



UNIVERSIDAD DE PAMPLONA
Una universidad incluyente y comprometida
Con el desarrollo integral



**DESARROLLO DE UN MANUAL DE PRÁCTICAS DE
LABORATORIO, BASADO EN EL SISTEMA EN CHIP
PROGRAMABLE DE SEÑAL MIXTA, PSoC, DE CYPRESS
SEMICONDUCTOR**

Autor

ROGER ALVAREZ FIGUEROA

Director

JULIO CÉSAR OSPINO ARIAS
INGENIERO ELECTRÓNICO. ESPECIALISTA

Ingeniería Electrónica
Departamento de Ingenierías Eléctrica, Electrónica, Sistemas y
Telecomunicaciones
Facultad de Ingenierías y Arquitectura
Universidad de Pamplona
Pamplona, Norte de Santander

Universidad de Pamplona
Facultad de Ingenierías y Arquitectura
Departamento de Ingenierías Eléctrica, Electrónica,
Sistemas y Telecomunicaciones
Programa de Ingeniería Electrónica
Trabajo presentado para optar por el título de
Ingeniero Electrónico

Tema:

**DESARROLLO DE UN MANUAL DE PRÁCTICAS DE LABORATORIO, BASADO
EN EL SISTEMA EN CHIP PROGRAMABLE DE SEÑAL MIXTA, PSoC, DE
CYPRESS SEMICONDUCTOR**

Fecha de inicio del trabajo: 1 de junio 2016

Fecha de culminación del trabajo: 30 de septiembre 2016

Nombres y firmas de autorización para la sustentación:

Roger Andrés Álvarez Figueroa
Autor

Julio César Ospino Arias
Director

Judith Cristancho Pabón
Directora de programa

Jurado calificador:

Jesús Ortiz

Luis Muñoz

Julio C. Ospino

PAMPLONA – NORTE DE SANTANDER
COLOMBIA
30 DE SEPTIEMBRE DE 2016

“Para las personas más bellas que he conocido.

*Los que llevan años levantándose en la madrugada
a hacer sus oraciones para que a mí me vaya bien.*

A mis papás y mi hermano, con todo el amor del mundo”

INDICE

PREFACIO	XII
AGRADECIMIENTOS	XIII
1 INTRODUCCIÓN	1
1.1 PROBLEMA	2
1.2 OBJETIVOS.....	2
1.2.1 OBJETIVO GENERAL	2
1.2.2 OBJETIVOS ESPECIFICOS	2
1.3 RESUMEN	3
1.4 DISTRIBUCIÓN DE LA BIOGRAFIA.....	3
2 MARCO TEÓRICO.....	4
2.1 PSoC 1	5
2.1.1 CARACTERISTICAS PSoC 1	6
2.2 PSoC 3	7
2.2.1 CARACTERSTICAS PSoC 3	9
2.3 PSoC 4	11
2.3.1 CARACTERISTICAS PSoC 4	11
2.4 PSoC 5LP.....	13
2.4.1 CARACTERISTICAS PSoC 5LP	13
3 DESARROLLO DEL PROYECTO	17
3.1 Revisión Teórica	18
3.1.1 CPU PSoC 3	18
3.1.2 CPU PSoC 5LP.....	22
3.1.3 GPIO	23
3.1.4 Timers, Contadores y PWM.....	24
3.1.5 ADC Sigma-Delta.....	25
3.1.6 Comparadores	26
3.1.7 LUT	27

3.1.8	Amplificadores operaciones	27
3.1.9	DAC	28
3.2	Manual de prácticas	29
3.2.1	Características de la CPU	29
3.2.2	GPIO	30
3.2.3	Manejo de interrupciones	43
3.2.4	Contador	49
3.2.5	Timer	57
3.2.6	Contador y Timer	65
3.2.7	Manejo de LCD	70
3.2.8	ADC	77
3.2.9	VDAC	85
3.2.10	VDAC y ADC	89
3.2.11	OpAmp	94
3.2.12	Comparadores	98
3.2.13	PWM	104
3.2.14	Control de velocidad de un motor DC con PWM	108
3.2.15	UART	117
3.2.16	LUT	122
3.3	Dispositivo Hardware para PSoC 5LP	125
4	Resultados	128
4.1	Pines digitales de salida	129
4.2	Pines digitales de entrada	130
4.3	Manejo de interrupciones	130
4.4	Contador	131
4.5	Timer	132
4.6	Contador y Timer	132
4.7	Manejo de LCD	133
4.8	ADC Delta Sigma	135
4.9	DAC de Voltaje	135
4.10	ADC Delta Sigma y VDAC	136

4.11	Amplificador Operacional	136
4.12	Comparadores.....	137
4.13	PWM	137
4.14	Control de Velocidad de un motor usando PWM.....	138
4.15	UART	139
4.16	LUT	140
5	Conclusiones.....	141
6	Bibliografía	144

INDICE DE FIGURAS

Figura 1.	Arquitectura PSoC 1 [2]	5
Figura 2.	Arquitectura PSoC 3 [4]	8
Figura 3.	Diagrama de tiempo del DMA [5].....	20
Figura 4.	Flujo típico de eventos cuando se desencadena una interrupción [5]	21
Figura 5.	Diagrama de bloques del ARM Cortex-M3 [7].....	22
Figura 6.	Timer/Contador/PWM [5][7].....	24
Figura 7.	ADC Sigma-Delta, Frecuencia de muestreo, Rango $\pm 1.024V$ [5][7]	25
Figura 8.	Diagrama de bloques del ADC Sigma-Delta [5][7].....	25
Figura 9.	Comparador análogo. [5][7]	27
Figura 10.	Amplificador Operacional [5][7]	28
Figura 11.	Configuraciones de Opamp [5][7]	28
Figura 12.	Diagrama en bloques del DAC. [5][7]	29
Figura 13.	Crear un nuevo proyecto.	31
Figura 14.	Escoger tarjeta de desarrollo.....	31
Figura 15.	Escoger tipo de esquemático	32
Figura 16.	Escoger un nombre y localización para el proyecto.....	32
Figura 17.	Ubicación de los componentes.....	33
Figura 18.	Ubicación de los pines digitales de salida	33
Figura 19.	Configuración del pin de salida Pin_1 Como LED	34
Figura 20.	Componentes en el WorkSpace	34
Figura 21.	Guardar y Compilar.....	34
Figura 22.	Archivo para asignar pines	35
Figura 23.	Pines asignados.....	35
Figura 24.	Archivo para ir al Main	36
Figura 25.	Código para del Main los pines de salida	36
Figura 26.	Guardar, Compilar, Generar Aplicación y Programar.....	36

Figura 27. Crear un nuevo proyecto.....	37
Figura 28. Escoger tarjeta de desarrollo.....	37
Figura 29. Escoger tipo de esquemático	38
Figura 30. Escoger un nombre y localización para el proyecto.....	38
Figura 31. Ubicación de los componentes.....	39
Figura 32. Ubicación de los pines digitales de entrada	39
Figura 33. Configuración del pin de entrada Pin_1	39
Figura 34. Ubicación de los pines digitales de salida	40
Figura 35. Configuración del pin de salida Pin_2 Como LED	40
Figura 36. Componentes en el WorkSpace	40
Figura 37. Guardar y Compilar.....	41
Figura 38. Archivo para asignar pines	41
Figura 39. Pines asignados.....	42
Figura 40. Archivo para ir al Main	42
Figura 41. Código del Main para los pines de entrada.....	43
Figura 42. Guardar, Compilar, Generar Aplicación y Programar.....	43
Figura 43. Parámetros de una interrupción	44
Figura 44. Ubicación de los pines digitales de entrada	45
Figura 45. Configuración del Pin_1 como Pull Up	45
Figura 46. Ubicación de las interrupciones	45
Figura 47. Ubicación de los pines digitales de salida	46
Figura 48. Configuración del LED y esquema de componentes	46
Figura 49. Archivo para ir al Main	47
Figura 50. Código del Main para la función CY_ISR.....	47
Figura 51. Código para iniciar la interrupción.	47
Figura 52. Archivo para asignar pines	48
Figura 53. Pines Asignados	48
Figura 54. Guardar, Compilar, Generar Aplicación y Programar.....	49
Figura 55. Parámetros de un Contador	49
Figura 56. Ubicación de los contadores en el catálogo de Cypress	51
Figura 57. Configuración del contador	51
Figura 58. Ubicación de los relojes en el catálogo de Cypress.....	52
Figura 59. Ubicación y conexión de un '0' lógico	52
Figura 60. Ubicación y conexión del pin de entrada digital	52
Figura 61. Configuración del Pin_1 como pull up.....	53
Figura 62. Ubicación y conexión de una interrupción.....	53
Figura 63. Configuración de la interrupción.....	53
Figura 64. Configuración del pin de salida como LED en modo Strong drive	54
Figura 65. Archivo para ir al Main	54
Figura 66. Código para la Función CY_ISR	55
Figura 67. Código del main para iniciar componentes.....	55

