincronizar, ya que utilizar esta fuente de reloj casi no se producen desviaciones.

4. No sincronizado (Unsynchronized): Deberá usarse cuando se utiliza Sync to SysClk*2.

- Habilitación de inversión (InvertEnable): Determina si la habilitación se activara con un flanco positivo ("Normal") o con un flanco negativo("Inverso") .

3.3.3.4.2 Instrucciones C por defecto (API´s)

Debido a la existencia de los bloques digitales PWM de 8 y de 16 bits, existen instrucciones diferentes para cada uno de ellos. Aunque la diferencia entre instrucciones en C reside únicamente en el número 8 o 16, usaré como ejemplo las API´s para el PWM de 8 bits.

- *PWM8_PERIOD y PWM8_PULSE_WIDTH*: Estas dos API´S representan los valores constantes del periodo y del ancho de pulso respectivamente. El valor de ambos puede determinarse en el editor del PSoC Designer y oscilará entre 0 y 255.

- *PWM8_EnableInt*: Habilitará las operaciones de interrupción del bloque PWM. Modelo: `PWM8_OK_EnableInt();`

- *PWM8_DisableInt*: Deshabilitará las operaciones de interrupción del bloque PWM. Modelo: `PWM8_OK_DisableInt();`

- *PWM8_Start*: Esta instrucción desencadenará el inicio del bloque PWM. Si la habilitación del bloque es un nivel alto, el contador iniciara su contaje descendente. Modelo: `PWM8_OK_Start();`

- *PWM8_Stop*: Como su nombre indica, detendrá el bloque PWM ya que parará el contador del mismo. Modelo: `PWM8_OK_Stop();`

- *PWM8_WritePeriod:* Con esta API introduciremos el valor del periodo en el registro asociado y si el PWM es detenido o el contador de este llega a cero, este valor será inmediatamente transferido al registro del contador. Modelo: `PWM8_OK_WritePeriod(50);`

- *PWM8_WritePulseWidth:* Similar a la anterior, esta instrucción permitirá al usuario introducir el valor del ancho de pulso en su registro asociado. Modelo: `PWM8_OK_WritePulseWidth(24);`

- *PWM8_bReadPulseWidth* y *PWM8_bReadCounter:* Estas dos API´s permiten realizar la lectura del valor del ancho de pulso y del contador respectivamente.

Debido a la simplicidad de utilización de este bloque de usuario, un ejemplo de utilización de las instrucciones citadas, omitiré la realización de un ejemplo de las mismas.

3.3.3.5 Contadores

PSoC Designer pone a disposición del programador bloques de usuario contadores de 8, 16, 24 o 32 bits. Este módulo de usuario incorpora un contador decreciente con un periodo y ancho de pulso programables y tanto la fuente de reloj como la señal de habilitación pueden provenir de fuentes externas al PSoC. El diagrama de bloques de uno de estos contadores es el siguiente:

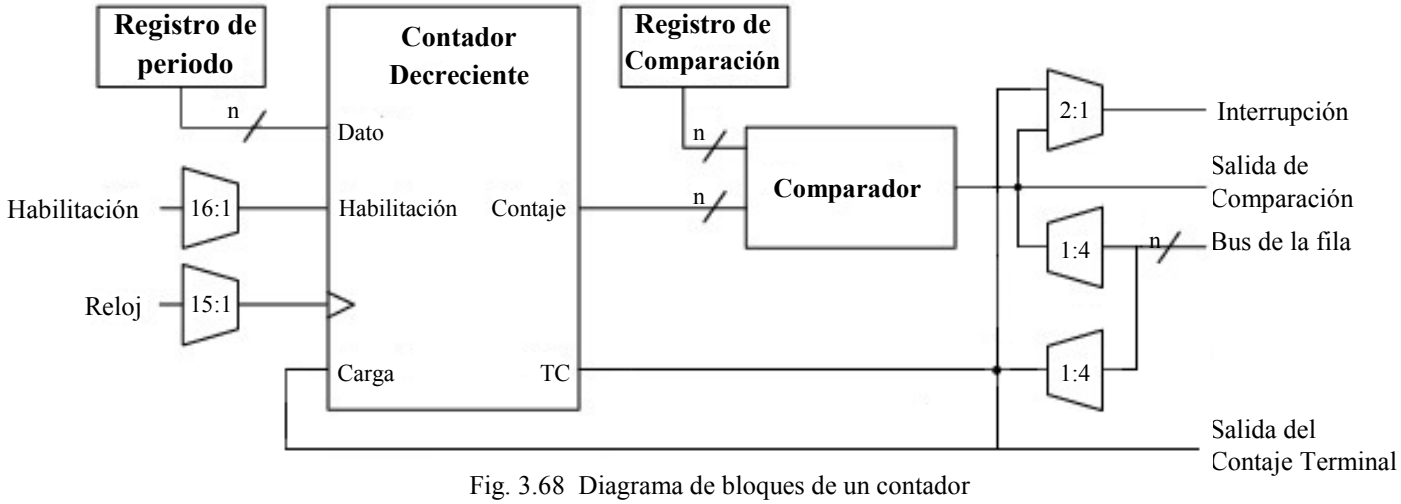


Fig. 3.68 Diagrama de bloques de un contador

El funcionamiento del contador es el siguiente: Una vez inicializado operará de forma continua recargando el valor del registro de periodo hasta que alcance el contaje terminal. En cada ciclo de reloj, el contador comparará el valor actual del contaje con el valor alojado en el registro de comparación. Como se puede observar en el diagrama, el contador ofrecerá diversas salidas y cada una con una función diferente. Las salidas son las siguientes: de interrupción, de comparación, de contaje terminal y otra hacia el bus de la fila en la que se encuentre el bloque.

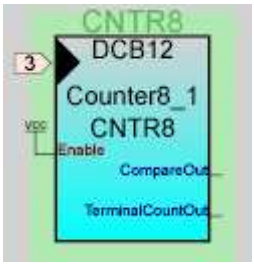


Fig. 3.69. Bloque digital del contador de 8 bits en PSoC Designer

El periodo del registro podrá ser modificado en cualquier momento siempre que el contador haya sido detenido, siendo actualizado también el contaje. Si el contador está en funcionamiento y se modifica el valor del periodo, el contador no será actualizado hasta el comienzo de un nuevo ciclo de contaje, tras la señal de contaje terminal. La duración en términos del periodo del reloj de entrada viene definido por la siguiente ecuación:

$$PeriodoSalida = (Valor\ de\ periodo + 1)t_{CLK}$$

Ya que el contaje terminal se alcanzará cuando el contaje llegue a cero, el periodo de operación y de la señal de salida serán una unidad mayor que el valor almacenado en el registro de periodo. Durante cada ciclo de reloj, el comparador comprobará los valores registro del contador respecto a los valores del registro de comparación y dependiendo de la configuración del usuario, la comparación realizada será "menor que" o "menor o igual que". La relación entre el valor de comparación y el periodo establece el ciclo de trabajo de la forma de onda de salida. La relación de ciclo de trabajo se puede calcular mediante la siguiente ecuación:

$$CiclodeTrabajo = \begin{cases} \dfrac{ValordeComparación}{ValordePeriodo+1} & \text{para la comparación "menor que"} \\ \dfrac{ValordeComparación+1}{ValordePeriodo+1} & \text{para la comparación "menor o igual que"} \end{cases}$$

3.3.3.5.1 Descripción/Configuración

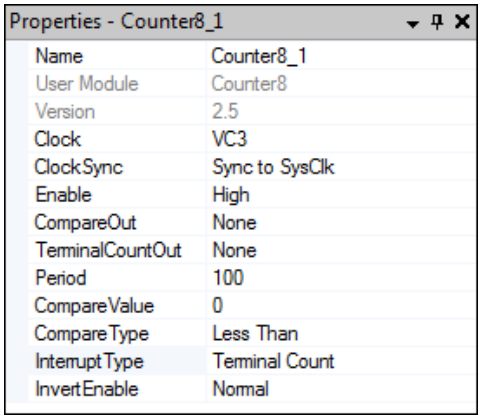


Fig. 3.70 Ventana de propiedades del bloque contador de 8 bits

Para definir las propiedades características del bloque contador, tomaré como ejemplo el cuadro de propiedades de un contador de 8 bits ya que no variará con respecto a los de 16, 24 y 32 salvo en el valor del periodo. Dichas propiedades son las mismas que fueron explicadas en el punto 3.3.3.4.1 correspondiente al bloque PWM, por lo tanto no volveré a repetir cada una de ellas. Cabe destacar que las propiedades serán idénticas independientemente del número de bits del contador con la diferencia de que los valores del periodo y del valor de comparación serán mayores cuanto mayor sea el numero de bits del contador

3.3.3.5.2 Instrucciones C por defecto (API´s)

Como sucede con el bloque PWM, también encontraremos API´s para cada uno de los contadores de 8, 16, 24 y hasta 32. Para enumerar las instrucciones más importantes utilizaré las de un contador de 8 bits:

- *COUNTER8_PERIOD y COUNTER8_COMPARE_VALUE*: Estas dos API´S representan los valores constantes del periodo y del ancho de pulso elegidos por el usuario, respectivamente. El valor de ambos puede determinarse en el editor del PSoC Designer y oscilará entre 0 y 255.

- *COUNTER8_EnableInt*: Habilitará las operaciones de interrupción del contador. Modelo: `COUNTER8_1_EnableInt();`

- *COUNTER8_DisableInt*: Deshabilitará las operaciones de interrupción del contador. Modelo: `COUNTER8_1_DisableInt();`

- *COUNTER8_Start*: Esta instrucción desencadenará el inicio del contador. Si la habilitación del bloque es un nivel alto, el contador iniciara su contaje descendente. Modelo: `COUNTER8_1_Start();`

- *COUNTER8_Stop*: Como su nombre indica, detendrá el contador. Modelo: `COUNTER8_1_Stop();`

- *COUNTER8_WritePeriod:* Con esta API el valor del periodo se transferirá directamente del registro de periodo al registro del contador cuando el contador llegue a cero o se detenga. Modelo: `COUNTER8_1_WritePeriod(50);`

- *COUNTER8_WriteCompareValue:* Similar a la anterior, esta instrucción permitirá al usuario introducir el valor del ancho de pulso en su registro asociado. Modelo: `COUNTER8_1_WritePulseWidth(24);`

- *COUNTER8_bReadCompareValue* y *COUNTER8_bReadCounter:* Estas dos API´s permiten realizar la lectura del valor del ancho de pulso y del contador respectivamente.

3.3.3.6 UART

El bloque de usuario UART (siglas en inglés de, Universal Asynchronous Receiver Transmitter) corresponde a un receptor-transmisor asíncrono universal de 8 bits que es compatible con el puerto serie RS-232 para comunicación sobre 2 hilos. Todos los datos tanto recibidos como enviados incluirán un bit de inicio, uno opcional de paridad y otro de detención. Al igual que con los demás bloques de usuario, se procederá a la explicación de su estructura interna, propiedades configurables por el usuario en el interfaz de usuario de PSoC Designer y las API´s más destacables.

El diagrama de bloques del bloque UART es el siguiente:

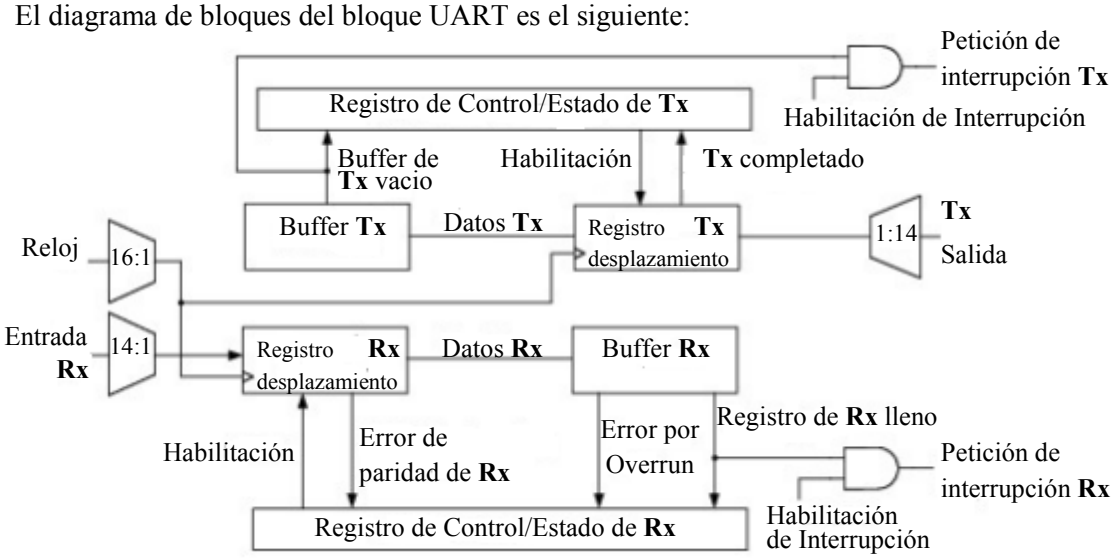


Fig. 3.71. Diagrama de bloques del UART

El bloque UART, formado por un receptor y un transmisor serie, ocupara dos bloques digitales PSoC llamados Rx y Tx respectivamente. Como se puede ver en el diagrama de bloques, cada uno de ellos operará de forma independiente ya que cada uno cuenta con su propio registro de control/estado, interrupciones programables, registros de desplazamiento y buffers etc. Lo único que comparten entre ellos será la habilitación, el reloj y el formato de los datos.

Cada dato recibido o transmitido consistirá en un flujo de datos que consistirá en un bit de inicio, los 8 bits de datos, un bit opcional de paridad y un bit de detención. La opción de paridad podrá ser escogida por el usuario y será explicada posteriormente en las propiedades del bloque. Antes de proceder a la explicación de las propiedades de este bloque, explicare por separado el funcionamiento de los bloques Rx, o receptor, y Tx, o transmisor, del UART:

- Rx: Receptor del UART

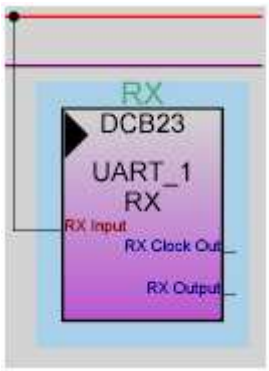


Fig. 3.72 Bloque Receptor en el editor de dispositivos en PSoC

El receptor usará los registros de control RX, de desplazamiento RX y de buffer RX de un bloque de comunicación digital del PSoC. Este bloque se inicializará y configurará utilizando las API´s que serán descritas posteriormente. Cuando se detecta un bit de comienzo en la entrada RX , un reloj de division por ocho bits comenzará y será sincronizado para muestrear los datos en el centro de los bits recibidos. En el flanco de subida del siguiente reloj de 8 bits, los datos de entrada son muestreados y movidos al registro de desplazamiento RX.

El muestreo del bit de detención resultará, al comienzo del siguiente ciclo de reloj, en la transferencia del byte de datos recibido al registro del buffer RX activándose entonces uno de los bits del registro de control RX , que controlará si la operación se ha realizado correctamente o ha sucedido algún problema como por ejemplo un error de paridad, de lectura del dato recibido o de fallo de lectura del bit de detención.

➢ Tx: Transmisor del UART

Fig. 3.73. Bloque Transmisor en el editor de dispositivos en PSoC

El transmisor usará los registros de control RX, de desplazamiento RX y de buffer RX de un bloque de comunicación digital del PSoC. De igual manera que con el bloque receptor, el transmisor se inicializará y configurará utilizando API´s. Cuando se activa el bit de habilitación en el registro de control TX, un reloj de división por ocho bits será generado. El flanco de subida del siguiente bit del reloj transferirá los datos del registro de desplazamiento y activara el bit de Buffer TX Vacio en el registro de control TX.

El bit de comienzo será transmitido al mismo tiempo que el byte de datos será transferido del registro de buffer TX al registro de desplazamiento TX. Los sucesivos bits del reloj desplazarán una cadena de bits en serie hacia la salida de menos a más significativos, un bit opcional de paridad y un bit final de detención. Hasta que se complete la transmission, un bit del registro de control TX permanecerá activo.

3.3.3.6.1 Descripción/Configuración

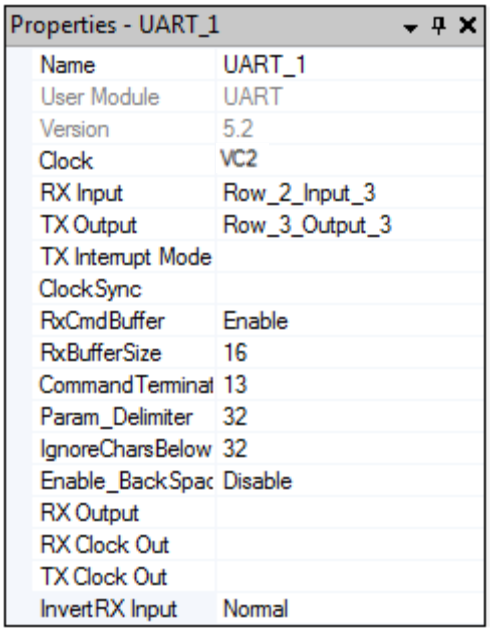


Fig. 3.74 Ventana de propiedades del bloque UART

Las propiedades configurables del bloque UART a través del editor de dispositivos de PSoC Designer son las siguientes:

- Nombre: Permite modificar el nombre que el usuario utilizara posteriormente para referirse al bloque UART.
- Reloj (Clock): Se puede escoger entre una gran variedad de fuentes de reloj (hasta un total de 16). La frecuencia del reloj deberá ser 8 veces la tasa de bits deseada ya que cada bit de datos será recibido o enviado cada 8 ciclos de reloj.
- Entrada de RX (RX Input): Con esta opción, el usuario elegirá la señal que entrará al UART. Esta señal puede provenir de diversos orígenes como otro bloque PSoC, la salida del bus comparador analógico o usando un bus global de entrada (esté ultimo posibilitará el uso de una señal externa proveniente de uno de los pines del integrado).
- Salida de TX (TX Output): La salida del transmisor puede ser conectada al bus de salida global y de esta manera conducir la salida del UART a cualquier otra región del PSoC, así como al exterior mediante un pin del integrado.
- Modo de interrupción de TX: Se podrá determinar el momento en el que el bloque Tx realizará sus interrupciones. Existen 2 opciones: "TxRegEmpty" que desencadenará la interrupción tan pronto como los datos se hayan transferido del registro de datos al de desplazamiento y "TxComplete" la cual retrasará la interrupción hasta que el dato haya sido totalmente transferido.
- Sincronización del reloj (ClockSync): Esta característica configurable es idéntica a la explicada en el punto 3.3.3.4.1 correspondiente al bloque PWM, por lo que no repetiré la explicación nuevamente.
- Buffer de comando RX (RxCmdBuffer): Esta propiedad permite habilitar el buffer de comando Rx. La interrupción RX debe estar habilitada para que el buffer de comando funcione correctamente.
- Tamaño del buffer RX (RXBufferSize): Este parámetro determina cuantas ranuras de memoria RAM son reservadas para el buffer de Rx y, por lo tanto, el mayor comando recibido será como máximo una unidad menor que el tamaño escogido.

- Finalizador de Comando (Command Terminator): El CT determina el carácter que supondrá el final de un comando. Cuando este sea recibido, un flag se activará denegándose más caracteres hasta que se llame a la función cmdReset(), que será explicada en el siguiente punto.

- Delimitador de parámetros (Param_Delimiter): Similar al anterior, este parámetro determina el carácter que se usará para separar el comando de los parámetros en el buffer recibidor de comandos.

- Discriminador de caracteres siguientes (IgnoreCharsBelow): Con esta opción el usuario podrá discriminar caracteres por debajo de uno predefinido aunque sólo tendra efecto si el buffer y la interrupción de RX está activa.

- Habilitar Barra Espaciadora (Enable_BackSpace): De igual forma, esta propiedad sólo tendrá efecto en las condiciones del anterior y permite asignar a un carácter un valor de retroceso o borrado. El usuario podrá elegir entre estas tres opciones: "Disable" (Deshabilitar), " BackSpace" (Retroceso) y "Delete" (Borrado).

- Salida de RX (RX Output): Esta propiedad habilita la transferencia de los datos de entrada a uno de los buses de una a de las 4 filas.

- Control de paso de RX (RX Clock Out): Con esta function se permite que el bit de reloj del bloque RX sea transferido a una de las filas de buses. El flanco de subida de la señal de Control de paso de RX se corresponde con el momento en el que el dato es estable y debe ser muestreado.
- Control de paso de TX (TX Clock Out): Idéntica a la anterior, pero en este caso con respecto a TX.
- Invertir entrada RX (Invert Input RX): Como su nombre indica, permite invertir la señal de entrada RX.

3.3.3.6.2 Instrucciones C por defecto (API´s)

Las API´s del bloque UART pueden dividirse entre las de bajo y alto nivel. Las instrucciones son las siguientes:

1. Instrucciones de bajo nivel

API	Descripción
void UART_Start(BYTE bParity)	Habilita el módulo UART y determina la paridad
void UART_Stop(void)	Deshabilita el módulo de usuario
void UART_EnableInt(void)	Habilita las interrupciones de RX y TX
void UART_DisableInt(void)	Deshabilita las interrupciones de RX y TX
void UART_SetTxIntMode(BYTE bTxIntMode)	Permite determinar la fuente de la interrupción de TX
void UART_SendData(BYTE bTxData)	Envia el byte sin comprobar el estado de TX
BYTE UART_bReadTxStatus(void)	Devuelve el estado del registro de estado de TX
BYTE UART_bReadRxData(void)	Devuelve datos al registro de datos de RX sin comprobar la validez de caracter
BYTE UART_bReadRxStatus(void)	Comprueba el estado del registro de estado de RX

2. Instrucciones de alto nivel

Function	Description
void UART_IntCntl(BYTE bMask)	Habilita/Deshabilita las interrupciones de RX y TX
void UART_PutString(char * szStr)	Envia fin de cadena del puerto TX
void UART_CPutString(const char * azStr)	Envia fin de cadena constante del puerto TX
void UART_PutChar(char bData)	Envia un caracter al puerto TX cuando el registro TX este vacio
char UART_cGetChar(void)	Devuelve un caracter desde el registro de datos de RX cuando datos validos estan disponibles
char UART_cReadChar(void)	Lee el registro RX al instante. Si no hay dato disponible devolvera 0, en otro caso, un caracter ASCII entre 1 y 255
int UART_ iReadChar(void)	Lee el regitro de datos RX al instante. Si no hay dato disponible o existe un error devuelve un estado de error.
void UART_PutSHexByte(BYTE bValue)	Envia dos caracteres hexadecimales representativos de bValue al puerto TX
void UART_PutSHexInt(int iValue)	Envia cuatro caracteres hexadecimales representativos de iValue al puerto TX
void UART_CmdReset(void)	Realiza un reset al buffer de comando de RX
BYTE UART_bCmdCheck(void)	Devuelve un valor distinto a cero si un finalizador de comando valido ha sido recibido
BYTE UART_bCmdLength(void)	Devuelve la longitud de comando actual
BYTE UART_bErrCheck(void)	Devuelve el estado de error en el buffer de comando

4. SISTEMA DE CONTROL ESTÁNDAR PARA MÁQUINAS CORTACÉSPED

4.1 Introducción al sistema: Requerimientos

Como ya he comentado en el capítulo 2.3, para la realización de esta aplicación basada en sistemas embebidos o empotrados (PSoC) he intentado crear un sistema de control estándar para las máquinas cortacésped producidas en la empresa Grillo SpA que perfectamente podría ser extrapolable a cualquier máquina del mercado con unas características similares.

Este sistema de control, sobre el que trataremos en los siguientes apartados, tratará de procesar variables físicas mediante diferentes sensores para añadir una mayor seguridad y comodidad de los usuarios de dichas máquinas, asi como mejorar los actuadores que en ese momento utilizaban y que, o bien, estaban obsoletos, o que por diversas causas no cumplían su función de manera eficaz.

Para crear un sistema que pueda utilizarse en diversos modelos, opte por trabajar con cada sensor por separado, creando sistemas específicos para cada uno de ellos facilitando la tarea de trabajar con las máquinas cortacésped que requerían esas mejoras o modificaciones.

Fig 4.1 Algunos de los prototipos realizados sobre placa de puntos

Cuando cada uno de estos sistemas específicos creados sobre placas de puntos a modo de prototipo cumplían los objetivos a alcanzar, estos eran montados sobre las máquinas cortacésped y se realizaban los test pertinentes para comprobar el correcto funcionamiento tanto del hardware como del software. Este punto realmente era el que más problemas conllevaba ya que los valores teóricos siempre requieren de ciertas modificaciones al llevarlos al mundo real. Todos los contratiempos así como las modificaciones realizadas para solventarlos serán detallados en los siguientes apartados.

En el siguiente apartado mostraré las variables físicas y los sensores montados sobre las máquinas cortacésped sobre los que posteriormente creare los sistemas de control específicos así como los sistemas de visualización pertinente de los datos procesados. Cada uno de estos "subsistemas" contará con sus diagramas de bloques así como de una explicación de las placas de puntos realizadas y del software asociado a cada una de ellas.

El diagrama de bloques de sistema final será desarrollado posteriormente y de manera introductoria, un diagrama de flujo conceptual de las tareas realizadas por el sistema es el siguiente:

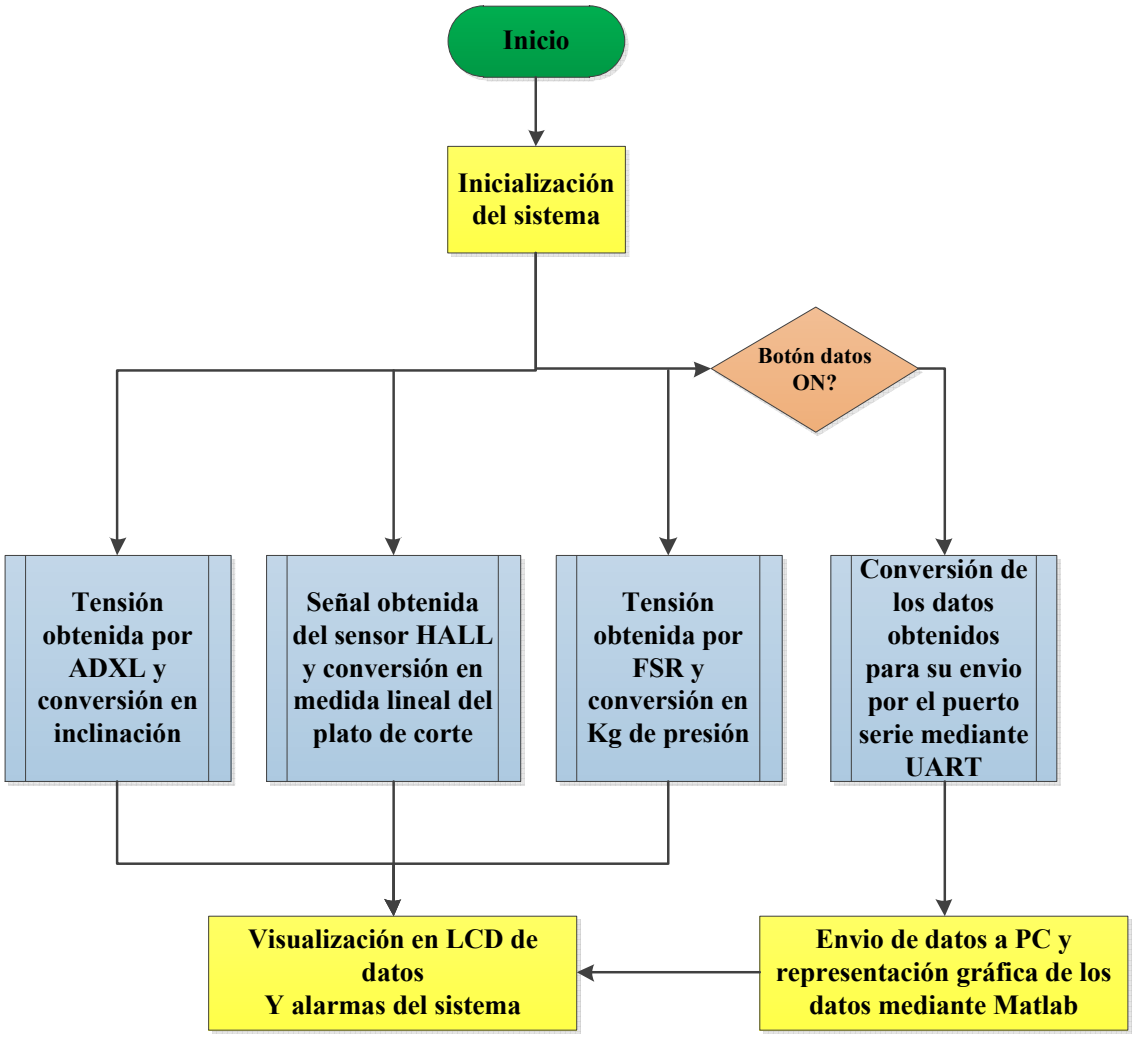


Fig. 4.2 Diagrama de flujo del sistema

Como se puede apreciar en el diagrama, tras una inicialización del sistema correspondiente con la declaración de todas las variables y bloques de usuario del PSoC (con su consiguiente inicialización), el sistema realizará de forma paralela las tareas de recepción y manipulación de las señales arrojadas por los sensores que recogerán diversos datos. Estos datos serán procesados y se podrá, visualizar mediante el LCD y un sistema de alarmas las variables de interés para el usuario de la máquina cortacésped. En el caso de que una condición se cumpla, el sistema enviará por el puerto serie los datos recogidos por los sensores para ser procesados en Matlab y obtener su representación gráfica.

En el siguiente capítulo se mostrará la descripción tanto del hardware como del software empleado para diseñar un sistema específico para cada sensor y posteriormente, un prototipo cuya operación es la correspondiente con el diagrama de flujo de la fig. 4.2

4.2 Descripción de variables/sensores a controlar

Las variables a controlar, a partir de las cuales, los subsistemas fueron creados variaron en función de las necesidades que cada máquina pudiera requerir. En un principio, sopesé controlar variables como la velocidad de la máquina cortacésped y de las cuchillas de corte o la creación de un emisor de ultrasonidos que ahuyentará los insectos de la misma pero una vez en la empresa y vistas las posibles necesidades reales de comodidad y seguridad de un futuro usuario descarté algunas de ellas y con la ayuda de los ingenieros de esta trabajé sobre algunos aspectos de las máquinas que requerían un cambio o mejora o que ni siquiera las tenían.

De esta manera comencé el estudio sobre la viabilidad del uso de los sensores que mejor se adaptarán a las máquinas o sustitución de los actuales por unos controlables mediante el PSoC, ya que en algunas ocasiones no era posible desarrollar la idea más simple por diversas situaciones (como por ejemplo, en el caso del depósito) que explicare en cada uno de los casos, y de qué manera se podrían incorporar a un sistema basado en el microcontrolador de Cypress.

Las variable que no contaba con ningún sistema de medición o de sensado era la inclinación de la máquina y, por otro lado, la altura del plato de corte y el sensor de llenado del depósito, contaban con sistemas analógicos que estaban obsoletos y que no funcionaban de manera eficiente respectivamente. En los siguientes sub apartados, procederé al desarrollo del trabajo realizado para el sensado y visualización de las mismas englobadas en un sistema microcontrolado.

4.2.1 Variable/Sensor de inclinación (ADXL335)

La primera de las variables a controlar era la inclinación máxima de seguridad que una máquina cortacésped podía alcanzar sin sufrir un vuelco que pudiera herir al usuario de la misma. En un primer momento, los ingenieros me recomendaron la creación de un sistema basado en un péndulo que ya estaban utilizando en algunos modelos con otros propósitos, y aunque lo desarrolle en un plano secundario, continue con mi idea que consistía en utilizar un sensor de aceleración (ADXL335) que realizaba perfectamente aquella tarea y que ya había utilizado anteriormente.

Para el desarrollo del sistema de control de este sensor, requeriría procesar 2 señales analógicas que me proporcionaba este correspondientes a los ejes X e Y que utilizaría para situar de manera digital a la máquina en el espacio. Para ello, debía utilizar 2 GPIO del PSoC como señales de entrada e idear un sistema de alarmas para que el usuario de la máquina sepa cuándo se encuentra en inclinaciones comprometidas.

Otra de las cuestiones a tener en cuenta es la alimentación por batería de 12V del cortacésped, que será nuestra fuente de alimentación y ya que los requerimientos de energía del PSoC son 5V y del sensor ADXL335 3.3V, deberemos añadir al circuito sendos reguladores de tensión(LM7805 y LM317) para que el circuito quede perfectamente suministrado.

Para presentar de una forma ordenada todas las características, requisitos y diseños asi como diagramas de bloque, esquema de circuito, etc, los separaré en diferentes puntos para desarrollar cada uno por separado: A. Sensor de inclinación ADXL335, B. Diagrama de bloques del sistema de control, C. Software del sistema y D. Desarrollo en placa de puntos, pruebas y resultados.

A. Sensor de inclinación ADXL335

El sensor ADXL es un sensor que detecta variaciones de aceleración en un rango de ±3g en la que se puede elegir el ancho de banda dependiendo de los condensadores utilizados a la salida de los pines correspondientes a los ejes X, Y y Z. Cada una de estas salidas, proporcionan una señal analógica proporcional a las aceleraciones detectadas en cada uno de los ejes, siendo el valor típico de unos 300 mV/g. El diagrama de bloques del mismo que puede encontrarse en su datasheet es el siguiente:

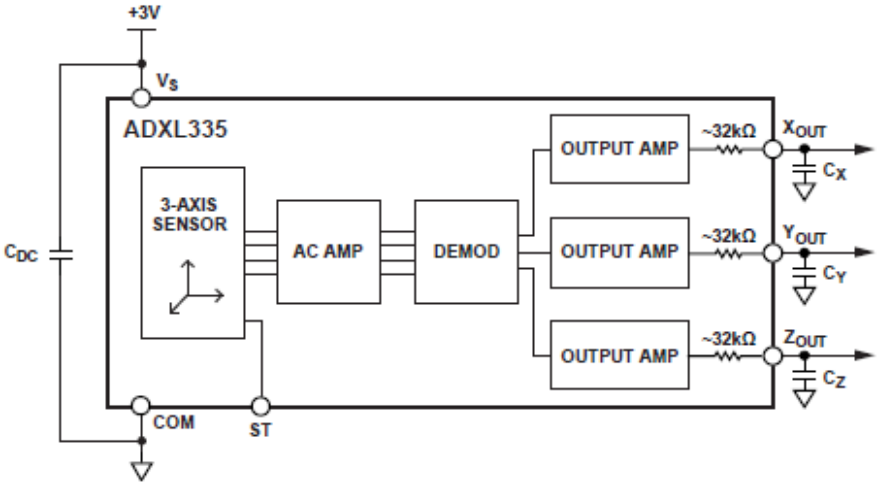


Fig 4.3 Diagrama de bloques del sensor ADXL 335

Como se puede ver en el diagrama, el sensor se debe alimentar a 3V, como valor típico, siendo el valor máximo admitido 3.6V. Con respecto a la alimentación, dependiendo del valor con el que alimentemos el sensor, este nos devolverá un valor de aceleración proporcional al mismo. Un condensador debe colocarse entre la alimentación y masa para desacoplar la alimentación del acelerómetro de la alimentación general. Normalmente y para la mayoría de las aplicaciones, un condensador de 0,1µF es suficiente.

Como ya he comentado, se deben incluir condensadores a las salidas de los pines de los ejes X, Y y Z funcionando como filtros paso bajo para reducir el ruido y problemas de aliasing. El fabricante recomienda que como mínimo, se utilicen condensadores de 4.7nF, mientras que para un ancho de banda de 50 Hz, lo recomendable es utilizar condensadores de 50Hz.
Además de los pines de alimentación, masa, eje X, eje Y y eje Z, el sensor incluye un pin de auto test (pin ST (Self Test)). Alimentándolo a Vs, permite al usuario comprobar el correcto funcionamiento del sensor.

Ya que para el prototipo utilice una placa de puntos, considere oportuno buscar un encapsulado de tipo THD para realizar las pruebas con el mismo. El problema es que este sensor se presenta con un encapsulado SMD de tipo LFCSP, por lo que tuve que buscar una alternativa que encontré en la página web www.sparkfun.com , donde encontré el sensor en una pcb con posibilidad de incorporarle pines, ya que sus salidas estaban routeadas a pads, incluyendo además los condensadores comentados anteriormente y con valor de 0,1µF.

Fig 4.4 ADXL 335

Con esta PCB que incluye el sensor, facilitaba sustancialmente mi labor para trabajar con él y añadiendo pines a los pads ruteados a las salidas del ADXL podía soldarlo a una placa de puntos y realizar todas las pruebas tanto hardware como software (con la placa de evaluación del PSoC asi como en la placa de puntos una vez realizada).

En la figura 4.2, podemos ver la PCB con el ADXL335 integrado y en la figura 4.3 vemos un esquema eléctrico de la misma. Una vez realizado el trabajo descrito, comprobé los valores de tensión que ofrecían a la salida los ejes para una inclinación/aceleración de reposo y ya que sólo requería de la información de los ejes X e Y (carece de sentido perder un puerto del PSoC para conocer conocer el estado del eje Z, que corresponde con, como muestra el dibujo impreso en la placa del ADXL, la posición boca arriba o boca debajo de la máquina. Posteriormente mostrare que este eje no será ruteado) me centré en estos, tomando unas lecturas similares para los dos ejes de 1.56V y 1.57V respectivamente.

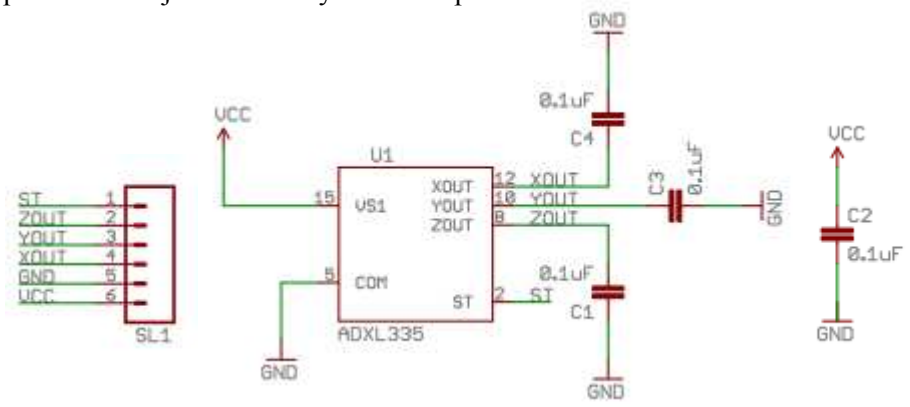


Fig 4.5 Esquemático de la PCB que integra el sensor ADXL335

Estas medidas me serán de gran ayuda a la hora de realizar el software en PSoC que determinarán los valores de inclinación críticos. Los ingenieros de la empresa me comentaron que los valores de inclinación que comprometían la estabilidad del modelo sobre el que trabajaría con el sensor (el cortacésped CLIMBER) era de unos 35º sobre la horizontal.
Por tanto, debía conocer los valores que arrojaba el sensor para diferentes valores de inclinación y después utilizar esos valores para realizar el software de sensibilización.

Este software, como comentaré en el siguiente apartado, deberá incluir un sistema de alarmas que avise al usuario del cortacésped en qué momento deberá variar su inclinación para evitar el vuelco de la misma. Para ello, en PSoC Designer debía utilizar diferentes bloques digitales asi como analógicos para realizar estas tareas.

En la placa de pruebas incluida en el kit de desarrollo (apartado 3.3.1) realicé las comprobaciones previas al montaje en placa de puntos para realizar las pruebas sobre la máquina. En este punto, debo comentar que al realizar las pruebas sobre la máquina ocurrió un problema de sensado ya que no tuve en cuenta la vibración de esta, que alteraba las lecturas del sensor y ofrecia datos erróneos, que al ser procesados obtenía respuestas inesperadas que no se correspondían con la inclinación real de la máquina.

B. Diagrama de bloques del sistema de control

El sistema de control del sensor ADXL 335 será el siguiente:

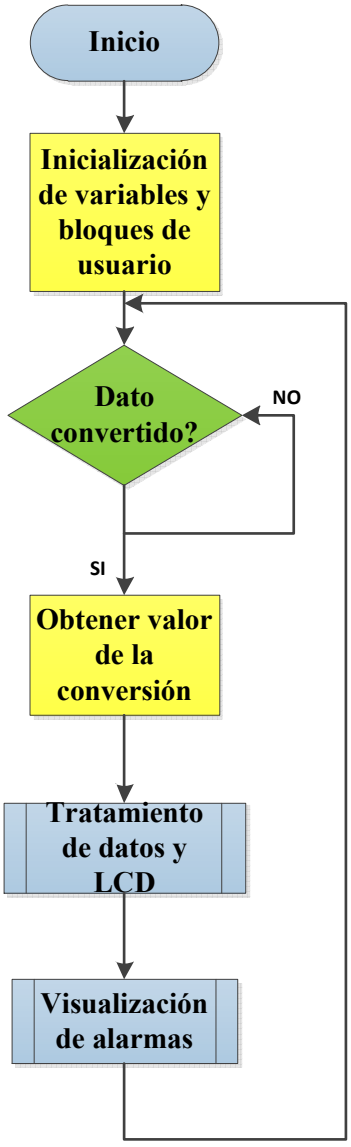


Fig 4.6 Diagrama de bloques del susbsistema ADXL

Este subsistema tiene como función captar las señales del sensor ADXL correspondientes a los ejes X e Y en los que se encuentra la máquina en tiempo real, transformar estas en señales digitales mediante un conversor AD y mostrar al usuario de la máquina, mediante una pantalla LCD y un par de diodos LED junto con un piezorresistivo, los grados de inclinación de la máquina y el grado de peligrosidad de dicha inclinación respectivamente. Para ello, como podemos ver en el diagrama de bloques de la fig. 4.5, el primer paso será realizar la inicialización y declaración de las variables que utilizaré así como los bloques de usuario.

Una vez completado este paso y mediante un bucle que se repetirá indefinidamente, esperaremos a que los Conversores AD (uno por eje) completen la conversión de los datos recogidos por el sensor ADXL. Cuando los datos arrojados por este se encuentren en formato digital, serán transformados en variables adecuadas para su posterior tratamiento mediante la subrutina de la fig. 4.6. Estos datos se compararán con los valores medidos para cada una de las inclinaciones y posteriormente se les asociará una variable, que será enviada a la subrutina de la fig 4.7, responsable del funcionamiento del sistema de alarmas.

Como podemos ver en la siguiente figura se muestra la subrutina de tratamiento de datos y LCD (fig. 4.6) que será comentada a continuación:

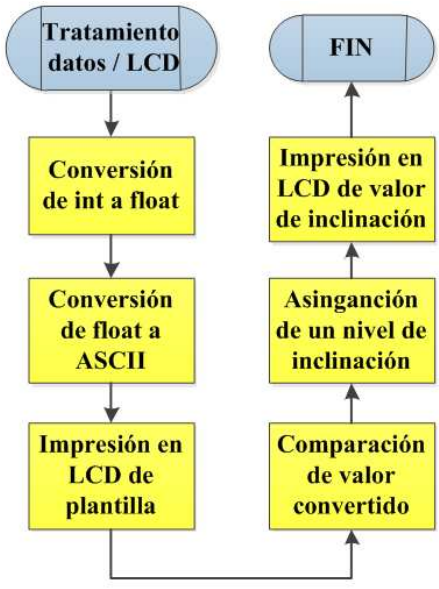


Fig. 4.7 Diagrama de bloques de la subrutina tratamiento de señales y LCD

Esta subrutina se encargará al mismo tiempo de varias tareas. Una de ellas, será convertir la señal digital del conversor AD con formato de entero (`int`) a formato flotante (`float`)para poder trabajar con el dato convertido por el AD, y de float a ASCII , posibilitando de esta forma escribir los datos recibido en el LCD. La otra función será utilizar las API´s del LCD para realizar una "plantilla" o carácteres que no se borrarán y actualizar en pantalla los datos variables recibidos del sensor.