Figura 68. Archivo para asignar pines	56
Figura 69. Pines asignados.....	56
Figura 70. Guardar, Compilar, Generar Aplicación y Programar.....	56
Figura 71. Parámetros del Timer	57
Figura 72. Ubicación del Timer en el catálogo de Cypress.....	59
Figura 73. Configuración de un nuevo reloj	59
Figura 74. Configuración del timer	60
Figura 75. Ubicación y conexión de la interrupción	60
Figura 76. Ubicación y conexión del pin de entrada y la interrupción.....	61
Figura 77. Configuración del pin de entrada como Pull up	61
Figura 78. Configuración del pin de salida como LED	62
Figura 79. Componentes en el WorkSpace	62
Figura 80. Archivo para ir al Main	62
Figura 81. Código para las funciones de las interrupciones.....	63
Figura 82. Código del main para iniciar los componentes	63
Figura 83. Archivo para asignar pines	64
Figura 84. Pines utilizados	64
Figura 85. Ubicación del Timer	65
Figura 86. Configuración del Contador.....	65
Figura 87. Ubicación y conexión del Clock	66
Figura 88. Ubicación y conexión del '0' lógico	66
Figura 89. Ubicación y conexión del pin de entrada y de la interrupción.....	66
Figura 90. Configuración de la interrupción del contador isrC	67
Figura 91. Configuración del pin de entrada como pull up	67
Figura 92. Ubicación y conexión del timer	67
Figura 93. Configuración del reloj del timer.....	68
Figura 94. Configuración del timer	68
Figura 95. Ubicación y configuración del pin digital de salida	69
Figura 96. Código de las funciones de las interrupciones	69
Figura 97. Código del main para iniciar componentes.....	70
Figura 98. Pines asignados.....	70
Figura 99. Parámetros de la LCD	71
Figura 100. Ubicación de la LCD	71
Figura 101. Pines utilizados	72
Figura 102. Distribución de pines para la LCD.....	72
Figura 103. Código para imprimir un mensaje	73
Figura 104. Código para parpadear la pantalla	73
Figura 105. Configuración de la LCD como un gráfico de barras horizontal.....	73
Figura 106. Código como barra de progreso	74
Figura 107. Configuración de la LCD para caracteres especiales	74
Figura 108. Código para imprimir los caracteres	75

Figura 109. Agregar librería	75
Figura 110. Nombre de librería	75
Figura 111. Ubicación del bloque LCD.....	76
Figura 112. Conexión de pines con la LCD.....	76
Figura 113. Distribución de pines	76
Figura 114. Pines asignados.....	77
Figura 115. Código para imprimir mensaje	77
Figura 116. Parámetros del ADC.....	78
Figura 117. Ubicación del ADC delta sigma en el catálogo de componentes	79
Figura 118. Configuración del ADC.....	79
Figura 119. Configuración del ADC.....	80
Figura 120. Ubicación del pin análogo	80
Figura 121. Conexión del pin análogo con el ADC.....	81
Figura 122. Ubicación de la LCD	81
Figura 123. Conexión de pines con la LCD.....	81
Figura 124. Configuración de la memoria para la LCD	82
Figura 125. Configuración para el buen funcionamiento de la LCD.....	82
Figura 126. Comando para la LCD	83
Figura 127. Código para declarar variables y agregar librería.....	83
Figura 128. Código para iniciar LCD.....	83
Figura 129. Código para iniciar ADC	84
Figura 130. Código para leer e imprimir resultado del ADC.....	84
Figura 131. Pines Asignados	84
Figura 132. Parámetros del VDAC.	85
Figura 133. Ubicación del VDAC en el catálogo.....	86
Figura 134. Configuración del VDAC.....	87
Figura 135. Ubicación del pin análogo	87
Figura 136. Pin análogo como salida	87
Figura 137. Código para iniciar el VDAC	88
Figura 138. Pines asignados.....	88
Figura 139. Ubicación del ADC en el catálogo	89
Figura 140. Ubicación del DAC en el catálogo	89
Figura 141. Configuración del ADC.....	90
Figura 142. Configuración del ADC.....	90
Figura 143. Conexión de componentes.....	91
Figura 144. Configuración del VDAC.....	91
Figura 145. Ubicación de la LCD.	91
Figura 146. Conexión de pines con la LCD.....	92
Figura 147. Conexión del pin de entrada con la interrupción	92
Figura 148. Código para declarar variables	92
Figura 149. Código para iniciar componentes.....	93

Figura 150. Código para leer ADC y mostrar resultado en LCD.....	93
Figura 151. Código para función de la interrupción.....	93
Figura 152. Pines asignados.....	94
Figura 153. Parámetros del Amplificador Operacional	94
Figura 154. Configuraciones del AmpOp	95
Figura 155. Ubicación del AmpOp en el catálogo	95
Figura 156. Configuración del Amplificador como seguidor	96
Figura 157. Ubicación del VDAC en el catálogo.....	96
Figura 158. Configuración del VDAC.....	97
Figura 159. Conexión de los componentes	97
Figura 160. Código para iniciar variables y componentes.....	98
Figura 161. Pines asignados.....	98
Figura 162. Parámetros del Comparador	99
Figura 163. Configuraciones del comparador	99
Figura 164. Ubicación del Comparador en el catálogo	100
Figura 165. Configuración del Comparador	101
Figura 166. Ubicación del VDAC en el catálogo.....	101
Figura 167. Configuración del VDAC.....	102
Figura 168. Ubicación de los pines análogos en el catálogo	102
Figura 169. Conexión del VDAC y el comparador.....	102
Figura 170. Conexión de todos los componentes	103
Figura 171. Código para iniciar componentes.....	103
Figura 172. Pines asignados.....	104
Figura 173. Parámetros del PWM	104
Figura 174. Ubicación del PWM en el catálogo.....	106
Figura 175. Configuración del PWM.....	106
Figura 176. Ubicación y conexión del clock para el PWM.....	107
Figura 177. Ubicación y conexión del '0' lógico para el PWM	107
Figura 178. Ubicación y conexión de un pin digital de salida al PWM	107
Figura 179. Código para establecer valor del PWM	108
Figura 180. Pines adignados.	108
Figura 181. Ubicación del bloque LCD	109
Figura 182. Conexión de pines con la LCD.....	109
Figura 183. Ubicación del PWM en catálogo.....	110
Figura 184. Configuración del PWM.....	110
Figura 185. Ubicación y conexión del pin digital de entrada al PWM.....	111
Figura 186. Ubicación de un pin digital de salida	111
Figura 187. Ubicación y conexión del reloj para el PWM.....	111
Figura 188. Configuración del reloj del PWM.....	112
Figura 189. Ubicación y conexión de un pin de salida para visualizar el PWM.....	112
Figura 190. Pines de entrada para cambiar el valor del PWM	113

Figura 191. Conexión de los pines de entradas con las interrupciones.	113
Figura 192. Pin de salida (LED).	113
Figura 193. Pines asignados.....	114
Figura 194. Parámetros de UART	118
Figura 195. Ubicación del componente UART.....	119
Figura 196. Configuración del UART.....	120
Figura 197. Ubicación y conexión de un '0' lógico	120
Figura 198. Configuración y ubicación del pin digital de salida	121
Figura 199. Código para la función CY_ISR.....	121
Figura 200. Código para iniciar componentes.....	121
Figura 201. Código para establecer comunicación serial mediante el bloque UART	122
Figura 202. Pines asignados.....	122
Figura 203. Parámetros del LUT	123
Figura 204. Ubicación del componente LUT.....	123
Figura 205. Configuración de la LUT.....	124
Figura 206. Conexión de pines de entrada y salida con el bloque LUT.....	124
Figura 207. Pines asignados.....	125
Figura 208. Esquema de componentes	126
Figura 209. Esquema de conexiones	126
Figura 210. Diseño del dispositivo hardware	127

PREFACIO

PREFACIO.....	XII
AGRADECIMIENTOS.....	XIII

AGRADECIMIENTOS

En primer lugar, quiero agradecer a Dios por darme la oportunidad de vivir, de tener una hermosa familia, por darme la oportunidad de estudiar y realizar este proyecto. Muchas gracias Dios no me alcanza la vida para agradecerte por todo lo que me has dado.

A mis padres, Lilianis Figueroa Flores y Roger Álvarez Betin, por traerme a este mundo, por darme palabras de aliento todas las mañanas para que pudiera llegar a cumplir mis sueños. Gracias por enseñarme a leer, escribir, sumar, restar. Gracias por enseñarme a ser una buena persona y por ser quien soy hoy. Gracias por darme la oportunidad de estudiar, por ese apoyo incondicional aun en los momentos más difíciles, gracias por sacrificarse a cada momento para que a mi hermano y a mí nunca nos hiciera falta nada.

A mi hermano, Roger Enrique Álvarez, gracias por existir, por todos los momentos que hemos vivido juntos, por siempre creer en mí. Por apoyarme y por todas y cada una de esas cosas que hizo para que yo siguiera estudiando. Gracias por siempre cuidar de mí.

Gracias al ingeniero Julio Cesar Ospino, por creer en este proyecto, por su apoyo y colaboración como director de trabajo de grado. Y también agradecer a todos y cada uno de los docentes que me brindaron sus conocimientos y me formaron como profesional de la ingeniería electrónica.

A mi familia, a mis amigos, a mis profesores y a todas esas personas que en este momento se me olvida mencionar, muchas, pero muchas gracias por creer en mí desde el primer momento que empecé a recorrer este camino como estudiante de Ingeniería Electrónica. Hoy culmino una de las mejores etapas de mi vida y eso es gracias al apoyo y cariño que siento de las personas que me rodean. Dios los bendiga.

1 INTRODUCCIÓN

1 INTRODUCCIÓN	1
1.1 PROBLEMA	2
1.2 OBJETIVOS.....	2
1.2.1 OBJETIVO GENERAL.....	2
1.2.2 OBJETIVOS ESPECIFICOS.....	2
1.3 RESUMEN	3
1.4 DISTRIBUCIÓN DE LA BIOGRAFIA.....	3

1.1 PROBLEMA

Este proyecto está basado en el desarrollo de un manual de prácticas basado en la tecnología PSoC, utilizando los kits de desarrollo para PSoC 3 (CY8CKIT-030), que el programa de Ingeniería Electrónica recibió como donación en 2009, dentro de la iniciativa University Alliance de Cypress Semiconductor, y utilizando también la PSoC 5 LP.

La necesidad de realizar este trabajo surgió de la evidencia acerca de las capacidades que el microcontrolador PSoC nos brinda, al contar con elementos específicos tanto análogos como digitales, que pueden ser implementados mediante programación gráfica, y de esta manera, suplir la no disponibilidad de algunos dispositivos en Pamplona, a la hora de realizar prácticas de instrumentación, electrónica analógica y digital.

En el caso del PSoC (Programable System on Chip), estos dispositivos son capaces de combinar la electrónica analógica y digital programable en un mismo chip, reduciendo el BOM (Listado de Materiales) al ofrecer más de 150 componentes (VDAC, IDAC, ADC Sigma Delta, Mezcladores, PGA, TIA, Op-Amps, Comparadores, Compuertas Lógicas, MUX Analógicos, Filtros Digitales, etc) que pueden ser programados en el entorno PSoC Creator, cuando se requieran. Esto reduce el tamaño de circuitos impresos, repercutiendo en un menor costo, menor espacio y mayor confiabilidad al momento de realizar prácticas de laboratorio, proyectos de investigación ó prototipos.

El programa de Ingeniería Electrónica de la Universidad de Pamplona, en su Semillero Sistemas Embebidos está interesado en apropiarse de esta tecnología y ha definido como prioridad, el aprovechar los kits de desarrollo disponibles en el laboratorio de electrónica digital, para las familias PSoC 1 (CY3210), PSoC 3 (CY8CKIT-030) y PSoC 5 (CY8CKIT-050). Esto demuestra la necesidad de comenzar a trabajar con estos dispositivos, a pesar de que la información disponible en español es escasa.

A nivel nacional, se está comenzando a utilizar estos dispositivos, cuya abanderada es la Universidad Distrital Francisco José de Caldas (Bogotá), que ha conformado un grupo de trabajo (PSoC Latinoamerica), que se ha encargado de realizar tutoriales básicos en Youtube, para llamar la atención de estudiantes y docentes, en colegios y universidades, buscando incentivar y masificar su uso.

1.2 OBJETIVOS

1.2.1 OBJETIVO GENERAL

- Desarrollar un manual de prácticas de laboratorio para el desarrollo de aplicaciones, utilizando el Sistema en Chip Programable de Señal Mixta, PSoC, de Cypress Semiconductor.

1.2.2 OBJETIVOS ESPECIFICOS

- Realizar una revisión teórica de la CPU, GPIO, interrupciones, contadores, temporizadores, amplificadores operacionales, VDAC, ADC Delta Sigma, PWM, UART, LUT, disponibles en el entorno PSoC Creator.

- Desarrollar las guías de laboratorio enfocadas en la CPU, GPIO, interrupciones, contadores, temporizadores, amplificadores operacionales, VDAC, ADC delta sigma, PWM, UART, LUT disponibles en el entorno de desarrollo PSoC Creator.
- Desarrollar un dispositivo hardware para las prácticas de laboratorio.
- Validar el manual de prácticas de laboratorio enfocado a los elementos anteriormente nombrados.

1.3 RESUMEN

El presente proyecto a realizar pretende desarrollar ejemplos, prácticas, la solución a dichas prácticas y una descripción de algunos de los componentes disponibles para dichas prácticas utilizando la tecnología PSoC, dentro del Kit de Desarrollo CY8CKIT-030 PSoC 5LP, la empresa Cypress Semiconductor.

Para comenzar se describirá la arquitectura interna de la PSoC, luego se procederá a la explicación de los módulos utilizados, sus características, sus funciones de programación, sus opciones de configuración (si es el caso) y los elementos que se deben configurar para el correcto funcionamiento de este dispositivo.

Por último, se presentarán las prácticas propuestas y la solución a cada una de ellas, explicando el esquemático en PSoC Creator y su respectivo código para su funcionamiento.

1.4 DISTRIBUCIÓN DE LA BIOGRAFIA

Este libro está organizado en 5 capítulos concernientes a la Introducción, Marco Teórico, Desarrollo del proyecto, Resultados y Conclusiones.

Cada capítulo tiene los ítems correspondientes al tema de interés.

2 MARCO TEÓRICO

- 2. MARCO TEÓRICO 4
 - 2.1 PSoC 1 5
 - 2.1.1 CARACTERISTICAS PSoC 1..... 6
 - 2.2 PSoC 3 7
 - 2.2.1 CARACTERSTICAS PSoC 3..... 9
 - 2.3 PSoC 4 11
 - 2.3.1 CARACTERISTICAS PSoC 4..... 11
 - 2.4 PSoC 5LP 13
 - 2.4.1 CARACTERISTICAS PSoC 5LP13

Cypress Semiconductor Corporation es una empresa de diseño y fabricación de semiconductores estadounidense fundada por T. J. Rodgers y otros de Advanced Micro Devices. Se formó en 1982 con el respaldo de Sevin Rosen y salió a bolsa en 1986. La compañía se centró inicialmente en el diseño y desarrollo de CMOS, SRAM, EEPROM de alta velocidad, dispositivos PAL, y los dispositivos de lógica TTL. Dos años después de la salida a bolsa de la compañía cambió desde el NASDAQ de la bolsa de Nueva York. En octubre de 2009, la compañía anunció que iba a cambiar su anuncio a la NASDAQ el 12 de noviembre de 2009. Tiene su sede en San José, California, y tiene divisiones en los Estados Unidos, Irlanda, India y Filipinas, así como una planta de fabricación en Minnesota.

La familia del Cypress PSoC ofrece una mezcla única de adaptabilidad en diseño a un precio relativamente bajo. En el mundo actual de creciente automatización y digitalización del mundo que nos rodea, los dispositivos PSoC permiten por ejemplo, agregar el cerebro al diseño y acondicionar una ó varias señales analógicas en un único chip. Este nivel de integración con anterioridad sólo existía en algunas partes mucho más grandes, haciendo palidecer ventajas de la integración en comparación con el precio añadido y la complejidad. Diseños anteriores con necesidades análogas de amplificar, filtrar y acondicionar señales utilizaban circuitos analógicos dedicados, utilizaban ADCs para digitalizar la señal para luego realizar su procesamiento digital en su CPU integrada.[1]

2.1 PSoC 1

Es el primer Sistema programable en un chip del mundo, integrando funciones periféricas digitales, análogas configurables, de memoria y microcontrolador en un solo chip. Con tecnología de núcleo de CPU M8C de 8 bits de Cypress, los dispositivos PSoC 1 proporcionan la integración del diseñado y configuración del sistema analógico, enrutamiento flexible de IO y periféricos digitales.