También, como se puede ver en el diagrama, mediante la comparación entre el valor recibido por el conversor AD y los valores medidos previamente de las inclinaciones críticas para ambos ejes, se asignará un valor de inclinación numerado del 1 al 10 a cada una de ellas y estos valores de inclinación serán posteriormente utilizados en la subrutina de alarmas.

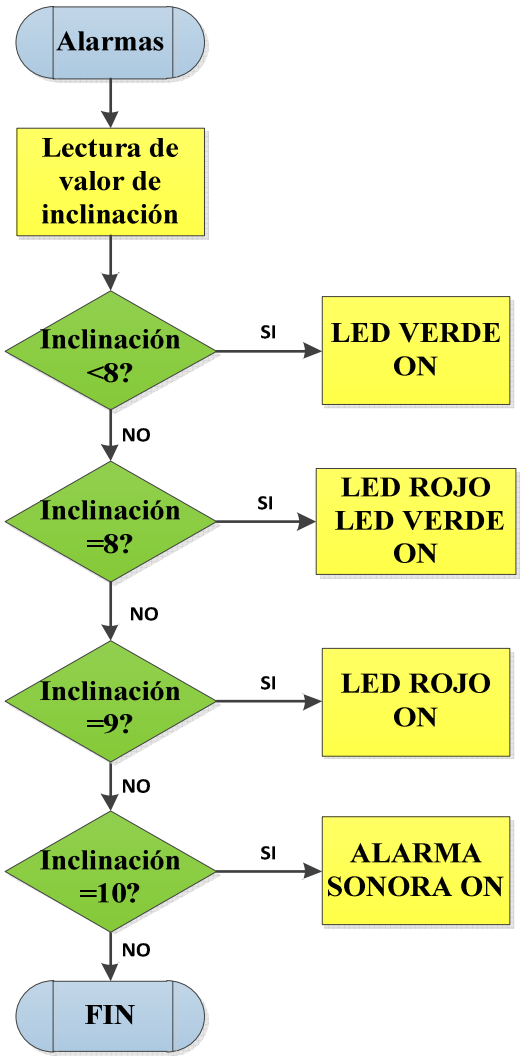


Fig. 4.8 Diagrama de bloques de la subrutina de alarmas

La figura 4.7 se corresponde con la subrutina encargada de activar los componentes asociados con el sistema de alarmas del subsistema. A través de los mencionados valores asignados del 1 al 10 a los datos proporcionados por el conversor AD, se conforma este diagrama basado en dichos valores.

Como se puede ver en el diagrama, una vez leído el valor de inclinación del 1 al 10, se realizarán una serie de comparaciones enlazadas resultando la activación de una de las alarmas, formadas por un diodo LED doble, es decir, con dos ánodos y un cátodo común resultando en la posibilidad de obtener 3 colores: verde, rojo y naranja(activando ambos ánodos al mismo tiempo) y por un piezorresistivo de 5V a modo de alarma sonora.

Cabe comentar, que aunque no se ve reflejado en el diagrama, la activación de cualquiera de las señales de alarma supondrá la desactivación de las restantes.

En el siguiente apartado se describirán todas las opciones de configuración, tanto del software como del hardware, del PSoC y del código C del subsistema en cuestión, desarrollando todos los pasos dados asi como imágenes del desarrollo del mismo en PSoC Designer.

C. Software del sistema

Para describir el software de programación del subsistema, realizare una serie de capturas de pantalla mostrando los aspectos más importantes de las configuraciones del hardware del PSoC así como la utilización de los bloques de usuario y las conexiones internas. Además, incluiré el código en C comentado. En la figura 4.8 se muestra el interfaz de PSoC Designer donde se muestran numerados los aparatados en los que modificaremos las propiedades del PSoC, tanto software como hardware. Cada apartado en el que se modifique alguna propiedad se comentará a continuación mediante las numeraciones que aparecen en la figura siguiente:

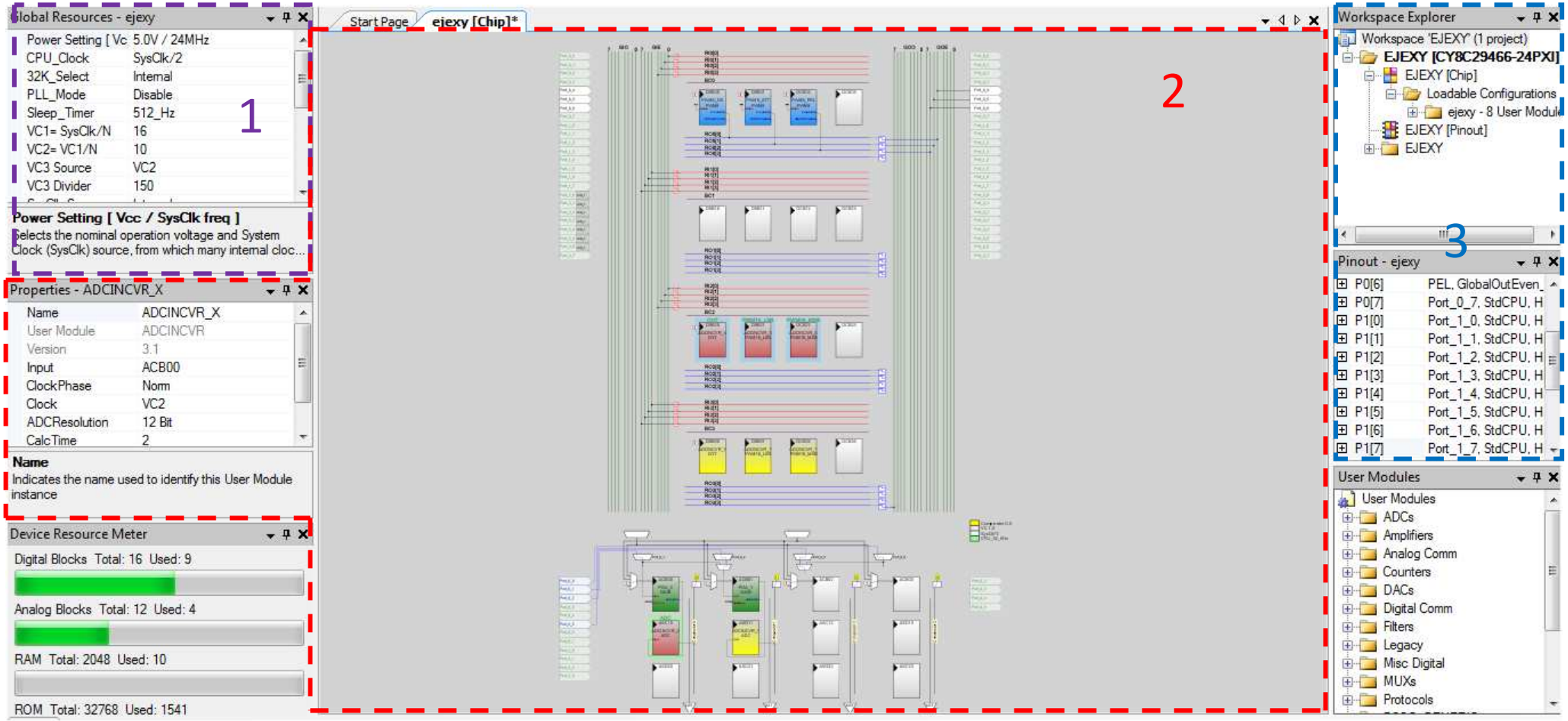


Fig. 4.9 Interfaz de usuario de PSoC Designer

1. Parámetros globales del PSoC

Para comenzar cualquier, proyecto, debemos configurar los parámetros globales más importantes del PSoC que son los siguientes:

Global Resources - ejexy

Power Setting [Vcc / SysClk freq]	5.0V / 24MHz
CPU_Clock	SysClk/2
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	10
VC3 Source	VC2
VC3 Divider	150
SysClk Source	Internal
SysClk*2 Disable	Yes
Analog Power	SC On/Ref Low
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

Fig. 4.10 Parámetros globales del subsistema ADXL

Tensión de alimentación y frecuencia de SysClk: Elegiré una tensión de alimentación de 5V y una frecuencia de 24MHz.

Frecuencia de reloj de la CPU: En este caso, la CPU funcionará a una velocidad de 12 MHz.

Fuente de SysClk: la elegiré interna ya que no requeriré del uso de cristal externo.

Deshabilitar SysClk*2: Al no requerir de SysClk multiplicado por dos, lo mantendremos deshabilitado.

Referencia analógica: Con esta opción seleccionamos el rango y la precisión de los bloques analógicos. Elegiré la opción (Vdd/2) / (-Vdd/2).

Frecuencias de trabajo VC1, VC2 y VC3: Estas frecuencias serán calculadas en función de las necesidades que requiere el circuito, como por ejemplo la frecuencia del PWM que activará el piezorresistivo o los diodos LED, o la frecuencia que necesitarán los Conversores AD. De esta manera, los valores utilizados serán los siguientes:

VC1= SysClk/N. Elegiré N=16 obteniendo VC1= 1.5 MHz

VC2=VC1/N. Eligiendo N=10 obtendré VC2= 150 KHz

Siendo VC3=VC2/N y eligiendo N=150, obtendremos VC3=1KHz (1ms)

Switch mode Pump: Esta opción la mantendremos deshabilitada al no requerirla.

Detección de bajo voltaje: Elegiré una tensión de 4.81V sobre 5V en la que el detector de bajo voltaje y el switch mode pump se activarán. Opción no relevante ya que no requeriré de estas utilidades.

Watchdog Enable: Al no requerir de perro guardián, la opción la mantendré desactivada.

Las demás opciones han sido descritas en el apartado 3.3.2.1, y su configuración se muestra en la fig. 4.9

2. *Mapa general del dispositivo. Bloques analógicos y digitales. Configuración*

En la siguiente imagen, la fig 4.11 se muestra el mapa general de la configuración de módulos, donde en la parte superior se pueden encontrar las 4 filas de bloques digitales y en la inferior las 4 columnas de bloques analógicos:

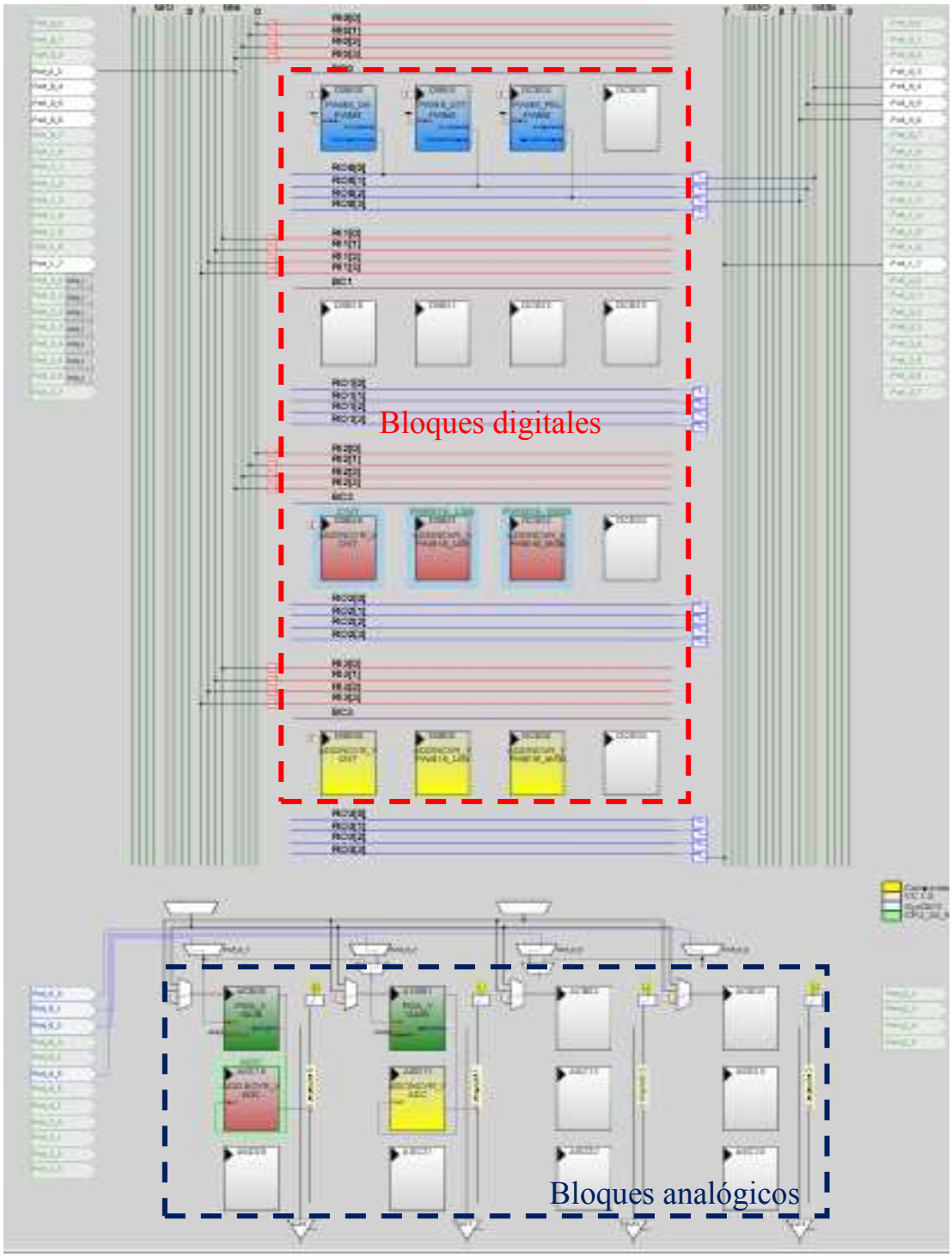


Fig. 4.11 Mapa de configuración de hardware del CY8C29466-24PXI

- Bloques analógicos

Los bloques analógicos que utilizaré serán 2 conversores AD que recibirán la señal desde un pin externo. La disposición de dichos bloques y su configuración en PSoC Designer son las siguiente:

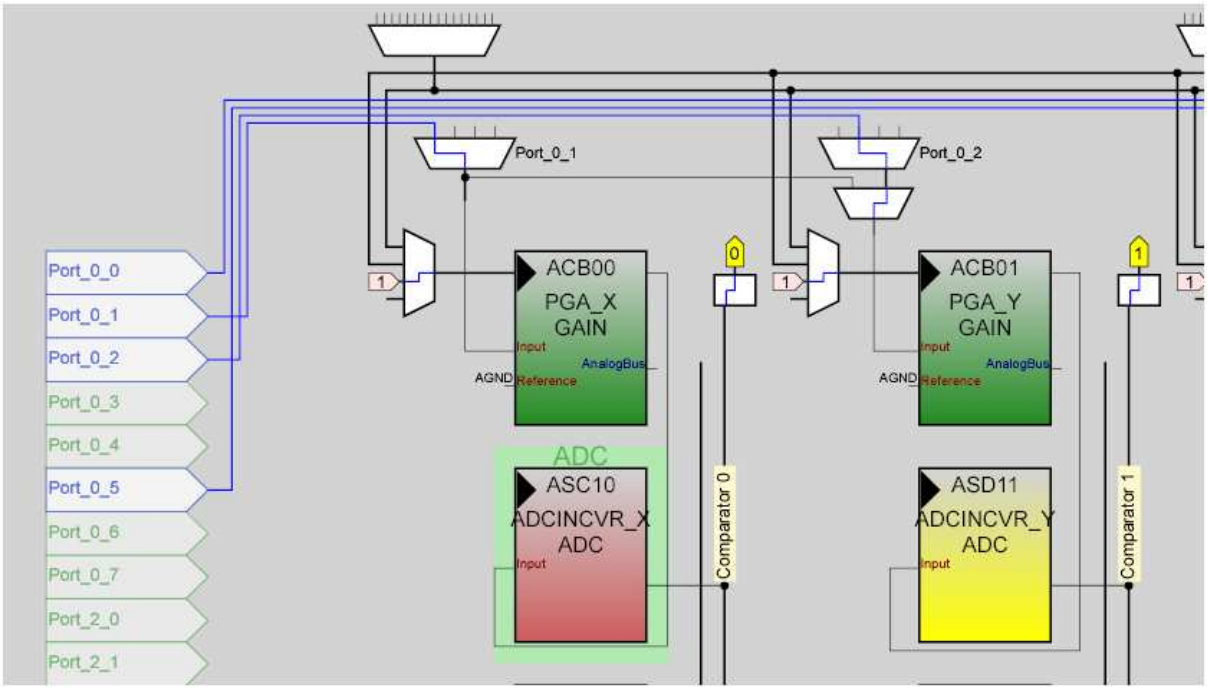


Fig. 4.12 Disposición de los bloques analógicos

Como se puede ver en la fig. 4.11, el sistema cuenta con dos conversores AD, correspondientes a los ejes X e Y, que se introducirán al sistema a través de los puertos P0_1 y P0_2 respectivamente. Ademas del bloque analógico que recibirá la señal externa a convertir, los conversores AD requieren de 3 bloques digitales mostrados en la fig. 4.12.

Ambos conversores cuentan con una configuración similar y como se puede observar, se requiere de sendos amplificadores de ganancia programable para conectar el pin con los bloques mencionados ya que no es posible conectar directamente la señal del sensor al conversor directamente.
Estos amplificadores como podremos ver a continuación, han sido configurados con una ganancia unidad para que la señal de entrada no se vea alterada. Por otro lado, la configuración de la parte digital de los conversores AD, que cuentan en este orden con un contador y un PWM de 16 bits que ocupa dos bloques (teniendo en cuenta que la configuración de ambos AD (fig.4.13) y amplificadores (fig. 4.14) es similar, solo mostraré la configuración de uno de ellos), así como su disposición en el PSoC, es la mostrada a continuación:

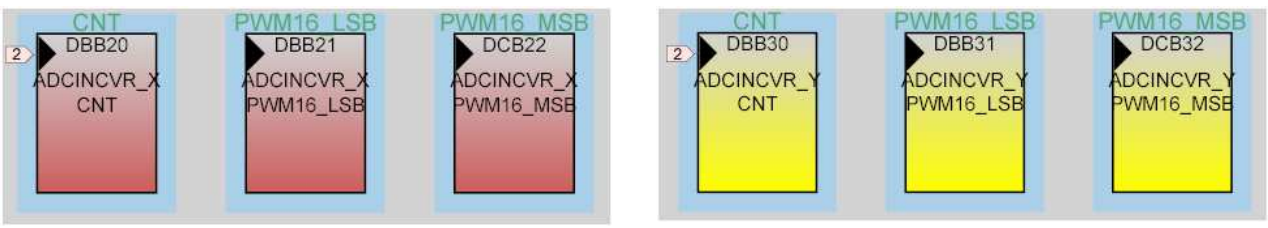


Fig. 4.13 Disposición de los bloques digitales de los conversores AD

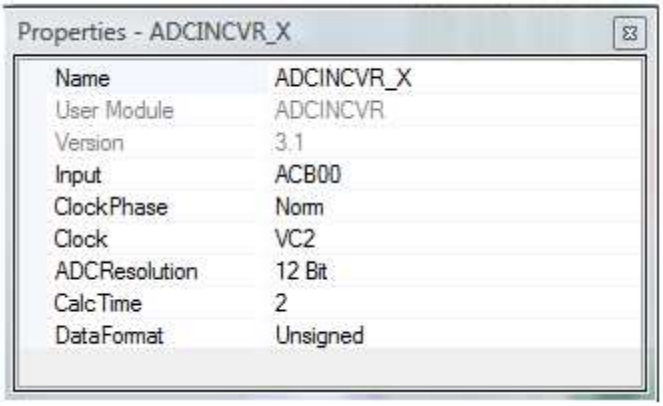


Fig. 4.14 Propiedades del conversor AD

La configuración de los conversores AD será la siguiente:

Entrada: Proveniente del PGA alojado en el bloque de tiempo continuo ACB00, y este a su vez ruteado al pin P0.1.

Fuente de reloj: Elegiré V2, que tiene un valor de 150 KHz o 6.66µs.

Resolución del conversor AD: Una resolución de 12 bits será adecuada.

CalcTime: Como ya he comentado anteriormente, este valor se calculara de la siguiente forma:

$$CalcTime \geq \frac{DataClock * 180}{CPUClock} = \frac{150KHz * 180}{24MHz} = 1.125$$

Por lo que elegiré calc valor 2. Entonces el valor de muestreo será el siguiente:

$$SampleRate = \frac{DataClock}{2^{bits+2} + CalcTime} = \frac{150K}{2^{14} + 2} = 9.1\ muestras/segundo$$

Obtendremos asi una velocidad de 9 muestras por segundo, más que suficiente para nuestro sistema.

Formato de la conversión: Ya que trabajaremos con valores que siempre serán positivos, el valor de la conversión lo configuraremos como sin signo.

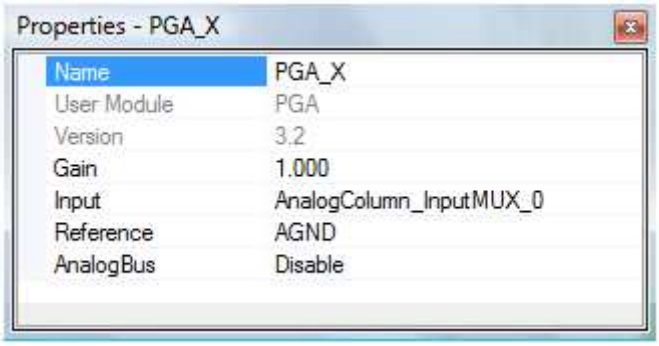


Fig. 4.15 Propiedades del Amplificador de ganancia programable

La configuración de los PGA o amplificadores de ganancia programable será la siguiente:

Ganancia: La ganancia tendrá un valor unidad para ambos casos, ya que no interesa perturbar la señal recibida por el sensor ADXL.

Entrada: Columna analógica del mux 0, ruteada a su vez con el puerto P0.1 en el caso del eje X. En el caso del eje Y, se encontrará ruteada con la columna analógica del mux 1y ruteada a su vez con el puerto P0.2

Referencia: Analog Ground. Tomaremos como referencia la masa analógica del sistema

De esta manera, las señales provenientes del sensor ADXL correspondientes a los ejes X e Y, serán recibidas por su conversor analógico digital sin alteración alguna, ya que al elegir una ganancia 1, no se verá adulterada.

- Bloques digitales

Además de los ya mencionados bloques digitales incluidos en los conversores AD, este subsistema requiere de 3 moduladores de anchura de pulso (PWM) de 8 bits para controlar la frecuencia de los componentes que funcionarán como alarmas del mismo. Como podemos ver en la fig. 4.16, la disposición de estos bloques es la siguiente:

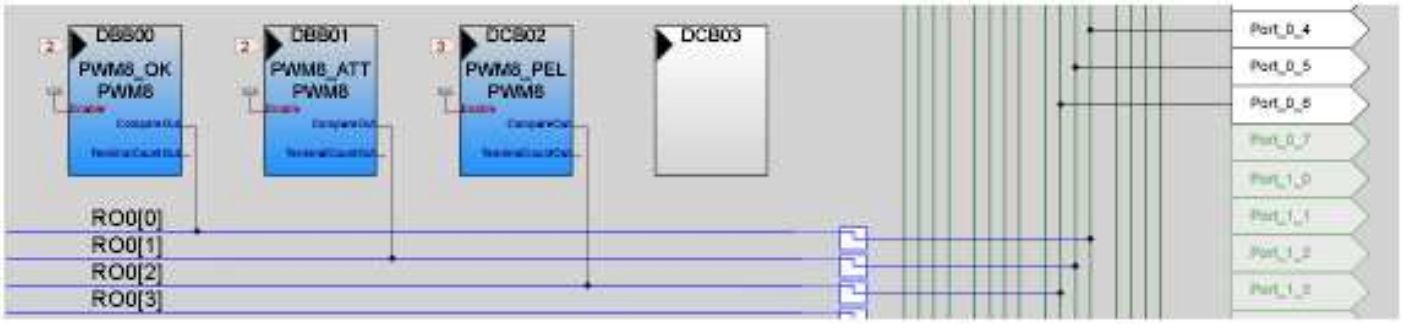


Fig. 4.16 Propiedades del Amplificador de ganancia programable

Cada uno de ellos, está conectado a uno de los pines de salida del PSoC, que serán configurados como Strong (explicado en el apartado 3.2.3). Dos de estos PWM activarán/desactivarán el diodo LED de doble ánodo, llamados PWM8_OK y PWM_ATT y el restante, PWM8_PEL se encargará de la activación/desactivación del piezorresistivo.

Los dos primeros tendrán una configuración similar a excepción de su conexión a los pines de salida del PSoC (Por lo que solo comentaré las propiedades configuradas de uno de ellos en la fig. 4.17), mientras que el restante (fig. 4.18) contará con una frecuencia menor ya que el piezorresistivo requiere de una menor velocidad de trabajo para obtener un sonido audible.

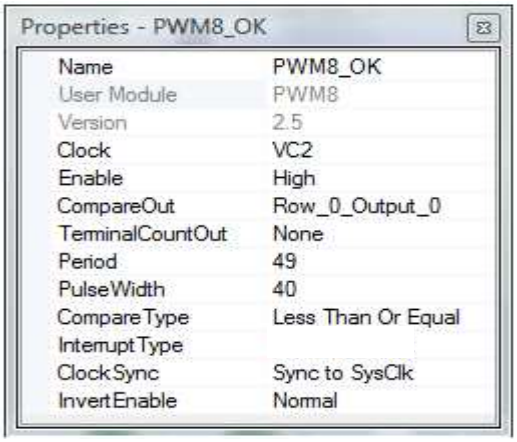


Fig. 4.17 Propiedades de los bloques PWM8_OK y PWM8_ATT

Fuente de reloj: VC2= 150KHz, proveniente de VC1/16. Esto equivale a 6.66µs.

Periodo: El periodo de contaje será 50 que completará un ciclo cada 50 * 6.66 µs = 0.33ms lo que equivaldrá a una frecuencia de señal de 3.03KHz.

Ancho de pulso: Eligiendo un valor 40, el ciclo de trabajo corresponderá al 80%.

Tipo de comparación: Elegiré un tipo de comparación "menor o igual que". Ya que el contaje tiene un carácter descendente, nos permitirá tener el valor alto del ciclo en la segunda parte de la onda.

Tipo de interrupción: Esta opción no será habilitada, ya que activaremos/desactivaremos estos bloques mediante las API´s especificas para dicha tarea.

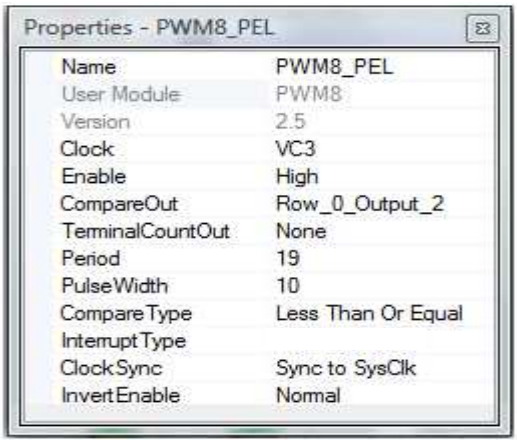


Fig. 4.18 Propiedades del bloque PWM8_PEL

Debido a que el piezorresistivo requiere de bajas frecuencias para emitir sonidos audibles, las únicas propiedades que variarán con respecto a los 2 anteriores son las siguientes.

Fuente de reloj: VC3= 1KHz, proveniente de VC2/150. Esto equivale a 1ms.

Periodo: El periodo de contaje será 20 que completará un ciclo cada 20 * 1ms = 20 ms lo que equivaldrá a una frecuencia de señal de 50Hz.

Ancho de pulso: Eligiendo un valor 10, el ciclo de trabajo corresponderá al 50%.

3. Situación final de los pines del PSoC (vista del integrado) y código C.

En estas dos ventanas, se muestran los pines utilizados y la situación global de los mismos en el integrado. En la fig. 4.19 se muestra la situación final del PSoC como último paso previo a la programación del software.
Los pines verdes se corresponden con las 2 entradas analógicas por las que el sistema recibirá las señales del sensor ADXL y tendrán una configuración del alta impedancia analógica.

Los pines rojos se corresponden a las salidas digitales mediante las cuales el PSoC activara, a través de sendos PWM, las alarmas del sistema y tendrán una configuración Strong.

Los pines azules del puerto 2, son los encargados de controlar el LCD mientras que los demás, que no tienen asignada función alguna, permanecerán con alta impedancia y como estándar del CPU.
El código C de este subsistema, se encuentra en el anexo correspondiente.

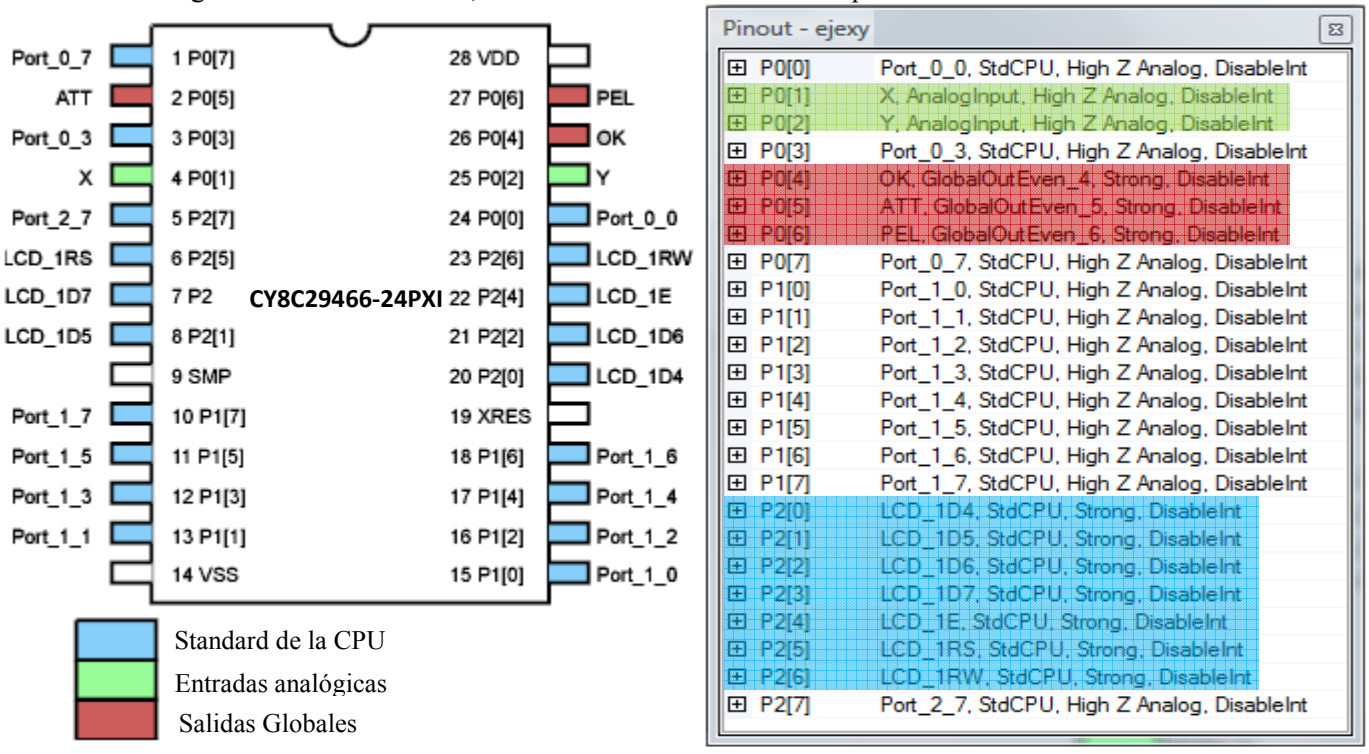


Fig. 4.19 Utilización de los pines del PSoC y configuración de los mismos

D. Desarrollo en placa de puntos, pruebas y resultados

Una vez realizado el software y comprobado su funcionamiento sobre el PSoC EVAL1, procedí a desarrollar el hardware del subsistema sobre una placa de puntos para realizar los test sobre la máquina cortacésped.
La comprobación de los valores de tensión para los cuales el sistema debía reaccionar mediante una de las alarmas dispuestas fue satisfactoria y ya que el usuario de la máquina no iba a estar pendiente constantemente de la inclinación marcada en el LCD sino que prestaría mayor atención a las alarmas visuales o sonoras, decidí no incluirlo en el diseño hardware.
De cualquier forma, el software no fue modificado ya que en función de los resultados obtenidos en los test, seria de utilidad para los programadores conocer en cada momento el valor de inclinación arrojado por el sensor para futuras modificaciones o correcciones.

Las placas de puntos correspondientes al subsistema creado son las que aparecen en la fig. 4.20 donde en la izquierda aparece la placa de puntos donde se encuentran los reguladores de tensión (a 5 y 3.3V respectivamente), el PSoC y el sensor ADXL que se colocarán en un lugar de la máquina protegido contra posible suciedad e inclemencias del tiempo, mientras que en la derecha podemos ver la placa que incluye el diodo LED tricolor y el piezorresistivo.

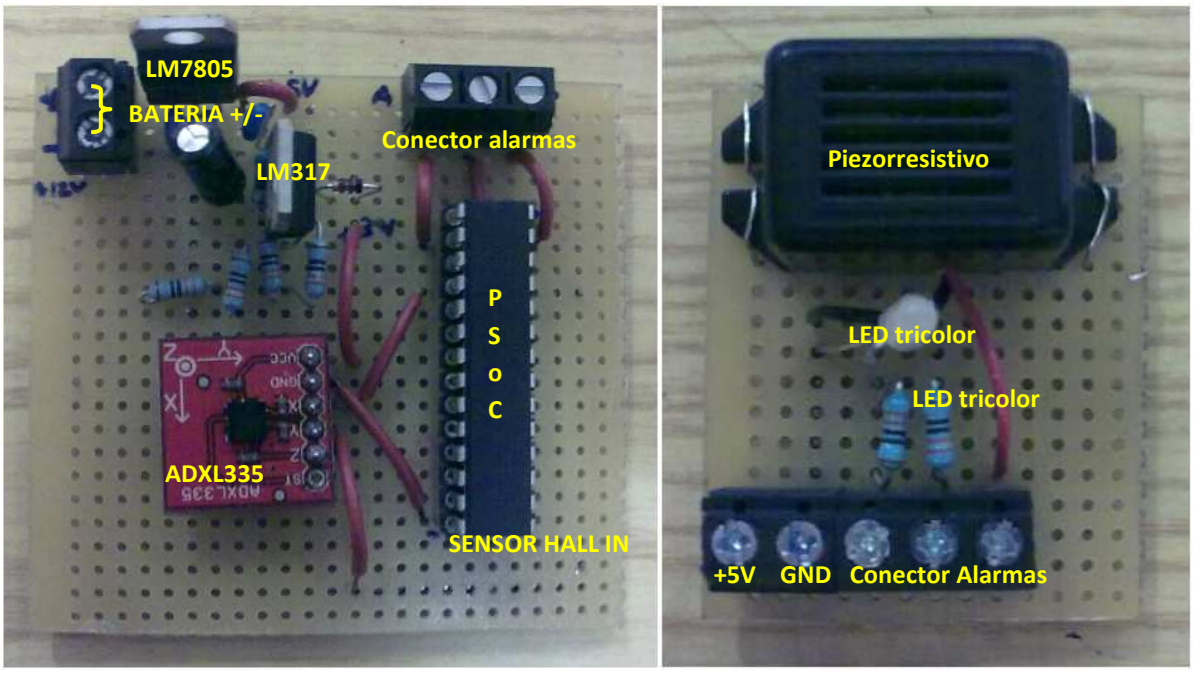


Fig 4.20 Hardware del subsistema basado en el sensor ADXL

Esta decisión de separar el sistema reside simplemente en la protección del sistema principal evitando fallos del sistema por problemas no relacionados con el normal funcionamiento de la misma así como para colocar la placa de las alarmas en un lugar visible para el usuario de la misma. En la fig. 4.21 se muestra la disposición del subsistema sobre la máquina a la hora de realizar los test sobre la misma para comprobar su correcto funcionamiento.

Fig 4.21 Montaje del subsistema sobre la máquina para la realización de los test de funcionamiento

Como se puede ver en la fig. 4.21 la inclusión de las placas de puntos sobre la máquina cortacésped la realicé de manera provisional mediante una pistola termoselladora y con cinta aislante aseguré los cables que conectaban ambas placas entre si y con la batería.

Con una mini grúa, realicé junto a algunos ingenieros de la empresa las primeras pruebas del sistema inclinando la máquina para comprobar si funcionaba correctamente. Cuando quedo comprobado que todo funcionaba de manera satisfactoria, nos dirigimos al exterior para realizar pruebas sobre el terreno con la máquina en funcionamiento. En este punto surgió un problema inesperado. La alta vibración de la máquina provocaba que el sensor recogiera datos que el PSoC falseaba produciendo errores en la visualización de las alarmas.

Debido a este problema debía realizar alguna modificación en el sistema para solventarlo. Se podía solventar mediante una modificación del software realizando una media de valores evitando el problema o incluyendo a la salida de los ejes X e Y del sensor sendos condensadores que filtrarán o evitaran esos cambios bruscos de la tensión solventando el problema.

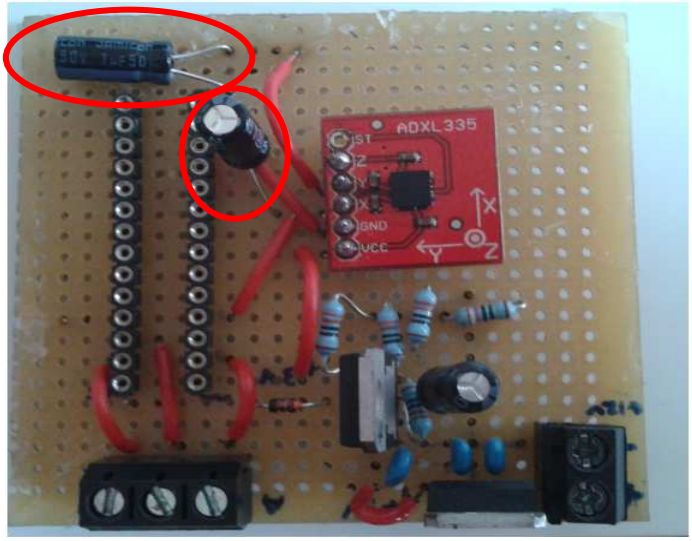


Fig 4.22 Modificación realizada sobre el hardware

Me decante por la segunda de opción incluyendo dichos condensadores en las salidas del sensor. En la fig. 4.22 se muestra la modificación realizada resaltando los condensadores incluidos.

Tras montar nuevamente la placa sobre el cortacésped, comprobé como el sistema de alarmas ya no se veía afectado por las vibraciones de la máquina y que el sistema funcionaba perfectamente.

De esta manera, este subsistema funcionaba correctamente y por tanto era válido para su inclusión en el diseño final.

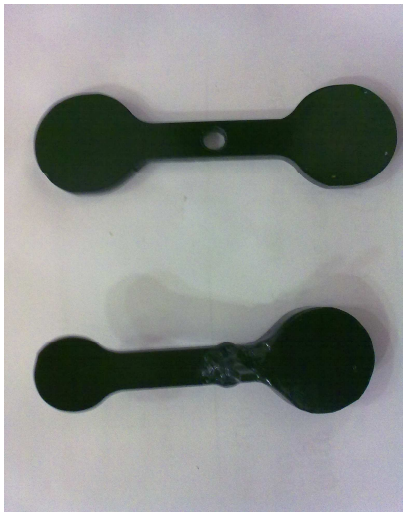


Fig. 4.23 Diferentes modelos de los péndulos usados

En la empresa me ofrecieron utilizar un péndulo, que ellos utilizan con otras aplicaciones, para utilizarlo como sensor de inclinación. El primer péndulo que me ofrecieron fue el que se encuentra en la parte superior de la figura 4.23 en el que ambos extremos eran idénticos y en la parte inferior el que utilicé posteriormente ya que con los extremos de diferente tamaño y peso se facilitaba el balanceo del mismo. A través de un potenciómetro quería medir las oscilaciones del péndulo y que el PSoC procesará dicha información devolviendo una señal luminosa o acústica en función de que la inclinación fuera aceptable, comprometida o peligrosa y mostrando los grados por el LCD.

Para ello, comencé un sencillo diseño que contaba con un regulador de tensión LM7805 que redujera los 12V de la batería y así alimentar el PSoC a su tensión de funcionamiento (5V), un potenciómetro para regular el brillo del LCD (con pines asociados a la inclusión del mismo) y otro que funcionaría como hipotético sensor de inclinación a través del péndulo que variaría con el ángulo de la máquina modificando así la resistencia del potenciómetro. Desde un primer momento, quedaba patente que el sistema carecía de la precisión necesaria y fue descartado. En lo referente al software, para realizar las pruebas utilicé el código C realizado para controlar el ADXL pero con unas modificaciones para controlar, en este caso, el potenciómetro y el LCD. En la fig. 4.24 se muestran unas fotografías del circuito realizado para comprobar el funcionamiento de este sistema basado en un péndulo.

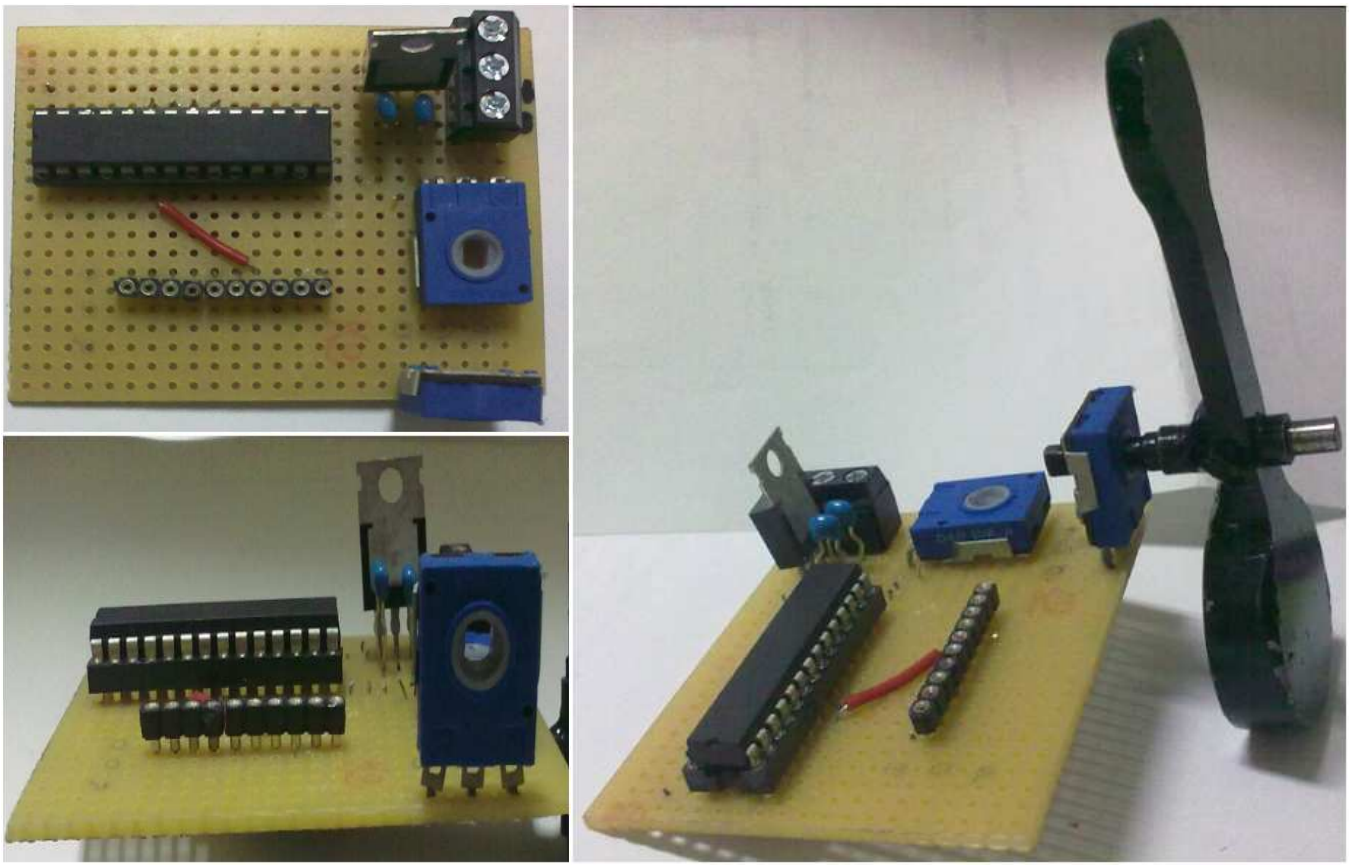


Fig. 4.24 Instantáneas del inclinómetro basado en un péndulo

4.2.2 Altura del plato de corte mediante sensor Hall

Otra de las variables a controlar era la altura que tenía el plato de las cuchillas de la máquina cortacésped con respecto al suelo en todo momento. Debido a la suciedad que se ocasiona de manera rutinaria al utilizar la máquina, debía encontrar un sensor adecuado para operar de manera eficaz bajo condiciones adversas severas de tiempo o de suciedad. Los ingenieros de Grillo SpA utilizaban un sensor con otras aplicaciones que me podría servir para controlar la altura en cuestión. Este sensor, que comentaré posteriormente, se basa en un sensor Hall de dos polos que permite conocer de manera precisa, dirección y distancia recorrida de su brazo móvil. Como dato adicional, el plato de corte se situará inicialmente en los 80mm y como máximo llegará a los 20mm del suelo, que será nuestro rango de operación.