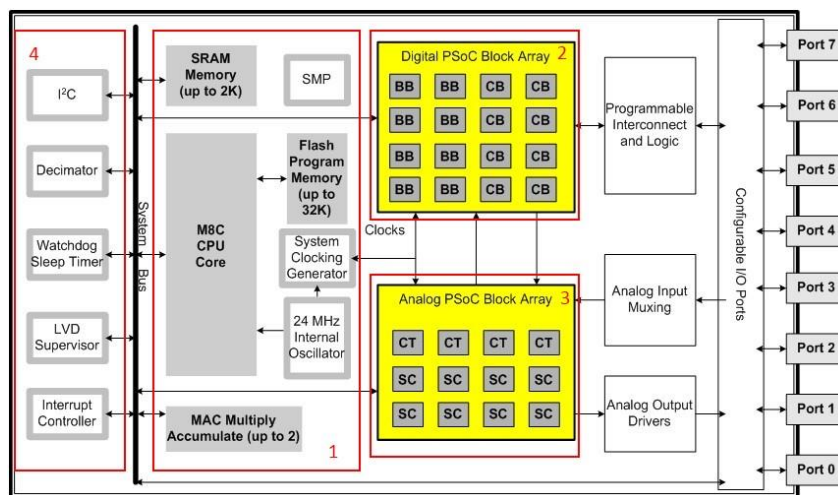


Figura 1. Arquitectura PSoC 1 [2]

La arquitectura PSoC 1 consta de cuatro áreas principales: (1) Núcleo PSoC, (2) sistema digital, (3) sistema analógico, y (4) los recursos del sistema. Tiene un busing global configurable que permite que todos los recursos del sistema puedan ser combinados en un sistema completo y personalizado. El PSoC 1 de Cypress ofrece dispositivos que tienen hasta 16 bloques digitales y 12 bloques analógicos. Los bloques digitales son bloques programables, el diseñador los puede utilizar para implementar la lógica, UART, PWMs, contadores, tablas de verdad, etc. Estos bloques contienen una ruta de datos programable, de estado y control registros, e incluye dos bloques básicos (BB) y bloques de Comunicaciones (CB). Los bloques analógicos son bloques programables que el diseñador puede utilizar para implementar OpAmps, PGA, comparadores, etc. E incluyen bloques de tiempo continuo (TC) y el condensador conmutado (SC) bloquea. Tanto los bloques digitales y analógicos son configurables mediante módulos de usuario (UMS) en PSoC Designer. [2]

2.1.1 CARACTERISTICAS PSoC 1

- Potencia del procesador y arquitectura de Harvard.
 - Procesador M8C, velocidad hasta 24 MHz.
 - Multiplicador de 8x8, acumulador de 32 bits.
 - Baja potencia a alta velocidad.
 - 3,0 V a 5,25 V. Voltaje de operación.
 - Rango de temperatura industrial: -40°C a +85°C.

- Periféricos avanzados (Bloques PSoC)
 - Cuatro bloques PSoC analógicos rail-to-rail proporcionan:
 - ADC hasta de 14 bits.
 - DAC hasta de 8 bits.
 - Amplificadores de ganancia programable.
 - Filtros programables y comparadores.
 - Cuatro bloques PSoC digitales proporcionan:
 - Timers y contadores de 8 – 32 bits, PWM de 8 – 16 bits.
 - Módulos CRC y PRS.
 - UART Full dúplex.
 - SPI múltiple, maestro o esclavo.
 - Se puede conectar a todos los pines GPIO.
 - Periféricos complejos, compuestos de bloques.
 - SAR ADC de alta velocidad de 8 bits, optimizado para control de motores.

- Reloj programable
 - Oscilador interno 24/48 MHz \pm 5%.
 - Alta precisión de 24 MHz, con cristal y PLL opcional de 32kHz.
 - Oscilador externo opcional, hasta 24 MHz.
 - Oscilador interno para watchdog y sleep.

- Memoria flexible en chip
 - Almacenamiento flash de 8KB, 50.000 ciclos de programa, borrado/escritura.
 - 256 bytes de almacenamiento de datos en SRAM.
 - Programación serial In-System (ISSP).
 - Actualizaciones parciales de flash.
 - Modos de protección flexibles.
 - Emulación de EEPROM en flash.

- Configuración de pines programable
 - 25 mA en todos los GPIO.
 - Pull up, pull down, alta impedancia, modo strong (alta corriente de salida), o modos de conducción abierta en todos los GPIO.
 - Hasta ocho entradas analógicas de GPIO más dos entradas analógicas adicionales con enrutamiento restringido.
 - Dos salidas analógicas de 30 mA en GPIO.
 - Interrupciones configurables en todos los GPIO.

- Recursos adicionales del sistema
 - I2C esclavo, maestro, y varios maestros de 400 kHz.
 - Watchdog y temporizadores sleep.
 - Detección de baja tensión configurable por el usuario.
 - circuito integrado de supervisión.
 - Precisión del voltaje de referencia en el chip.

- Herramientas de desarrollo completas
 - Desarrollo de software libre (PSoC Designer™).
 - Todas las funciones del emulador en circuito (ICE) y programador.
 - Emulación de máxima velocidad.
 - Complejo estructura de punto de interrupción.
 - 128 KB de memoria de trazas. [3]

2.2 PSOC 3

PSOC® 3 es un sistema programable en chip que integra funciones periféricas digitales, analógicas configurables y, la memoria y un microcontrolador en un solo chip incorporado. Cuenta con un gran avance, la nueva arquitectura del PSOC 3 aumenta el rendimiento a través de:

- Resolución de alta precisión integrados analógicos de 20 bits.

- Ultra baja potencia con el voltaje más amplia de la industria.
- La lógica programable basada en PLD.
- CPU 8051 hasta 67 MHz.

La arquitectura PSoC® consiste en bloques analógicos y digitales configurables y, un subsistema de la CPU y el encaminamiento programable y de interconexión. PSoC le permite predefinir la conexión y probar la biblioteca PSoC de funciones, o el código de su cuenta.

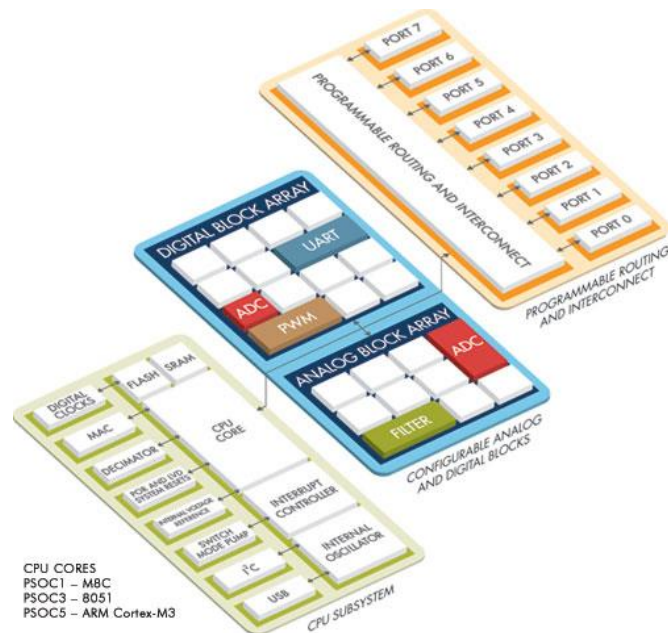


Figura 2. Arquitectura PSoC 3 [4]

- Enrutamiento programable y de interconexión: Esto libera señales puede iniciar una nueva ruta a los pines seleccionada por el usuario, desprendiéndose de las limitaciones de un controlador fijo-periférica.
- Bloques análogos y digitales configurables: La unión de análogo configurable y los circuitos digitales es la base de la plataforma PSoC. Mediante la combinación de varios bloques digitales, se pueden crear amplios recursos lógicos de 16, 24 o incluso de 32 bits. Los bloques analógicos se componen de una variedad de condensador de conmutación, op-amp, comparador, ADC, DAC, y bloques de filtros digitales
- Subsistema de la CPU: PSoC ofrece un subsistema de la CPU sofisticado con SRAM, EEPROM, y la memoria flash, múltiples opciones básicas y una variedad de recursos de sistema esenciales, incluyendo:
 - Oscilador interno principal y baja velocidad.

- Conectividad con oscilador de cristal externo para la precisión, de reloj programable.
- Watchdog y timer sleep.
- Fuentes múltiples de reloj que incluyen un PLL.

Los dispositivos PSoC también han dedicado interfaces de comunicación como el I2C, Full-Speed USB 2.0, CAN 2.0, y en el chip utilizando capacidades de depuración JTAG.[4]

2.2.1 CARACTERÍSTICAS PSoC 3

➤ Características de operación

- Rango de voltaje: 1.71 a 5.5 V, hasta seis dominios de poder.
- Rango de temperatura (ambiente) -40 a 85 ° C.
- DC a 67 MHz operación
- Modos de operación
 - Modo activo de 1,2 mA a 6 MHz, y de 12 mA a 48 MHz.
 - Modo de espera 1uA.
 - 200 nA modo de hibernación con la retención de la memoria RAM.
- Regulador elevador de entrada de 0,5 V hasta salida de 5 V

➤ Rendimiento

- CPU 8051 de 8 bits, 32 entradas de interrupción,
- 24 canales de acceso directo a memoria (DMA) del controlador.
- 24 bits del procesador filtro digital de punto fijo 64-tap (DFB)

➤ Memorias

- Hasta 64 KB de memoria flash de programa, con las características del cache y de seguridad.
- Hasta 8 KB de memoria flash adicional para el código de corrección de errores (ECC).
- Hasta 8 KB de RAM.
- Hasta 2 KB EEPROM.

➤ Periféricos digitales

- Cuatro timers, contadores, PWM's de 16 bits.
- I2C, 1 Mbps de velocidad de bus.
- Certificación USB 2.0 de Full-Speed (FS) 12 Mbps interfaz periférica (TID # 40770053) usando oscilador interno.
- CAN 2.0B Full, 16 Rx, Tx 8 buffer.
- 16 a 24 bloques digitales universales (UDB), programable para crear cualquier número de funciones:
 - Timers, contadores y PWM de 8, 16, 24, y 32 bit.