Como en el caso anterior, requería procesar 2 señales analógicas correspondiente a cada uno de los polos del sensor Hall, utilizando ambas para conocer en qué dirección y a qué altura se estaba moviendo el plato en todo momento dependiendo siempre de un interruptor de 3 posiciones que el usuario de la máquina accionaría para subir o bajar el plato. En este caso, era necesario el uso del LCD para que el usuario conociera en todo momento la altura del plato de corte. Toda la alimentación del circuito será de 5V por lo que con un solo regulador de tensión (el LM7805), para reducir los 12V de la batería, se aseguraría un suministro estable y suficiente.

Al igual que en él capitulo anterior, desarrollaré en diferentes apartados todos los aspectos teóricos y prácticos del subsistema basado en el sensor Hall. Estos apartados serán los siguientes: A. Sensor Hall Linak LA23, B. Diagrama de bloques del sistema de control, C. Software del sistema y D. Desarrollo en placa de puntos, pruebas y resultados.

A. Sensor Hall Linak LA23

El LINAK LA23 (Fig. 4.25) es un actuador eléctrico que convierte el movimiento de rotación de un motor CC (en este caso del motor de desplazamiento del plato de corte de la máquina cortacésped) en movimiento lineal. Este modelo en particular, cuenta con un sensor Hall de dos polos que recuerda en funcionamiento a un encoder, proporcionando unas ondas que permiten al usuario conocer con precisión la posición en la que se encuentra la carga movida por el motor.

Fig.4.25 Actuador LINAK LA23

Este sensor se coloca en el eje del motor que mueve el plato de la cuchilla. Cuando el motor sea accionado a través del interruptor del panel de mando (Fig.4.26), el eje del actuador girará solidario con él. Esto provoca que los imanes dentro del sensor se pongan en movimiento. Y es, en ese momento, cuando los 2 sensores Hall empiezan proporcionar señales cuadradas que son equivalentes a la velocidad y al sentido de giro del eje del motor.

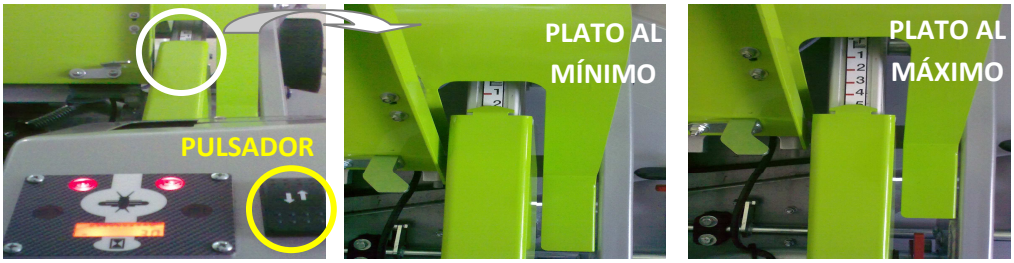


Fig.4.26 Pulsador en el panel del mando del cortacésped y posición máx./mín. del indicador de altura del plato de corte actualmente

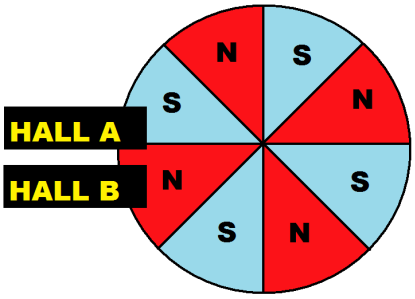
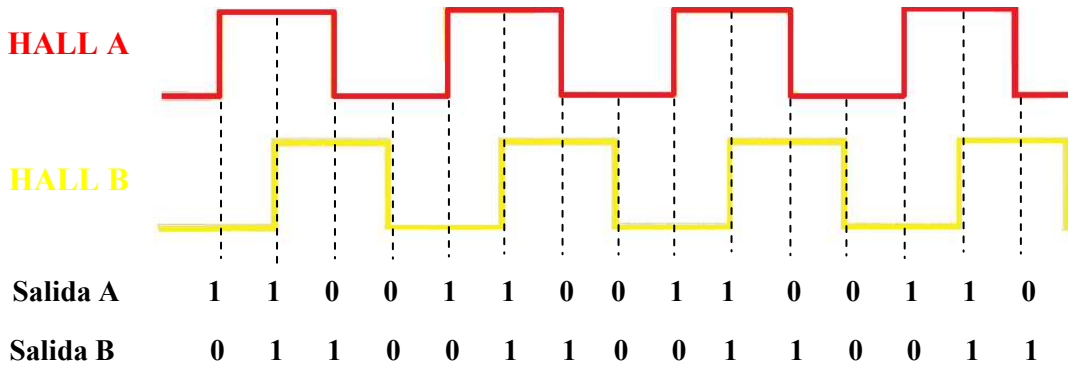


Fig.4.27 Detalle del interno del actuador LA23

Las Fig. 4.27 muestra un detalle del interior del LA23 dónde se pueden ver los dos sensores Hall alineados sobre un cilindro que cuenta con imanes con polaridades intercaladas. Colocados de esta forma, proporcionan las señales cuadradas mencionadas que se muestran en la Fig. 4.23 correspondientes a las ondas proporcionadas por el mismo cuando se encuentra en funcionamiento. De acuerdo con las señales cuadradas, el SENSOR HALL A proporciona información acerca de la velocidad del motor (en forma de frecuencia) y el SENSOR HALL B permite determinar el sentido de giro del motor (comparando con la onda proporcionada por el sensor A, si cuando el sensor B proporciona un nivel alto, A está en alto, el motor estará girando en sentido horario. En caso contrario, el giro será en sentido antihorario).

Actuador Entrando (Sentido horario)



Actuador Saliendo (Sentido antihorario)

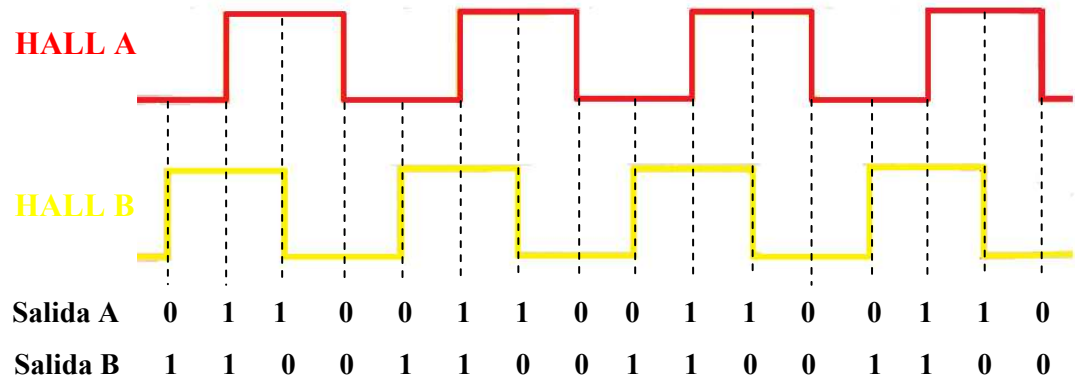


Fig.4.28 Ondas de salida del actuador LA23

La siguiente tabla (Fig. 4.29) muestra algunos de los datos más relevantes con respecto al actuador LA23, a partir de la cual, pude realizar los cálculos necesarios para configurar correctamente tanto el hardware como el software del sensor.

Descripción	Especificaciones	Comentarios
Voltaje de entrada	**8-24VDC**	El circuito de realimentación es alimentado 1seg. antes del arranque del motor y hasta un 1seg. después de su detenimiento
Voltaje de salida	NPN en colector abierto con 10KΩ a VCC	Voltaje de salida típico: **HIGH: Output=Input-1V** **LOW: Output=0-1.5V (27Ω serie)**
Consumo de corriente	El consumo máximo de corriente son 20mA	Incluso cuando el actuador no está funcionando
Resolución	El sistema de realimentación ofrece 16 pulsos por cada vuelta del eje: **3mm/rotación: 0.1875mm por pulso**	Mínimo de 10ms entre pausa/pulso. En un recorrido de 100mm se obtendrán el siguiente número de pulsos: **3mm/rotación=533pulsos**
Conexiones	Fig. 4.25	Uso de cable estándar de 6 hilos

Fig. 4.29 Tabla de características del LA23

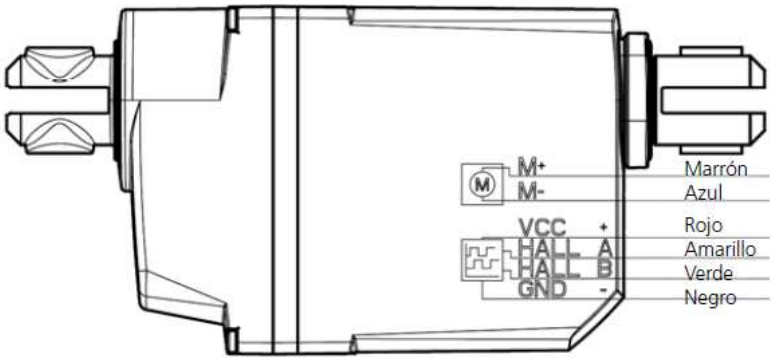


Fig. 4.30 Conexiones internas del LA23

Mediante la mencionada tabla, donde he resaltado los datos mas relevantes, era posible realizar todos los cálculos necesarios así como las correctas conexiones del actuador (que entre cada pulso transcurren 10ms y que redondeando, con cada pulso se produce un desplazamiento de 0.2mm). Ya que el sistema es lo bastante preciso escogería como referencia los 0.1875mm por pulso y realizando (mediante software) un contador de 53 ciclos obtendríamos aproximadamente el contaje de un centímetro, que usaremos para conocer la posición relativa del plato de cuchillas. Ya que el plato se accionará únicamente si el interruptor de subida o bajada del mismo es accionado (condicionando el sentido de movimiento del motor), se podía prescindir de la señal de uno de los dos sensores Hall, eligiendo por comodidad el sensor HALL A.

Como la alimentación del LA23 se realizaría a 12V y sus salidas tendrían como máximo una tensión de 11V, mediante un divisor de tensión transformaría esos 11V en un máximo de 5V y así poder procesar dichas señales en el PSoC. Al igual que con el sensor ADXL, previo montaje del LA23 sobre la placa de puntos realice una serie de comprobaciones sobre el PSoC EVAL-1 para contrastar los datos del datasheet así como el correcto funcionamiento de este.

B. Diagrama de bloques del sistema de control

El sistema de control correspondiente al actuador LA23 será el siguiente:

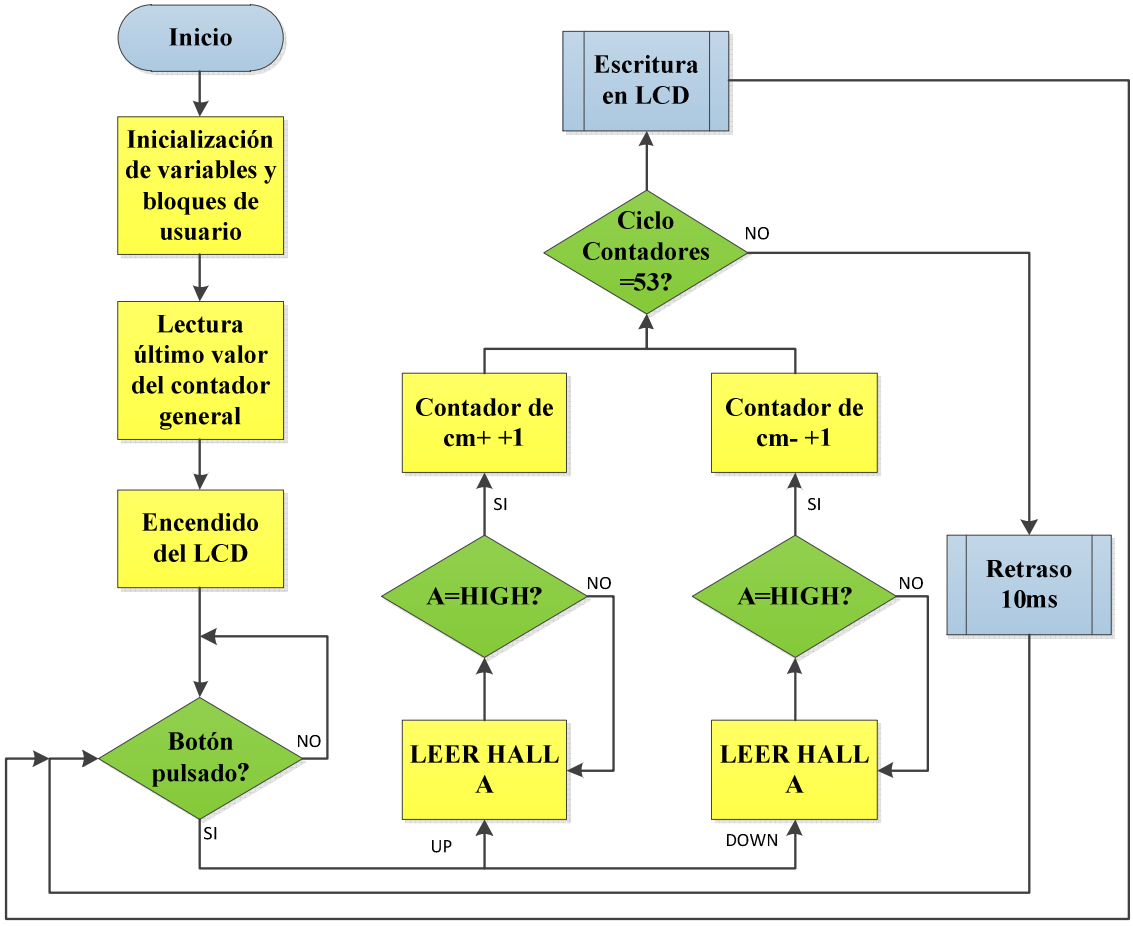


Fig. 4.31 Diagrama de bloques del sistema basado en el sensor Hall

Este subsistema tiene como función, mostrar en el LCD la posición horizontal del plato de corte expresado en centímetros. Como ya ha sido comentado, el sistema se dependerá en un primer momento del accionamiento del interruptor de subida o bajada del plato, el cual desencadenará todo el proceso de control del sensor HALL. Al contar con un interruptor para cada movimiento del plato (subida o bajada) se podía prescindir de una de las dos señales proporcionadas por el sensor Hall ya que con una de ellas, a partir de la cual obtener la altura relativa del plato en cada momento, era suficiente. Como se puede ver en el diagrama de bloques, una vez que el usuario de la máquina ha pulsado el botón de subida o de bajada del plato de cuchillas, se leera el pin del PSoC donde se haya asignado la señal del sensor Hall y se comprobará si tiene un nivel alto.

En caso afirmativo, se incrementará en una unidad el contador correspondiente al interruptor presionado y se realizará una comparación del valor resultante con el valor calculado (equivalente a un centímetro). Si el contaje ha alcanzado dicho valor se procederá a la escritura en el LCD del valor actual del plato y, en caso contrario, se realizará un retraso de 10ms correspondiente a la frecuencia de trabajo del sensor Hall y se comprobará entonces si el interruptor sigue pulsado para repetir la operación.

La subrutina correspondiente a la escritura en el LCD se muestra a continuación en la Fig. 4.32:

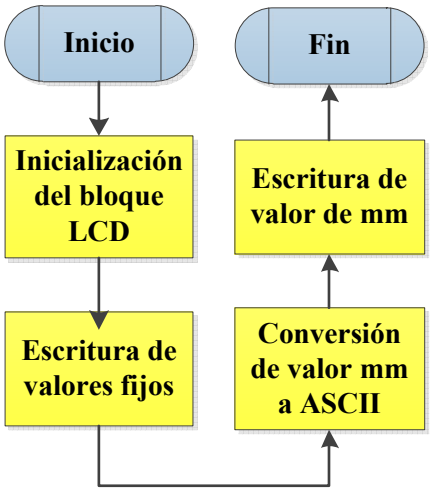


Fig. 4.32 Diagrama de bloques de la subrutina del LCD

Una vez inicializado el bloque LCD, se escribirán las cadenas de caracteres que no variarán a lo largo de la ejecución del subsistema, que serán el nombre de la empresa, el indicador "LAMA" tras el cual aparecerá el valor en mm del plato de corte y "mm" tras dicho valor. Este valor deberá ser convertido en ASCII para poder escribirlo en el LCD, previa conversión del valor int a valor float. Una vez realizadas estas conversiones será posible escribir el valor en milímetros calculado a través de la pantalla. Esta subrutina como se puede ver es mucho más sencilla que la utilizada para la escritura en el LCD en el subsistema basado en el sensor ADXL. La subrutina que figura con el nombre de Retraso 10ms", consiste simplemente en un contador realizado mediante software en PSoC Designer que será descrito posteriormente.

C. Software del sistema

Al igual que con el subsistema anterior, realizare una serie de capturas de pantalla mostrando los aspectos más importantes de las configuraciones del hardware del PSoC, la utilización de los bloques de usuario, las conexiones internas e incluiré el código C correspondiente. Tomando como referencia nuevamente la figura 4.9 procederé a comentar las configuraciones realizadas para este subsistema basado en el sensor Hall:

1. Parámetros Globales del PSoC

Global Resources - hall

Power Setting [Vcc / SysClk freq]	5.0V / 24MHz
CPU_Clock	SysClk/1
32K_Select	Internal
PLL_Mode	Disable
Sleep_Timer	512_Hz
VC1= SysClk/N	16
VC2= VC1/N	15
VC3 Source	VC2
VC3 Divider	100
SysClk Source	Internal
SysClk*2 Disable	Yes
Analog Power	SC On/Ref High
Ref Mux	(Vdd/2)+/-(Vdd/2)
AGndBypass	Disable
Op-Amp Bias	Low
A_Buff_Power	Low
SwitchModePump	OFF
Trip Voltage [LVD (SMP)]	4.81V (5.00V)
LVDThrottleBack	Disable
Watchdog Enable	Disable

Fig. 4.33 Parámetros globales del subsistema ADXL

Los parámetros globales del PSoC para este subsistema serán los siguientes:

Tensión de alimentación y frecuencia de SysClk: Elegiré una tensión de alimentación de 5V y una frecuencia de 24MHz.

Frecuencia de reloj de la CPU: En este caso, la CPU al máximo, esto es, 24MHz

Fuente de SysClk: La elegiré interna ya que no requeriré del uso de cristal externo.

Deshabilitar SysClk*2: Al no requerir de SysClk multiplicado por dos, lo mantendremos deshabilitado.

Referencia analógica: Con esta opción seleccionamos el rango y la precisión de los bloques analógicos. Elegiré la opción (Vdd/2) / (-Vdd/2).

Frecuencias de trabajo VC1, VC2 y VC3: Es necesario calcular estos tres valores para satisfacer las necesidades del bloque contador y del bloque PWM, que describiré más adelante:

VC1= SysClk/N. Elegiré N=16 obteniendo VC1= 1.5 MHz

VC2=VC1/N. Eligiendo N=15 obtendré VC2= 100 KHz

Siendo VC3=VC2/N y eligiendo N=100, obtendremos VC3=1KHz (1ms)

Switch mode Pump: Esta opción la mantendremos deshabilitada ya que no la necesitaré.

Detección de bajo voltaje: Elegiré una tensión de 4.81V sobre 5V en la que el detector de bajo voltaje y el switch mode pump se activarán. Opción no relevante ya que no requeriré de estas utilidades.

Watchdog Enable: Al no requerir de perro guardián, la opción la mantendré desactivada.

Las demás opciones han sido descritas en el apartado 3.3.2.1, y su configuración se muestra en la fig. 4.33

2. *Mapa general del dispositivo. Bloques analógicos y digitales. Configuración*

En la Fig. 4.34 se muestra el mapa general de la configuración de módulos, donde en la parte superior se pueden encontrar las 4 filas de bloques digitales y en la inferior las 4 columnas de bloques analógicos:

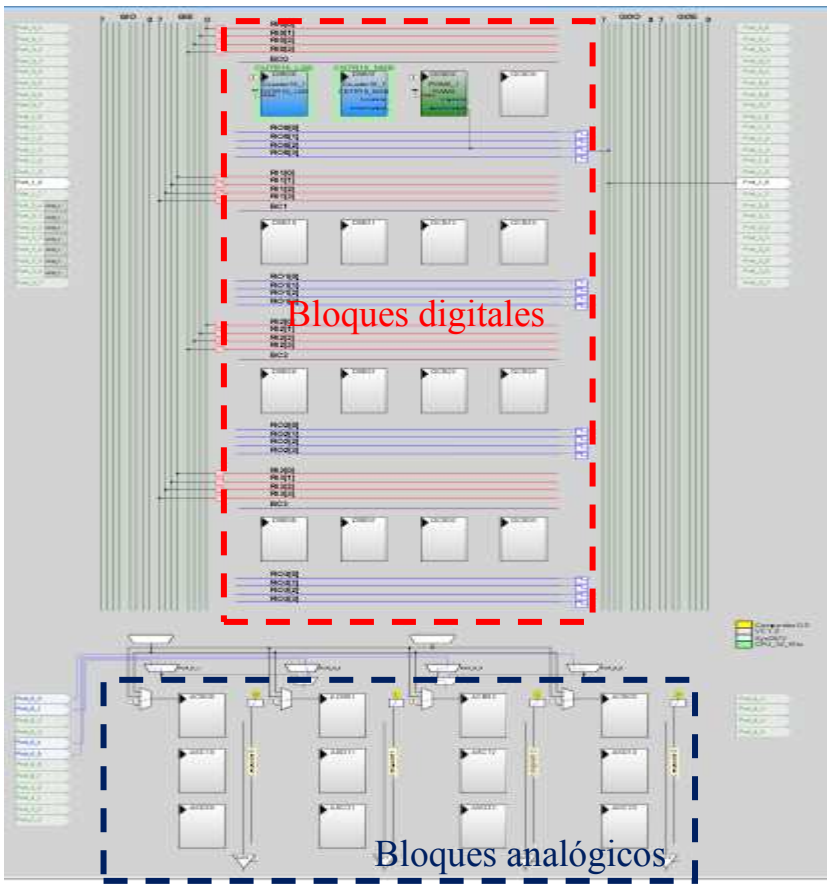


Fig. 4.34 Mapa de configuración de hardware del CY8C29466-24PXI

- Bloques analógicos

En este caso, el sistema no requerirá de bloques analógicos, ya que la recepción de las señales analógicas recibidas, tanto por parte del sensor Hall así como del interruptor se realizarán mediante el software realizado en código C.

- Bloques digitales

Este sencillo substistema, cuenta únicamente con un contador de 16 bits para realizar la rutina de retraso de 10ms. Si bien podría haber escogido uno de 8 bits, para realizar los test de funcionamiento preferí usar el mencionado. Además del contador, incluí un modulador de anchura de pulso de 8 bits para realizar los primeros test en ausencia del sensor hall, que realizaría la función de este. Como podemos ver en la Fig. 4.35, la disposición de los bloques mencionados es la siguiente:

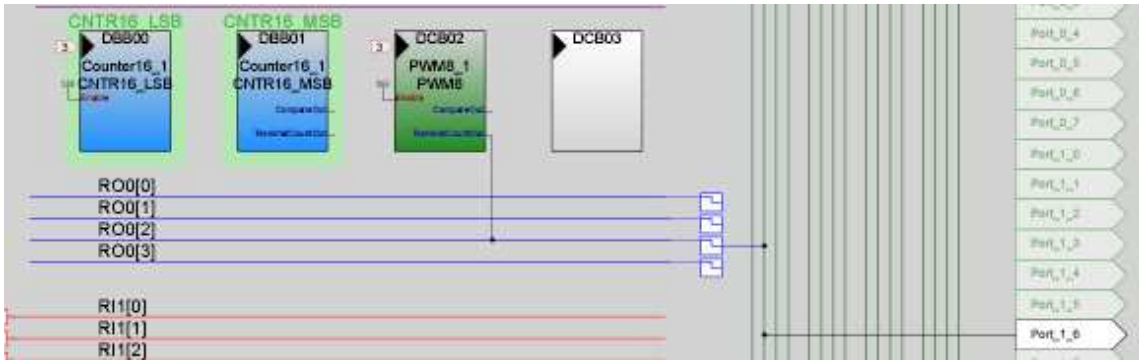


Fig. 4.35 Disposición de los bloques digitales

Además del LCD, el PWM de 8 bits será el único bloque que será asociado a un pin de salida del PSoC, el cual será realimentado al pin de entrada correspondiente con el de entrada del sensor Hall. A continuación mostrare la configuración de los mencionados bloques. En la Fig. 4.36 se puede ver el cuadro de configuración del bloque PWM y en la Fig. 4.37 la del contador de 16 bits.

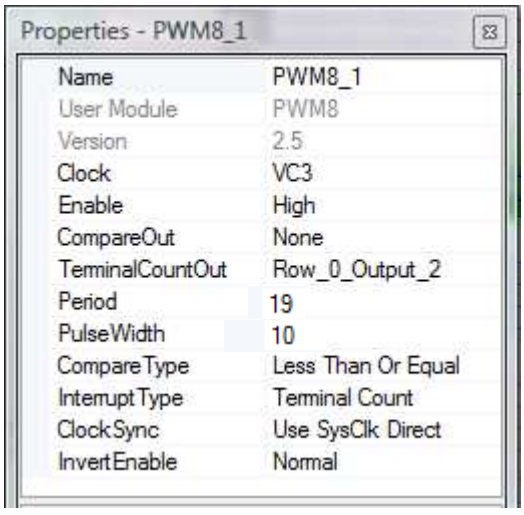


Fig. 4.36 Propiedades de los bloques PWM8_1

Fuente de reloj: VC3= 1KHz, proveniente de VC2/100. Esto equivale a 1ms.

Periodo: El periodo de contaje será 20 que completará un ciclo cada 20ms lo que equivaldrá a una frecuencia de señal de 50Hz.

Ancho de pulso: Eligiendo un valor 10, el ciclo de trabajo corresponderá al 50%.

A través de estas dos propiedades, conseguiremos una onda similar a la que obtendría del sensor Hall, con un mínimo de 10 ms entre los niveles alto y bajo de la onda.

Tipo de comparación: Elegiré un tipo de comparación "menor o igual que". Ya que el contaje tiene un carácter descendente, nos permitirá tener el valor alto del ciclo en la segunda parte de la onda.

Tipo de interrupción: Esta opción no será habilitada, ya que activaremos/desactivaremos estos bloques mediante las API´s específicas para dicha tarea.

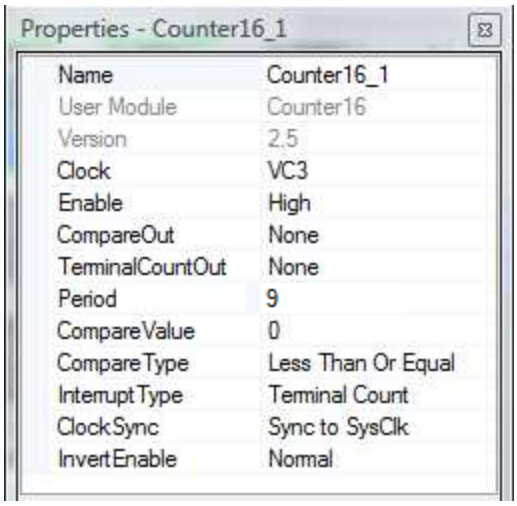


Fig. 4.37 Propiedades del bloque contador de 16 bits

Como ya he comentado, se podría elegir indistintamente un contador de 8 bits ya que para realizar el retraso de 10 ms es suficiente.

Fuente de reloj: VC3= 1KHz, proveniente de VC2/100. Esto equivale a 1ms.

Periodo: El periodo de contaje será 10 que completará un ciclo cada 10ms lo que equivaldrá a una frecuencia de señal de 100Hz.

Tipo de comparación: Elegiré un tipo de comparación "menor o igual que". Ya que el contaje tiene un carácter descendente, nos permitirá tener el valor alto del ciclo en la segunda parte de la onda.

3. Situación final de los pines del PSoC (vista del integrado) y código C.

Como se puede observar en la Fig 4.38, la disposición final de los pines utilizados para la creación del subsistema basado en el sensor Hall es la siguiente:

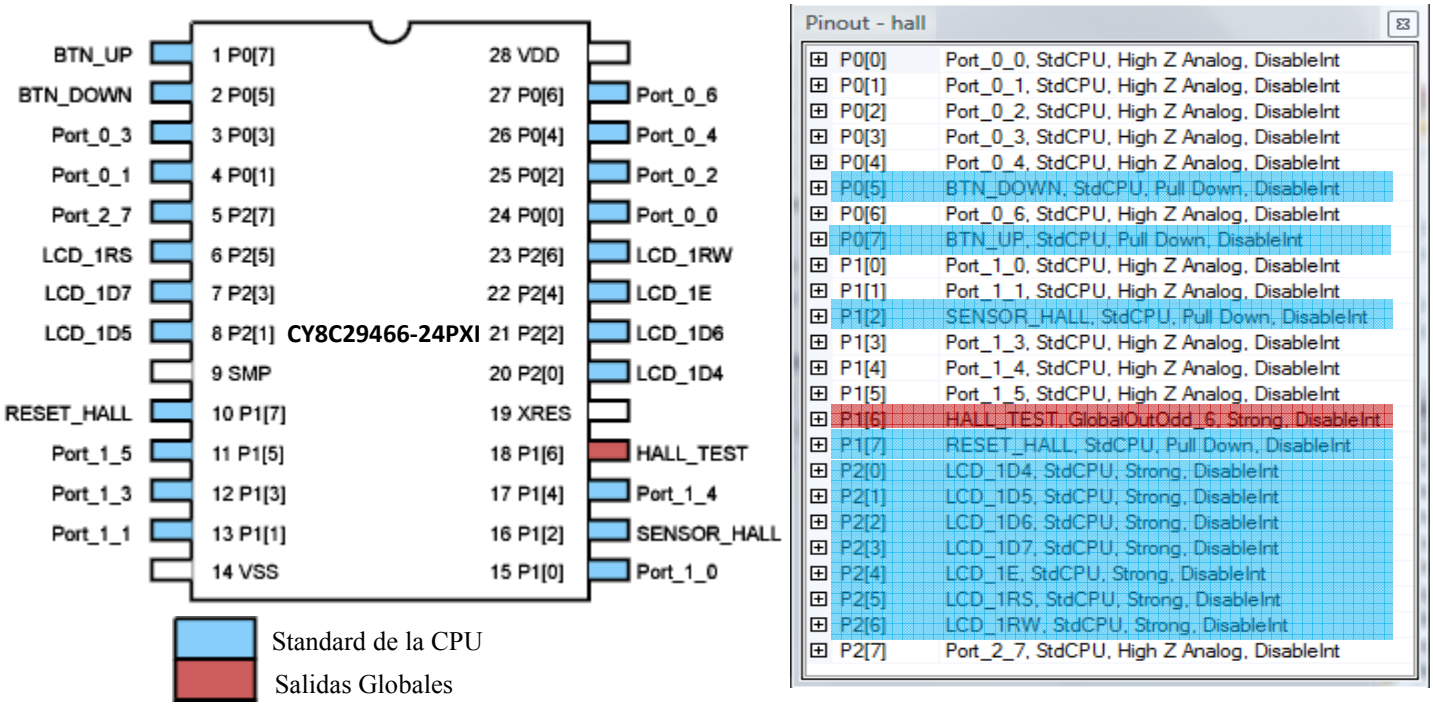


Fig. 4.38 Utilización de los pines del PSoC y configuración de los mismos

Además de los pines del puerto 2 destinados al LCD, se pueden observar los pines correspondientes a los botones de reset del LCD, los botones de subida y bajada del plato de cuchillas y el de entrada del sensor hall, todos ellos configurados como pull down y como estándares de la CPU.
Como único pin configurado como salida global, vemos el pin P1 [6] correspondiente a la salida del bloque PWM que simulará la onda del sensor Hall para realizar los test previos a su montaje en el circuito. Por ello, este pin se encontrará puenteado al P1 [2].
El código C del subsistema, se encuentra en el anexo correspondiente.

D. Desarrollo en placa de puntos, pruebas y resultados

Como con el subsistema anterior, una vez comprobado que el funcionamiento del software era el esperado, realice un prototipo en una placa de puntos para realizar los test pertinentes sobre la máquina cortacésped.
A diferencia del subsistema basado en el sensor ADXL, el usuario de la máquina si tendrá en cuenta el valor que aparezca en la pantalla LCD, tanto al comenzar a utilizarla y en cualquier momento en el que deseara modificar la altura del plato de corte para, por ejemplo, elegir un corte más apurado. Por ello, tomé la decisión de separar la pantalla LCD de la placa principal y, de esta manera, poder colocarlo en un lugar de fácil lectura por parte del usuario y facilitar su trabajo.

Como se puede observar en la Fig. 4.35, se encuentra por un lado la placa donde está montado el PSoC, el regulador de tensión (LM7805), el potenciómetro para regular el brillo del LCD, los bornes de alimentación de la batería, los correspondientes a los botones de subida y bajada del plato de corte y el de entrada de la señal del sensor Hall.

Como se puede observar de cada uno de los bornes correspondientes con los interruptores de subida y bajada, así como del sensor Hall cuentan con resistencias que realizarán un divisor de tensión para disminuir la tensión a valores no peligrosos para la integridad del PSoC.

Las señales que llegan a estos bornes son de 12V para el caso de los interruptores y de 11V para el caso de la señal del sensor Hall. Un pequeño diagrama del divisor realizado sobre la placa es el siguiente (Fig. 4.34). Como ya he comentado la resistencia R1 será diferente dependiendo de si la entrada es de 12V o en su defecto de 11V. Además, también añadí los pines necesarios para conectar esta placa al LCD.

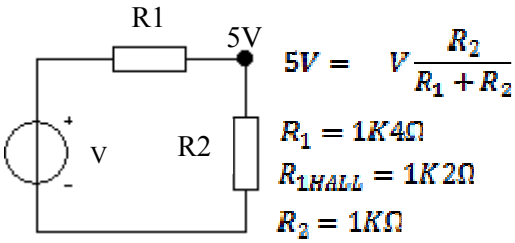


Fig. 4.39 Divisor de tensión utilizado en el subsistema

Por otro lado, la placa que alberga el LCD contaba con su propio borne para alimentarlo directamente desde la batería y con el divisor de tensión (Fig. 4.39). De esta manera, evitaba el uso de otro cable para llevar los 5V de alimentación del LCD desde la placa principal, facilitando el montaje del subsistema.

El subsistema de control del sensor Hall se puede ver en la Fig. 4.40:

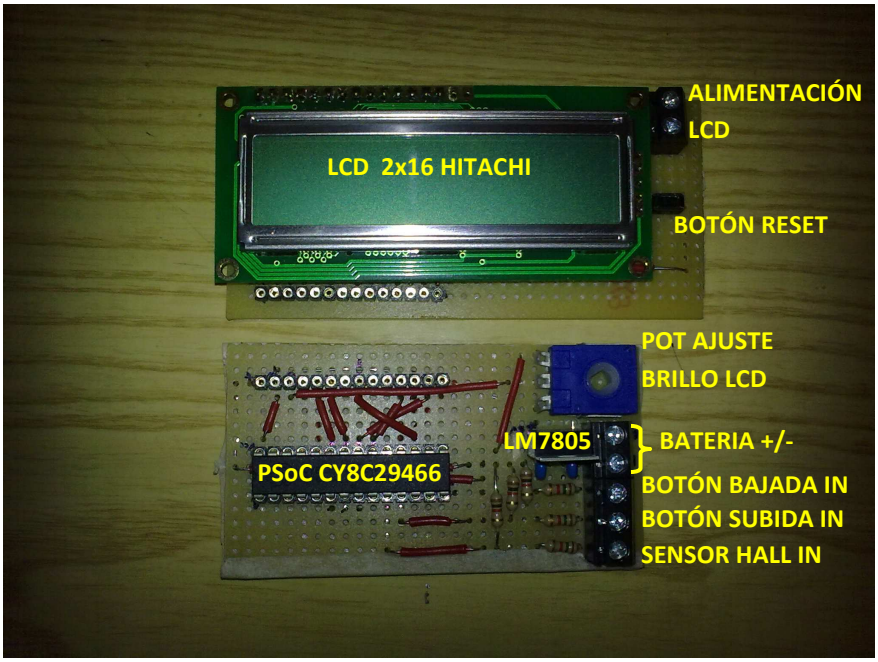


Fig. 4.40 Hardware del subsistema basado en el sensor Hall

En la Fig. 4.41 se muestra el montaje provisional del subsistema sobre la máquina cortacésped:

Fig. 4.41 Montaje del subsistema sobre la máquina para la realización de los test de funcionamiento

En la primera imagen, se puede observar como el actuador LA23 tiene uno de sus dos ejes fijo (el superior) mientras que el otro está unido al plato de cuchillas que se moverá en función del accionamiento del interruptor de subida o bajada del mismo. En la segunda imagen, se puede ver el indicador analógico que montaba la máquina cortacésped para conocer la altura del plato de corte y, en la tercera imagen, se muestra el montaje provisional del subsistema para realizar los test de funcionamiento del mismo.

En este caso, los test de funcionamiento fueron los esperados y no se produjeron errores ni contratiempos como con el subsistema ADXL. Si bien, antes de montarlo sobre la máquina ya realice simulaciones con el kit del PSoC para comprobar que en el LCD aparecieran los datos correctamente, era necesario comprobar que también funcionaba sobre la máquina cortacésped con unos resultados satisfactorios. En la Fig. 4.42, se muestran capturas de dichas simulaciones con diferentes capturas del LCD.

Fig. 4.42 Capturas del LCD en la realización de las simulaciones previas al montaje sobre la máquina

4.2.3 Sensor de llenado del cesto de recogida basado en sensor FSR

La medición del llenado del cesto de recogida de hierba de la máquina cortacésped es la que me ocasiono más dificultades debido al medio en el que se debían tomar las mediciones. Este depósito, al recibir la hierba recién cortada, tiende a ensuciarse con facilidad. En el caso de que la hierba este seca, al entrar en el depósito tiende a ocasionar polvo y si está mojada, en contacto con el agua tiende a convertirse en una sustancia ácida.

En cualquier caso, era necesaria la utilización de sensores robustos, o bien, idear un encapsulado o sistema de protección para los mismos y así evitar que se dañaran o tomaran medidas falseadas. Actualmente el cesto de recogida contaba con un pseudo sensor en forma de palanca que cuando era accionada por la hierba, una vez que el depósito estaba lo suficientemente lleno, mandaba una señal que detenía el movimiento de las cuchillas para detener el flujo de césped hacia el mismo. Esta palanca es la que se muestra en la Fig. 4.43.

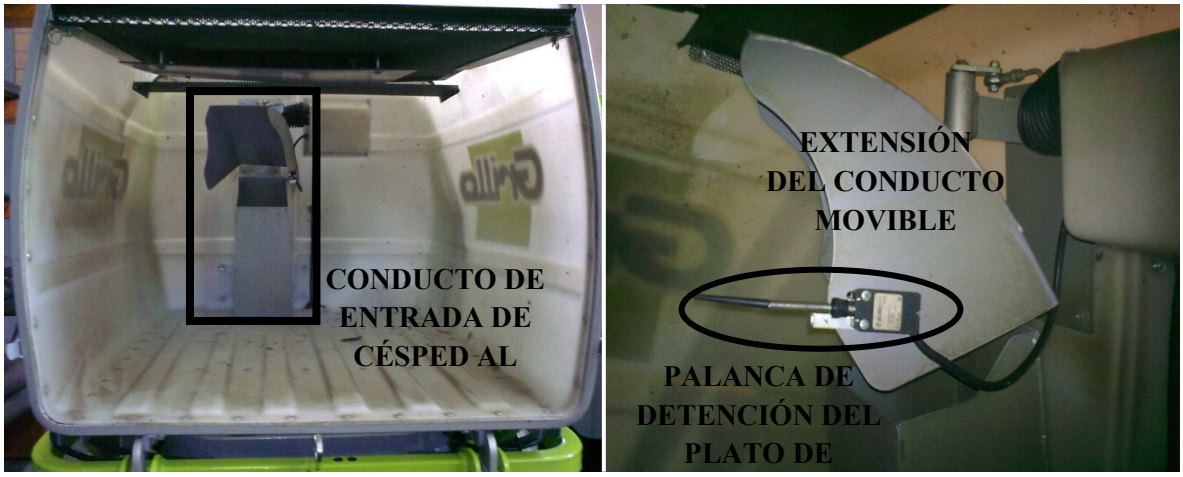


Fig 4. 43 Cesto de recogida y palanca de detención de las cuchillas del plato de corte

El dispositivo que se encarga de distribuir el césped de manera uniforme por el cesto consiste en un brazo que se encuentra al final de un tubo que comunica con el plato de las cuchillas el cual se mueve de manera constante. En uno de los laterales de este brazo se encuentra la palanca responsable de detener el acceso de la hierba al cesto. Una vez que el cesto está lo suficientemente lleno, el césped comienza a empujar hacia la salida del brazo y en un determinado momento, esta palanca se acciona.
Los sensores en los que en un primer momento pensé en utilizar para sustituir al existente eran los siguientes:

1. Sensores sensibles a la luz (LDR): Las LDR(Light Dependent Resistors) son como su nombre indican resistencias sensibles a la luz que reciben. Por lo tanto, basándome en su principio de funcionamiento y teniendo en cuenta que el cesto de recogida esta fabricado de un material translucido, la utilización de estos sensores constituían una buena posibilidad. Colocando cuatro LDR como se muestra en la Fig. 4.44 sería posible detectar el nivel de césped en el cesto a medida que se fuera llenando.

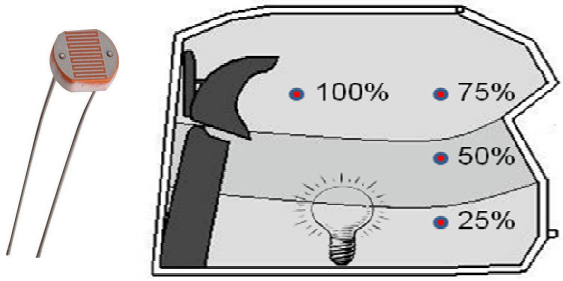


Fig. 444 Sensor LDR y recreación de su disposición en el cesto de recogida

En el caso de que la luz externa no fuera suficiente, la colocación de una bombilla o un grupo de diodos LED solventaría el problema de la luz en el interior del cesto. Sin embargo, la colocación de estos sensores sin un encapsulado, como una caja fabricada con un material transparente o de

similares características (como por ejemplo de metacrilato) en el depósito, expuesto a mucha suciedad incluso a un medio ácido en el caso de albergar césped mojado, podría deteriorar e incluso estropear los LDR dejando el sistema inservible. Debido a esta razón, descarte el uso de realizar este subsistema de control con estos sensores.

2. Alternativa a los LDR (Fotointerruptores): Este sensor creado por Sharp consiste en un emisor infrarrojo que envía señales a un receptor y que funciona de manera similar a un interruptor ya que conmutará cuando el receptor no capte la señal enviada por el emisor. Conceptualmente funcionarían de la misma manera que los LDR y requerirían de un encapsulado de protección especial y más complejo que con los comentados debido a estructura pero también lo descartaría ya que este sistema podría funcionar correctamente un número limitado de veces , ya que cuando el emisor o el receptor se obstruyeran con césped dejaría de funcionar. En la figura siguiente se muestra el sensor fotointerruptor y la idea conceptual de su montaje en el depósito.

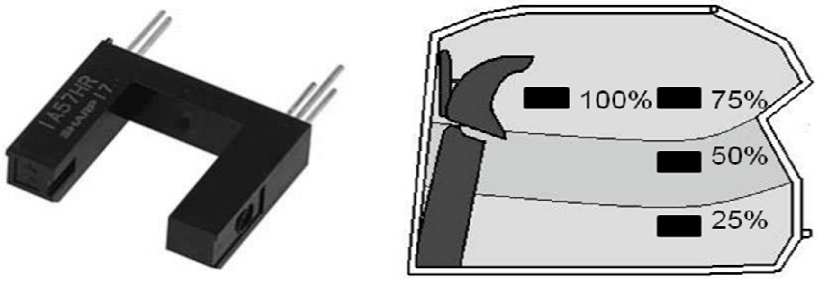


Fig. 4.45. Sensor fotointerruptor de Sharp y recreación del montaje en el cesto de recogida

3. Sensores de proximidad infrarrojos (GP2DXX): También creado por Sharp, este sensor se alejaba un poco de la idea conceptual de los 2 anteriores. Este sensor emite luz infrarroja por medio de un LED emisor de IR, que pasa a través de una lente que concentra el haz en un único rayo.
Este haz de luz seguirá en línea recta hasta que encuentre un obstáculo reflectante, que lo hará rebotar de vuelta hacia el sensor con un cierto grado de inclinación, que dependerá de la distancia. El haz rebotado será nuevamente concentrado por una lente que incidirá sobre un sensor de luz infrarroja. Este sensor lineal, dependiendo del ángulo de recepción, ofrecerá en su salida un valor de tensión proporcional al ángulo del haz de luz recibido. Como se puede observar en la Fig. 4.46, dependiendo de la distancia en la que el haz de luz infrarroja.

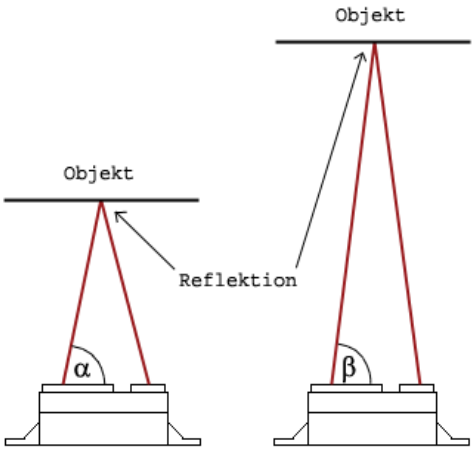


Fig. 4.46. Funcionamiento del sensor de proximidad GP2DXX

Dependiendo de la distancia en la que el haz de luz infrarroja concentrada rebotara contra un objeto reflectante, el ángulo registrado por el sensor infrarrojo sería diferente. Por ello, se presentaba como un sensor bastante preciso y no muy difícil de proteger contra el medio en el que iba a trabajar. El problema residía en la colocación de una superficie reflectante que fuera movida por el césped mientras este fuera ocupando el cesto de recogida. El sensor en cuestión es el que se puede ver en la Fig. 4.47

Fig. 4.47. Sensor GP2D12 de Sharp

Fig. 4.48 Recreación del montaje del sensor GP2D12 en el cesto del cortacésped

La necesidad de una superficie reflectante para que el sensor funcionara correctamente, me llevaba a idear un sistema mediante el cual una superficie con las características mencionadas se desplazará hacia el sensor conforme el volumen de césped en el depósito fuera creciendo. En la Fig. 4.48 se puede ver como una superficie reflectante montada sobre raíles se encuentra colocada de forma paralela al sensor que se desplazará de manera paulatina conforme el cesto se llene de césped.

A cada desplazamiento de esta superficie, el sensor arrojará una medida de tensión proporcional a su distancia real con ella, que yo procesaría mediante el PSoC, pudiendo determinar para que valores de tensión el cesto se podría considerar como lleno. En este caso, crear una carcasa de protección para el sensor no suponía un gran problema, ya que el sensor se colocaría en la pared frontal del interior del cesto y con una pequeña caja transparente quedaría lo suficientemente protegido. El problema del uso de este sensor reside en la colocación de la superficie reflectante sobre unos hipotéticos raíles que se podrían obstruir fácilmente provocando falsas lecturas del sensor y, por lo tanto, fallos en el sistema. Debido a lo comentado, descarte el uso de este tipo de sensores.

4. Resistencias sensibles a la presión: FSR (Force Sensing Resistor): Al ser este tipo de sensores los que utilicé para realizar el subsistema y para seguir la estructura de los apartados anteriores, la explicación de los mismos se realizará de la siguiente manera: A. Sensores FSR , B. Diagrama de bloques del sistema de control, C. Software del sistema y D. Desarrollo en placa de puntos, pruebas y resultados.

A. Sensores FSR: Los FSR (Fig.4.49) son unos sensores cuya resistencia disminuye conforme se incrementa la presión aplicada sobre los mismos y aunque no se puedan considerar como galgas extensiométricas ni células de carga, sus propiedades son muy similares. Estos sensores cuentan con una curva característica de fuerza-resistencia como la que se muestra en la imagen siguiente:

Fig. 4.49 Sensor FSR

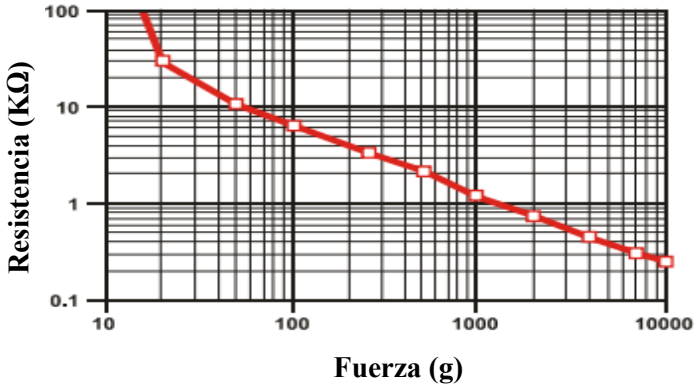


Fig 4.50 Curva característica del FSR

Como se puede ver en el diagrama, desde los 100KΩ a aproximadamente los 10KΩ la respuesta del sensor es no lineal variando bruscamente entre dichos valores y desde ese valor, la señal disminuye de manera prácticamente proporcional a la fuerza aplicada sobre el sensor (lo que equivale a una fuerza entre los 100g y los 10Kg).

El rango de medida del FSR oscila entre los 100g y los 10Kg de fuerza y la fuerza mínima de funcionamiento del mismo oscilará entre los 20 y los 100g aunque para la aplicación a desarrollar estos últimos valores no son demasiado relevantes. Tras preguntar a los ingenieros el rango de fuerza que el césped aplicaba al sensor existente sobre la máquina para activarlo y por lo tanto desencadenar el proceso de detención de las cuchillas, me comentaron que oscilaba alrededor de los 4kg de fuerza aproximadamente.

En el datasheet del FSR se incluyen recomendaciones de montaje del sensor, adaptando la siguiente al subsistema en cuestión. Consiste en colocar el sensor en un divisor de tensión y llevar su tensión resultante a un amplificador operacional. En la siguiente figura se muestra este montaje y una gráfica que relaciona la fuerza que recibe el sensor con la tensión arrojada por el

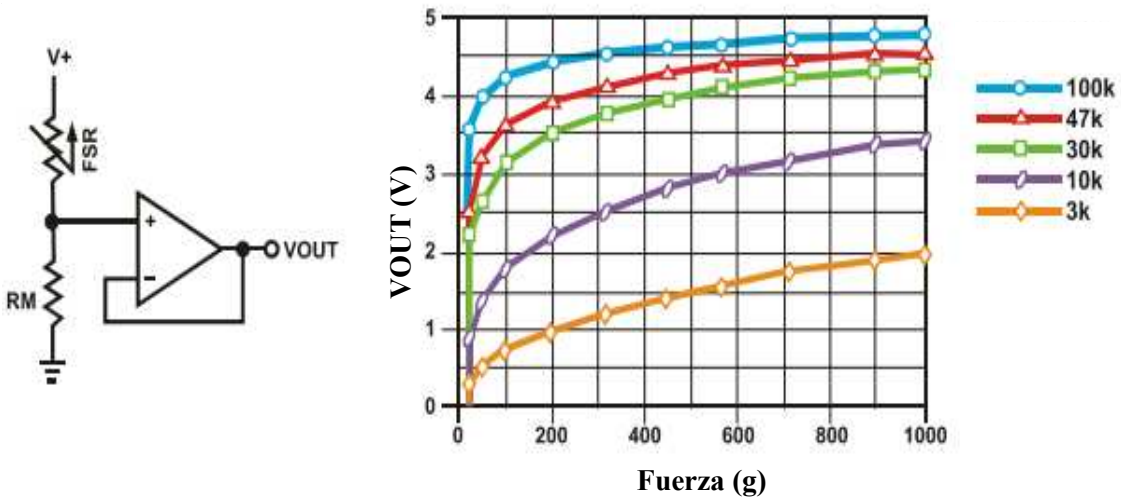


Fig 4.51. Montaje recomendado del FSR y curva que relaciona tensión, fuerza aplicada y resistencia usada en el montaje

En la gráfica se muestra la relación de la tensión de salida con la fuerza aplicada al sensor en función de la resistencia usada en el divisor de tensión. El voltaje de salida variará dependiendo del valor de las mismas, si bien el valor de la resistencia es bajo (3KΩ, 10KΩ) ofrecerá cambios fácilmente apreciables conforme aumente la presión mientras que si la resistencia aumenta de valor (30KΩ, 47KΩ o 100KΩ) se obtendrán valores mucho más lineales pero con cambios poco apreciables.

Como la precisión necesaria para este sistema no debe ser especialmente precisa, decidí escoger un valor de 10KΩ para realizar el montaje del FSR, ya que cuenta con una mayor variación de tensiones para los diferentes valores de presión. Por tanto, montando el FSR en un divisor de tensión y llevándolo a una de las entradas analógicas del PSoC podría realizar el control del llenado del cesto de una manera sencilla.

En el siguiente apartado mostrare el diagrama de bloques correspondiente al subsistema basado en este sensor, asi como una explicación del mismo.

B. Diagrama de bloques del sistema de control

El sistema de control del sensor FSR será el siguiente:

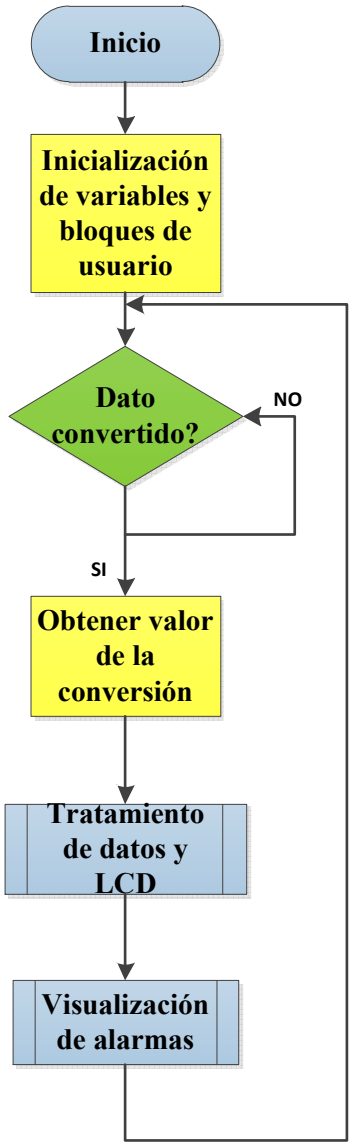


Fig 4.52 Diagrama de bloques del susbsistema del FSR

El diagrama de bloques del sistema de control del FSR es idéntico al diagrama del sensor ADXL. En este caso la diferencia reside, en la necesidad de la conversión de una sola señal analógica para su procesamiento a través del PSoC. Mediante uno de los conversores AD incluidos en el mismo, convertiría la variable analógica en datos digitales para poder trabajar con ellos y mostrar, en la realización de los test previos al montaje del circuito sobre la máquina, los resultados obtenidos en el LCD al ejercer diversos grados de presión sobre la superficie del sensor FSR. Como se puede ver en la figura 4.52 los pasos realizados por el subsistema serán los mismos que para el caso del ADXL, variando relativamente las subrutinas.

Tras inicializar las variables el sistema entrara en un bucle infinito en el que continuamente se convertirán los datos arrojados por el FSR y procesados mediante las subrutinas. La primera de ellas (fig. 4.53) será la encargada de, con los datos convertidos ya en formato digital, proporcionar un nivel de llenado a dichos datos y mostrarlo por el LCD (como ya he comentado, solo para realizar los tests) Como podemos ver en la siguiente figura se muestra la subrutina de tratamiento de datos y LCD que será comentada a continuación:

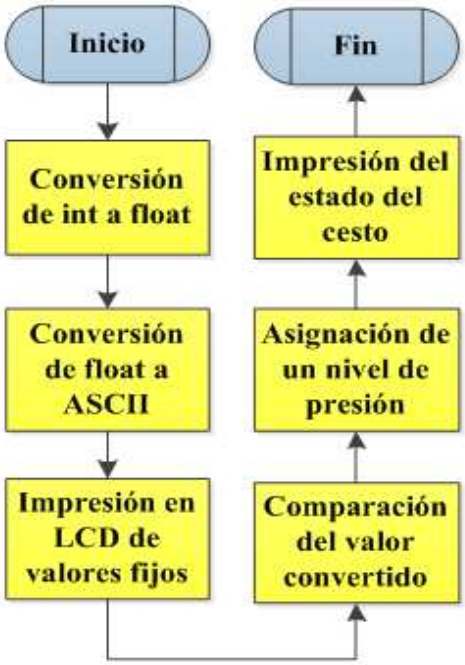


Fig. 4.53 Diagrama de bloques de la subrutina tratamiento de señales y LCD

Al igual que con la subrutina del sistema ADXL, la conversión de los datos digitales a los formatos necesarios para su procesamiento son iguales. Posteriormente se escribe en el LCD los datos que aparecerán constantemente y, después, con los datos convertidos se realizara una comparación con valores medidos previamente para conocer el nivel de llenado del cesto de recogida. Una vez conocido el valor del nivel de llenado, se imprimirá en pantalla y se repetirá de nuevo el proceso.

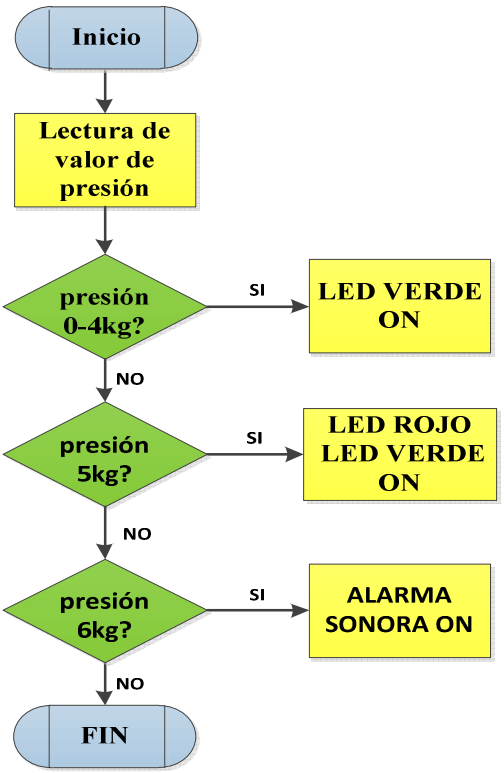


Fig 4.54 Diagrama de bloques de la subrutina de alarmas del susbsistema del FSR

En la figura 4.54 se muestra la subrutina de alarmas del subsistema. Al utilizar el mismo hardware que con el sistema ADXL, el diagrama de bloques y el funcionamiento del mismo será similar.

Una vez conocido el valor de presión actual registrado por el sensor FSR, se comparará con diferentes valores de presión y dependiendo del resultado de esta comparación la respuesta de la alarma será diferente.

Los ingenieros de la empresa sabían que el sensor antiguo se activaba cuando era sometido a una presión de aproximadamente 6 kilos de fuerza. Por lo tanto, la escala de alarmas se basaría en ese valor. Para valores de presión comprendidos entre 0 y 4 kilos de fuerza, el subsistema devolvería una respuesta de normalidad (LED verde).

En el caso de una fuerza de 5 kilos, el sistema mostraría encendería el LED naranja correspondiente al nivel de precaución y finalmente si se detectaran 6 kilos de fuerza, se activaría la alarma sonora y se detendría la entrada de césped en el cesto.

C. Software del sistema.

Para mostrar la configuración del software del sistema realizado en PSoC Designer, tomaré una serie de capturas de las ventanas de dicho programa. Tomando como referencia nuevamente la figura 4.9 y siguiendo la dinámica de los subsistemas anteriores, procederé a comentar las configuraciones realizadas en este sistema de control del sensor FSR:

1. Parámetros Globales del PSoC

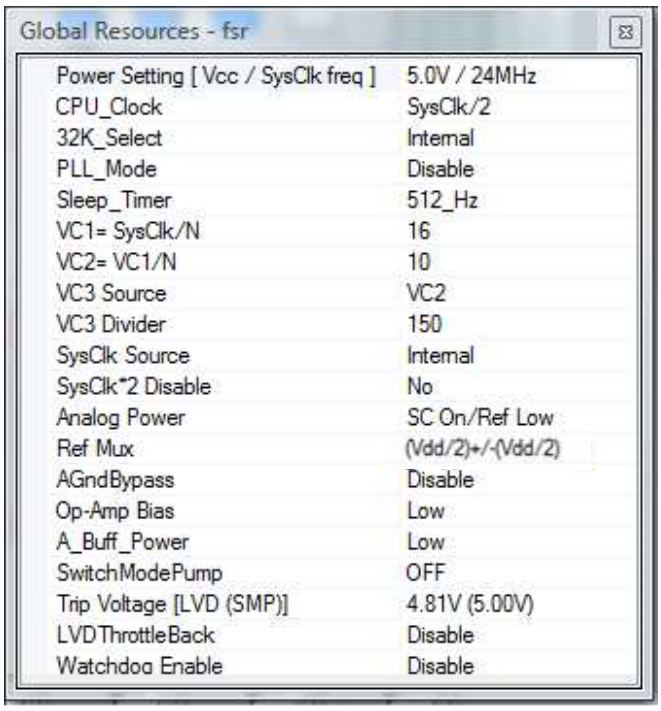


Fig. 4.55 Parámetros globales del subsistema FSR

Los parámetros globales del PSoC para este subsistema serán los siguientes:

Tensión de alimentación y frecuencia de SysClk: Elegiré, como en los sistemas anteriores, una tensión de alimentación de 5V y una frecuencia de 24MHz.

Frecuencia de reloj de la CPU: La frecuencia de reloj de la CPU, para realizar las pruebas será de 12MHz.

Fuente de SysClk: No usaré un cristal externo, por lo que la fuente de SysClk será la propia del PSoC.

Deshabilitar SysClk*2: Esta función la mantendré deshabilitada nuevamente.

Referencia analógica: Con esta opción seleccionamos el rango y la precisión de los bloques analógicos. Elegiré la opción (Vdd/2) / (-Vdd/2).

Frecuencias de trabajo VC1, VC2 y VC3: Los valores de estas frecuencias de trabajo coinciden con las calculadas para el subsistema ADXL:

VC1= SysClk/N. Elegiré N=16 obteniendo VC1= 1.5 MHz

VC2=VC1/N. Eligiendo N=10 obtendré VC2= 150 KHz

Siendo VC3=VC2/N y eligiendo N=150, obtendremos VC3=1KHz (1ms)

Switch mode Pump: Esta opción la mantendré deshabilitada nuevamente.

Detección de bajo voltaje: Esta opción no será relevante para la realización de las pruebas del sistema. Mantendré la opción utilizada en el subsistema ADXL.

Watchdog Enable: Al no requerir de perro guardián, la opción la mantendré desactivada.

Las demás opciones han sido descritas en el apartado 3.3.2.1, y su configuración se muestra en la fig. 4.55

2. *Mapa general del dispositivo. Bloques analógicos y digitales. Configuración*

En la Fig. 4.56 se muestra, como en los subsistemas anteriores una captura en PSoC Designer del mapa general de la configuración de módulos, donde quedan resaltadas las filas y las columnas de bloques digitales y analógicas, respectivamente:

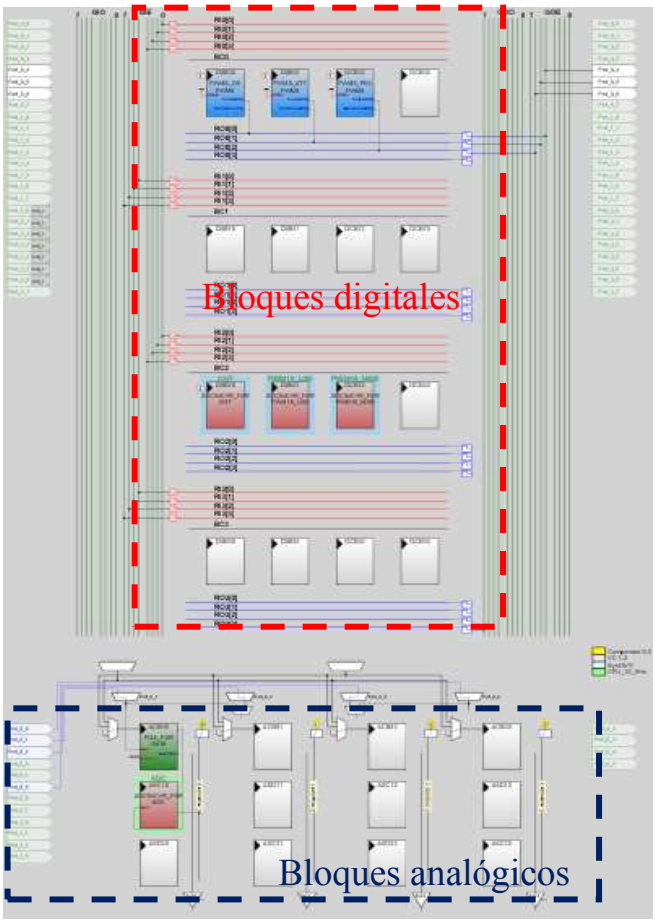


Fig. 4.56 Mapa de configuración de hardware del CY8C29466-24PXI

- Bloques analógicos

El bloque analógico usado en este subsistema será un conversor AD que utilizaré para recibir la señal analógica proveniente del divisor de tensión del que forma parte el FSR que será conectado a un pin del PSoC. En la siguiente figura se muestra su disposición en PSoC Designer:

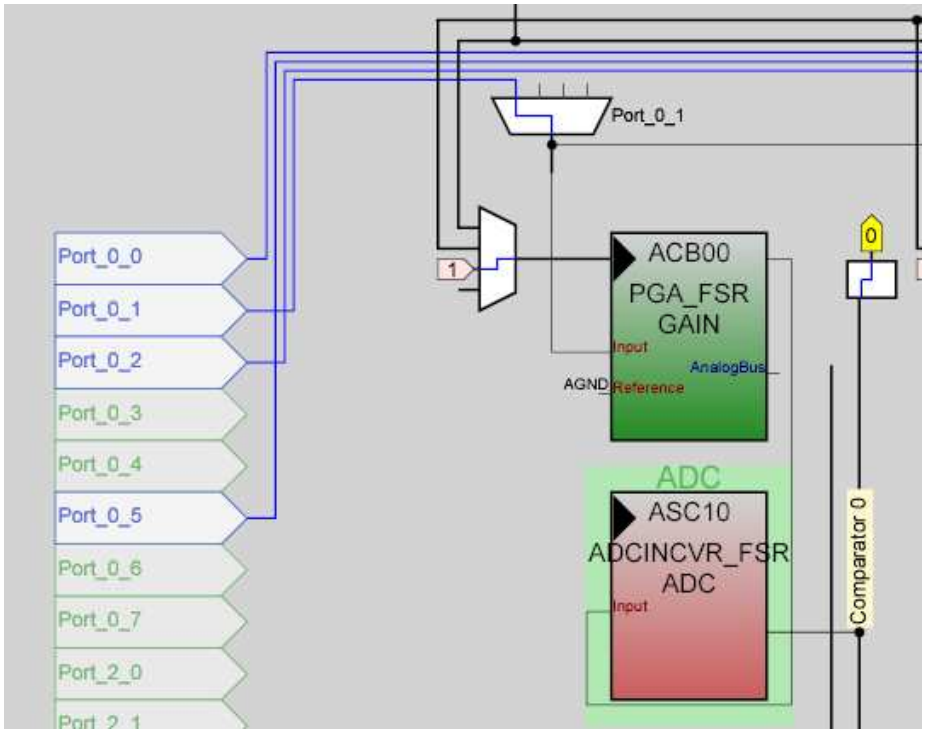


Fig. 4.57 Disposición de los bloques analógicos

Se puede ver la parte analógica del conversor AD (bloque rojo), que recibirá la señal del FSR a través de un amplificador de ganancia programable de valor 1 (bloque verde) (ya que como he comentado anteriormente, no es posible conectar directamente la señal del sensor al conversor AD), al cual le introduciré la señal del sensor por el puerto P0_1.

Por otro lado, la parte digital del conversor AD estará formada como ya he comentado por 3 bloques (un contador y un PWM de 16 bits que ocupa dos bloques) digitales los cuales no tendrán entradas ni salidas configurables, ya que únicamente forman parte del bloque conversor AD. El usuario únicamente podrá configurar las variables que mostraré a continuación. Una captura de pantalla de PSoC Designer donde se muestra la disposición de los bloques mencionados es la siguiente:

Fig. 4.58 Disposición de los bloques digitales de los conversores AD

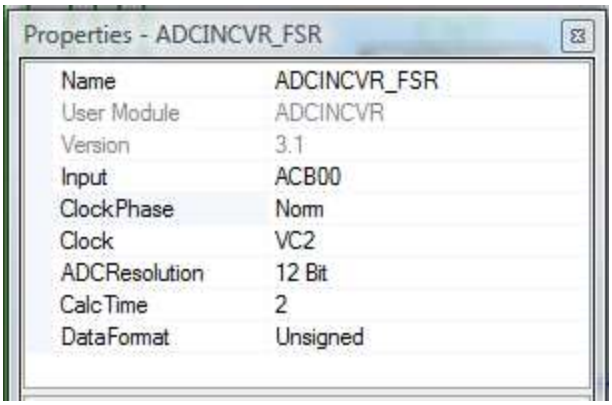


Fig. 4.59 Propiedades del conversor AD

Las variables del conversor AD configurables por el usuario, y que coincidirán con las elegidas para el subsistema basado en el ADXL, serán las siguientes:

Entrada: Proveniente del PGA alojado en el bloque de tiempo continuo ACB00, y este a su vez ruteado al pin P0_1.

Fuente de reloj: Elegiré V2, que tiene un valor de 150 KHz o 6.66µs.

Resolución del conversor AD: Una resolución de 12 bits será adecuada.

CalcTime: Como ya he comentado anteriormente, este valor se calculara de la siguiente forma:

$$CalcTime \geq \frac{DataClock * 180}{CPUClock} = \frac{150KHz * 180}{24MHz} = 1.125$$

Por lo que elegiré calc valor 2. Entonces el valor de muestreo será el siguiente:

$$SampleRate = \frac{DataClock}{2^{bits+2} + CalcTime} = \frac{150K}{2^{14} + 2} = 9.1\ muestras/segundo$$

Obtendremos asi una velocidad de 9 muestras por segundo, más que suficiente para nuestro sistema.

Formato de la conversión: Ya que trabajaremos con valores que siempre serán positivos, el valor de la conversión lo configuraremos sin signo.

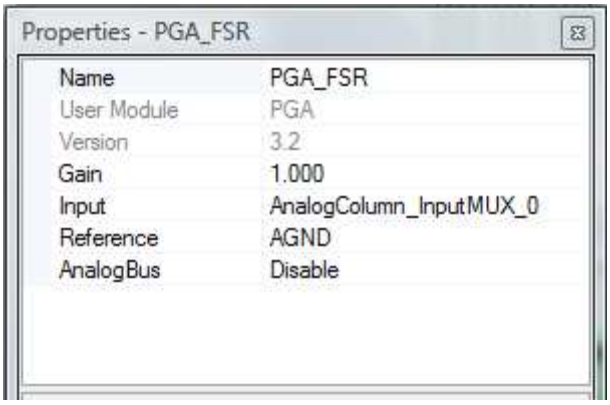


Fig. 4.60 Propiedades del Amplificador de ganancia programable

La configuración del PGA o amplificador de ganancia programable que también coincidirá con la realizada anteriormente en el susbistema ADXL será la siguiente:

Ganancia: Nuevamente, elegiremos ganancia 1 para no perturbar la señal recibida.

Entrada: Columna analógica del mux 0, ruteada a su vez con el puerto P0.1.

Referencia: Analog Ground. Tomaremos como referencia la masa analógica del sistema.

De esta forma, quedarán configurados los bloques analógicos de subsistema FSR.

- Bloques digitales

Para realizar los test pertinentes al subsistema basado en el FSR, además de los comentados bloques digitales pertenecientes al conversor AD, incluiré nuevamente 3 bloques PWM de 8 bits que se encargarán de activar las alarmas dependiendo de los niveles recibidos por el sensor. La disposición de los bloques es la misma que la del sistema ADXL como se puede apreciar en la figura 4.61:

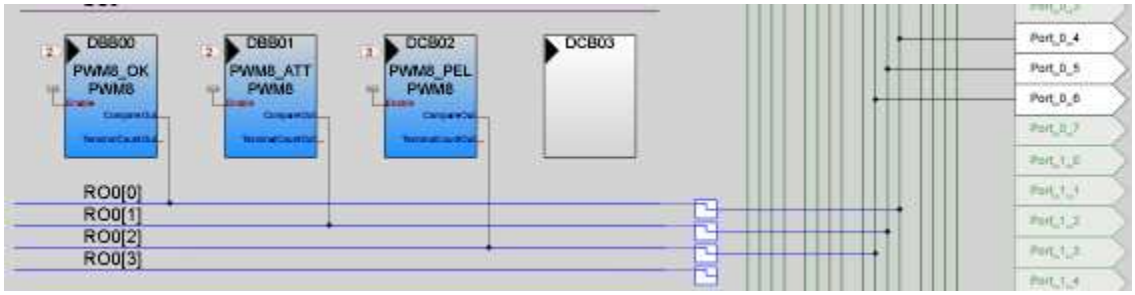


Fig. 4.61 Disposición de los bloques digitales

Como ya comenté anteriormente, cada uno de estos tres bloques estará conectado a uno de los pines de salida del PSoC, con una configuración Strong (explicado en el apartado 3.2.3). Dos de estos PWM activarán/desactivarán el diodo LED de doble ánodo, llamados PWM8_OK y PWM_ATT y el restante, PWM8_PEL se encargará de la activación/desactivación del piezorresistivo. Al no variar la configuración de estos bloques con respecto al subsistema ADXL obviare su explicación.

3. Situación final de los pines del PSoC (vista del integrado) y código C.

Como ya he mostrado en los capítulos anteriores, la situación final de los pines del PSoC se encuentra en la siguiente imagen. Igualmente, el pin verde corresponde con la entrada analógica, por la que recibirá la señal del FSR, los pines rojos son los correspondientes a las salidas digitales que se encargarán de activar las alarmas y los pines azules, que permanecerán inactivos y configurados por defecto como estándar de la CPU y con alta impedancia.
El código C del subsistema, se encuentra en el anexo correspondiente.

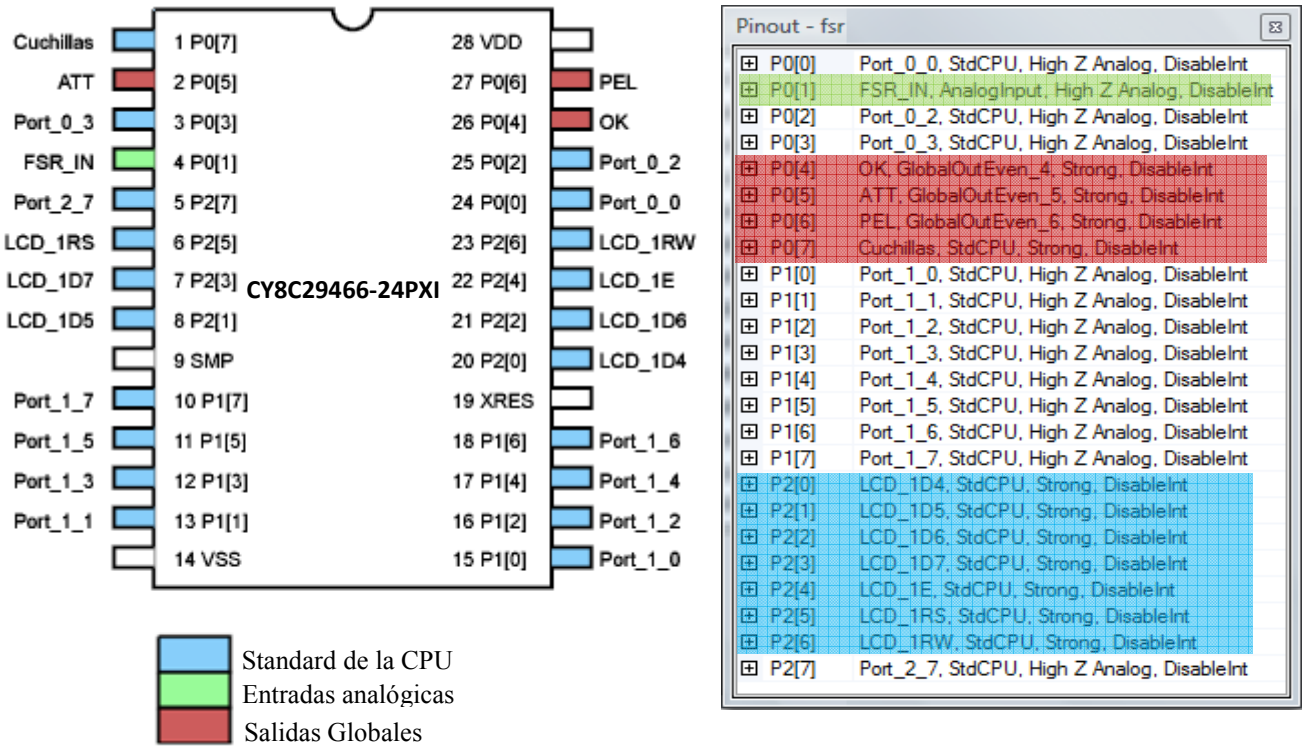


Fig. 4.62 Utilización de los pines del PSoC y configuración de los mismos

D. Desarrollo en placa de puntos, pruebas y resultados

Una vez más, tras comprobar el funcionamiento del software del subsistema sobre el kit de evaluación, procedí a la realización del prototipo sobre placa de puntos para realizar los test pertinentes sobre la máquina cortacésped.
Como en el caso del subsistema basado en el ADXL, no requerirá de los datos arrojados por el sistema sobre el LCD y por ello no debería aparecer en el diseño del mismo, pero para comprobar más fácilmente la fuerza que ejercía el césped sobre el sensor cuando estaba en funcionamiento, decidí incluir los pines asociados al LCD.

Como la placa de puntos iba a estar fuera del cesto de recogida del césped (únicamente el FSR se colocaría dentro del cesto) el LCD lo montaría sobre la placa de puntos y asi ver las lecturas del sensor fácilmente.

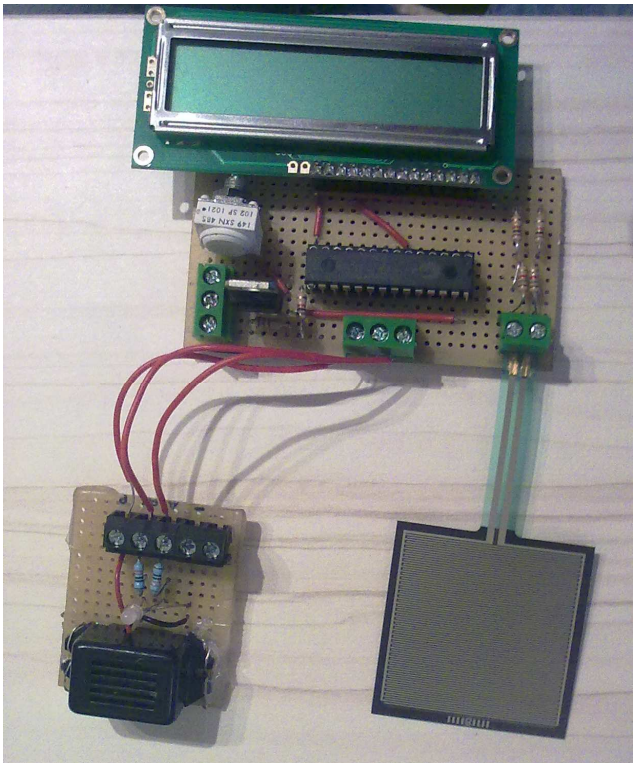


Fig 4.63. Montaje de comprobación del funcionamiento del hardware basado en el sensor FSR

Como se puede ver en la fig. 4.63, se muestra el montaje de la placa de puntos para realizar las comprobaciones pertinentes previas al montaje sobre la máquina. Junto a la borna de alimentación a 12V, se encuentra un regulador de tensión LM7805, que será la alimentación del PSoC.

También se pueden ver las borna correspondiente a las alarmas (las mismas que las utilizadas en el subsistema ADXL) y la correspondiente con la que se unirá el FSR. Como ya he comentado, esta placa albergará el LCD, la cual también incluye el potenciómetro para regular la intensidad del mismo.

Las pruebas realizadas sobre la misma fueron satisfactorias ya que los datos que arrojaba el sensor y que posteriormente procesaba el PSoC ofrecían datos consecuentes a las presiones ejercidas sobre el FSR.

Fig 4.64 FSR protegido para realizar las pruebas sobre la máquina

Como he comentado anteriormente, el medio en el que trabajará el sensor, tanto por acidez como por suciedad, debía ser protegido para evitar que se dañara.

De esta manera, plastifiqué el FSR, le añadí una placa de metacrilato de base, añadí sendos cables a sus terminales y plastifiqué el conjunto, considerando que así estaría lo suficientemente protegido del medio de trabajo.

Tras colocar el sensor donde se encontraba el anterior y realizar las pruebas sobre la máquina, el sensor no sufrió daño alguno y las lecturas realizadas por el mismo fueron correctas.

4.3 Comunicación de datos a PC por puerto serie (UART)

Para añadir al sistema final la posibilidad de enviar datos a través de un puerto serie a un PC, de datos recogidos durante su funcionamiento que puedan ser de interés para el usuario de la máquina, diseñé un pequeño programa para aprender a utilizar tanto el hardware del bloque UART del que dispone el PSoC así como su configuración en PSoC Designer.

El sensor que utilizaría para realizar esta prueba seria el FSR por ser el que cuenta con una mayor facilidad de configuración. La idea previa a comenzar el diseño, consistiría en conseguir una comunicación serie bidireccional entre el PSoC y el PC a través del puerto serie en la que al comenzar la comunicación el PSoC escribiera al hiperterminal del PC un mensaje de bienvenida con las opciones que el usuario podría realizar y, a la vez, un mensaje al LCD con la situación en la que se encuentra el sistema.

Como más adelante se podrá visualizar en el código C realizado para el control del hardware que también será explicado posteriormente, las opciones que el usuario podrá visualizar en el mensaje de bienvenida en el hiperterminal serán las siguientes:

1. presionando la f → El sistema proporcionará datos arrojados por el sensor FSR. Para realizar las pruebas será la única opción configurada totalmente.

2. presionando la h → El sistema proporcionará datos arrojados por el sensor Hall.

3. presionando la a → El sistema proporcionará datos arrojados por el sensor ADXL.

Una vez dentro de una de estas 3 opciones principales el usuario deberá elegir si desea tomar y enviar una sola muestra o si, en cambio, prefiere realizar un muestreo continuo tomando y enviando muestras al hiperterminal. Para realizar este muestreo de los datos arrojados por el sensor también debí configurar una interrupción para dicha tarea mediante un bloque PWM y un TIMER cuyas configuraciones explicaré a continuación. La intención de incluir estos bloques no es otra que determinar el intervalo de tiempo que transcurrirá entre el envio de un dato y el siguiente.

Para seguir la estructura de los apartados anteriores, la configuración del sistema será la siguiente: A. Hardware USB a RS232 , B. Diagrama de bloques del sistema de control, C. Software del sistema y D. Comprobación de funcionamiento, pruebas y resultados.