- I2C, UART, SPI, I2S, interfaz LIN 2.0.
 - Comprobación de redundancia cíclica (CRC).
 - Generadores de secuencia pseudo aleatoria (PRS).
 - decodificadores de cuadratura.
 - Funciones de compuertas lógicas de nivel.
- Reloj programable
- Oscilador interno de 3 hasta 62 MHz, 1% de precisión a 3MHz.
 - Oscilador de cristal externo de 4 hasta 25MHz.
 - PLL interno de generación de reloj de hasta a 67 MHz.
 - Oscilador interno de baja potencia a 1, 33 y 100 kHz.
 - Oscilador de cristal de reloj externo de 32.768 kHz.
 - 12 divisores de reloj se pueden enrutar a cualquier periférico o E/S.
- Periféricos análogos
- ADC delta-sigma configurable, 8 hasta 20 bits.
 - Hasta cuatro DAC de 8 bits.
 - Hasta cuatro comparadores.
 - Hasta cuatro amplificadores operacionales.
 - Hasta cuatro bloques analógicos programables, para crear:
 - amplificador de ganancia programable (PGA).
 - Amplificador de transimpedancia (TIA).
 - Mezclador.
 - El circuito de muestreo y retención.
 - CapSense®, hasta 62 sensores.
 - 1,024 V referencia de tensión interna $\pm 0,1\%$.
- Versátil sistema de E/S
- 29 a 72 pins de E/S. Hasta 62 de propósito general (GPIO).
 - Hasta ocho pines de rendimiento especial (SIO).
 - 25 mA sumidero de corriente
 - Umbral de entrada y de salida programable a altos voltajes.
 - Puede actuar como elemento de comparación de propósito general.
 - Capacidad de intercambio en caliente y la tolerancia de sobretensión.
 - Dos pines de USBIO que se pueden utilizar como GPIOs.
 - Ruta de cualquier periférico digital o analógico a cualquier GPIO.
 - Accionamiento directo desde cualquier pantalla LCD GPIO, hasta 46 × 16 segmentos.
 - CapSense de cualquier GPIO.
 - 1.2 V para tensiones de interfaz de 5.5 V, hasta cuatro dominios de poder.

- Programación y depuración
 - JTAG (4 hilos), depuración de alambre de serie (SWD) (2 hilos), y las interfaces de visualización de un solo cable (SWV).
 - Programación del cargador de arranque a través de I2C, SPI, UART, USB y otras interfaces.

- Opciones del paquete: SSOP de 48 pines, QFN de 48 pines, 68 pines QFN, TQFP de 100 pines y 72 pines WLCSP.

- Apoyo para el desarrollo con la herramienta gratuita PSoC Creator™
 - Esquemático y soporte de diseño de firmware.
 - Más de 100 componentes PSoC™ para integrar múltiples circuitos integrados y las interfaces del sistema en una PSoC. Los componentes son circuitos integrados incorporados libres representados por iconos.
 - Incluye conexión Keil compilador 8051
 - Soporta la programación del dispositivo y la depuración. [5]

2.3 PSoC 4

PSoC® 4 es una arquitectura de plataforma escalable y reconfigurable para una familia de controladores programables del sistema integrado de señal mixta con una CPU ARM Cortex™ -M0. Combina bloques digitales con enrutamiento automático y programable analógico flexibles y reconfigurables. La familia de productos PSoC 4200, basado en esta plataforma, es una combinación de un microcontrolador con lógica programable digital, de alto rendimiento de conversión de analógico a digital, amplificadores operacionales con el modo de comparador, y los periféricos de comunicación y sincronización estándar. Los PSoC 4200 productos serán totalmente compatible hacia arriba con los miembros de la plataforma PSoC 4 para nuevas aplicaciones y necesidades de diseño.

2.3.1 CARACTERISTICAS PSoC 4

- Sub-sistema de MCU 32 bits
 - CPU de 48 MHz ARM Cortex-M0 con solo ciclo multiplicador.
 - hasta 32 kB de flash con Read acelerador.
 - Hasta 4 KB de SRAM

- Periféricos Analógicos
 - Dos amplificadores operacionales con la unidad reconfigurable de alta duro externo y de alto ancho de banda interno, modos de comparadores, y ADC capacidad de almacenamiento en búfer de entrada.

- SAR ADC de 12 bits, 1 Msps con los modos diferenciales y de un solo término indefinido; Canal Secuencial con el promedio de señal.
 - Dos DACs de corriente (IDACs) para fines generales o las aplicaciones de sensores capacitivos en cualquier pin.
 - Dos comparadores de baja potencia que operan en modo de hibernación.
- Periféricos Digitales
 - Cuatro bloques lógicos programables llamados bloques digitales universales, (UDBs), cada uno con 8 Macro células y ruta de datos.
 - Proporcionado por la biblioteca de componentes periféricos, máquinas de estados definidos por el usuario, y la entrada de Verilog.
- Bajo consumo de operación 1.71-V a 5,5 V
 - 20 nA en modo STOP con activación de los pines GPIO.
 - Modo hibernación y sueño profundo permiten que los tiempos de activación de energía en comparación con las compensaciones.
- Sensor capacitivo
 - El CapSense Sigma-Delta de Cypress (CSD) proporciona la mejor SNR en su clase (> 5: 1) y la tolerancia al agua.
 - Los componentes de software suministrados por Cypress hace que en el diseño sea fácil la detección capacitiva.
 - Ajuste automático de hardware (SmartSense™).
- Controlador para LCD
 - Unidad de LCD compatible con todos los pines (común o segmento)
 - Funciona en modo de suspensión reforzada con 4 bits de memoria por pin.
- Comunicación Serial
 - Dos bloques de comunicación serial (SCB) de tiempo de ejecución independiente reconfigurables con I2C, SPI o funcionalidad UART.
- Sincronización y PWM
 - Cuatro timers, contadores y PWM (TCPWM bloques) de 16 bits.
 - Modos: alineado al centro, Edge, y pseudo-aleatorios
 - Activación basada en Comparador de las señales de interrupción para la unidad de motor y otras aplicaciones de lógica digital de alta fiabilidad.
- Hasta 36 GPIO programables
 - Cualquier pin GPIO puede ser CapSense, LCD, analógicos o digitales.

- Modos de manejo, fortalezas y velocidades de respuesta son programables cinco paquetes diferentes.
 - 48 pines TQFP, 44 pines TQFP de 40 pines QFN, 35-ball WLCSP, y el paquete de SSOP de 28 pines.
 - Paquete WLCSP 35-Ball se envía con I2C cargador de arranque en flash.
- Temperatura de operación industrial
 - 40 ° C a + 105 ° C.
 - PSoC Creator
 - Entorno de desarrollo integrado (IDE) proporciona entrada de diseño esquemático y construir (con analógico y cálculo automático digital).
 - Componente de interfaz de programación de aplicaciones (API) para toda función fija y periféricos programables.
 - Estándar de la industria herramienta de compatibilidad
 - Después de la introducción del esquema, el desarrollo se puede hacer con las herramientas de desarrollo estándar de la industria basados en ARM. [6]

2.4 PSoC 5LP

PSoC[®] 5LP es un sistema programable en chip que integra funciones periféricas digitales, analógicas configurables y, la memoria y un microcontrolador en un solo chip incorporado. Cuenta con un gran avance, la nueva arquitectura del PSoC 5LP aumenta el rendimiento a través de:

- CPU ARM Cortex-M3 de 32 bits, con controlador DMA procesador de filtro digital, a velocidades de hasta 80 MHz.
- Ultra baja potencia con el voltaje más amplia de la industria.
- Periféricos digitales y analógicos programables permiten funciones personalizadas.
- Enrutamiento flexible de cualquier función periférica analógica o digital a cualquier pin.

Los dispositivos PSoC emplean una arquitectura altamente configurable sistema-en-chip para el diseño de control embebido. Un dispositivo PSoC solo puede integrar un máximo de 100 funciones periféricos digitales y analógicas, lo que reduce el tiempo de diseño, espacio en la placa, el consumo de energía, y el coste del sistema y mejorar la calidad del sistema.

2.4.1 CARACTERÍSTICAS PSoC 5LP

- Características de operación
 - Rango de voltaje: 1,72 hasta 5,5 V. hasta 6 dominios de poder.