A. Hardware USB a RS232: Para realizar la conexión de la placa de pruebas del PSoC (CY3210) con el PC, adquirí un conversor de puerto USB a puerto DB9 que incluía el integrado RS232 para comunicaciones serie. En la siguiente figura se muestra el conversor conectado a la placa de evaluación del PSoC:

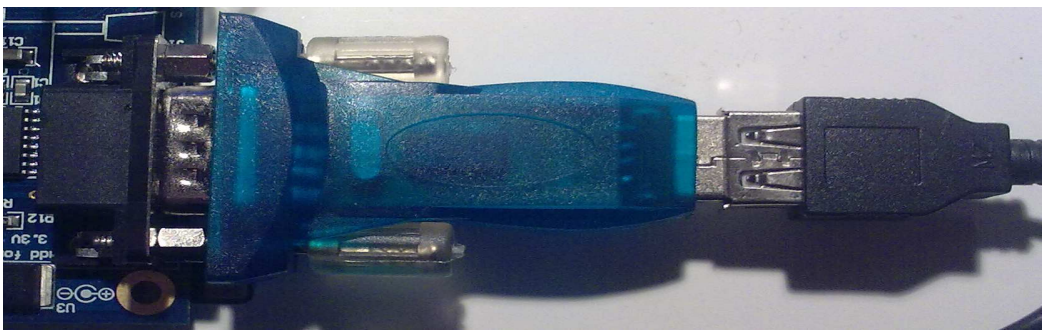


Fig. 4.65. Conversor de puerto USB a DB9 con RS232

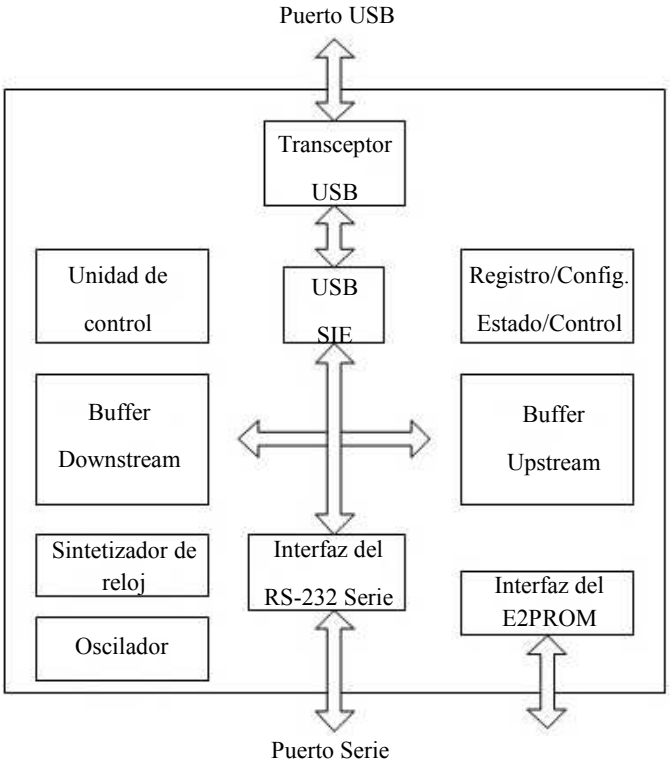


Fig. 4.66. Diagrama de bloques del conversor USB a DB9

El diagrama de bloques correspondiente con el conversor es el que podemos ver en la fig. 4.66. Además de los bloques internos que incluye tales como la unidad de control, el registro de estado/control, un sintetizador de reloj, un oscilador y un interfaz de E2PROM podemos ver como entre el puerto USB y el puerto serie con conector DB9 existen los siguientes:

Transceptor USB: Se encarga de recibir los datos recibidos a través del puerto USB y enviarlos al bloque USB SIE (siglas en inglés Serial Interface Engine, o motor de interfaz serie).

USB SIE: Este bloque se encargará de realizar la conversión de los datos recibidos por el USB y enviados al DB9 y viceversa.

Interfaz del RS-232 Serie: Mediante este bloque se transformarán los datos para ser transmitidos por el puerto serie, o bien, para ser conducidos al USB SIE.

Buffers: El conversor también dispone de dos buffers, uno para cada sentido de la conversión.

Para poder utilizar este conversor es necesaria su configuración en el PC. Una vez instalados los drivers del conversor, debía configurar el puerto serie del PC para conseguir una comunicación bidireccional. Para ello, me dirigí al administrador de dispositivos y comprobé en que puerto COM se encontraba. Después, debía crear una nueva conexión (fig. 4.67) y configurar los parámetros de, en este caso, el COM2:

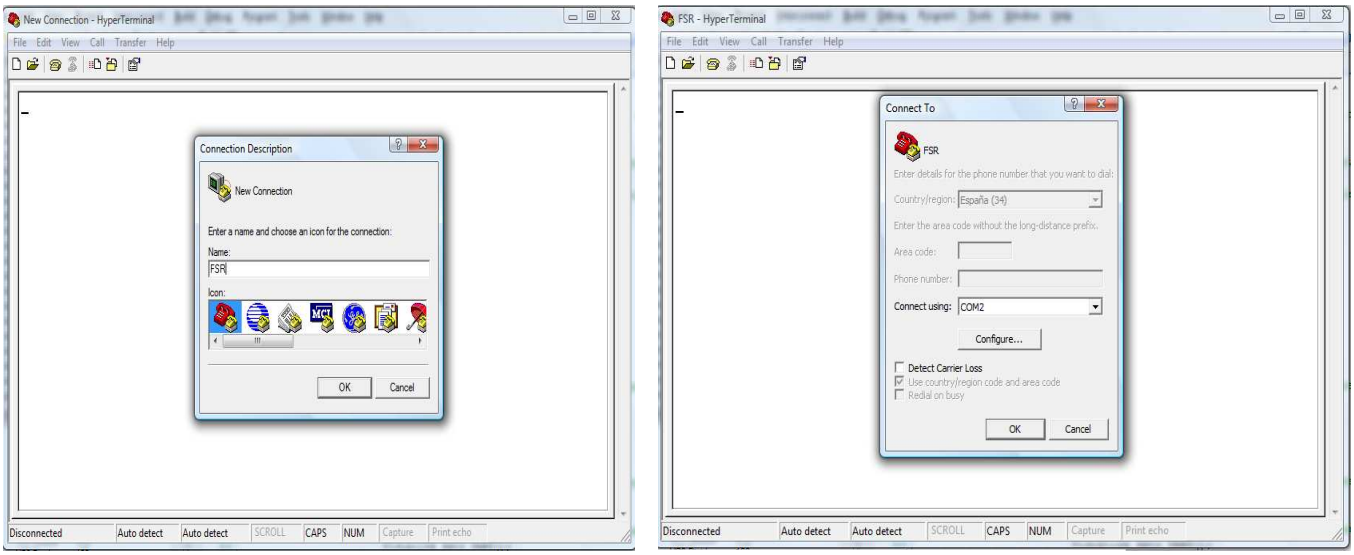


Fig. 4.67. Creación de una nueva conexión al hiperterminal

Una vez elegido el puerto COM2, procedí a la configuración de los parámetros del mismo como se puede ver a continuación (fig.4.68):

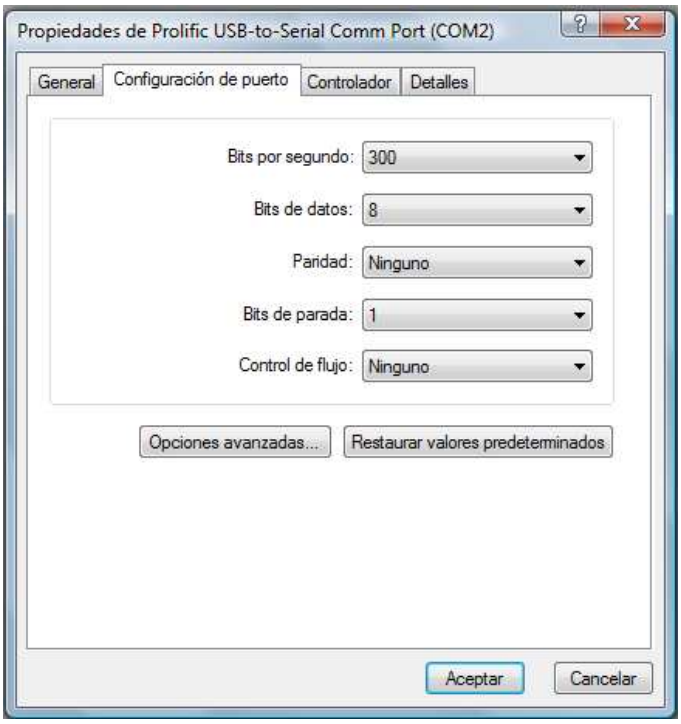


Fig. 4.68. Configuración del puerto COM2

Bits por segundo: La velocidad máxima de transferencia realizada por el sistema coincidirá con la opción del envío de una muestra por segundo a través de un TIMER que provocará una interrupción. Por lo tanto, la velocidad de comunicación serie será baja y será fijada a 300bps

Bits de datos: Para realizar el test del UART, sólo incluí el sensor FSR que configuré mediante un conversor A/D de 8 bits. Por tanto, se transferirán 8 bits por transmisión

Paridad: No requeriré de control de paridad. Por tanto, ninguno.

Bits de parada: Elegiremos un bit como indicador de fin de byte.

Control de flujo: No usaré ningún control de flujo. Por tanto, ninguno

Una vez configuradas las características del puerto COM, deberemos modificar una última propiedad de la conexión con el hiperterminal. En la ventana del hiperterminal, accederemos a través de FILE→PROPERTIES→SETTINGS→ASCII SETUP y, en ese sub menú, marcaremos la opción "Echo typed characters localy" ("Eco de los caracteres escritos localmente") para poder visualizar en el hiperterminal los caracteres que le envíe mediante el PSoC (fig. 4.69.).

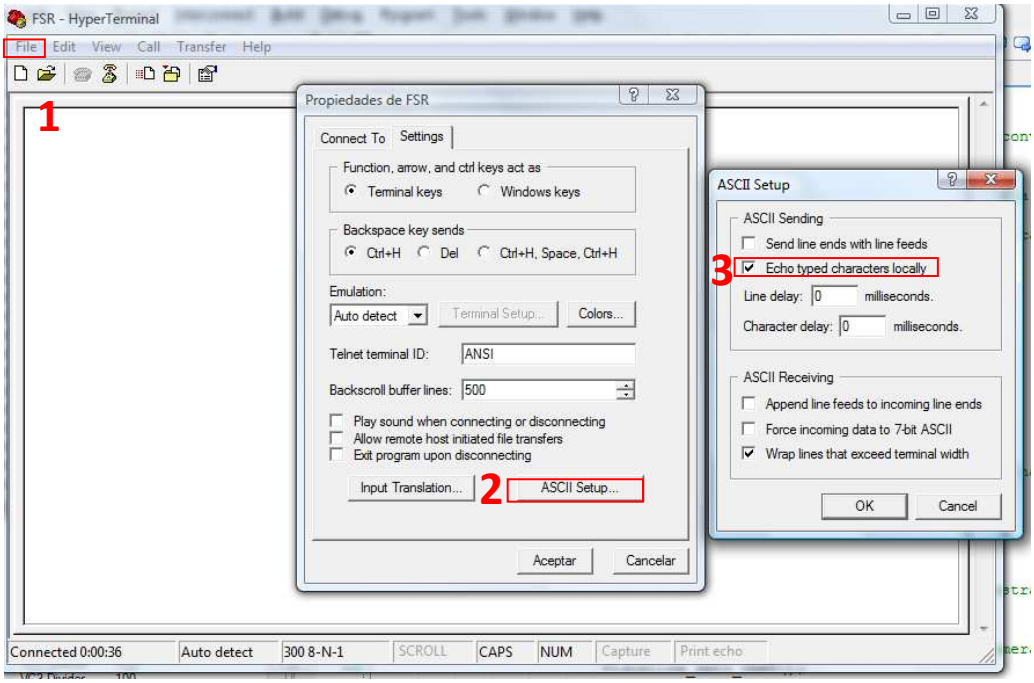


Fig. 4.69. Configuración del puerto COM2 para visualizar caracteres enviados al hiperterminal

B. Diagrama de bloques del sistema de control: El diagrama de bloques correspondiente con el sistema realizado para conocer y comprobar el funcionamiento del bloque UART es el siguiente:

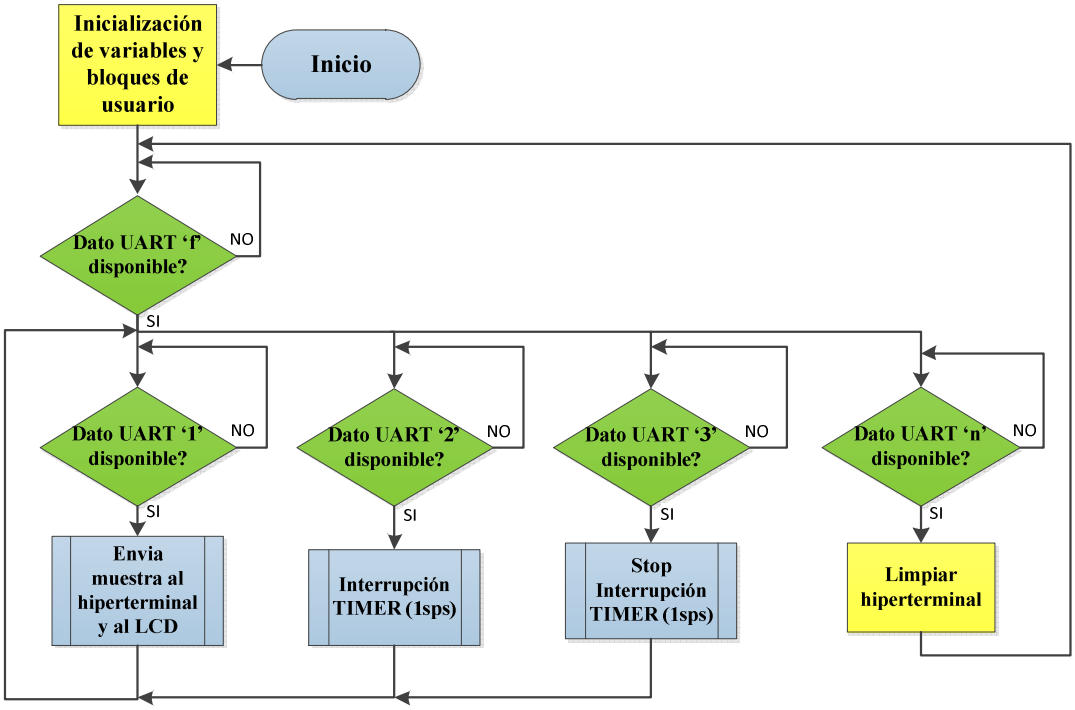


Fig. 4.70. Diagrama de bloques del sistema de control del UART

Como se muestra en la figura 4.70, tras la inicialización de tanto las variables del sistema como los bloques de usuario involucrados en el mismo, tras escribir el mensaje de inicio tanto en el LCD como en el hiperterminal, este esperará hasta recibir el carácter 'f' (como he comentado anteriormente, para comprobar el funcionamiento del UART, solo implementé en el sistema el sensor FSR pero fue realizado con vistas a poder implementar los otros sensores, y por eso, sólo programé las opciones relativas a dicho sensor) ya que en el caso de recibir un carácter diferente aparecerá un mensaje en el hiperterminal pidiendo la introducción de los caracteres permitidos. Una vez lo haya recibido, aparecerá un nuevo submenú en el hiperterminal en el que ofrecerá al usuario las opciones de obtener datos relativos al sensor FSR:

- 1- El sistema enviará al hiperterminal una sola muestra en hexadecimal del sensor FSR y su valor analógico en V al LCD. Permanecerá en ese submenú a la espera de recibir otro dato.
- 2- Al recibir este valor, el sistema activara la interrupción del Timer de 8 bits que incorpora, accediendo a ella y enviando tanto al hiperterminal como al LCD una muestra por segundo. Enviará una muestra por segundo mientras no se reciba un '3' que detendrá la entrada en la interrupción.
- 3-Cuando se reciba, desactivará la entrada en la interrupción del Timer deteniendo el envio por segundo de un dato tanto al hiperterminal como al LCD.
- n-Si se recibe este dato, se borrará el hiperterminal y se mantendrá el sistema a la espera de recibir de nuevo el sensor a controlar(en este caso, se volverá a esperar el carácter 'f')

El contenido de las subrutinas se puede consultar en el anexo correspondiente a códigos C.

C. Software del sistema

Nuevamente mostraré la configuración, basada en la figura 4.9, del software en este caso del ejemplo basado en el UART. La configuración de los parámetros y bloques tanto digitales como analógicos del PSoC serán los siguientes:

1. Parámetros Globales del PSoC

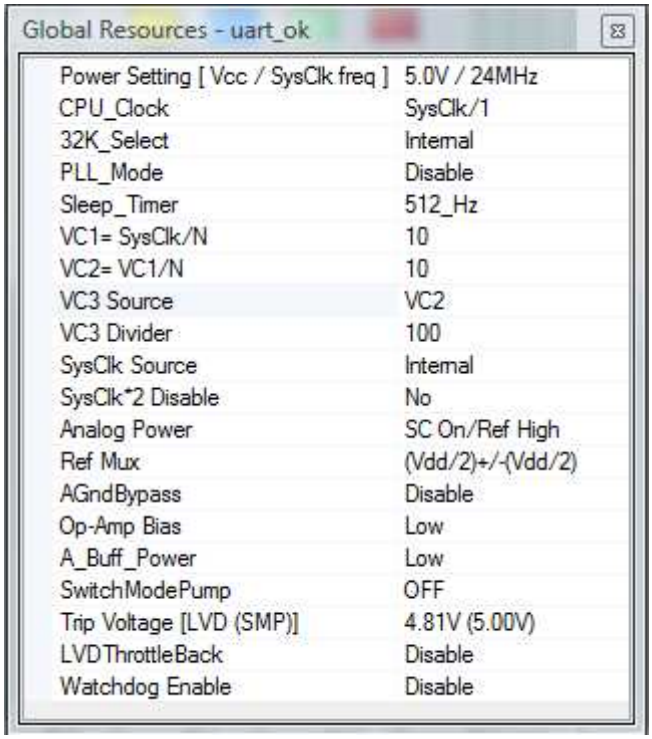


Fig. 4.71 Parámetros globales del test UART

Los parámetros globales del PSoC para este subsistema serán los siguientes:

Tensión de alimentación y frecuencia de SysClk: Como en sistemas anteriores, una tensión de alimentación de 5V y una frecuencia de 24MHz.

Frecuencia de reloj de la CPU: La frecuencia de reloj de la CPU elegida en este caso serán los 24 MHz.

Fuente de SysClk: No usaré un cristal externo, por lo que la fuente de SysClk será la propia del PSoC.

Deshabilitar SysClk*2: Esta función la mantendré deshabilitada nuevamente.

Referencia analógica: Con esta opción seleccionamos el rango y la precisión de los bloques analógicos. Elegiré la opción (Vdd/2) / (-Vdd/2).

Frecuencias de trabajo VC1, VC2 y VC3: Como ya he comentado, estos valores representarán las fuentes de reloj de las que dispondrá el sistema, pudiendo elegirse indistintamente:

VC1= SysClk/N. Elegiré N=10 obteniendo VC1= 2.4 MHz

VC2=VC1/N. Eligiendo N=10 obtendré VC2= 240 KHz

Siendo VC3=VC2/N y eligiendo N=100, obtendremos VC3=2.4KHz (0.416 ms)

Switch mode Pump: Esta opción la mantendré deshabilitada nuevamente.

Detección de bajo voltaje: Esta opción no será relevante para la realización de las pruebas del sistema. Mantendré la opción utilizada en los subsistemas anteriores.

Watchdog Enable: Al no requerir de perro guardián, la opción la mantendré desactivada.

Las demás opciones han sido descritas en el apartado 3.3.2.1, y su configuración se muestra en la fig. 4.71

2. *Mapa general del dispositivo. Bloques analógicos y digitales. Configuración*

En la Fig. 4.72 se muestra, como en los subsistemas anteriores una captura en PSoC Designer del mapa general de la configuración de módulos, donde quedan resaltadas las filas y las columnas de bloques digitales y analógicas, respectivamente:

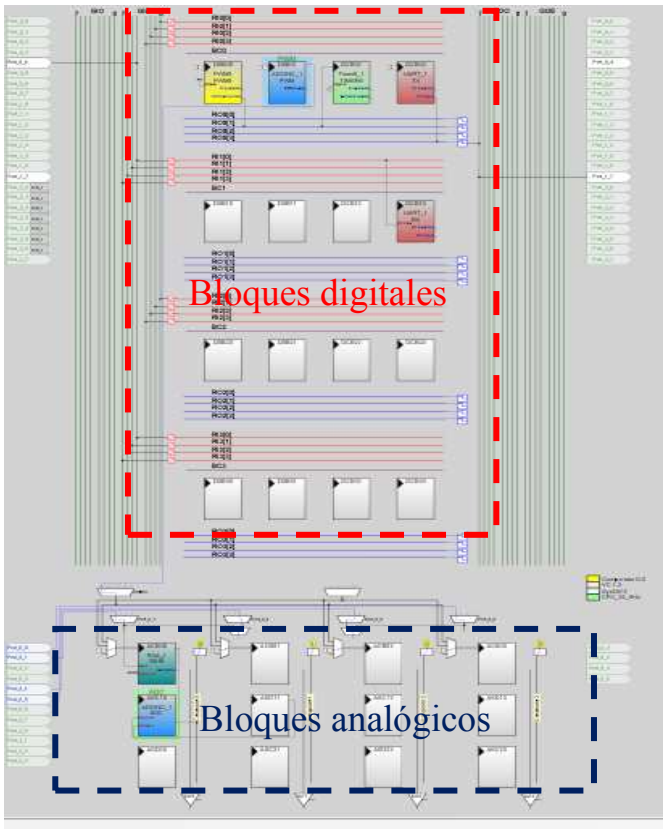


Fig. 4.72 Mapa de configuración de hardware del CY8C29466-24PXI

- Bloques analógicos

A diferencia de los sistemas anteriores, el bloque conversor AD que introduje no era de resolución variable, con vistas al sistema final, ya que el consumo de bloques digitales de este conversor es elevado ("2" bloques analógicos y 3 bloques digitales), eligiendo un conversor AD incremental simple. La señal analógica que recibiría sería la proveniente del divisor de tensión del que forma parte el FSR para realizar las pruebas pertinentes con el UART. En la siguiente figura se muestra su disposición en PSoC Designer:

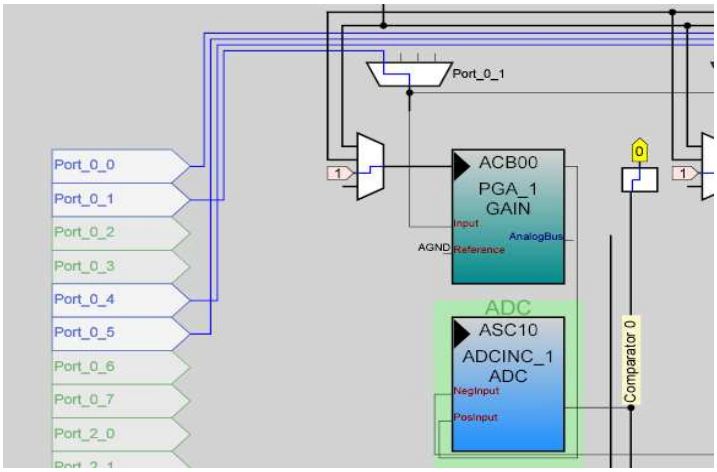


Fig. 4.73 Disposición de los bloques analógicos

De igual forma que con el conversor AD incremental de resolución variable, se necesita de un amplificador de ganancia programable (PGA) con ganancia igual a uno para conducir la señal analógica del FSR desde el pin de entrada al PSoC (P0_1) hasta la entrada del conversor AD.

La diferencia del espacio ocupado entre el ADCINCVR y el ADCINC es evidente. Mientras que el primero incluye 3 bloques digitales (uno para el contador y dos para el PWM de 16 bits) el que utilizaremos en este caso sólo incluye un único bloque digital correspondiente con un bloque PWM.

Este bloque PWM, si tendrá configurables sus salidas a diferencia de los incluidos en el ADCINCVR, aumentando la capacidad de configuración. Una captura de pantalla y la configuración en PSoC Designer donde se muestra la disposición del bloque digital del ADCINC comparado con sus homólogos del ADCINCVR, y su configuración es la siguiente:

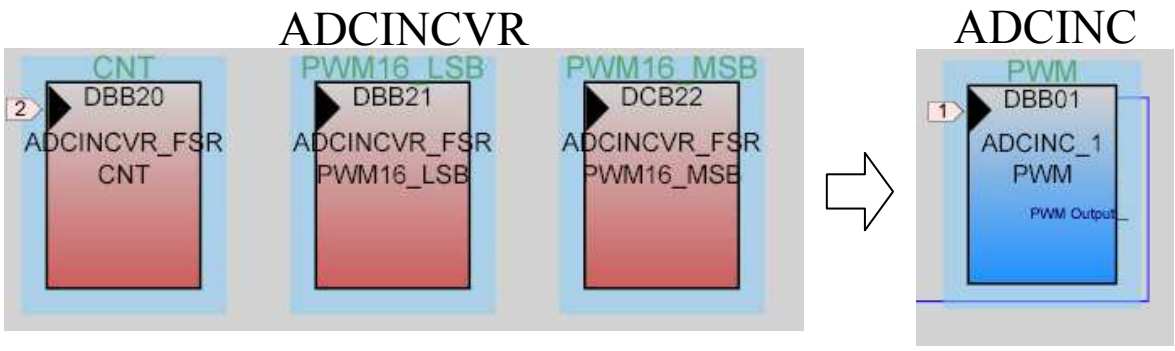


Fig. 4.74 Diferencia entre los bloques digitales ocupados por los diferentes conversores AD

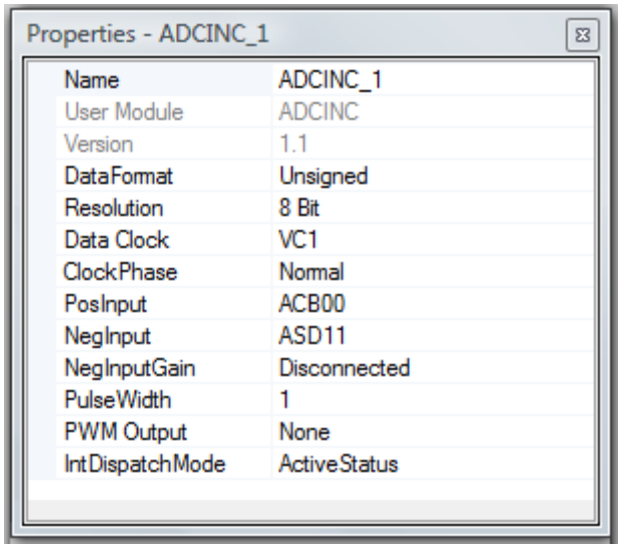


Fig. 4.75 Propiedades del conversor AD

Las variables del conversor AD incremental serán las siguientes:

Formato de datos: Elegiré la opción de datos sin signo.

Resolución: Escogí una resolución de 8 bits en este caso.

Reloj de datos: El reloj elegido para el conversor será VC1=2.4 MHz.

Entrada Positiva: Proveniente del PGA alojado en el bloque de tiempo continuo ACB00, y este a su vez conectado al pin P0_1.

Salida del PWM: El PWM que incluye el AD no será utilizado por lo que no lo conectaremos.

De esta manera el conversor AD incremental será configurado con una resolución de 8 bits, suficiente para la realización del test basado en la UART, cuya configuración de bloques será explicada posteriormente junto a los bloques digitales utilizados para dicha tarea.

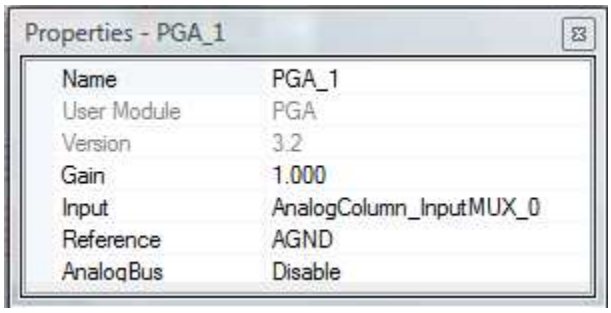


Fig. 4.76 Propiedades del Amplificador de ganancia programable

La configuración del PGA o amplificador de ganancia programable no variará con respecto a las configuraciones anteriores:

Ganancia: La ganancia tendrá un valor unidad, ya que no interesa perturbar la señal recibida.

Entrada: Columna analógica del mux 0, ruteada a su vez con el puerto P0.1 en el caso del eje X.

Referencia: Analog Ground. Tomaremos como referencia la masa analógica del sistema

- Bloques digitales

Los bloques digitales que utilizaré además de los dos correspondientes con la UART serán los mostrados en la siguiente captura de pantalla en PSoC Designer:

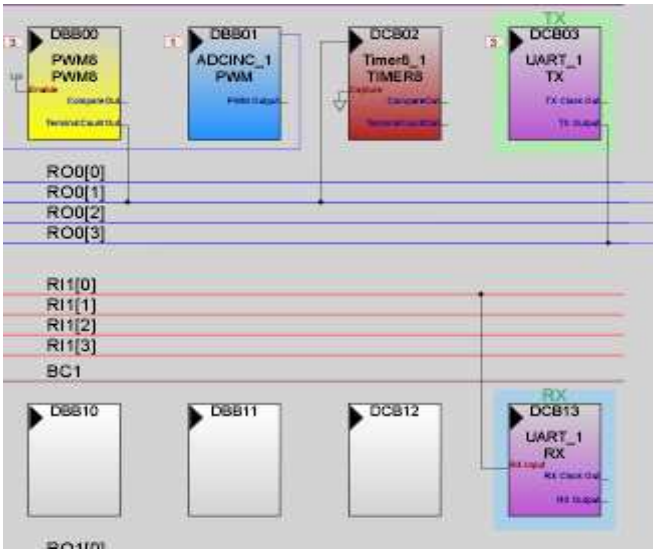


Fig. 4.70 Disposición de los bloques digitales

La configuración de cada uno de estos bloques digitales se mostrará a continuación:

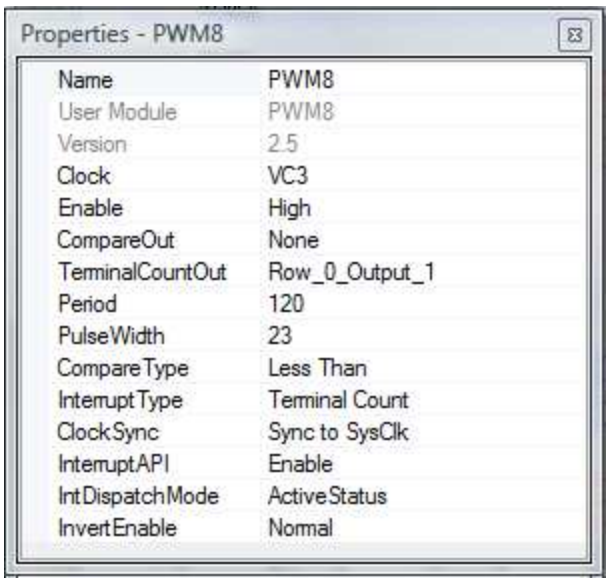


Fig. 4.77 Configuración del bloque PWM

Este bloque PWM se usará como fuente de reloj del TIMER para realizar la interrupción de 1 segundo y su configuración es la siguiente:

Fuente de reloj: VC3= 2.4KHz, proveniente de VC2/100. Esto equivale a 0.41ms.

Periodo: El periodo de contaje será 120 que completará un ciclo cada 50ms lo que equivaldrá a una frecuencia de señal de 20Hz.

Ancho de pulso: Eligiendo un valor 23 y una comparación menor que, el ciclo de trabajo corresponderá al 20%.

Salida de contaje terminal: servirá como reloj del módulo de usuario Timer_8.

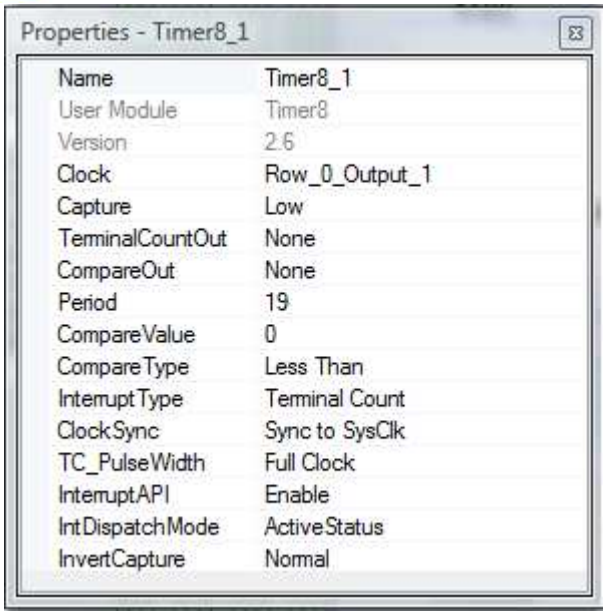


Fig. 4.78 Configuración del bloque Timer

El bloque timer lo utilizaré para generar una interrupción cada segundo:

Fuente de reloj: La fuente de reloj proviene de la salida del PWM. Por lo tanto, 20 Hz (50ms)

Periodo: El periodo de contaje será 20 para conseguir que cada ciclo tarde un segundo: 50ms*20=1s (1Hz).

Tipo de comparación: Es indiferente para la tarea que quiero realizar.

Sincronización con el reloj: Sincronizaremos con el reloj del sistema SysClk.

API de interrupción: Estará habilitada en este caso, Timer8_1_EnableInt();

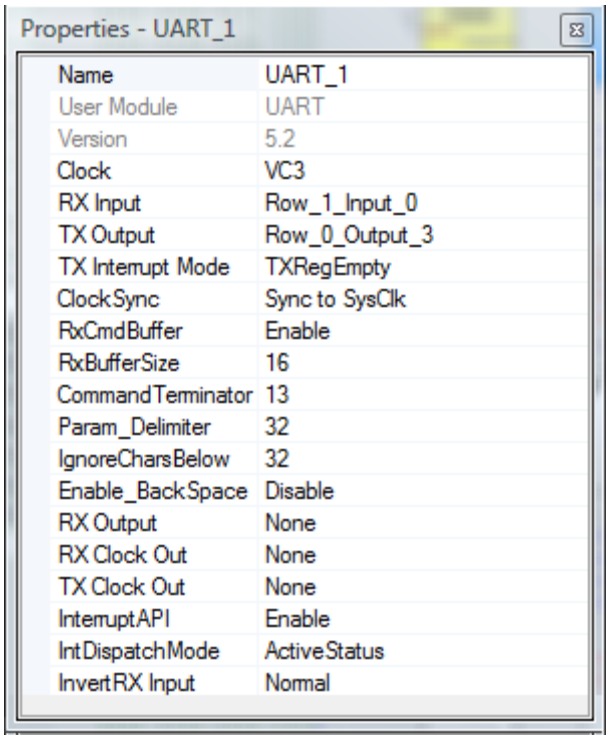


Fig. 4.79 Configuración del bloque UART

La configuración del bloque UART, que estará formada por dos bloques digitales, UART_RX que recibirá los datos arrojados por el hiperterminal y UART_TX que se encargará de enviar los datos desde el PSoC.

Ya que no realicé un prototipo en la placa de puntos para este test, lo realice sobre la placa de pruebas del PSoC, por lo que la configuración de las entradas y salidas de los bloques la ajusté a los predeterminados, los cuales estaban conectados al RS232 y este a su vez al conector DB9. Las propiedades configurables más importantes del bloque UART son las siguientes:

Fuente de reloj: La fuente de reloj escogida VC3=2.4 MHz equivalente a 0.41ms.

Entrada de RX: La entrada del bloque de recepción del UART será la fila1-entrada0 que estará conectada a su vez con el puerto P0_4 que será el predeterminado del PSoC Eval1.

Salida de TX: La salida del bloque UART será la fila0-salida3 que está conectada con el puerto P1_7 que será como en el caso anterior, el puerto predeterminado del PSoC Eval1.

Tamaño del buffer de RX: Elegiremos un tamaño del buffer de 16 bits.

Invertir la entrada RX: No necesitaré que la entrada RX esté invertida, por lo que lo mantendré en modo normal.

De esta manera, el bloque UART estará configurado para funcionar correctamente.

3. Situación final de los pines del PSoC (vista del integrado) y código C.

Como ya he mostrado en los capítulos anteriores, la situación final de los pines del PSoC se encuentra en la siguiente imagen. En este caso se pueden ver los pines resaltados en diferentes colores dependiendo de su naturaleza.

El pin verde corresponde con la entrada analógica del sensor FSR, el rosa con la entrada global del receptor del UART (RX), el rojo con la salida global por la que el UART transmitirá los datos(TX) y los resaltados en azul, que se corresponden con los utilizados por el LCD que tienen una configuración estándar.

En la siguiente figura se muestra dicha situación:

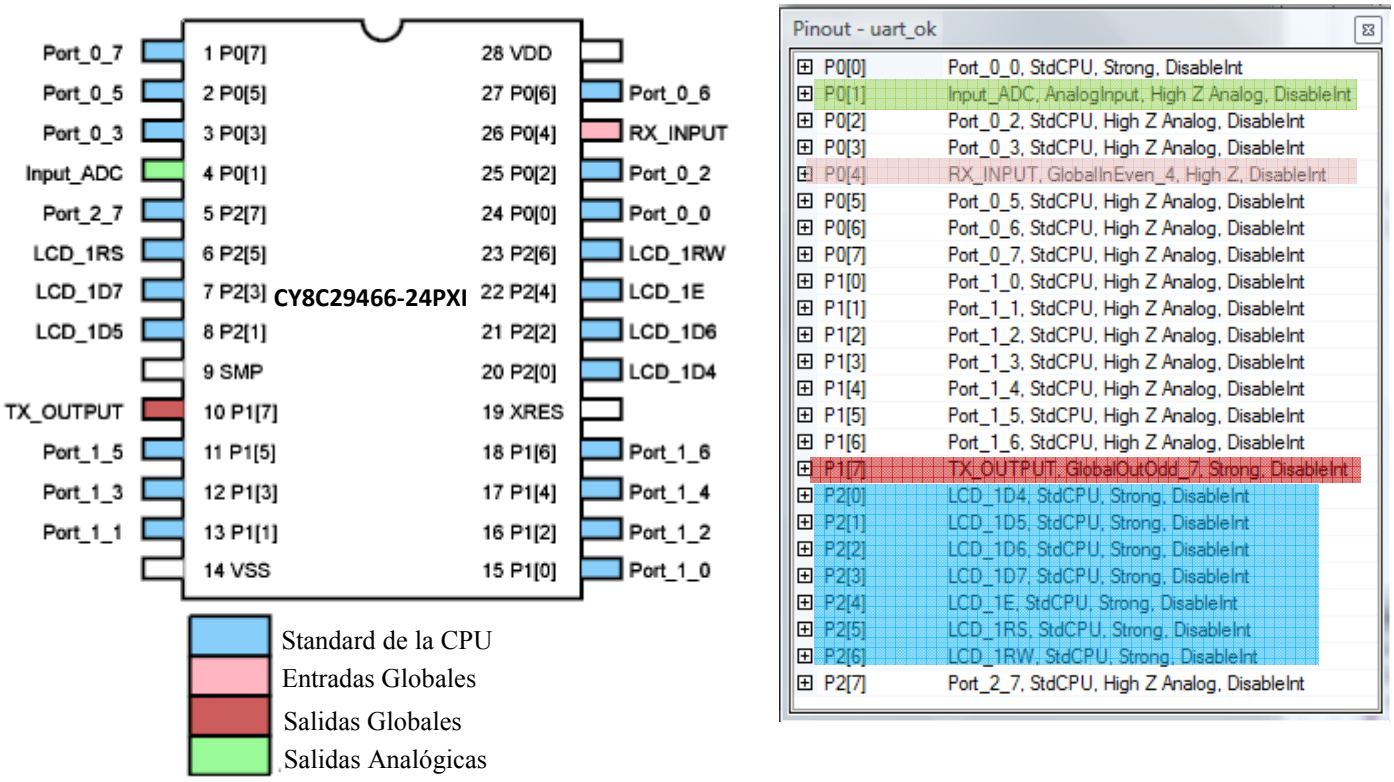


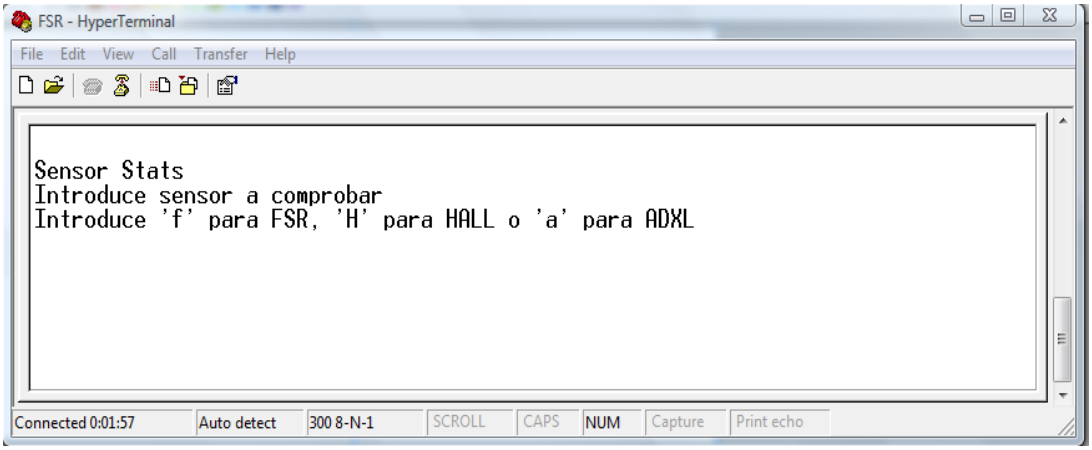
Fig. 4.80 Utilización de los pines del PSoC y configuración de los mismos

El código C de este subsistema se puede consultar en el anexo correspondiente.

D. Comprobación de funcionamiento, pruebas y resultados.

Una vez realizada la configuración previa del puerto COM en el PC y, por consiguiente, del hiperterminal como he mostrado en el punto 1 de este apartado, procedí a la comprobación del funcionamiento del sistema. Para ello, primero realizaremos la conexión con el hiperterminal presionando sobre el icono del teléfono y, a continuación, encenderemos el PSoC desde PSoC Designer como explique en el punto B. del apartado 3.3.2.2. Realicé una serie de capturas de pantalla para mostrar los diferentes estados en los que se puede encontrar tanto en el hiperterminal como en el LCD y que mostraré a continuación:

1. Inicio del sistema: Se visualizará en el hiperterminal el mensaje de inicio y en el LCD la situación del PSoC (en este caso, esperando dato del usuario sobre el hiperterminal)



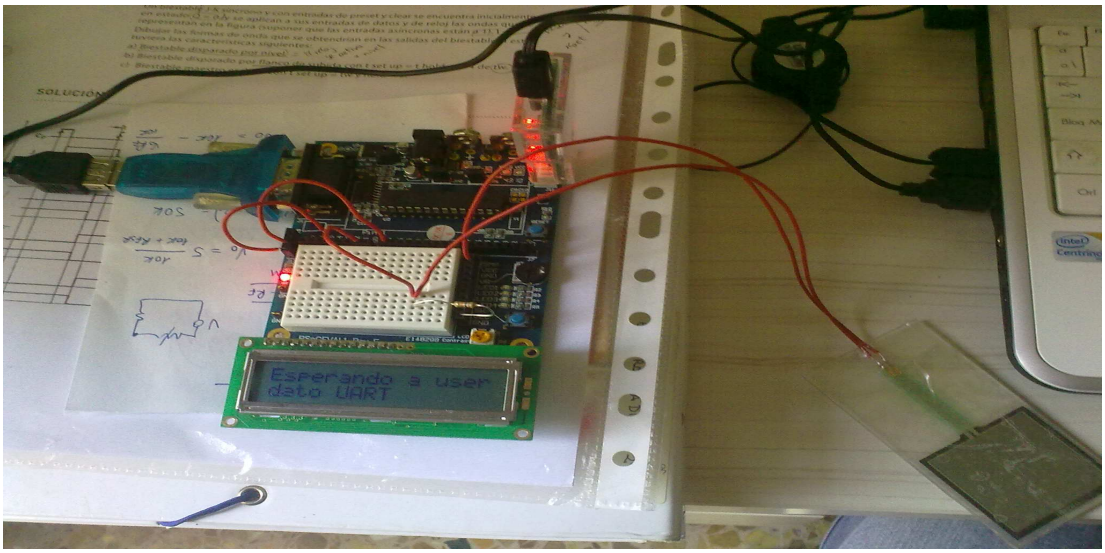


Fig. 4.81 Pruebas sobre el sistema UART: Inicio

2. Dato recibido en el hiperterminal: Como he explicado en el inicio de este capítulo y como se ha mostrado en el código C, los caracteres válidos que desencadenarán los demás eventos serán 'f', 'h' y 'a' correspondientes a FSR, Hall y ADXL respectivamente.