- Rango de temperatura (ambiente): -40 hasta 80 °C.
 - Modos de operación:
 - Modo activo 3,1 mA a 6MHz, 15,4 mA a 48 MHz.
 - Modo sleep: 2uA.
 - Mono hibernar: 300 nA con retención de RAM
- Desempeño
- Procesador ARM Cortex de 32 bits, con 32 interrupciones.
 - 24 canales de acceso directo a memoria (DMA).
 - 64 DFB (procesadores de filtro de punto fijo) de 24 bits.
- Memorias
- Hasta 256 KB de memoria flash, con características de caché y seguridad.
 - Hasta 32 KB adicionales de memoria flash para código de corrección de errores (ECC).
 - Hasta 64 KB de RAM.
 - Hasta 2 KB de EEPROM.
- Periféricos digitales
- Cuatro temporizadores, contadores y bloques de PWM (TCPWM) de 16 bits.
 - I2C, 1 Mbps de velocidad del bus.
 - Certificación USB 2.0 de velocidad completa, 12 Mbps para la interfaz periférica (TID # 10840032) utilizando oscilador interno.
 - Full CAN 2.0b, 16 buffers Rx, 8 buffers Tx.
 - 20 hasta 24 UDB (bloques digitales universales), programables para crear cualquier número de funciones:
 - Timers, contadores y PWMs de 8, 16, 24, y 32 bits.
 - I2C, UART, SPI, I2S, LIN 2.0 interfaces.
 - Chequeo de redundancia cíclica (CRC).
 - Generadores de pseudo secuencia aleatoria (PSR).
 - Decodificadores de cuadratura.
 - Nivel de funciones para compuertas lógicas.
- Reloj programable
- Oscilador interno de 3 hasta 74 MHz, 1% de exactitud a 3 MHz.
 - Oscilador externo de cristal de 4 hasta 25 MHz.
 - PLL interno para generación de reloj de hasta 80 MHz.
 - Oscilador interno de bajo consumo a 1, 33 y 100KHz.
 - Oscilador de reloj de cristal externo 32.768 kHz.

- 12 divisores de reloj se pueden enrutar a cualquier periférico de E / S.
- Periféricos análogos
 - ADC delta sigma configurable de 8 hasta 20 bits.
 - Hasta 2 SAR ADCs de 12 bits.
 - Hasta cuatro DACs de 8 bits.
 - Hasta cuatro comparadores.
 - Hasta cuatro opamps.
 - Hasta cuatro bloques análogos configurables, para:
 - Amplificador de ganancia programable (PGA).
 - Amplificador de transimpedancia (TIA).
 - Mezclador.
 - Circuito de muestreo y retención.
 - CapSense, hasta 62 sensores.
 - 1,024 V \pm 0,1% voltaje de referencia interno.
- Sistema versátil de E/S
 - 46 hasta 72 pines de I/O – hasta 62 de propósito general (GPIOs)
 - Hasta 8 pines de función especial (SIO):
 - 25 mA corriente de sink (current sink).
 - umbral de entrada programable y salida de altas tensiones.
 - Puede actuar como un comparador para fines generales.
 - Capacidad de intercambio en caliente y la tolerancia de sobretensión.
 - Dos pines de USBIO que pueden ser utilizados como GPIOs.
 - Ruta cualquier periférico digital o analógico a cualquier GPIO.
 - LCD de transmisión directa desde cualquier GPIO, hasta 46 \times 16 segmentos.
 - CapSense para cualquier GPIO.
 - Voltaje de interface: 1,2V hasta 5,5V. Hasta cuatro dominios de poder.
- Programación y depuración
 - JTAG (4 hilos), depuración de alambre de serie (SWD) (2 hilos), solo hilo visor (SWV), y Traceport interfaces (5 hilos).
 - Módulos de depuración y traza ARM incrustado en el núcleo de la CPU.
 - programación del cargador de arranque a través de I2C, SPI, UART, USB y otras interfaces
- Opciones del paquete: 68 PINES QFN Y 100 PINES TQFP.

- Apoyo para el desarrollo con la herramienta gratuita PSoC Creator™
 - Esquemático y soporte de diseño de firmware.
 - Más de 100 componentes PSoC™ para integrar múltiples circuitos integrados y las interfaces del sistema en una PSoC. Los componentes son circuitos integrados incorporados libres representados por iconos.
 - Incluye compilador GCC libre, apoya Keil / ARM MDK compilador.
 - Soporta la programación del dispositivo y la depuración. [7]

3 DESARROLLO DEL PROYECTO

3 DESARROLLO DEL PROYECTO	17
3.1 Revisión Teórica	18
3.1.1 CPU PSoC 3	18
3.1.2 CPU PSoC 5LP	22
3.1.3 GPIO.....	23
3.1.4 Timers, Contadores y PWM	24
3.1.5 ADC Sigma-Delta	25
3.1.6 Comparadores	26
3.1.7 LUT.....	27
3.1.8 Amplificadores operaciones.....	27
3.1.9 DAC.....	28
3.2 Manual de prácticas.....	29
3.2.1 Características de la CPU.....	29
3.2.2 GPIO.....	30
3.2.3 Manejo de interrupciones.....	43
3.2.4 Contador.....	49
3.2.5 Timer	57
3.2.6 Contador y Timer.....	65
3.2.7 Manejo de LCD	70
3.2.8 ADC.....	77
3.2.9 VDAC.....	85
3.2.10 VDAC y ADC	89
3.2.11 OpAmp	94
3.2.12 Comparadores.....	98
3.2.13 PWM.....	104
3.2.14 Control de velocidad de un motor DC con PWM.....	108
3.2.15 UART.....	117
3.2.16 LUT.....	122
3.3 Dispositivo Hardware para PSoC 5LP	125

Realizar una revisión teórica de la CPU, GPIO, interrupciones, contadores, temporizadores, amplificadores operacionales, VDAC, ADC Delta Sigma, PWM, UART, LUT, disponibles en el entorno PSoC Creator

3.1 REVISIÓN TEÓRICA

3.1.1 CPU PSoC 3

Los dispositivos CY8C38 utilizan una CPU 8051 de un solo ciclo, que es totalmente compatible con el conjunto de instrucciones original de MCS-51. La familia CY8C38 utiliza una arquitectura RISC para segmentar, que ejecuta la mayoría de las instrucciones en 1 a 2 ciclos para proporcionar un rendimiento máximo de hasta 33 MIPS con un promedio de 2 ciclos por instrucción. El subsistema de la CPU 8051 incluye las siguientes características:

- Solo un ciclo 8051 de la CPU.
- Hasta 64 KB de memoria flash, hasta 2 KB de memoria EEPROM, y hasta 8 KB de SRAM.
- Caché de instrucciones de 512 bytes entre la CPU y el flash.
- Controlador de interrupciones programable vectorial anidado.
- Controlador DMA.
- HUB periférica (PHUB).
- Interfaz de memoria externa (EMIF).

3.1.1.1 Modos de direccionamiento

Los siguientes modos de direccionamiento son compatibles con el 8051:

- Direccionamiento directo: El operando se especifica mediante una dirección directa en el campo de 8 bits. Sólo la RAM interna y los registros SFR se puede acceder mediante esta modalidad.
- Direccionamiento indirecto: La instrucción especifica el registro que contiene la dirección del operando. Los registros R0 o R1 se utilizan para especificar la dirección de 8 bits, mientras que el registro de puntero de datos (DPTR) se utiliza para especificar la dirección de 16 bits.
- Direccionamiento de registros: Ciertas instrucciones de acceso a uno de los registros (R0 a R7) en el banco de registros especificado. Estas instrucciones son más eficientes porque no hay necesidad de un campo de dirección.
- Registro de instrucciones específicas: Algunas instrucciones son específicas para determinados registros.
- Las constantes inmediatas: Algunas instrucciones llevan el valor de las constantes directamente en lugar de una dirección.
- El direccionamiento indexado: Este tipo de direccionamiento sólo se puede utilizar para una lectura de la memoria del programa.
- Bit Direccionamiento: En este modo, el operando es uno de los 256 bits.

3.1.1.2 Conjunto de instrucciones

El conjunto de instrucciones 8051 está muy optimizado para la manipulación de 8 bits y operaciones booleanas. Los tipos de instrucciones soportadas incluyen:

- Instrucciones aritméticas.
- Instrucciones lógicas.
- instrucciones de transferencia de datos.
- Instrucciones booleanas
- Programa de ramificación instrucciones

3.1.1.2.1 Resumen de instrucciones

3.1.1.2.1.1 Instrucciones aritméticas

Instrucciones aritméticas soportan los registros directos, indirectos, constantes inmediatas, y la instrucción de registros específicas.

3.1.1.2.1.2 Instrucciones lógicas

Las instrucciones lógicas realizan operaciones booleanas como AND, OR, XOR.

3.1.1.2.1.3 Instrucciones de transferencia de datos

Las instrucciones de transferencia de datos son de tres tipos: núcleo de la memoria RAM, XDATA RAM, y las tablas de búsqueda.

3.1.1.2.1.4 Instrucciones Booleanas

El conjunto de instrucciones incluye todo el menú de operaciones de bits, tales como movimiento, establecer, eliminar de palanca, OR y AND las instrucciones y las instrucciones de salto condicional.

3.1.1.2.1.5 Instrucciones condicionales

El 8051 es compatible con un conjunto de instrucciones de salto condicional e incondicional que ayudan a modificar el flujo de ejecución del programa.