Para realizar el test, solo implementé lo necesario para hacer operar correctamente al sensor FSR y comprobar el funcionamiento del UART, por lo que cualquier valor distinto a 'f' el sistema lo consideraría como erróneo ofreciendo nuevamente las opciones disponibles.

De esta manera, una vez recibido un dato válido en el hiperterminal ('f'), aparecerá un nuevo menú en el hiperterminal donde se podrá elegir qué modo de operación deseamos visualizar, mientras que en el LCD aparecerá el sensor elegido (en este caso, "FSR"):

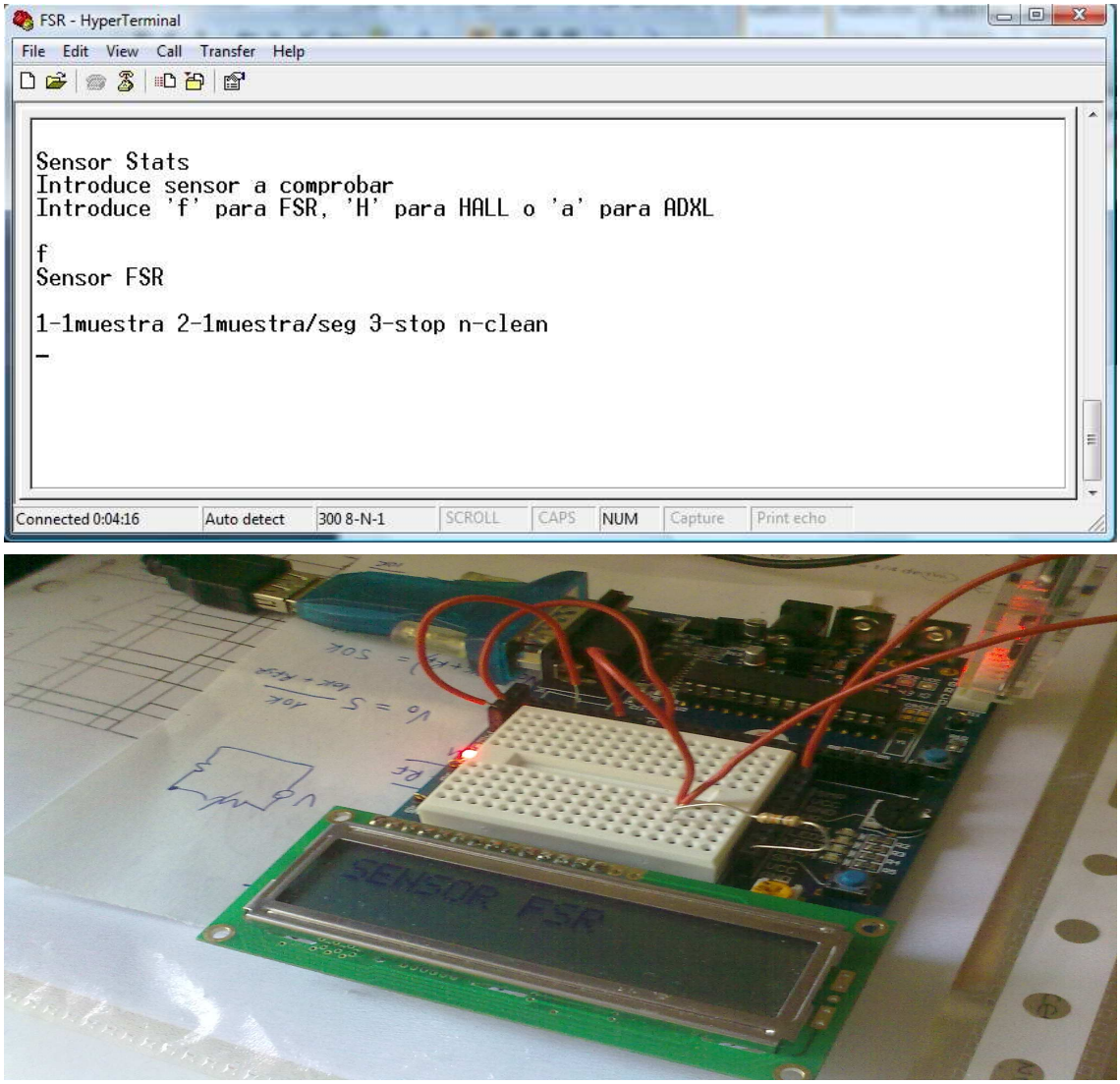


Fig. 4.82 Pruebas sobre el sistema UART: Dato recibido en el hiperterminal

3. Modo de operación 1: Tras aparecer el segundo menú, eligiendo la opción '1' el sistema obtendrá una muestra del conversor AD mostrando lo siguiente:

- LCD: Se visualizará en el LCD la opción escogida en el hiperterminal ('1'en este caso) además del valor del conversor AD en formato de coma flotante.
- Hiperterminal: En el hiperterminal además de los datos mostrados anteriormente, se mostrará la opción escogida y, seguidamente, el valor del conversor AD en formato hexadecimal de 4 bits.

En la siguiente imagen se muestra el resultado de escoger la opción '1':

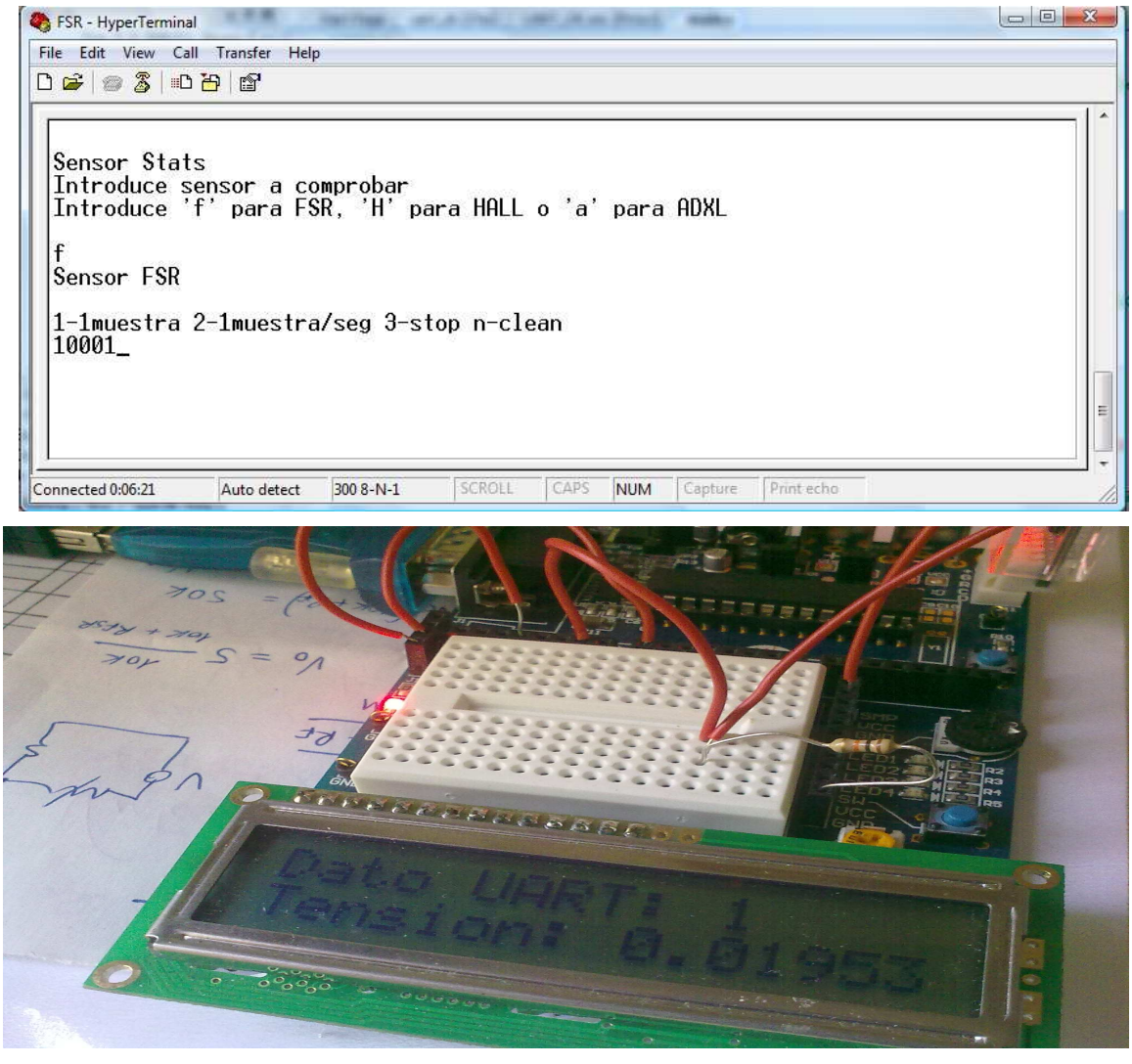


Fig. 4.83 Pruebas sobre el sistema UART: Modo de operación 1

4. Modo de operación 2: Tras aparecer el segundo menú, eligiendo la opción '2' el sistema obtendrá una muestra cada segundo (gracias a la configuración de la interrupción mediante TIMER) del conversor AD mostrando lo siguiente:

- LCD: Se visualizará en el LCD la opción escogida en el hiperterminal ('2'en este caso) además del valor del conversor AD en formato de coma flotante cada segundo.
- Hiperterminal: En el hiperterminal además de los datos mostrados anteriormente, se mostrará la opción escogida y, seguidamente, un valor del conversor AD en formato hexadecimal de 4 bits cada segundo que variará en función de los valores registrados por el mismo, como se puede apreciar.

En la siguiente imagen se muestra el resultado de escoger la opción '2':

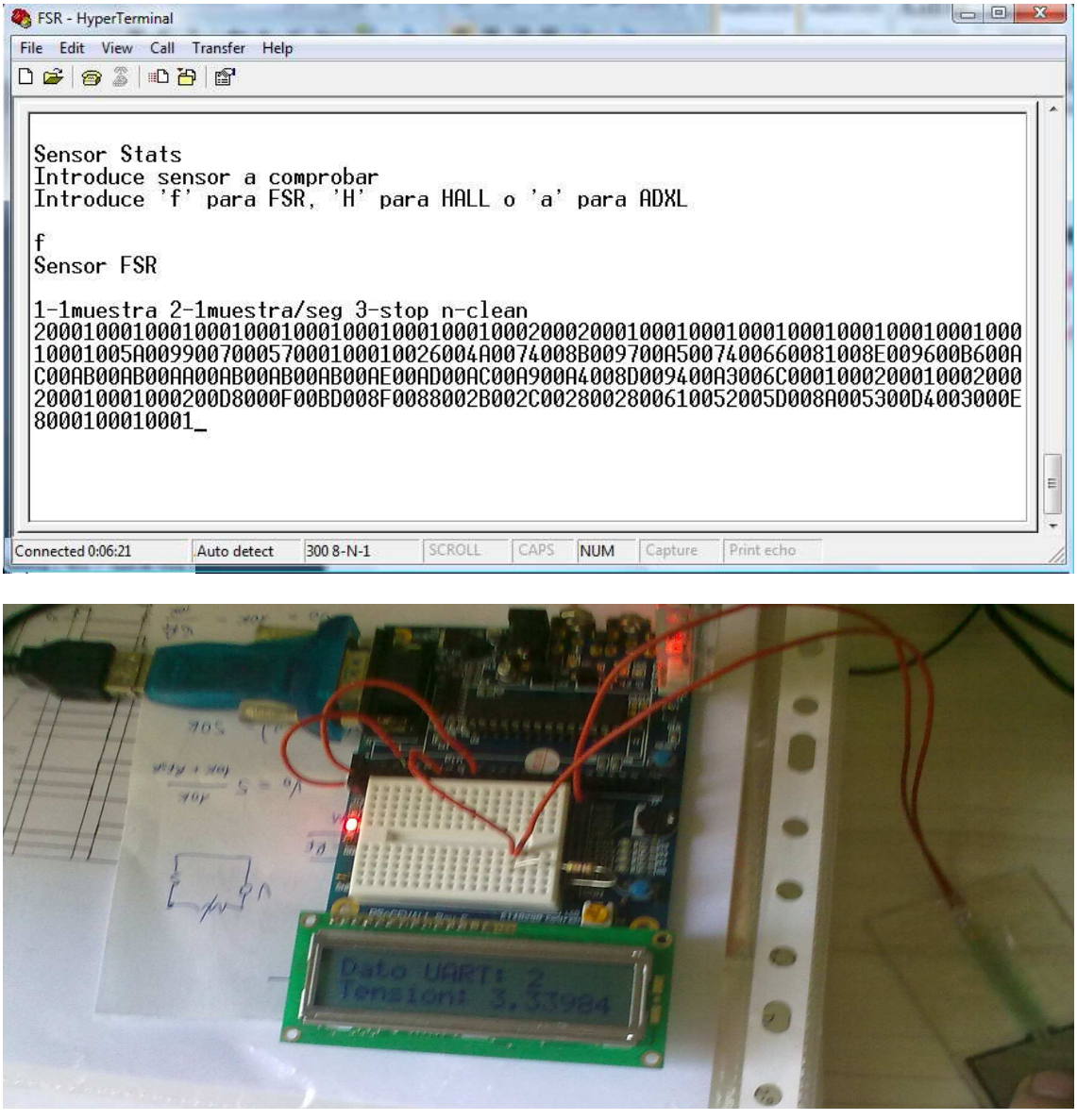


Fig. 4.84 Pruebas sobre el sistema UART: Modo de operación 2

5. Modo de operación 3: Tras aparecer el segundo menú, eligiendo la opción '3' el sistema segundo menú, eligiendo la opción '3' el sistema detendrá la toma de muestras cada segundo, en el caso de que haya sido activada y se mostrará lo siguiente:

- LCD: Se visualizará en el LCD la opción escogida en el hiperterminal ('3'en este caso) y la palabra 'STOP' para confirmar que la desactivación del modo de datos continuo se ha detenido.

- Hiperterminal: En el hiperterminal únicamente se mostrará la opción elegida.

En la siguiente imagen se muestra el resultado de escoger la opción '3':

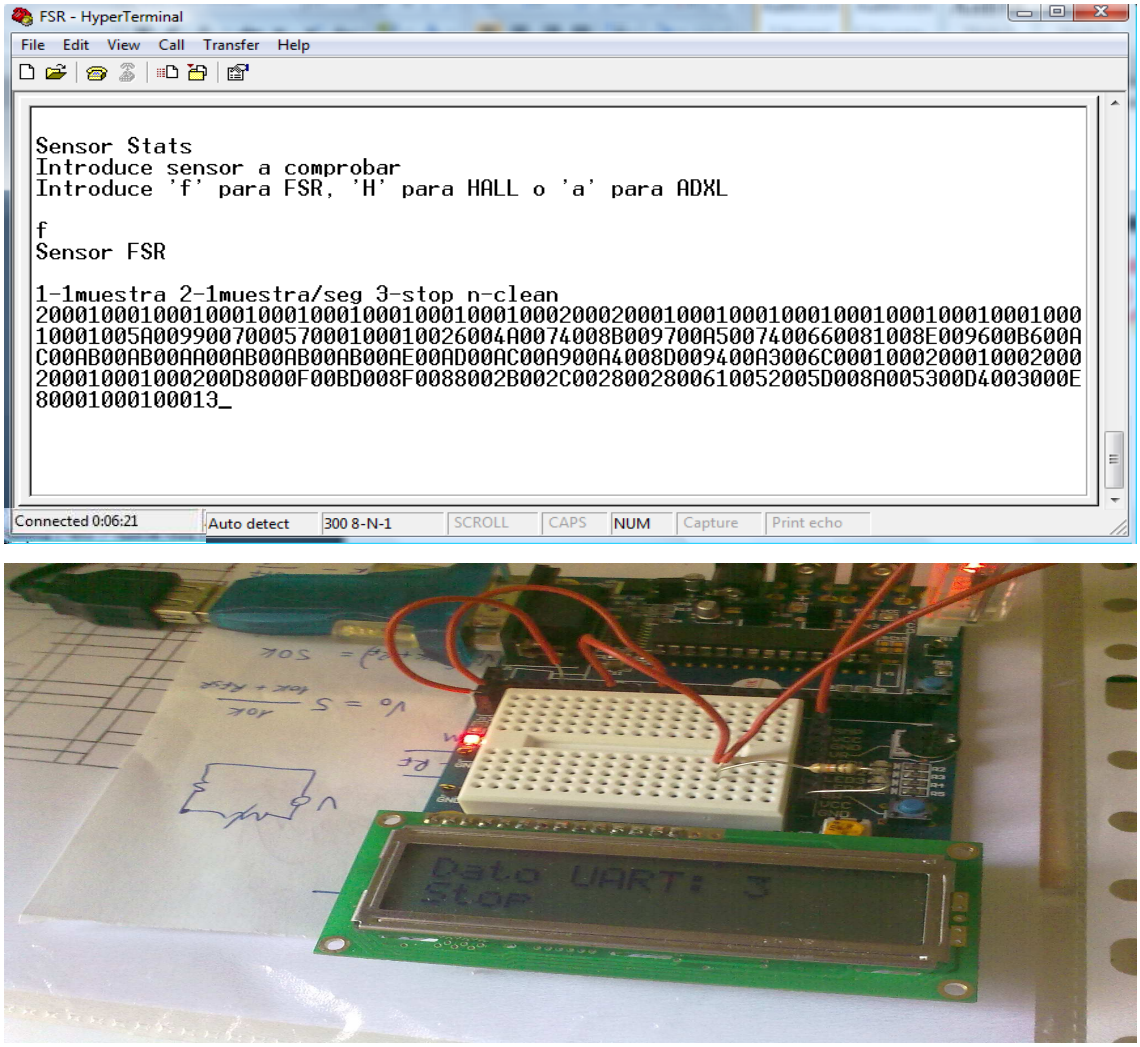


Fig. 4.85 Pruebas sobre el sistema UART: Modo de operación 3

6. Modo de operación 4: En el caso de escoger la opción 'n', se borrará la pantalla del hiperterminal y aparecerá de nuevo el mensaje de inicio del punto 1.

Como se puede comprobar, el test de comprobación de funcionamiento del bloque UART se realizó satisfactoriamente. Por tanto, todos los elementos necesarios ya habían sido comprobados y ya podía realizar su implementación en el sistema final.

4.4 Prototipo de sistema conjunto

Una vez concluida la creación de un subsistema de control concreto para cada uno de los sensores a utilizar en el prototipo final, además de la comunicación serie para la representación gráfica en Matlab de las señales arrojadas por los mismos, podía comenzar el desarrollo del software de control del prototipo final.

El software del prototipo final cuenta con las siguientes funcionalidades:

- Visualización en tiempo real de la inclinación de la máquina cortacésped en el LCD, mediante la realización de un baremo de inclinaciones (OK,ATT y PEL) visibles en el mismo y de un sistema de alarmas formado por dos diodos LED y un piezorresistivo que se activarán en función de este baremo.

- Visualización en tiempo real de la altura del plato de corte de la máquina cortacésped a través del LCD, que se podrá variar entre los 80mm y los 20 mm y que contará con dos interruptores para subir/bajar el plato.

- Visualización en tiempo real de la situación de llenado del cesto de recogida del césped en el LCD, con un baremo similar al realizado para la visualización de la inclinación pero con valores diferentes (OK, MED y FULL) y con un sistema de alarmas formado por dos diodos LED. Cuando el sistema detecte un número determinado de alarmas de cesto lleno, enviará una señal que detendrá las cuchillas de corte acabando con la entrada de más césped al cesto.

- A través de un interruptor, entrada en modo UART en el que el sistema enviará cada segundo señales convertidas en formato de cadena (string) a través del puerto serie que serán procesadas y representadas gráficamente mediante Matlab.

Para continuar la dinámica de los apartados anteriores, desarrollaré la explicación del software del prototipo del sistema de control en los siguientes apartados: A. Hardware del sistema, B. Diagrama de bloques del sistema de control, C. Software del sistema y D. Comprobación de funcionamiento, pruebas y resultados.

A. Hardware del sistema

El hardware de este sistema será detallado en el apartado 4.7 ya que para la realización del prototipo utilizaré el programa Altium Designer en el que desarrollaré una placa de circuito impreso que será controlada por el software descrito en las siguientes páginas. Para el desarrollo del software asociado al prototipo, y las pruebas previas para la comprobación del funcionamiento de este, las realizaré con la placa de evaluación del PSoC, el CY3210, descrito en el apartado 3.2.2.

Por tanto, el desarrollo de este apartado se centrará en la descripción del software de control del prototipo, concretamente en el desarrollo del diagrama de bloques del sistema y el código C relativo al sistema.

B. Diagrama de bloques del sistema de control: El diagrama de bloques del sistema de control final será el siguiente: Como se puede ver en la fig. 4.86 el diagrama de bloques del prototipo varia considerablemente con respecto a los correspondientes de los subsistemas anteriores. En este caso, el bucle principal estará formado únicamente por subrutinas para agilizar la velocidad de este. En este caso, evitaré el uso de la instrucción `goto` ya que puede provocar bucles infinitos indeseados.

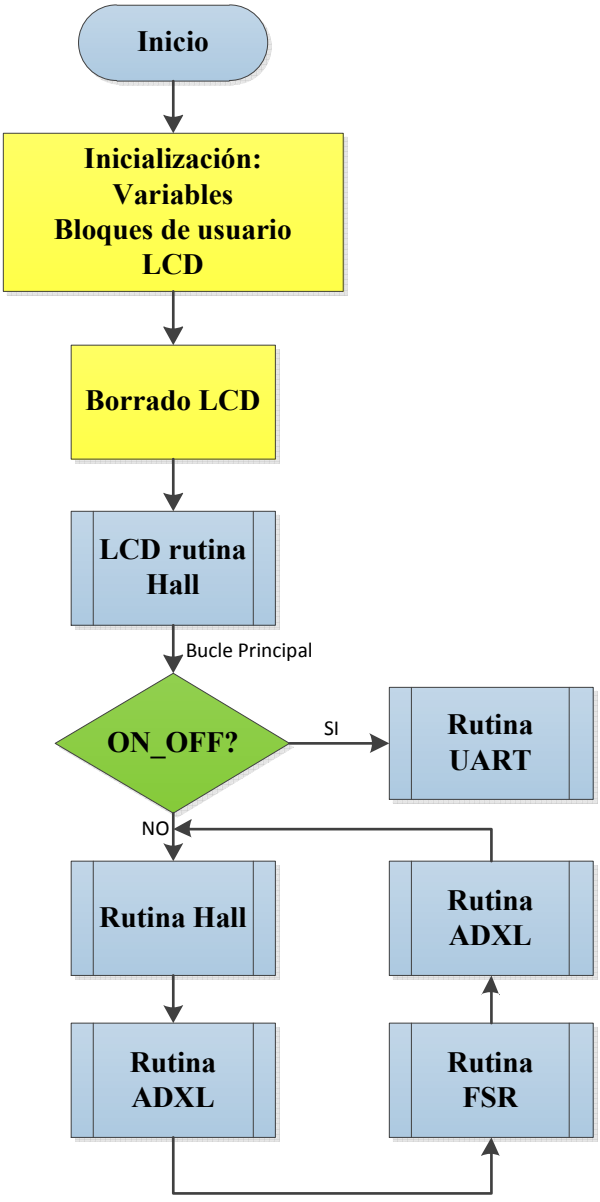


Fig. 4.86 Diagrama de bloques del prototipo final

Dependiendo del estado de un interruptor que llamaré ON_OFF, el sistema realizará su tarea principal o enviara los datos de los sensores por el puerto serie para representarlos gráficamente mediante Matlab. En el bucle principal, es decir, cuando el interruptor ON_OFF no esté activado, el sistema se dedicará continuamente a captar las señales arrojadas por los sensores refrescando los datos de pantalla y activando las alarmas de cada uno de ellos en caso de que se detecten valores determinados como no recomendados.

Estas subrutinas se corresponden con la captación de las señales recibidas por el sensor hall, los ejes X e Y del sensor ADXL y el sensor FSR respectivamente.

En el momento que el usuario desee enviar los datos de estos sensores por el puerto serie, activará el interruptor ON_OFF y instantáneamente comenzará el envío de datos, apareciendo en la pantalla del LCD la confirmación del sistema de entrada en el modo UART. En el momento en el que se desactive dicho interruptor, el sistema volverá nuevamente al bucle principal mostrando el estado de los sensores mencionados.

Aunque las subrutinas que aparecen en el diagrama de bloques pueden variar sensiblemente con respecto a las desarrolladas en los apartados anteriores, de forma general realizan la misma función y por lo tanto no las desarrollaré nuevamente. Los cambios que presentan se podrán apreciar posteriormente en la exposición del código C del sistema, donde se detallarán los mismos.

C. Software del sistema

Basándome en el diagrama de bloques mostrado en el punto B y, a partir nuevamente de la figura 4.9, mostraré la configuración del software del PSoC y posteriormente el código C relativo a la misma. La configuración de los parámetros y bloques tanto digitales como analógicos del PSoC serán los siguientes:

1. Parámetros Globales del PSoC

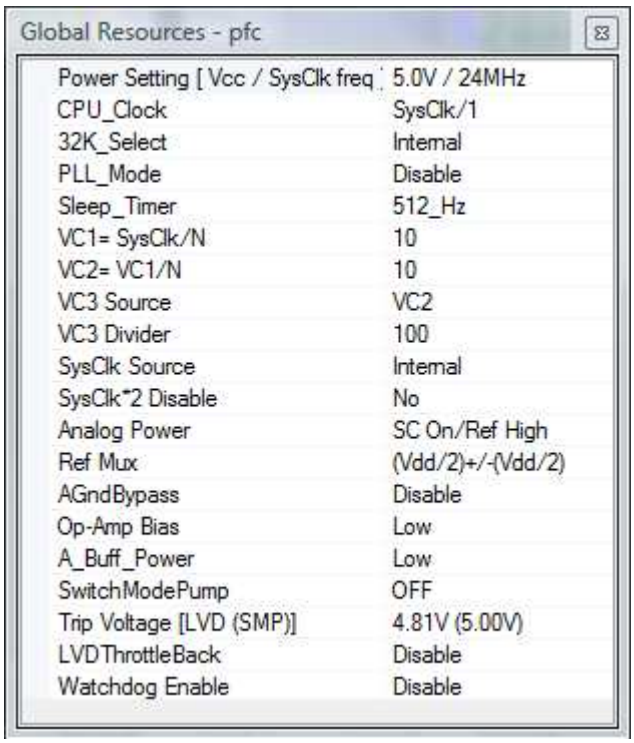


Fig. 4.87 Parámetros globales del prototipo

Los parámetros globales del PSoC para este prototipo final serán los siguientes:

Tensión de alimentación y frecuencia de SysClk: Al igual que en los subsistemas descritos, la tensión de alimentación de será de 5V y con una frecuencia de 24MHz.

Frecuencia de reloj de la CPU: No realizaré ninguna división del reloj principal y la CPU tendrá la frecuencia de SysClk.

Fuente de SysClk: No usaré un cristal externo, por lo que la fuente de SysClk será la propia del PSoC.

Deshabilitar SysClk*2: Esta función la mantendré deshabilitada nuevamente.

Referencia analógica: Con esta opción seleccionamos el rango y la precisión de los bloques analógicos. Elegiré la opción (Vdd/2) / (-Vdd/2).

Frecuencias de trabajo VC1, VC2 y VC3: Como ya he comentado, estos valores representarán las fuentes de reloj de las que dispondrá el sistema, pudiendo elegirse indistintamente:

VC1= SysClk/N. Elegiré N=10 obteniendo VC1= 2.4 MHz

VC2=VC1/N. Eligiendo N=10 obtendré VC2= 240 KHz

Siendo VC3=VC2/N y eligiendo N=100, obtendremos VC3=2.4KHz (0.416 ms)

Switch mode Pump: Esta opción la mantendré deshabilitada nuevamente.

Detección de bajo voltaje: Esta opción no será relevante para la realización de las pruebas del sistema. Mantendré la opción utilizada en los subsistemas anteriores.

Watchdog Enable: Al no requerir de perro guardián, la opción la mantendré desactivada.

Las demás opciones han sido descritas en el apartado 3.3.2.1, y su configuración se muestra en la fig. 4.87

2. *Mapa general del dispositivo. Bloques analógicos y digitales. Configuración*

En la Fig. 4.88 se muestra, como en los subsistemas anteriores una captura en PSoC Designer del mapa general de la configuración de módulos, donde quedan resaltadas las filas y las columnas de bloques digitales y analógicos, respectivamente:

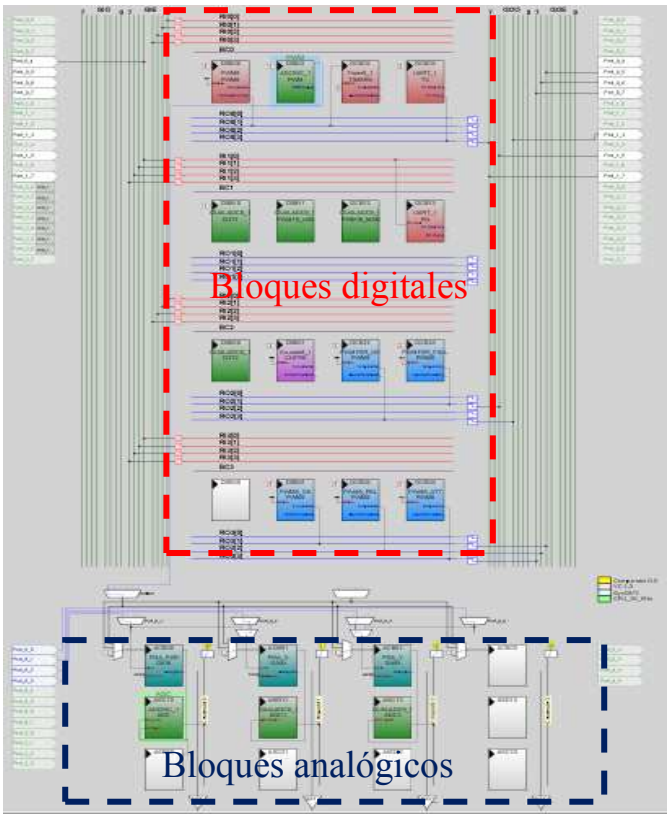


Fig. 4.88 Mapa de configuración de hardware del CY8C29466-24PXI

Como se puede apreciar en la figura 4.88, se han utilizado prácticamente todos los bloques disponibles para la configuración del sistema. Si bien, en la zona correspondiente a los bloques digitales se pueden encontrar los bloques asociados al UART en rojo (bloques TX, RX, Timer de 8 bits y PWM de 8 bits), los bloques digitales de los conversores AD en verde (tanto del ADC incremental dual como del ADC de 8 bits), los bloques correspondientes con los sistemas de alarmas del sistema en azul (los cuales serán bloques PWM de 8 bits que activarán los diodos LED y el piezorresistivo) y en morado un bloque asociado a la temporización del ciclo del sensor hall (correspondiente con un contador de 8 bits para realizar un retraso de 530ms), en la zona de bloques analógicos encontraremos en verde la parte analógica del los conversores ADC incremental dual y ADC incremental de 8 bits además de los correspondientes amplificadores de ganancia programable (como ya he comentado anteriormente configurados con valor 1) para recibir las señales analógicas a través de los pines del PSoC.

Comenzaré mostrando en primer lugar la configuración de los bloques analógicos y posteriormente los digitales.

- Bloques analógicos

Para la captación de señales de todos los sensores a controlar, en el prototipo requeriremos de tres conversores AD. Si bien, podría haber escogido utilizar 3 bloques AD de 8 bits con bajo gasto de bloques, preferí aunar la recepción de las señales arrojadas por el sensor ADXL en un solo conversor AD, el conversor AD incremental Dual. Este conversor es idéntico al de 8 bits con la diferencia de que tiene una entrada doble y puede realizar la conversión de dos señales analógicas a la vez. Este conversor utilizará un bloque analógico para cada una de las 2 señales a convertir y cuatro bloques digitales correspondientes a dos contadores de 8 bits y un PWM de 16 bits (byte significativo y byte no significativo) los cuales no se podrán configurar, ya que forman parte del funcionamiento interno del propio conversor.

Por otro lado, usaré el citado conversor AD incremental de 8 bits para la captación de las señales arrojadas por el sensor FSR, ya que sólo consumirá un bloque analógico y otro digital. En la siguiente imagen se puede ver la disposición de los bloques analógicos del prototipo final:

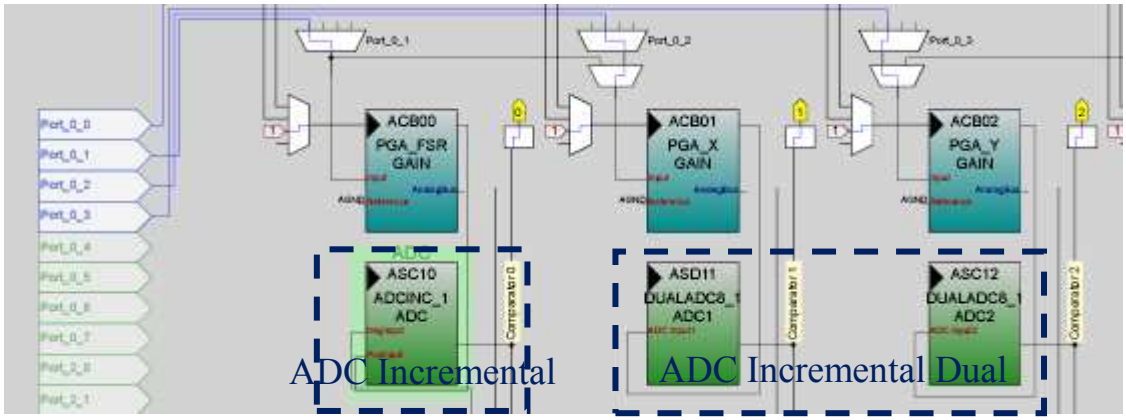


Fig. 4.89 Disposición de los bloques analógicos

Cada una de las entradas de estos conversores AD requieren de la incorporación de un amplificador de ganancia programable, ya que no es posible realizar la conexión directa entre el pin del integrado y el bloque del conversor. Por tanto, la configuración de estos amplificadores, como se podrá ver a continuación, contarán con una ganancia unidad para no perturbar la señal que reciban los conversores.

Los bloques digitales que forman parte del conversor AD incremental dual son los mostrados en la fig. 4.85. Como se puede ver, cuenta con dos contadores (uno para cada entrada de señal analógica) y un modulador de 16 bits. Estos bloques no serán configurables ya que formarán parte de la configuración interna del conversor. Los bloques digitales correspondientes al conversor AD de 8 bits ya se han mostrado en el apartado 4.3.

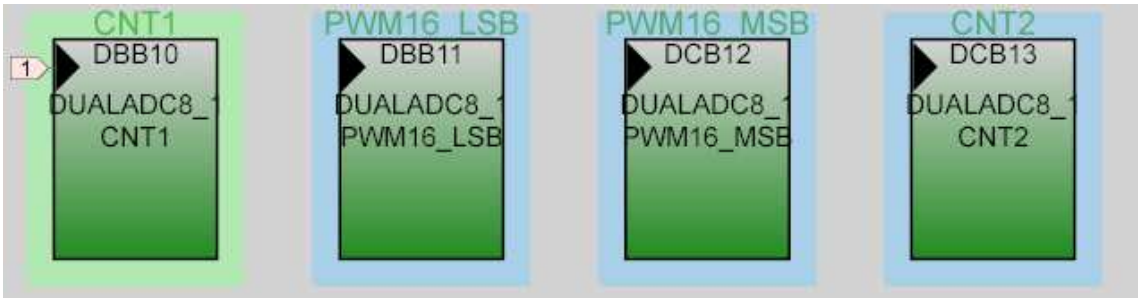


Fig. 4.90 Bloques digitales del conversor AD incremental dual

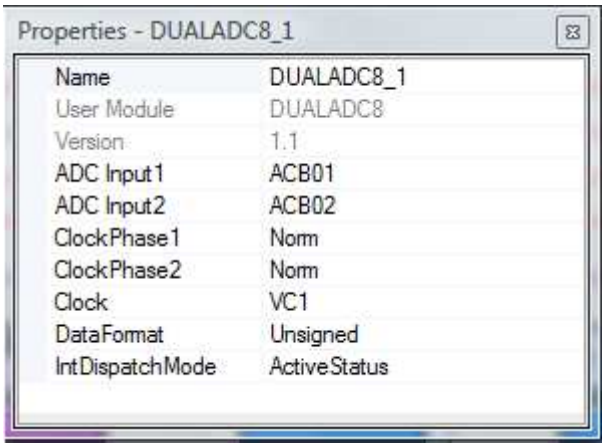


Fig. 4.91 Propiedades del conversor AD Dual

La configuración del conversor AD dual de 8 bits serán las siguientes:

Entrada 1 del ADC: Escogí una columna analógica libre. En este caso, y como se puede ver en la fig. 4.84. en la "ranura" ACB01, correspondiente con la entrada de uno de los PGA.

Entrada 2 del ADC: Igual que con la opción anterior pero en este caso la entrada procede del PGA localizado en el bloque ACB02.

Fase del reloj 1 y 2: Podremos elegir entre una fase de reloj normal o intercambiada. Elegiré la opción normal para ambas entradas.

Reloj: La fuente de reloj elegida será VC1 equivalente a SysClk/10, es decir, 2.4 MHz.

Formato de datos: El formato de datos, ya que no trabajaré con datos negativos, será configurado sin signo.

Un valor de configuración importante que sólo se puede configurar a través de una API en el código C del sistema, es el valor CalcTime, que permitirá establecer la frecuencia de muestreo. En este caso, la ecuación para determinarlo variará ligeramente de la mostrada en apartados anteriores. De esta manera, la frecuencia de muestreo vendrá determinada por las siguientes ecuaciones:

$$CalcTime \geq \frac{248 \times DataClock}{CPU\_Clock} \qquad SampleRate = \frac{DataClock}{1024 + CalcTime} \qquad DataClock = \frac{1024}{IntegrateTime}$$

Mediante el valor Integrate Time rechazaré el ruido y los armónicos de la frecuencia de trabajo que en este caso será 50Hz. Por tanto:

$$IntegrateTime = 5 \times \frac{1}{50} = 100ms \qquad DataClock = \frac{1024}{100ms} = 10.24KHz$$

Sustituyendo valores, obtendré los siguientes resultados

$$CalcTime \geq \frac{248 \times 10.24KHz}{24MHz} = 0.105$$

Elegiré un valor de 1, por lo que la frecuencia de muestreo será la siguiente:

$$SampleRate = \frac{10.24KHz}{1024 + 1} = 9.99\ muestras/s$$

Con una frecuencia de muestreo de casi 10 muestras por segundo, aseguraremos el correcto funcionamiento del sistema. Dicha configuración del valor CalcTime se podrá ver posteriormente en el código C del sistema.

Por otro lado, la configuración de los amplificadores de ganancia programable es idéntica a la mostrada en los subsistemas anteriores, por lo que no volveré a mostrarla.

- Bloques digitales

Los bloques digitales que conforman el sistema, además de los mostrados y que forman parte de los conversores AD, son los siguientes:

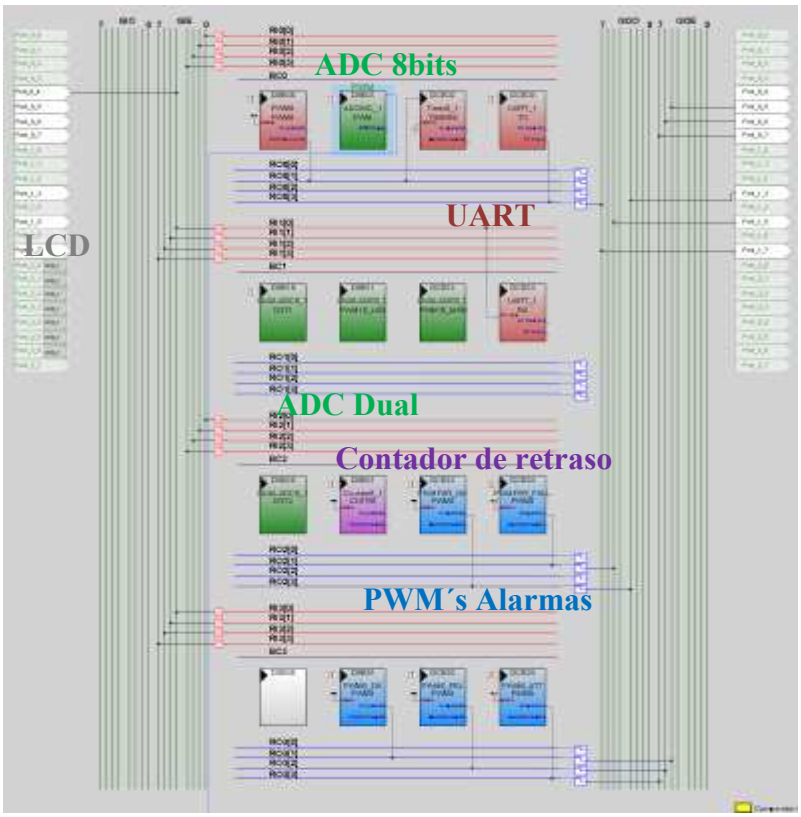


Fig. 4.92 Disposición de los bloques digitales

En la figura 4.92 se puede ver como los bloques con una función similar o como parte de un conjunto de trabajo tienen el mismo color. De esta manera, y agrupados por funciones, los bloques digitales del sistema final son los siguientes:

- Bloques UART: Los bloques utilizados para el buen funcionamiento del UART, asi como del temporizador utilizado para que el sistema genere una interrupción cada segundo en el modo de envío de datos de los sensores por el puerto serie serán los siguientes: un PWM de 8 bits que será la base de tiempos del TIMER de 8 bits y los dos bloques asociados al sistema de envío de datos por el puerto serie, UART_RX y UART_TX.

- Bloques AD: Como ya he comentado, los bloques digitales correspondientes a los conversores AD serán 5, cuatro del conversor AD Dual y uno al conversor AD de 8 bits.

- Alarmas: Para la activación de las alarmas que serán 4 diodos LED y el piezorresistivo utilizaré sendos PWM de 8 bits.

- Retraso: Para realizar un retraso necesario para el contaje del sensor Hall, he incorporado un contador de 8 bits.

Las configuraciones de los bloques pertenecientes al grupo UART son las siguientes:

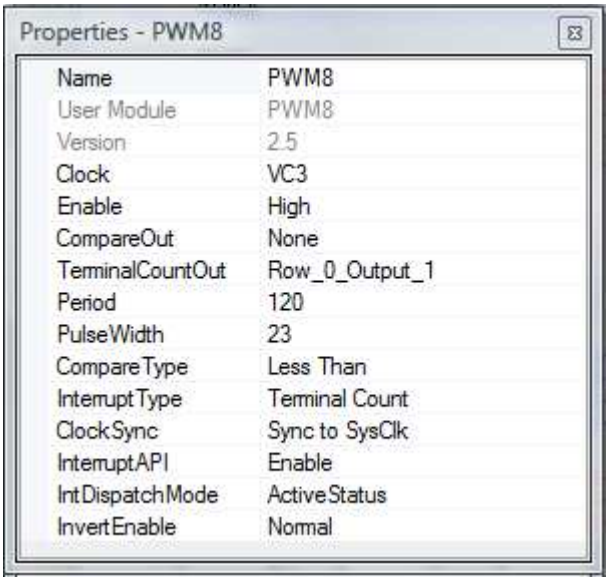


Fig. 4.93 Configuración del bloque PWM

Este bloque PWM se usará como fuente de reloj del TIMER para realizar la interrupción de 1 segundo y su configuración es la siguiente:

Fuente de reloj: VC3= 2.4KHz, proveniente de VC2/100. Esto equivale a 0.41ms.

Periodo: El periodo de contaje será 120 que completará un ciclo cada 50ms lo que equivaldrá a una frecuencia de señal de 20Hz.

Ancho de pulso: Eligiendo un valor 23 y una comparación menor que, el ciclo de trabajo corresponderá al 20%.

Salida de contaje terminal: servirá como reloj del módulo de usuario Timer_8.

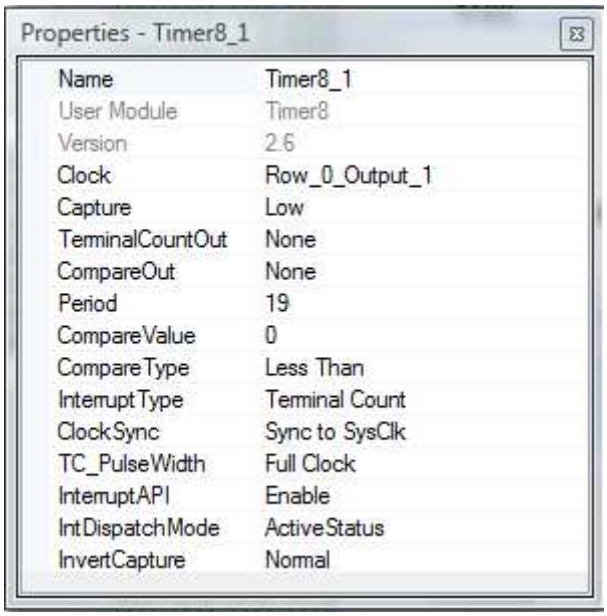


Fig. 4.94 Configuración del bloque Timer

El bloque temporizador lo utilizaré para generar una interrupción cada segundo:

Fuente de reloj: La fuente de reloj proviene de la salida del PWM. Por lo tanto, 20 Hz (50ms)

Periodo: El periodo de contaje será 20 para conseguir que cada ciclo tarde un segundo: 50ms*20=1s (1Hz).

Tipo de comparación: Es indiferente para la tarea que quiero realizar.

Sincronización con el reloj: Sincronizaremos con el reloj del sistema SysClk.

API de interrupción: Estará habilitada en este caso, Timer8_1_EnableInt();

El sistema se encargará de mostrar en tiempo real la situación de los sensores comentados y en el caso de que el interruptor ON_OFF fuera activado, se entrará en modo UART y se activará la interrupción por timer, enviando cada segundo muestras de cada uno de los sensores a través del puerto serie incorporado en el prototipo final.

Estos datos, serán procesados a través del programa Matlab mostrando las gráficas realizadas a través de los datos enviados por el sistema.

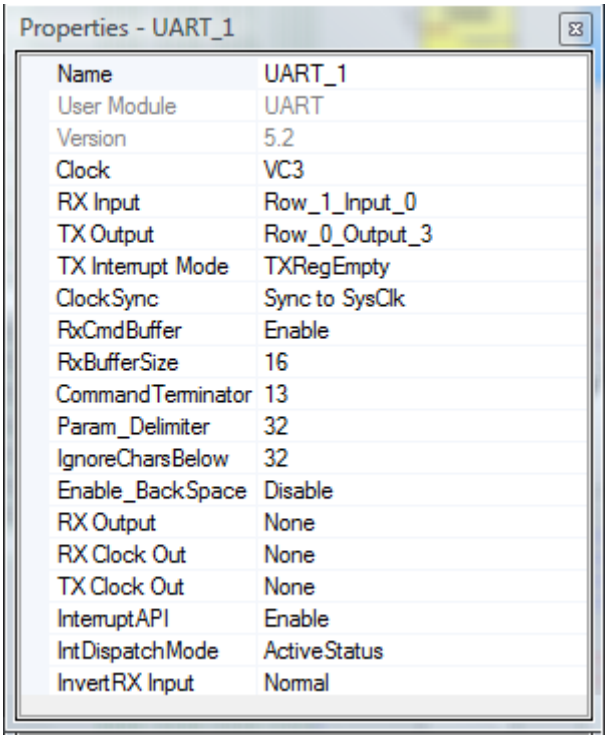


Fig. 4.95 Configuración del bloque UART

Las propiedades configurables más importantes del bloque UART son las siguientes:

Fuente de reloj: La fuente de reloj escogida VC3=2.4 MHz equivalente a 0.41ms.

Entrada de RX: La entrada del bloque de recepción del UART será la fila1-entrada0 que estará conectada a su vez con el puerto P0_4, manteniendo el puerto predeterminado del PSoC.

Salida de TX: La salida del bloque UART será la fila0-salida3 que está conectada con el puerto P1_7.

Tamaño del buffer de RX: Elegiremos un tamaño del buffer de 16 bits.

Invertir la entrada RX: No necesitaré que la entrada RX esté invertida, por lo que lo mantendré en modo normal.

Las configuraciones del grupo de conversores AD ya se han mostrado en el apartado de los bloques analógicos, por lo que no mostraré nuevamente la configuración.

Las configuraciones de los bloques pertenecientes al grupo de alarmas, son idénticas a las utilizadas en los subsistemas. De los 5 bloques PWM, 4 se encargarán de controlar diodos LED y el restante del piezorresistivo. Por lo tanto, los valores correspondientes al periodo y el ciclo de trabajo de los mismos serán proporcionales a los calculados anteriormente y por ello no mostraré nuevamente su configuración.

La configuración del contador de 8 bits para generar el retraso que se necesita para el correcto funcionamiento del control del sensor Hall es la siguiente:

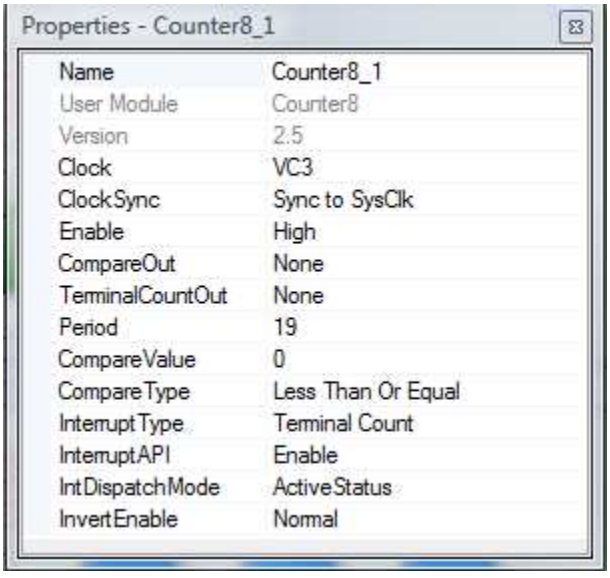


Fig. 4.96 Configuración del bloque contador

Fuente de reloj: VC3= 2.4KHz, proveniente de VC2/10. Esto equivale a 0.41ms.

Periodo: El periodo de contaje será 20 que completará un ciclo cada 8.3 ms lo que equivaldrá a una frecuencia de señal de 120Hz.

Valor de comparación: El valor de comparación será 0. Por lo que el contador se detendrá cuando llegue a este valor.

Tipo de comparación: Elegiré un tipo de comparación "menor o igual que". Ya que el contaje tiene un carácter descendente, nos permitirá tener el valor alto del ciclo en la segunda parte de la onda.

3. Situación final de los pines del PSoC (vista del integrado) y código C.

La situación final de los pines del PSoC para la creación del PCB del prototipo es importante, ya que la elección de estos puede suponer una gran ventaja a la hora del ruteado de los mismos. En la siguiente imagen se muestra la situación final de los pines:

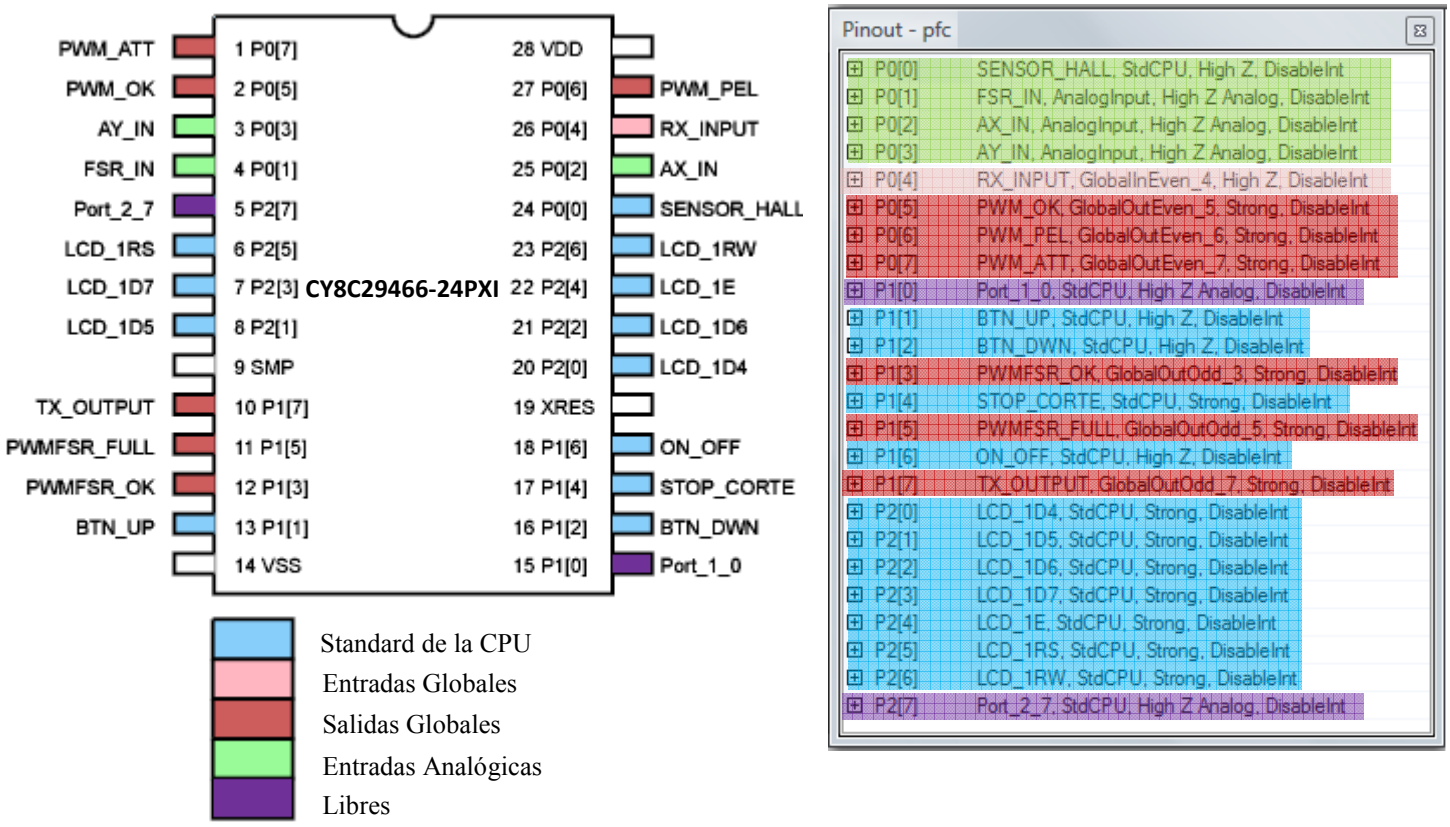


Fig. 4.97 Utilización de los pines del PSoC y configuración de los mismos

La disposición de los bloques asociados a los pines será la siguiente:

PUERTO P0: la parte baja del puerto recibirá las señales analógicas de los sensores (P0[0]-P0[3]) mientras que en la parte alta se sitúa el receptor del UART (P0[4]) y los PWM´s que activarán las alarmas del sensor ADXL (P0[5], P0[6] y P0[7]).

PUERTO P1: El pin P1[0] quedará libre, los pines P1[1] y P1[2] recibirán la situación de los interruptores de subida y bajada del plato respectivamente, los pines P1[3], P1[5] recibirán las salidas de los PWM´s correspondientes con las alarmas del sensor FSR, el pin P1[4] invertirá su valor lógico en caso de recibir la alerta de llenado del cesto (por defecto en nivel bajo), el pin P1[6] recibirá la situación del interruptor ON_OFF y el pin P1[7] estará conectado con el transmisor del UART.

PUERTO P2: Este puerto albergará únicamente a las señales del LCD (P2[0]–P2[6]), quedando el pin P2[7] libre.

El código C del prototipo, se encuentra en el anexo correspondiente.

D. Comprobación de funcionamiento, pruebas y resultados.

Previamente a la creación del prototipo sobre una PCB, las pruebas previas de funcionamiento del software fueron llevadas a cabo mediante la placa de evaluación del PSoC. Utilicé en innumerables ocasiones la técnica de ensayo y error ya que me permitía conocer en qué puntos el código fallaba o que estructuras me permitían mejorarlo.

Como ya he comentado, la instrucción `goto` la he suprimido de la programación ya que podría causar bucles infinitos ocasionando el bloqueo y consecuentemente, el fallo del sistema. Por tanto, tanto la subrutina del sensor Hall como la del UART debieron ser modificadas del código original de sus subsistemas para adaptarlos a la programación del prototipo.

Aunque fue un costoso trabajo adaptarlos, mediante la técnica mencionada conseguí que el programa funcionara perfectamente. Además, como se puede apreciar en el código, la rutina principal sólo cuenta con llamadas a subrutinas lo que agiliza enormemente la velocidad del sistema y permite obtener un código "limpio" (sin fragmentos de código que ralentizarían la velocidad del sistema pudiendo llegar a producir errores indeseados).

A continuación mostraré una serie de capturas de pantalla de los test realizados para la comprobación de funcionamiento del sistema tanto del LCD como de los datos enviados al hiperterminal.

Al alimentar el sistema aparecerá un mensaje de inicio tanto en el LCD como en el hiperterminal

Fig. 4.98. Mensajes de inicio en LCD e Hiperterminal

Tras unos instantes, en el LCD aparecerá la pantalla de explotación mostrando la altura del plato de corte y el estado de los sensores ADXL y FSR. También se muestra la variación de la inclinación:

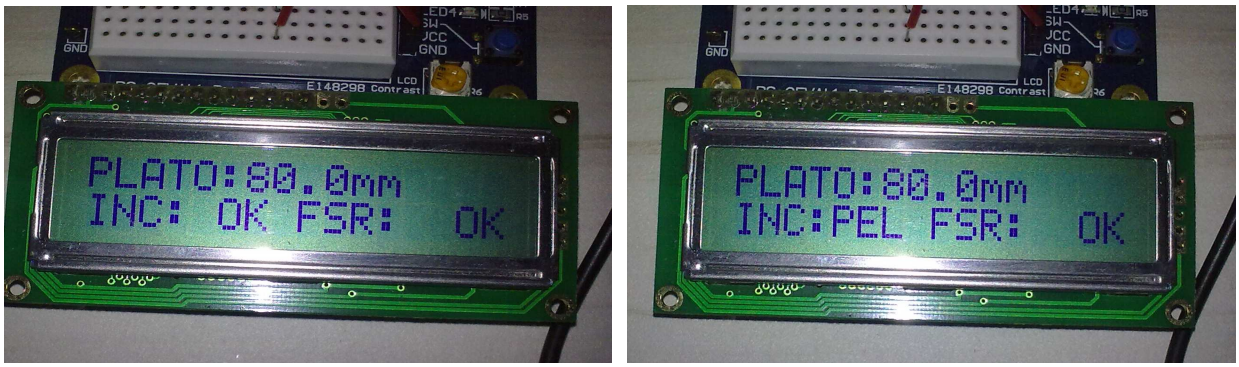


Fig. 4.99. Estado del LCD con el sistema en funcionamiento

En el caso de que el interruptor ON_OFF sea activado, en el LCD se mostrará confirmación y comenzará el envio de datos de los sensores al hiperterminal. Cuando se descative este interruptor, en el hiperterminal se mostrará un mensaje de finalización y el sistema volverá a la pantalla anterior conservando las lecturas previas.

Fig. 4.100. Confirmación en LCD y datos en el hiperterminal

Todos los test pertinentes habían sido realizados, incluso el funcionamiento del puerto serie y la visualización de los datos por el hiperterminal, de manera satisfactoria. Por tanto, y llegado a este punto, era momento de comenzar el diseño de la PCB del prototipo así como la programación en Matlab de un sistema de representación gráfica de las señales arrojadas por el puerto serie a través del PSoC.

4.5 Diseño en Protel de prototipo final

Altium Designer (Protel DXP) es un software diseñado para la creación de placas de circuito impreso. Ya había tenido contacto con este software a lo largo de las asignaturas de la carrera pero nunca había profundizado en la utilización del mismo.

Fig.4.101. Pantalla de inicio de Altium Designer

El primer paso para la creación de un proyecto consiste en la creación del plano esquemático o esquema de circuito, que deberá incluir todos los elementos que formarán parte del sistema final, para posteriormente importar ese plano (junto con los footprints o planta física de los componentes) al apartado de diseño de la PCB física, que permitirá rutear (conexión de las pistas o caminos del circuito) los componentes automática o manualmente y finalmente, procedería a la fabricación de la misma.

Por tanto, este apartado será dividido en varias partes: A. Diseño del plano esquemático, B. Diseño de la PCB y C. Resultados finales.

A. Diseño del plano esquemático.

El diseño del plano esquemático lo realicé solo después de haber concluido la creación del software del sistema, que determinaría los componentes y la disposición de los mismos. Para abordar la explicación del diseño de este plano lo realizaré por partes, mostrando cada uno de los bloques operativos hasta concluir con el plano conjunto.

- Alimentación:

El prototipo será alimentado a través de la batería de 12V de la máquina cortacésped. Por un lado, el PSoC debe ser alimentado a 5V y por el otro, el sensor ADXL puede ser alimentado como máximo a 3.3V, por lo que requería de una fuente de alimentación que redujera dichos 12V a los valores deseados. A través de sendos reguladores de tensión, el LM7805 y el LM317 respectivamente, se conseguiría reducir esa tensión a los valores que requiere el sistema. Consultando la configuración adecuada de estos componentes en sus datasheets, la fuente de alimentación del prototipo será la que se puede apreciar en la siguiente imagen:

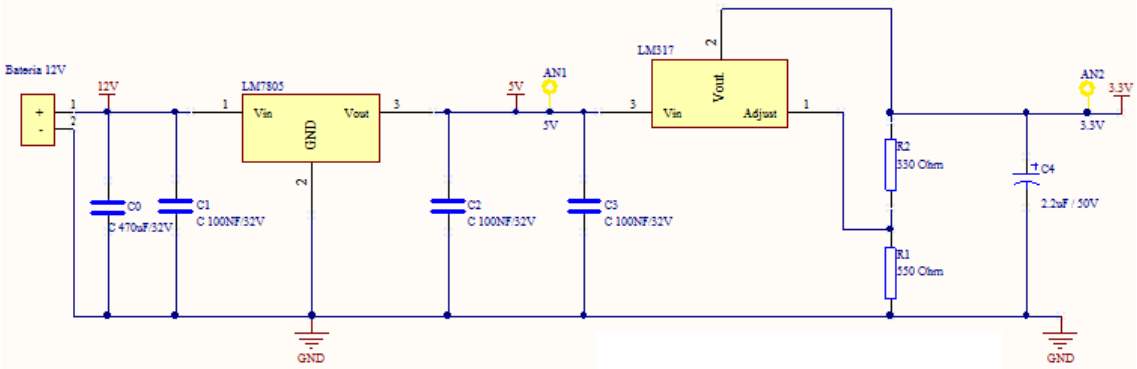


Fig.4.102. Fuente de alimentación del prototipo

Mediante una conector de 2 entradas, introduciremos tanto la señal de 12V como la masa de la batería en el circuito. Para desacoplar la entrada y filtrar frecuencias parásitas no deseadas colocaré un condensador de 470µF.

Después está colocado el primer regulador de tensión, el LM7805, que reducirá la tensión de entrada a 5V con sendos condensadores de desacoplo tanto a la entrada como a la salida del mismo y con valor de 100nF. Conectado a la salida de 5V se situa el LM317, cuya configuración es prácticamente inversa a la del anterior.

La entrada de este regulador será la última patilla (la 3), la salida será la segunda y en la tercera debemos añadir un divisor de tensión para determinar la tensión a la salida del regulador. Eligiendo los valores de 330Ω y 550Ω conseguiré una tensión en la salida de 3.3V. También he incluido sendos condensadores de desacoplo, siendo en este caso en la salida de 2.2µF (recomendado por el fabricante).

La inclusión de dos anillas de medida resultará útil para comprobar en cualquier momento la tensión resultante a la salida de ambos reguladores.

- Sensor de inclinación ADXL330

Este sensor, como ya he explicado anteriormente, detecta aceleraciones en sus tres ejes. Yo solo necesito tomar las lecturas de los ejes X e Y, por lo que el pin correspondiente con el eje Z no será utilizado. Nuevamente, a través del datasheet realicé la conexión de sus pines:

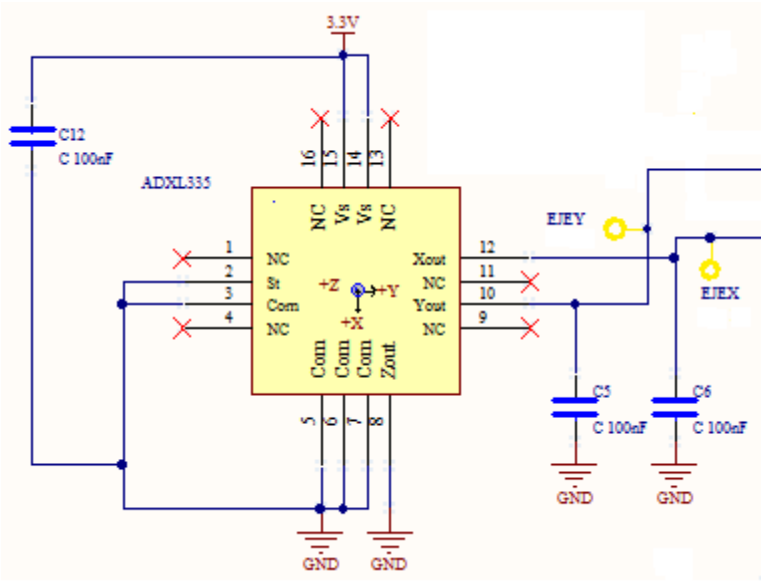


Fig.4.103. Conexiones del ADXL330

Como se puede observar en la fig.4.95, los pines que no tienen conexión interna en el integrado están marcados con una X roja, indicando que no serán conectados (pines 1, 4, 9, 11, 13 y 16).

Entre los pines 14 y 15(alimentación del integrado, a 3.3V) y los pines 5, 6 y 7 (masa del integrado) colocaré un condensador de desacoplo de 100nF

El pin 2, nombrado como St (self test) no me será necesario y por tanto, lo conectaré a masa.

El pin 3, con el nombre Com, debe ser conectado con masa, de acuerdo con el datasheet.

Como ya he comentado, no requiero de la medición del eje Z (pin 8), por lo que lo conectaré a masa. Los pines 10 y 12 se corresponden con las señales de los ejes Y e X respectivamente, los cuales tienen conectados sendos condensadores de 100nF a masa para desacoplar estas señales de la alimentación (ambos están recomendados por el fabricante).

También he añadido 2 anillas de medida para comprobar en todo momento las señales arrojadas por el sensor.

- Sensor FSR: Este sensor se alojará en el cesto de recogida del césped, por lo que en la PCB sólo colocaré un conector para insertarlo.

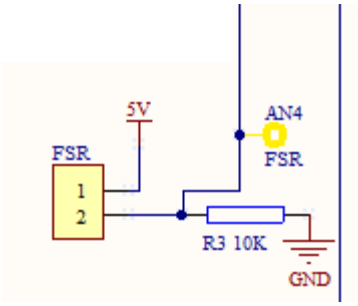


Fig.4.104. Conexiones del FSR

Como ya fue explicado en el apartado relativo a este sensor, para su correcto funcionamiento se colocará en un divisor resistivo con una resistencia de 10KΩ.

También he añadido una anilla de medida para comprobar la señal a la salida de dicho divisor resistivo y comprobar el correcto funcionamiento del sensor.

- Sensor Hall: Al igual que con el sensor FSR, este sensor se colocará en uno de los laterales del plato de corte de la máquina, por lo que en la PCB solo aparecerá un conector.

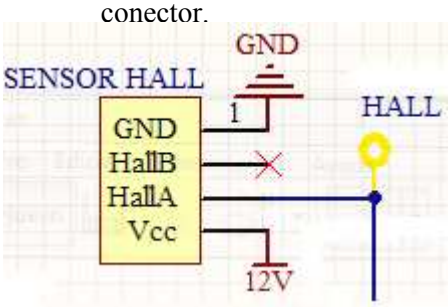


Fig.4.105. Conexiones del Hall

Como ya se explico en el correspondiente capítulo, sólo usaremos una de las señales arrojadas por el sensor, por lo que la otra señal no la rutearé. Este sensor se alimenta a 12V por lo que el conector contendrá sendas señales de Vcc y masa para alimentarlo.

En este sensor también he añadido una anilla de medida para comprobar el funcionamiento de la señal aroojada por el sensor.

- Comunicación serie: Para la comunicación serie, he tenido que añadir un integrado RS232 y un conector DB9 hembra.

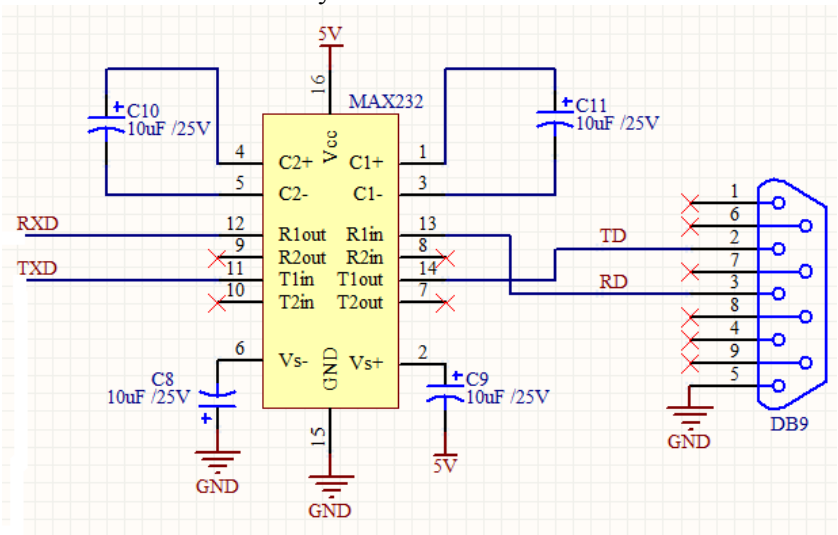


Fig.4.106. Conexiones para comunicación serie

Las conexiones son idénticas a las existentes en la placa de pruebas del PSoC. El RS232 requiere de 4 condensadores de 10µF colocados en los pines 1, 2, 3, 4, 5 y 6.Los pines 11 y 12 serán las señales TXD y RXD respectivamente, que se conectarán a los pines del PSoC, y por otro lado, los pines 13 y 14 se corresponderán con las señales que se conectarán al conector DB9, quedando los demás pines sin conexión, a excepción del pin 5, que estará conectado a masa.

- Sistema de alarmas: Las alarmas del sistema las conforman cuatro diodos LED y un piezorresistivo. Cada uno de los diodos LED tendrán una resistencia de 330Ω colocada en serie para asegurar el suministro de corriente.

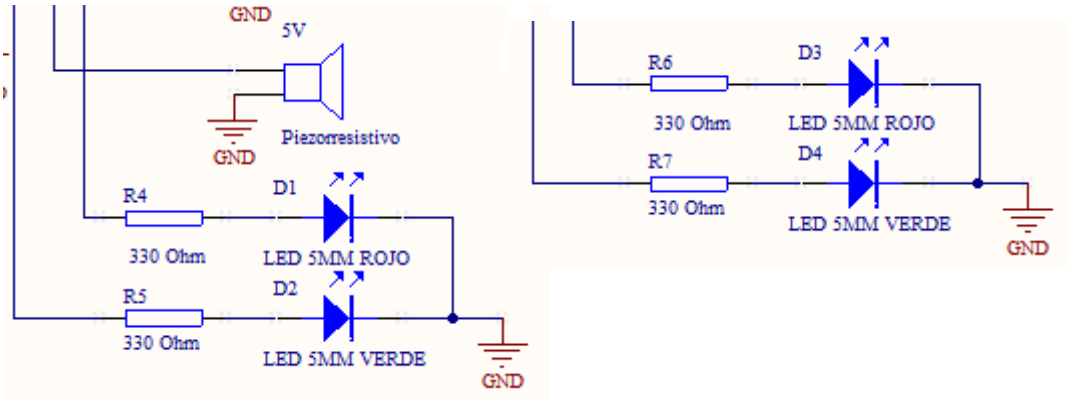


Fig.4.107. Alarmas del sistema

- Botones de interacción y LCD: Es necesario colocar 2 botones para subir o bajar el plato de corte y un interruptor encender el modo de envío de datos por el puerto serie. El LCD se representará por un conector de 14 pines

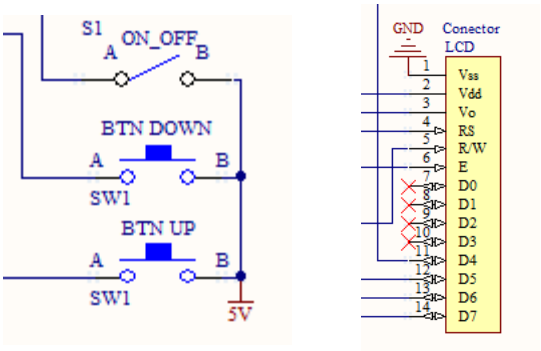


Fig.4.108. Botones de interacción

- PSoC: Una vez colocados todos los elementos, resta la colocación del microcontrolador y la conexión de los mismos.

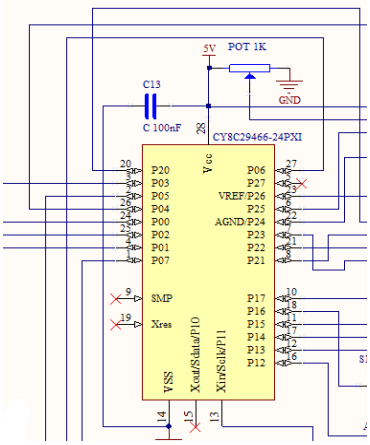


Fig.4.109. Conexiones del PSoC

Como ya se ha visto en la situación final de los pines del PSoC en el apartado anterior, los puertos P2 [7] y P1 [0] no se usarán por lo que no serán ruteados. De igual manera, los pines 9 y 10 no serán necesarios.

Entre el pin de alimentación y masa he añadido un condensador de desacoplo de 100nF.

Una vez mostrados todos los elementos que conforman el plano esquemático, mostraré el plano conjunto del sistema donde aparecen todos los elementos y seguidamente mostraré el desarrollo de la PCB a partir de la importación de los elementos mostrados en el esquemático.

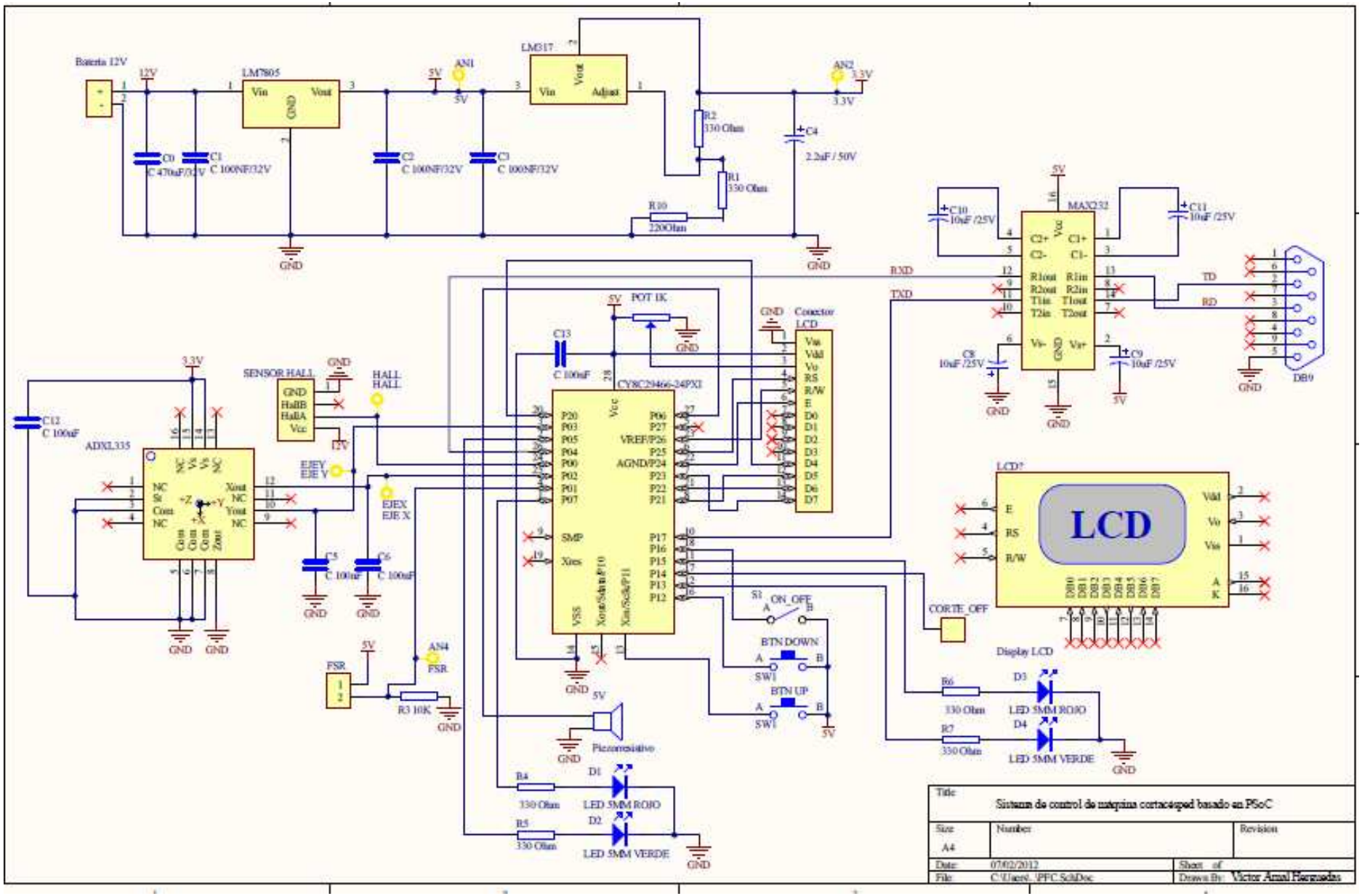


Fig.4.110. Esquemático de los componentes del diseño del prototipo

B. Diseño de la PCB

Una vez configurados los footprints de cada uno de los componentes del plano esquemático y a su importación al plano PCB, el primer paso consiste en la colocación o placement de los componentes sobre la placa. Es importante colocar los elementos por bloques para facilitar su posterior ruteado. Una primera colocación de los componentes es la siguiente:

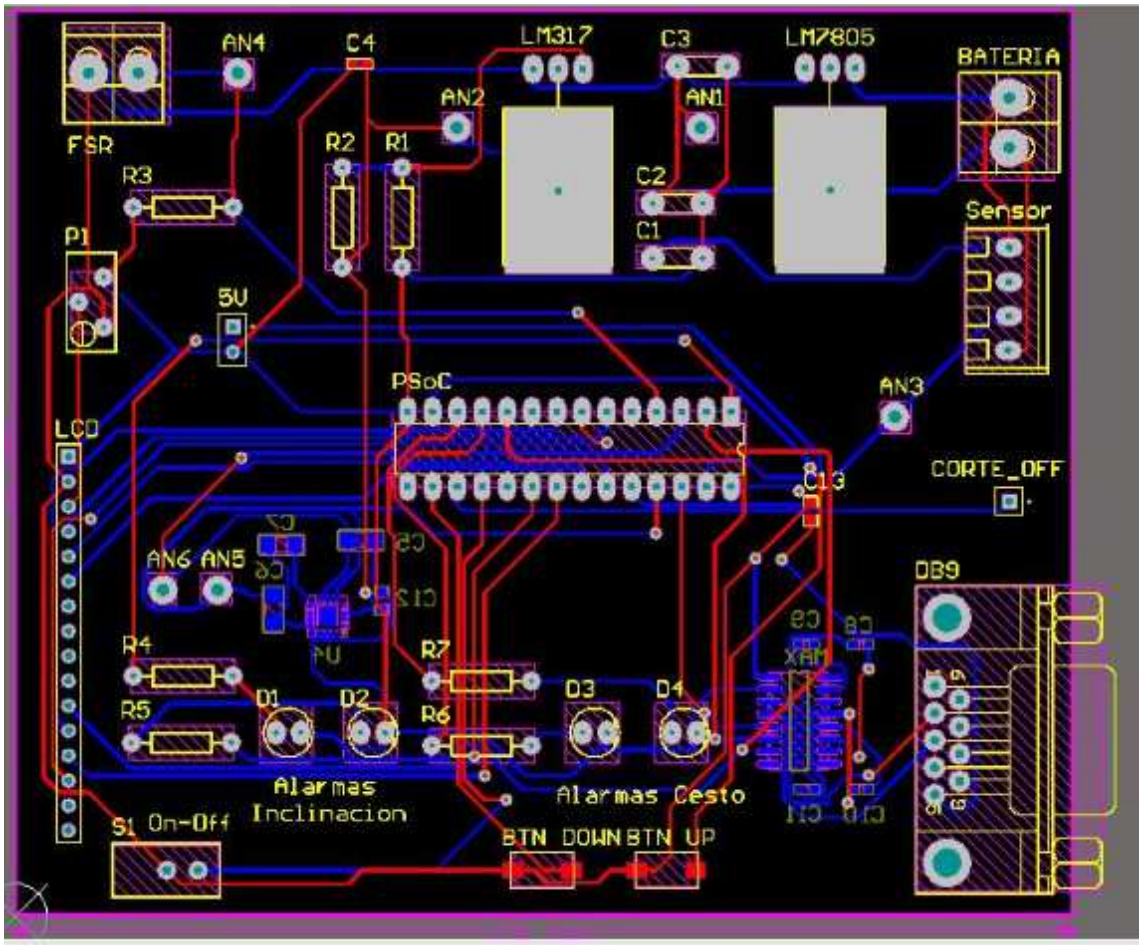


Fig. 4.111 Primera colocación de los componentes sobre la PCB

Como se puede, existe mucho espacio vacío entre los elementos, por lo que todavía se podía mejorar considerablemente la colocación. También se puede apreciar que los condensadores de desacoplo de los componentes que los requieren no están los suficientemente cerca de estos, lo que puede disminuir la eficacia de los mismos y por ello, deben colocarse lo más cerca posible a los pines de los componentes que los requieran.

En este primer placement, los footprints tanto de las resistencias como de la mayoría de los condensadores los había elegido THD (Through Hole Device), es decir, resistencias y condensadores comunes de patillas los cuales ocupaban un espacio considerable. En este aspecto la PCB también podía ver reducido su tamaño escogiendo resistencias y condensadores SMD (Surface Mount Device) cuyo tamaño es muchísimo más pequeño y, normalmente, mucho más precisos.

Por otro lado, el ruteado de las pistas en este primer placement fue realizado de manera automática por la función de AutoRoute, o ruteado automático, de Altium. Mientras que muchas de las rutas elegidas por el programa no eran las más adecuadas, este ruteado presentaba innumerables vías para conseguir conducir todas las señales a sus respectivos componentes.

En definitiva, tras realizar una nueva colocación, y a pesar de un mayor esfuerzo, el ruteo de las pistas lo haría de forma manual, eligiendo en todo momento las mejores opciones para conectar todas las señales.

Otro de los aspectos a tener en cuenta, son las anchuras y diámetros tanto de las pistas, como de los pads y de las vías. Es necesario controlar estos valores para que, por ejemplo, la máquina que realiza la PCB pueda crear correctamente lo diseñado en Altium o que cuando realice los agujeros de la placa no quite ningún pad con el taladro (aspecto que ya sufrí anteriormente por no tener en cuenta estos valores).

Estos valores mínimos que aseguren una correcta creación de la PCB y su posterior montaje son los siguientes:

- Clearances: Son las distancias mínimas que existirán entre una pista y un pad, sin que pueda suponer un problema de cortocircuito o similar. Este valor he situado a un valor mínimo de 0.3mm

- Anchura de las pistas: Para asegurar que las señales circulan correctamente debemos elegir un valor suficiente de las pistas pero además, hay que tener en cuenta que la máquina que las plumas que realizan la PCB tienen una precisión máxima. Por tanto, he elegido para las pistas un valor mínimo de 0.3 mm y un valor máximo de 0.6 mm, considerando un valor preferente de las mismas de 0.4 mm.

  Tras consultar con los maestros de laboratorio la precisión máxima de la máquina de creación de las PCB, la anchura mínima de las pistas no podía ser menor de 0.257mm por lo que la elección de un valor de 0.4 mm era aceptable. El único componente que podría comprometer este valor es el integrado del ADXL335. Ya que al establecer el valor de 0.4mm de las pistas como regla de diseño, Altium considera como error el tamaño de los pads del sensor.

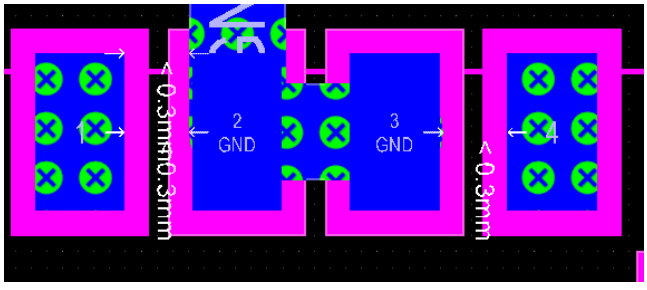


Fig. 4.112 Detalle de los pads del sensor ADXL

Como es un valor que no se puede modificar, comprobé si era posible la utilización de este integrado estableciendo como prueba la anchura mínima de las pistas a 0.257mm

Al realizar el cambio, todos los elementos de la placa aparecen como correctos y por tanto podía utilizarlo, ya que se puede realizar en la PCB física.

Algo similar ocurre con el condensador de desacoplo de 470 uF colocado entre el conector de la batería y el regulador LM7805. Esta pista, la he realizado con una anchura muy superior a la de sus análogas para asegurar un buen apantallamiento de la señal de masa del sistema y evitar la entrada de posibles señales parasitas.