3.1.1.3 DMA Y PHUB

El PHUB y el controlador de DMA son responsables de la transferencia de datos entre la CPU y los periféricos, y también las transferencias de datos entre los periféricos. El PHUB consiste en:

- Un eje central que incluye el controlador de DMA, árbitro, y el router.
- Múltiples rayos que irradian hacia fuera del cubo para la mayoría de los periféricos.

Hay dos maestros PHUB: la CPU y el controlador de DMA. Ambos maestros pueden iniciar transacciones en el bus. Los canales de DMA pueden manejar la comunicación periférica sin intervención de la CPU.

3.1.1.3.1 Niveles de prioridad

Los canales DMA de mayor prioridad (número de prioridad inferior) pueden interrumpir transferencias corrientes DMA. En el caso de una interrupción, se permite que la transferencia de corriente para completar su transacción actual.

3.1.1.3.2 Modos de transacción soportados

La configuración flexible de cada canal de DMA y la capacidad de la cadena de múltiples canales permiten la creación de los dos casos simples y complejos de uso. casos de uso general incluyen, pero no se limitan a:

3.1.1.3.2.1 DMA simple:

En un caso simple DMA, un solo transfiere TD datos entre una fuente y sumidero (periféricos o posición de memoria).

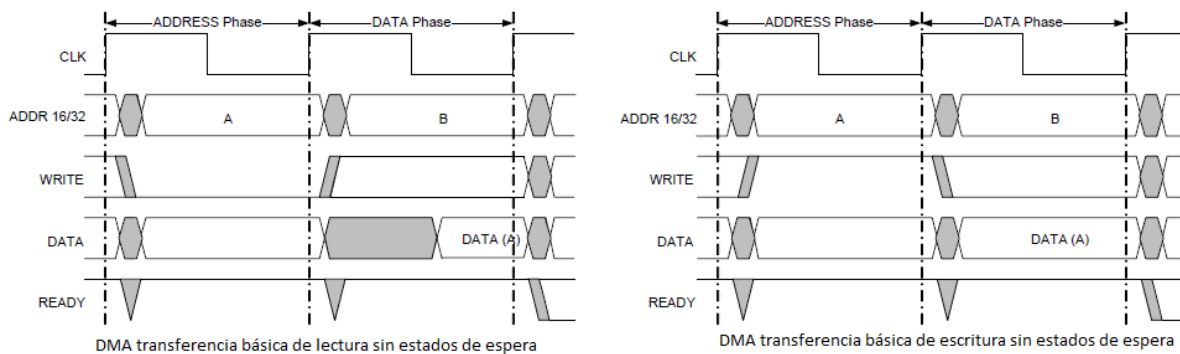


Figura 3. Diagrama de tiempo del DMA [5]

3.1.1.3.2.2 Repetición automática de DMA

DMA de repetición automática se usa típicamente cuando un patrón estático se lee repetidamente desde la memoria del sistema y se escribe en un periférico.

3.1.1.3.2.3 DMA ping pong

Utiliza el búfer doble para permitir una memoria intermedia para ser llenado por un cliente, mientras que otro cliente está consumiendo los datos recibidos previamente en la otra memoria intermedia.

3.1.1.3.2.4 DMA circular

El DMA circular es similar al DMA de ping pong, excepto que contiene más de dos buffers. En este caso hay varios TDs; después de que el ultimo TD se completa, la cadena regresa de nuevo al primer TD.

3.1.1.3.2.5 Dispersión reunida del DMA

En el caso de la dispersión reunida DMA, hay múltiples fuentes o destinos que se requieren para llevar a cabo eficazmente una operación global DMA no contiguos.

3.1.1.3.2.6 Cola de paquetes en el DMA

Es similar a la dispersión reunida, pero se refiere específicamente a los protocolos de paquetes. Con estos protocolos, puede ser de configuración independiente, los datos, y las fases de estado asociados con el envío o recepción de un paquete.

3.1.1.3.2.7 DMA anidado

Un TD puede modificar otro TD, como el espacio de configuración del TD es mapeado en memoria similar a cualquier otro periférico.

3.1.1.3.2.8 Control de interrupciones

El controlador de interrupción proporciona un mecanismo para los recursos de hardware para cambiar la ejecución del programa a una nueva dirección, independientemente de la tarea actual que está siendo ejecutado por el código principal.

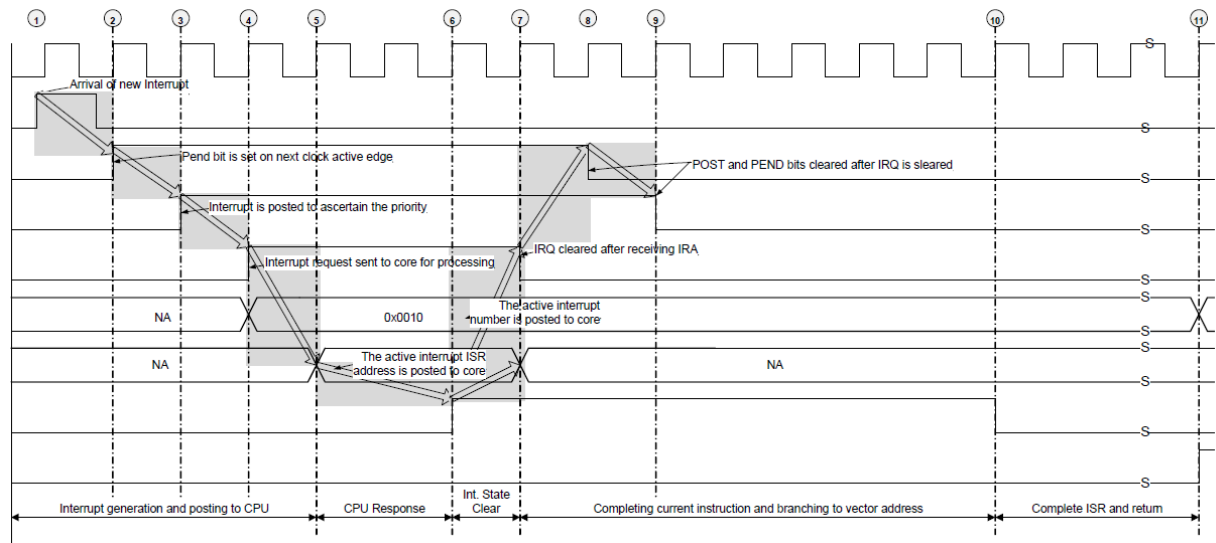


Figura 4. Flujo típico de eventos cuando se desencadena una interrupción [5]

Cuando una interrupción está pendiente, la instrucción en curso se ha completado y el contador de programa se inserta en la pila. Ejecución de código a continuación, salta a la dirección del programa proporcionado por el vector. Después de que se completó la ISR, una instrucción RETI se ejecuta y la ejecución vuelve a la instrucción siguiente interrumpido la previamente de instrucciones. Para ello la instrucción del RETI aparece el contador de programa de la pila.

3.1.2 CPU PSoc 5LP

La familia de dispositivos CY8C58LP tienen una CPU ARM CORTEX-M3. El Cortex-M3 es de bajo consumo de 32 bits, de tres etapas de segmentación de CPU, Harvard-arquitectura que ofrece 1,25DMIPS/MHz. Está destinado a aplicaciones profundamente arraigados que requieren características de manejo de interrupción rápida.

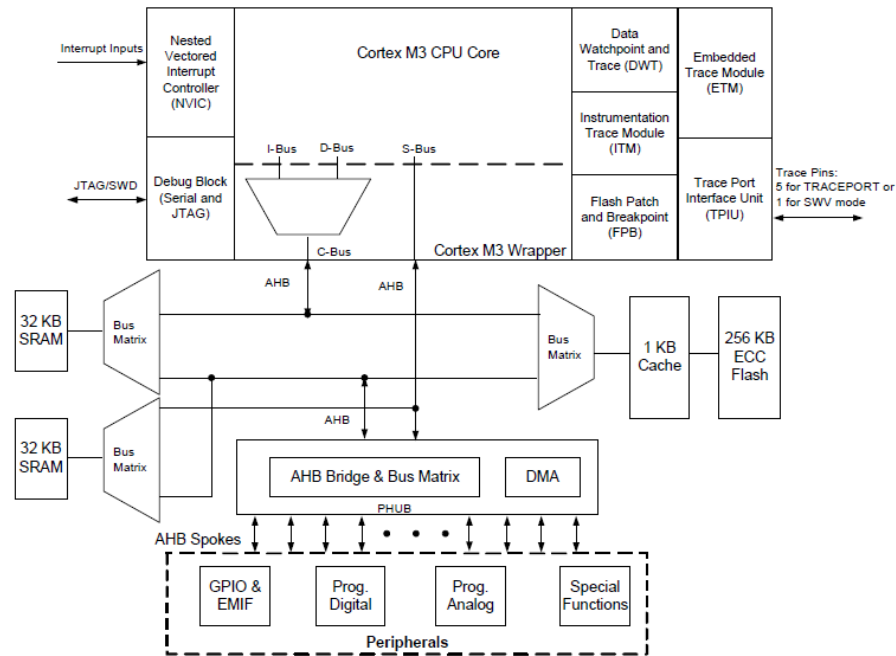


Figura 5. Diagrama de bloques del ARM Cortex-M3 [7]

El subsistema de la CPU ARM Cortex-M3 incluye las siguientes características:

- CPU ARM Cortex-M3.
- Controlador de interrupciones programable con vectores anidados (NVIC), estrechamente integrado con el núcleo de la CPU.
- Módulos de depuración y traza de funciones completo, estrechamente integrado con el núcleo de la CPU.
- Hasta 256 KB de memoria flash, 2 KB de memoria EEPROM, y 64KB de RAM.
- Controlador de memoria caché.
- Periféricos HUB (PHUB).
- Controlador DMA.
- Interfaz de memoria externa (EMIF).