Una vez considerados y realizados los cambios, la disposición de los componentes sobre la PCB será la siguiente. Además se muestra el ruteado de la cara TOP, de la cara BOTTOM y una vista conjunta de la PCB:

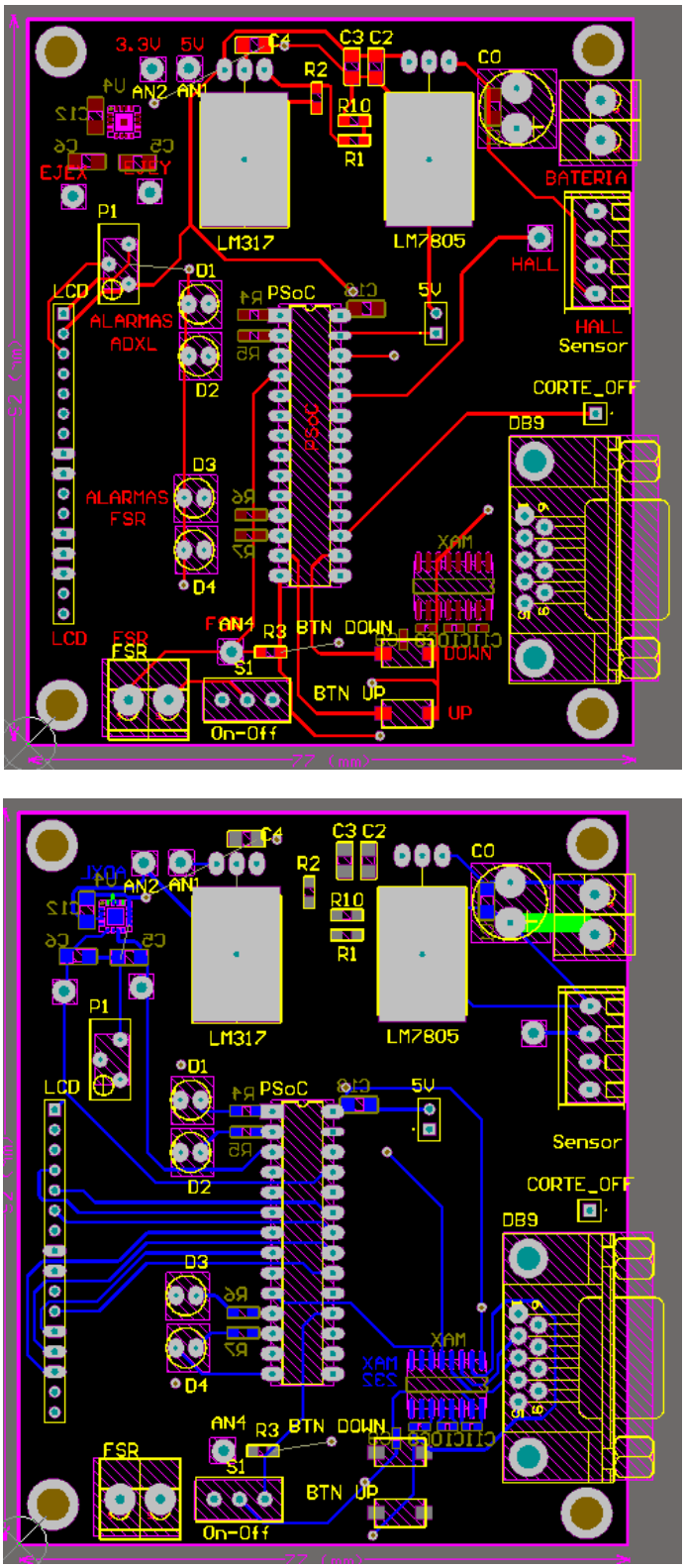


Fig. 4.113 Pistas en cara TOP (rojo) y en cara BOTTOM (azul)

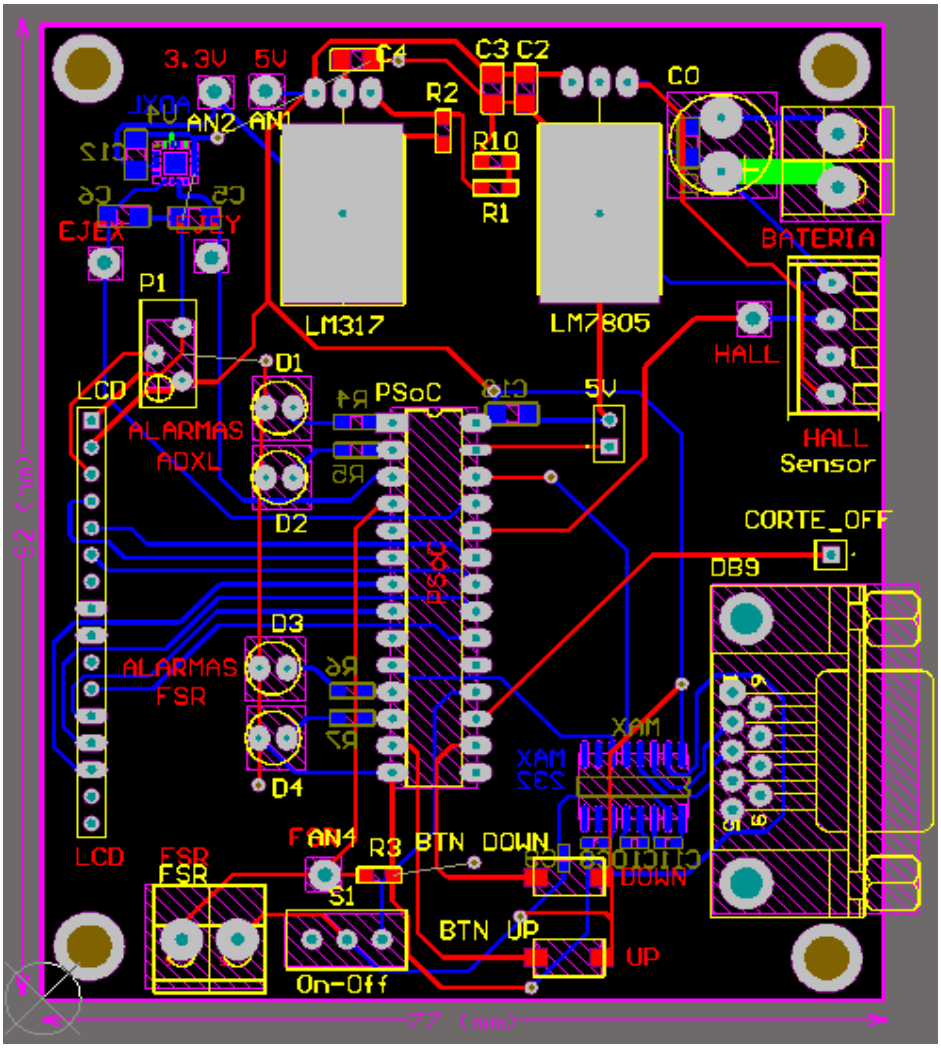


Fig. 4.114 Conjunto de pistas de la PCB

Como se puede apreciar, el tamaño se ha disminuido considerablemente, los condensadores de desacoplo se encuentran, en la mayoría de los casos, pegados a su pin correspondiente asegurando el cumplimiento de su función, el ruteado realizado manualmente consigue la reducción de los pads utilizados y una mejor distribución de las mismas. Al haber realizado un diseño a doble cara, facilita la compresión de los componentes sobre las superficies de la PCB.

Como se puede apreciar, aparecen en color verde la pista de masa del condensador de desacoplo de 470 µF y de las pistas que unen los pads del sensor ADXL. Como ya he explicado con anterioridad, no es necesario tenerlos en cuenta ya que, aunque están fuera de los valores establecidos, no suponen ningún problema para la realización de las mismas.

Realizado el placement y el ruteado de los componentes, he procedido a la creación de los planos de masa de ambas caras de la PCB. También he añadido algunas vías que conecten ambos planos de masa para asegurar que queda totalmente aislada de señales parasitas o ruido.

En la siguiente imagen se muestra una imagen del plano de masa de la cara TOP:

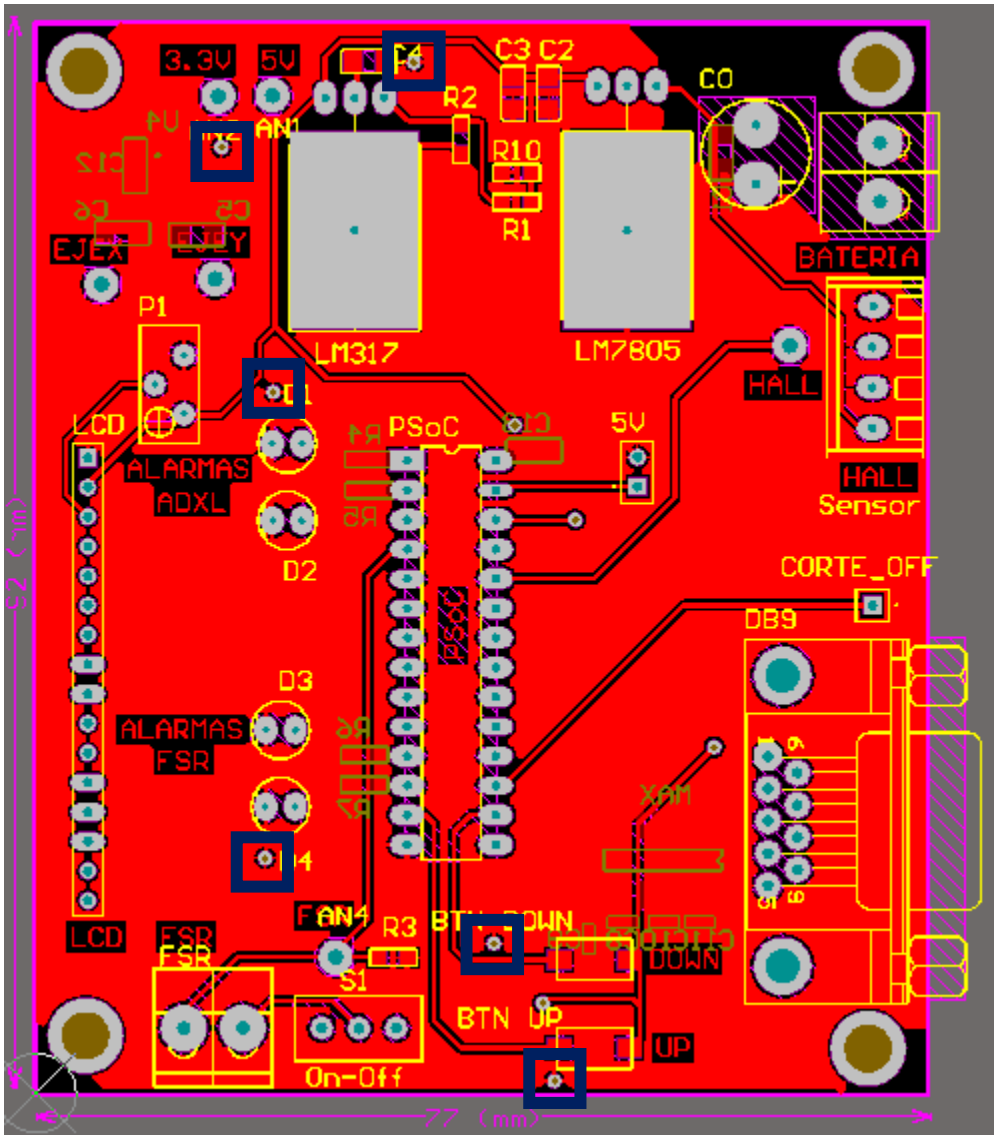


Fig. 4.115 Plano de masa de la cara TOP

En total he colocado un total de 6 vías para conectar los planos de masa de ambas caras. Cada uno de los componentes y conectores de interés han sido etiquetados para poder localizarlos a simple vista. En este caso no ha sido necesaria la corrección de las posibles islas que aparecen cuando se realizan los planos de masa.

Estas islas consisten en regiones aisladas del plano de masa que pueden actuar como antena y producir señales parásitas que podrían desestabilizar el sistema. Como se puede ver, en el extremo superior derecho de la PCB no se ha realizado plano de masa, si no que únicamente como se puede ver en el plano de masa de la cara BOTTOM, la pista que une el conector de la batería con el plano de masa la he agrandado considerablemente para asegurar que la señal de masa llega correctamente a todo el circuito.

Ahora mostraré una captura del plano de masa de la cara BOTTOM:

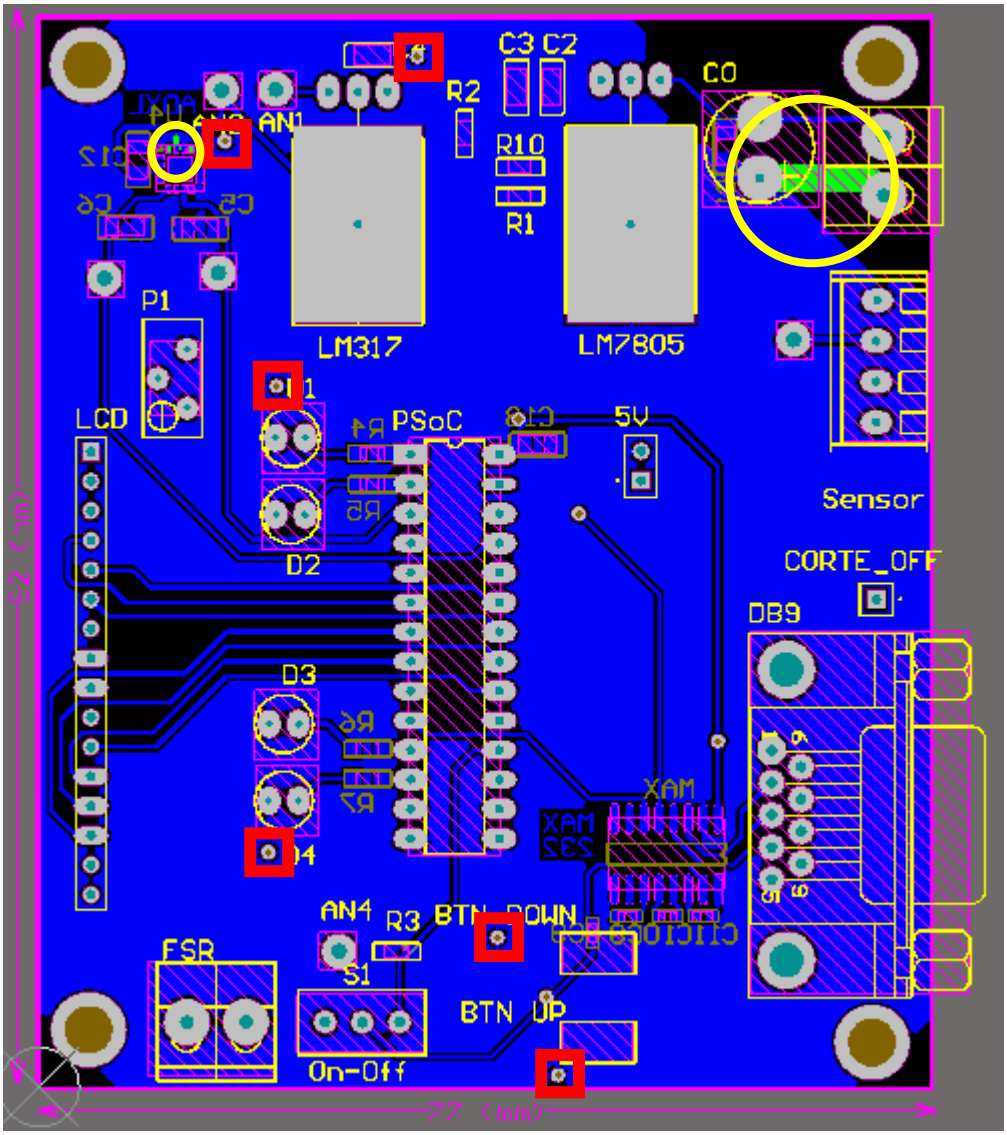


Fig. 4.116 Plano de masa de la cara BOTTOM

Además de las comentadas vías incluidas para conectar ambos planos de masa he señalado en la PCB con dos círculos amarillos los errores detectados por Altium que he citado anteriormente. Este error, como ya he dicho se debe a que, por un lado, la distancia mínima se ve comprometida en el caso de de los pines del sensor ADXL y, por el otro, la anchura máxima de la pista que conecta el conector de la batería con ambos planos de masa en considerablemente superior. No se debe prestar mayor atención, ya que en ambos casos he comprobado que no existe ningún tipo de problema.

Cabe destacar que en el plano de la cara BOTTOM he tenido que eliminar una isla que se había ocasionado entre las pistas correspondientes con las señales de control del LCD. Una vez eliminada, la PCB quedaba preparada para su fabricación.

Para finalizar este apartado, mostraré una imagen del estado final del PCB y de una vista en 3D que permite hacerse una idea del resultado final de la misma:

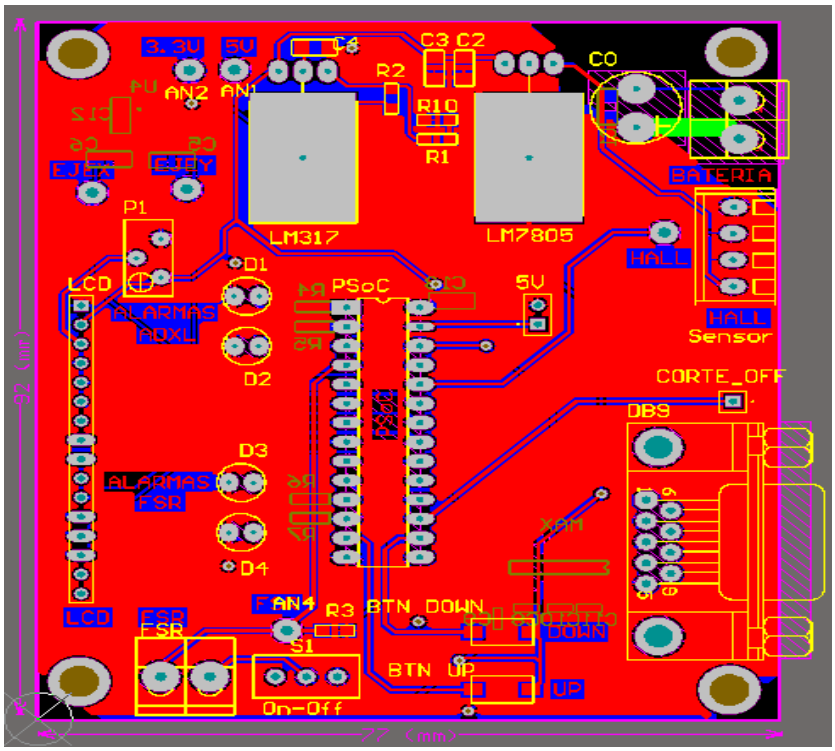


Fig. 4.117 Plano final de la PCB

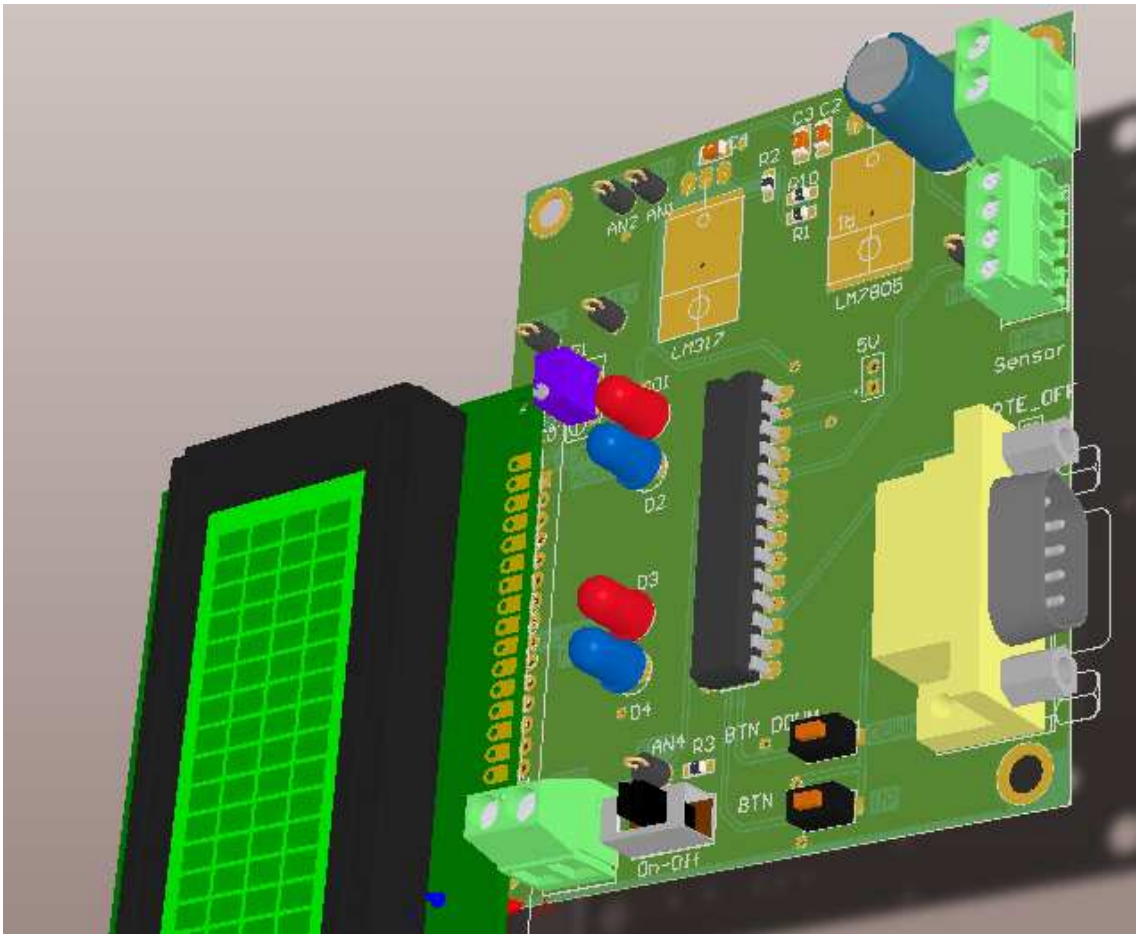
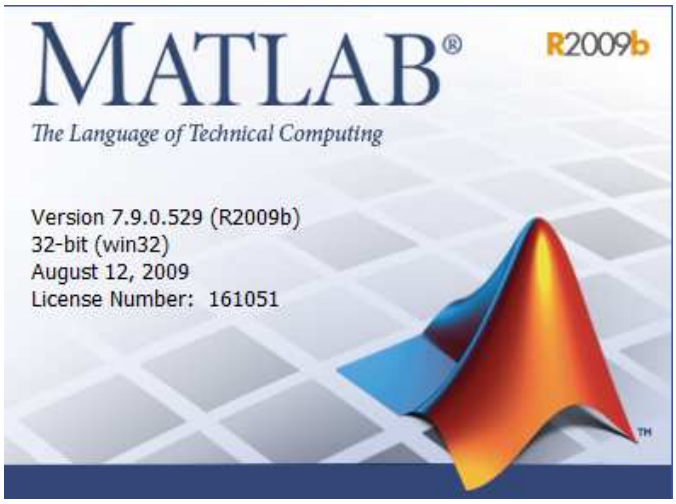


Fig. 4.118 Vista 3D del PCB

4.6 Aplicación software en Matlab

Matlab es un software matemático con un entorno de desarrollo integrado que permite manipular matrices de datos, representar gráficamente datos y funciones, creación de interfaces gráficas de usuario y comunicación con otros dispositivos hardware.



A partir de los datos de los sensores enviados por el PSoC a través del hiperterminal, como he explicado en el apartado 4.3, realizaré un pequeño programa en Matlab para representar gráficamente dichos datos y facilitar su interpretación. Como he comentado anteriormente, es un software que solo había utilizado anteriormente de manera superficial, por lo que debía consultar los archivos de ayuda incluidos en el mismo.

Fig. 4.119 Pantalla de inicio de Matlab

Una vez iniciado el programa, el interfaz de trabajo es el siguiente, donde detallaré sus apartados más importantes:

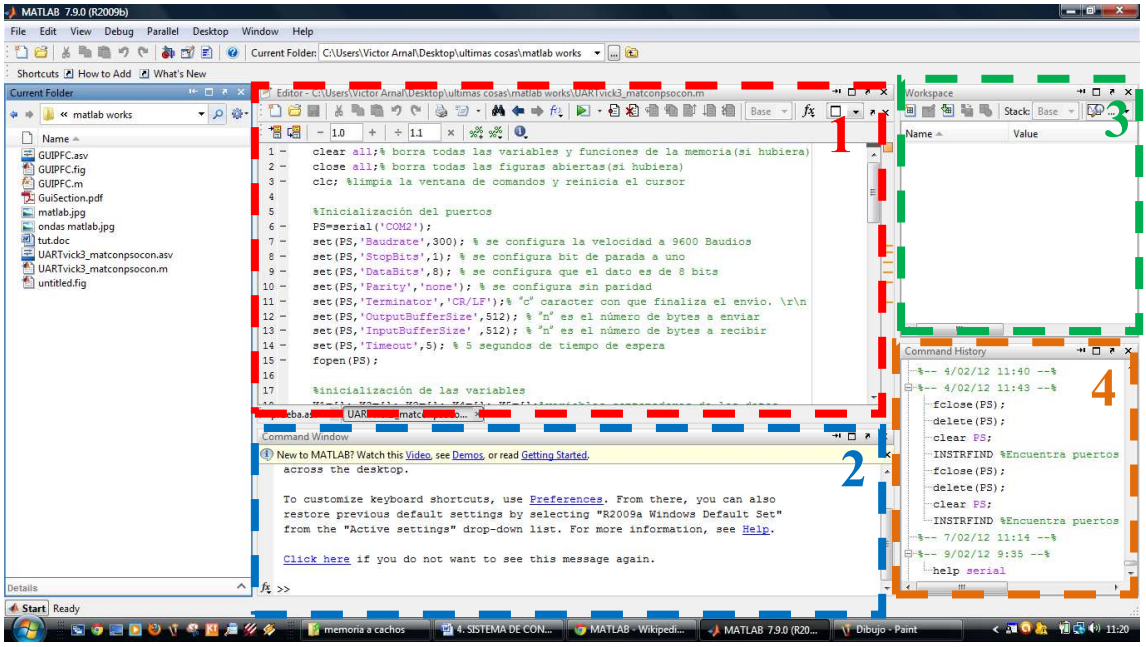


Fig. 4.120 Pantalla de explotación de Matlab

1. Editor de código: En esta ventana se confecciona el código propio de Matlab, el código M, que posteriormente se puede ejecutar a través del botón RUN que se sitúa en la parte superior de la ventana y tiene un icono como el siguiente: 

2. Ventana de comandos: En el caso de querer ejecutar únicamente una serie de instrucciones se utilizará esta ventana. Además mediante la introducción del comando help + palabra clave como por ejemplo, `help com`, accederemos a la ayuda de Matlab donde podremos encontrar extensos manuales de uso de las instrucciones consultadas. Estos manuales de ayuda me han sido de gran ayuda para poder desarrollar el código necesario para representar gráficamente los datos de los sensores.

3. Workspace: Durante la ejecución de un programa o fragmento de código, en esta ventana podremos ver todas las variables involucradas en el mismo, mostrando los valores que van adoptando. Resulta realmente útil porque puedes ver en tiempo real el estado de las variables del código realizado, lo que es muy ventajoso para la detección de posibles errores.

4. Historial de comandos: En esta ventana, se pueden consultar cuales han sido las últimas instrucciones utilizadas.

Una vez comentadas las ventanas del interfaz del programa, comentaré el software realizado para la representación gráfica de las variables recibidas por el puerto serie. Primero, he consultado la ayuda de Matlab con respecto a la captación de datos del puerto COM mediante el comando `help serial`. En este documento de ayuda, he conocido las instrucciones involucradas con la recepción y configuración de los datos recibidos por el puerto serie del PC.

Estas instrucciones serán las siguientes:

```
PS=serial('COM2'); %atribuimos los datos del puerto a una variable
set(PS,'Baudrate',300);%se configura la velocidad a 300 Baudios
set(PS,'StopBits',1); %se configura bit de parada a uno
set(PS,'DataBits',8); %se configura que el dato es de 8 bits
set(PS,'Parity','none'); %se configura sin paridad
set(PS,'Terminator','CR/LF');%carácter con que finaliza el envío(\r\n)
set(PS,'OutputBufferSize',512);%se configura número de bytes a enviar
set(PS,'InputBufferSize' ,512);%se configura número de bytes a recibir
set(PS,'Timeout',5); % 5 segundos de tiempo de espera
fopen(PS); %realizo la conexión entre el puerto COM físico y virtual
```

Una vez configuradas las características del puerto COM, tengo que crear una serie de variables a las que le introduciré los datos recibidos y después una rutina que los represente gráficamente:

```
%inicialización de las variables
K1=[]; K2=[]; K3=[]; K4=[]; K5=[];%variables contenedoras de los datos
encendido=1;                      %recibidos por el puerto serie
```

Desde el PSoC, enviare los datos en formato de cadena (STRING), por lo que las instrucciones en Matlab de recepción de datos estarán condicionadas a este. Los datos se enviarán sin haber escalado el dato, por lo que lo realizaremos.

```
while (encendido==1)

 Instruccion=fscanf(PS,'%s');
 Dato=fscanf(PS,'%s');
 switch(Instruccion)%dependiendo de la instrucción recibida introduce
  case 'FSR'       %los datos en cada una de las variables declaradas
        K1=[K1 str2num(Dato)*0.01953125]

  case 'AX'
        K2=[K2 str2num(Dato)*0.01953125]

  case 'AY'
        K3=[K3 str2num(Dato)*0.01953125]

  case 'HALL'
        K4=[K4 str2num(Dato)]

  case 'fin'
        encendido=0;% Mensaje que le hace salir del bucle

  otherwise
        K5=[K5 str2num(Dato)]

  end
   %Instrucciones para representar graficamente las variables
   figure(1)
   hold off;
   plot(K1,'g')
   hold on;
   plot(K2,'r')
   hold on;
   plot(K3,'b')
   hold on;
   plot(K4,'y')
end
```

Como se puede observar, y de manera condicionada por los datos recibidos por el PSoC, primero buscaré la recepción de la instrucción que me dirá a quien le corresponde el dato recibido. Mediante una estructura switch enumeraré las opciones posibles que corresponderán con los sensores de los que tomaré las señales y una opción de finalización de envío de datos. En el caso de que no se esté recibiendo ninguna de las "etiquetas" establecidas, el código introducirá esos datos en otra variable.

En el caso de recibir una instrucción válida, capturaré el dato (`Dato=fscanf(PS,'%s')`) recibido en formato STRING, lo convertiré mediante la instrucción `str2num(Dato)` a un array de números para poder representarlos gráficamente y lo escalaré para obtener el valor real arrojado por los sensores. En el caso del sensor hall no será necesario debido a la naturaleza de dicha señal, ya que se que será un valor que oscilara entre 0 y 5V.

Por lo tanto, mientras el interruptor citado este activado, se representarán constantemente los datos recibidos por el puerto serie a través de la declaración de la grafica mediante la instrucción `figure(1)`y mediante la escritura en la misma de los valores de los sensores a través de esta otra instrucción `plot(K1,'g')` donde estableceremos que variable y con qué color quiero dibujarla.

Por otro lado, cuando el botón ON_OFF de la PCB sea colocado en la posición OFF el PSoC finalizará el envio de datos, y por tanto, deberé detener la representación gráfica. Debido a esto, cuando se reciba la instrucción fin, saldremos del bucle while finalizando la ejecución del programa. Antes de finalizar la ejecución del programa, es necesario cerrar el puerto COM para que para una posterior ejecución, Matlab detecte como disponible el puerto.

Esto me conllevó problemas para ejecutar más de una vez el código sin la necesidad de reiniciar Matlab, ya que sin estas instrucciones entre una ejecución y la siguiente, me detectaba un error en el puerto COM, señalándome que el puerto estaba en uso y Matlab no podía acceder a él. Por ello, y mediante la introducción de las siguientes instrucciones el problema queda resuelto:

```
%Instrucciones para cerrar el hiperterminal
fclose(PS); %Cierra el  puerto COM2
delete(PS); %Borra la variable asociada al puerto COM2
clear PS;   %Borra la variable del Workspace
INSTRFIND %Encuentra puertos serie con los valores especificados
```

La gráfica que aparece al ejecutar el código es la siguiente, que variará dependiendo de cómo varíen los sensores:

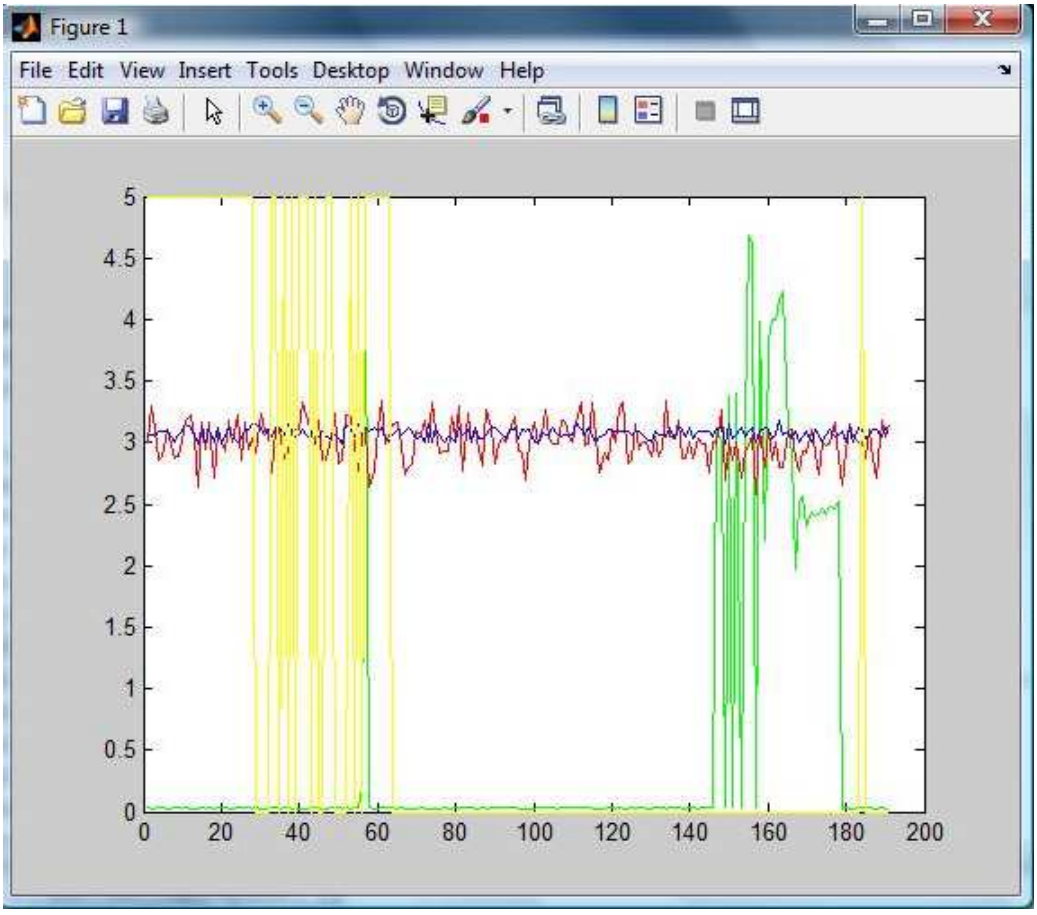
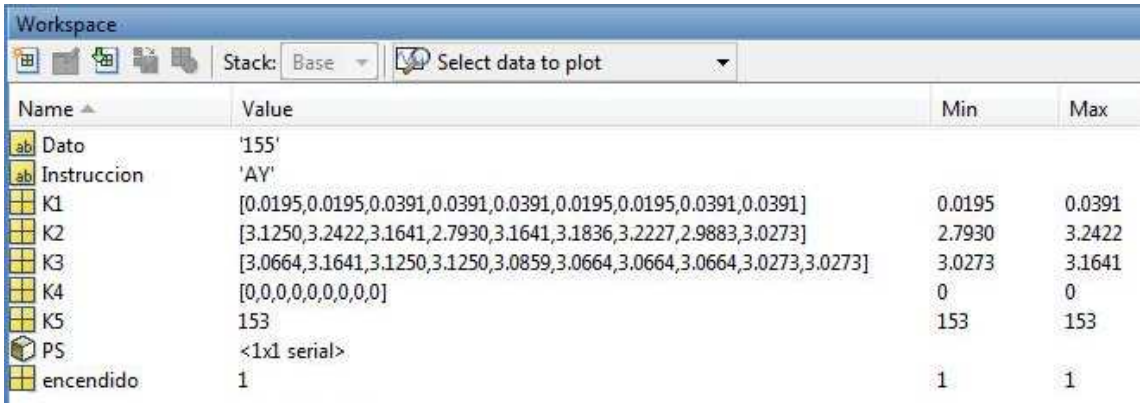


Fig. 4.121 Representación gráfica de los valores arrojados por los sensores

La gráfica representa la variación de la tensión de los sensores con respecto al tiempo transcurrido: Como se puede observar los valores oscilarán entre 0 y 5V.

Una vez realizada la ejecución y habiendo observado la representación de los valores, puede resultar de utilidad consultar los valores que aparecen en la ventana del Workspace para consultar el estado de las variables utilizadas. En el caso de la representación gráfica mostrada, los datos aparecidos en esta ventana son los siguientes:



Name	Value	Min	Max
Dato	'155'		
Instruccion	'AY'		
K1	[0.0195,0.0195,0.0391,0.0391,0.0391,0.0195,0.0195,0.0391,0.0391]	0.0195	0.0391
K2	[3.1250,3.2422,3.1641,2.7930,3.1641,3.1836,3.2227,2.9883,3.0273]	2.7930	3.2422
K3	[3.0664,3.1641,3.1250,3.1250,3.0859,3.0664,3.0664,3.0664,3.0273,3.0273]	3.0273	3.1641
K4	[0,0,0,0,0,0,0,0,0]	0	0
K5	153	153	153
PS	<1x1 serial>		
encendido	1	1	1

Fig. 4.122 Valores de las variables tras la ejecución del programa en el Workspace

Se puede observar, que tras la recepción de una instrucción válida (en este caso la correspondiente con el eje Y del sensor ADXL, AY) y posteriormente del dato (en este caso 155, evidentemente sin escalar), en los contenedores K1-K5 aparecerán los valores entre 0 y 5V de los valores de los sensores.

También se puede observar, que en ausencia de una instrucción válida, un dato no ha sido aceptado y se ha guardado en el contenedor K5.

Como conclusión, he considerado Matlab como una herramienta muy útil para el desarrollo de pequeñas aplicaciones que permitan visualizar gráficamente el correcto funcionamiento de un sistema. En este caso, esta aplicación es útil para que un técnico pueda realizar un diagnostico de los sensores colocados sobre la máquina cortacésped y visualizar fácilmente su correcto funcionamiento o si alguno de ellos presenta alguna anomalía.

5. CONCLUSIONES

5.1 Resultados obtenido

Después del trabajo realizado en este proyecto final de carrera, puedo decir que he logrado alcanzar los siguientes objetivos:

- He conseguido identificar las necesidades y especificaciones necesarias para realizar un sistema que controlará una serie de variables de interés para el usuario de una máquina cortacésped.
- He diseñado un sistema de control lo suficientemente genérico para poder incluirlo en cualquier máquina cortacésped.
- He realizado una placa PCB propia con un PSoC sin tener que depender de ninguna tarjeta externa comercial.
- He realizado, soldado y utilizado hasta un total de 6 placas de puntos correspondientes al conocimiento del manejo de cada uno de los sensores de forma independiente: Sensor ADXL, sensor FSR, sensor Hall, sensor de inclinación basado en el péndulo, sistema de alarmas y sistema de visualización mediante LCD.
- He realizado la programación en C de cada uno de los sistemas mencionados comentándolos adecuadamente para que no resulte un problema el entendimiento del mismo a un nuevo usuario.
- Se han asentado las bases de desarrollo para una futura línea de avance de modernización de las máquinas cortacésped actuales

La realización de este proyecto final de carrera perseguía el objetivo final del diseño e implementación de un sistema de control para mejorar la seguridad y calidad del trabajo del usuario de una máquina cortacésped. Este objetivo final se ha conseguido.

5.2 Mejoras y ampliaciones propuestas

A lo largo del presente proyecto final de carrera han sido observadas posibles mejoras que en un futuro podrían ser realizadas para mejorar este primer prototipo de un sistema de control para una máquina cortacésped:

- La realización de una PCB con unas dimensiones reducidas al máximo posible. Esto permitiría reducir el coste de la misma mediante la optimización del placement de los componentes sobre la misma.
- Depurar el código C del sistema de control e incluir un sistema de representación gráfica de los valores de llenado del cesto y del grado de inclinación en el LCD para una lectura más fácil por parte del usuario.
- Desarrollo de una interfaz gráfica de usuario (GUI) en Matlab para que la representación gráfica de las señales arrojadas por los sensores ofrezca una mayor estética.
- Incrementar el número de sensores sobre la máquina para, aumentando las prestaciones del sistema de control y mejorando la experiencia de uso para el usuario de la máquina.

5.3 Cronograma

La siguiente tabla se corresponde con los periodos de tiempo que he empleado para la confección de la memoria de este proyecto final de carrera. Como se puede apreciar, existe un periodo de dos meses, concretamente junio y julio de 2011, en los que estuve realizando trabajos para la empresa que no tenían relación con este proyecto, por lo que aparecen marcados en rojo. El periodo de estancia en la empresa italiana aparece marcado en verde.

Punto del desarrollo/Fecha	Nov '10	Dic '10	Ene '11	Feb '11	Mar '11	Abr '11	May '11	Jun '11	Jul '11	Ago '11	Sep '11	Oct '11	Nov '11	Dic '11	Ene '11	Feb '11
Estancia en empresa																
Estudio de sensores existentes																
Conocimiento y estudio del PSoC																
Realización primeros subsistemas y placa de puntos																
Pruebas de los subsistemas sobre las máquinas																
Realización de otras tareas en la empresa																
Confección de documentación sobre el PSoC																
Diseño y realización del software del prototipo																
Primer diseño conceptual del prototipo PCB																
Realización de la aplicación en Matlab																
Realización y tests del PCB del prototipo																
Confección demás aspectos de la memoria																
Finalizar la Documentación																

Fig. 5.1. Cronograma del Proyecto Final de Carrera

5.4 Conclusiones personales

Las conclusiones personales que considero más importantes podrían ser las siguientes:

- He adquirido una visión más cercana y real del mundo laboral
- He estado trabajando casi nueve meses en un equipo de diseño de prototipos muy activo y he valorado la experiencia como algo motivador y gratificante.
- He podido aprender distintas herramientas informáticas que me serán de mucha ayuda en un futuro profesional (Altium Designer, Matlab, PSoC Designer).
- He aprendido a utilizar un tipo nuevo de microcontrolador y, en mi opinión, mi nivel de programación en lenguaje C ha aumentado considerablemente.
- He podido experimentar en primera persona todo el proceso de elaboración de una placa de circuito impreso, desde el diseño del esquemático y selección de componentes hasta la soldadura de éstos.
- He tenido la posibilidad de abracar la totalidad de los frentes que presenta un proyecto y disfrutar de la madurez profesional que esto supone.
- La estancia durante nueve meses en Italia, me ha permitido adquirir un nivel tanto escrito como hablado aceptable, considerándolo como un gran avance personal.

6. BIBLIOGRAFIA

6.1 Bibliografía

"Programmable System on Chip Technical Reference Manual", Cypress Semiconductor 2011

"PSoC CY3120- PSoC Eval1 Guide Reference Manual", Cypress Semiconductor 2011

"C language Complier User Guide", Cypress Semiconductor 2005

"Schematic Editor Basics", Altium Designer

"PCB Editor Basics", Altium Designer

"Design Rules", Altium Designer

"Routing and polygons", Altium Designer

6.2 Páginas de internet consultadas

http://www.google.es/

http://es.wikipedia.org/wiki/

http://www.psocdeveloper.com/forums/

http://www.mathworks.es/

http://www.cypress.com/

http://www.sparkfun.com/

6.3 Agradecimientos

Me gustaría agradecer a todas las personas que de algún u otro modo han estado involucradas en la realización de este proyecto fin de carrera.

En primer lugar me gustaría agradecer a mi director de proyecto Antonio Bono por asesorarme ofrecerme ayuda en la medida de lo posible a lo largo de la confeccion de este PFC.

Me gustaría agradecer también a los maestros de laboratorio Álvaro Gragera por toda la ayuda prestada y por la celeridad en la fabricación del PCB.

Como mención especial, agradecer infinitamente a mi compañero y amigo Tomás Cabeza por su predisposición incondicional para echarme una mano cuando la necesitaba. ¡Muchas gracias amigo!

Por último el agradecimiento infinito para mis padres, mi hermano y mi novia que me apoyaron en todo momento para alcanzar el sueño de convertirme en lo que deseo ser.

De nuevo, muchas gracias a todos.