3.1.2.1 Modos de operación

El Cortex-M3 opera ya sea a nivel de privilegio o el nivel de usuario, y ya sea en el modo de hilo o el modo de controlador.

A nivel de usuario, el acceso a ciertas instrucciones, registros especiales, los registros de configuración y componentes de depuración se bloquea. Los intentos de acceder a ellos causan una excepción de fallo. A nivel privilegiado, acceder a todas las instrucciones y registros es permitido. El procesador funciona en el modo de controlador (siempre en el nivel de privilegio) al manipular una excepción, y en el modo de hilo cuando no.

3.1.2.2 Registro de la CPU

Los registros de la CPU Cortex-M3 se enumeran en la Tabla 10. Los registros R0-R15 son los 32 bits de ancho.

3.1.2.3 Controlador de memoria caché

La familia CY8C58LP tiene un 1 KB, caché de instrucciones asociativa en conjunto de 4 vías entre la CPU y la memoria flash. Esto mejora la velocidad de ejecución de instrucciones y reduce el consumo de energía del sistema al requerir el acceso de destello menos frecuente.

3.1.2.4 DMA Y PHUB

La PSoC 5LP contiene las mismas características que la PSoC 3 (Capítulo 3.1.3). La única diferencia es que la PSoC 5LP contiene un modo de transacción adicional que es *DMA indexada*. En este caso, un maestro externo requiere el acceso a ubicaciones del bus de sistema como si esos lugares eran compartidos memoria.

3.1.2.5 Control de interrupciones

El controlador de interrupción Vectored Nested (NVIC) maneja las interrupciones de los periféricos, y pasa los vectores de interrupción a la CPU. Las características incluyen:

- 32 interrupciones. Múltiples fuentes para cada interrupción.
- Ocho niveles de prioridad, con control de prioridad dinámica.
- Prioridad agrupación. Esto permite la selección de adelantarse y los niveles no apropiante de interrupción.
- El apoyo a la cola en cadena, y la llegada tardía, de las interrupciones. Esto permite el procesamiento de interrupciones de regreso a la espalda sin el añadido del ahorro y la restauración del estado entre las interrupciones.
- Estado del procesador guarda automáticamente en la entrada de interrupción, y restaurado a la salida de interrupción, sin gastos de instrucciones.

3.1.3 GPIO

Proporciona interfaces a la CPU, periféricos digitales, periféricos analógicos, interrupciones, controlador LCD y CapSense.

3.1.3.1 Sistema de E/S y enrutamiento

Las E/S PSoC son extremadamente flexibles. Cada GPIO tiene la capacidad de E/S analógica y digital. Todas las E/S tienen un gran número de modos de manejo, que se establecen en

el POR. PSoC también proporciona hasta cuatro dominios individuales de tensión de E/S a través de los pines VDDIO.

3.1.3.2 Características de los pines GPIO

- Controlador de LCD en dispositivos.
- CapSense.
- Capacidad analógica y digital de entrada y salida.
- 100 uA de capacidad de corriente continua.
- Fuerza de accionamiento estándar hasta 1,71 V.

3.1.4 Timers, Contadores y PWM

Los timers, contadores y PWMs con periféricos dedicados de 16. PSoC Creator le permite elegir el temporizador, contador, y PWM con las características que necesita. El conjunto de herramientas utiliza los recursos disponibles más óptimos.

Los timers, contadores y PWMs pueden seleccionar entre múltiples fuentes de reloj, con las señales de entrada y salida conectados a través de la ruta de DSI. El enrutamiento DSI permite conexiones de entrada y salida a cualquier pin del dispositivo y cualquier señal digital interna accesible a través de la DSI.

3.1.4.1 Características

- Timer, contador y PWM de 16 bits (solo contador de bajada).
- Fuente de reloj seleccionable.
- Recarga Período en el arranque, reinicio, y el recuento de terminales.
- Interrumpir el contador de terminal, comparar verdadera, o la captura.
- Contador dinámico de lectura.
- Modo de captura de temporizador.
- Modo de marcha libre.
- Modo de un impulso (parada al final del período).
- Salidas PWM complementarias con banda muerta.
- Salida de PWM muerta.

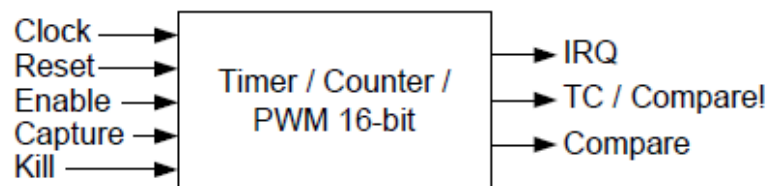


Figura 6. Timer/Contador/PWM [5][7]

3.1.5 ADC Sigma-Delta

Las familias de dispositivos CY8C58LP y CY8C38 contienen un ADC sigma-delta. Este ADC ofrece entrada diferencial, de alta resolución y una excelente linealidad, lo que es una buena opción de ADC tanto para aplicaciones de procesamiento de señales de audio. El ADC puede ser configurado para emitir una resolución de 20 bits a velocidades de datos de hasta 187 sps.

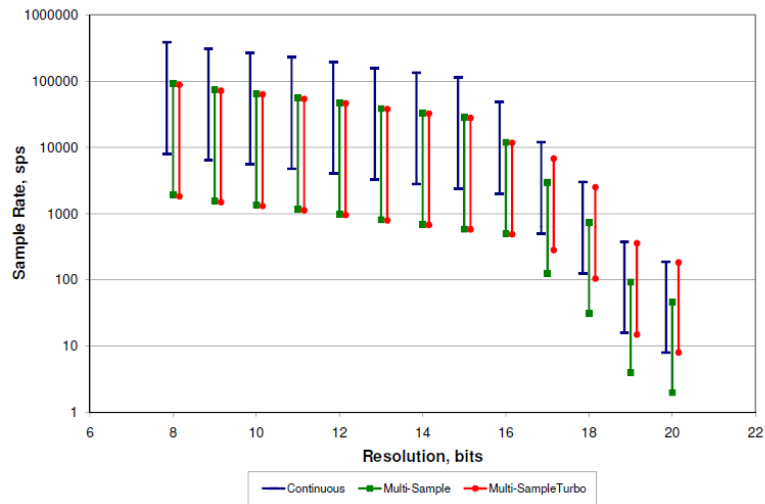


Figura 7. ADC Sigma-Delta, Frecuencia de muestreo, Rango $\pm 1.024V$ [5][7]

3.1.5.1 Descripción funcional

El ADC se conecta y configura tres componentes básicos, buffer de entrada, modulador delta-sigma, y Decimador. El modulador delta-sigma realiza la conversión real de análogo a digital. El decimador convierte el flujo de datos serie de alta velocidad en resultados de ADC en paralelo.

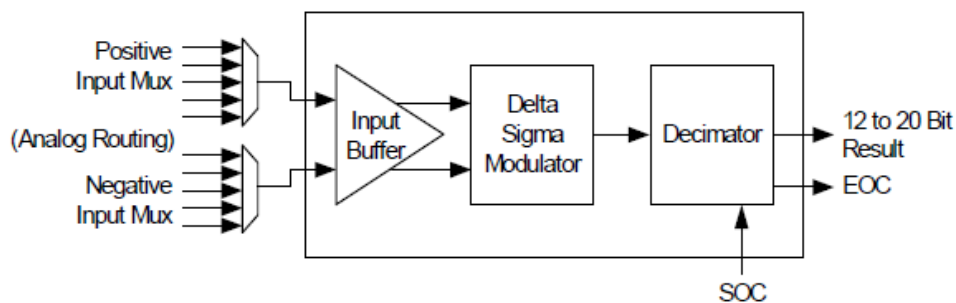


Figura 8. Diagrama de bloques del ADC Sigma-Delta [5][7]

La resolución y frecuencia de muestreo son controlados por el Decimador. Los datos se segmentan en el Decimador; la salida es una función de las últimas cuatro muestras.

3.1.5.2 *Modos de operación*

El ADC puede ser configurado por el usuario para operar en uno de cuatro modos: Una sola muestra, muestra múltiples, continuas o multimuestra (Turbo).

3.1.5.3 *Una sola muestra*

El ADC realiza una conversión de la muestra en un disparador. En este modo, el ADC se mantiene en estado de espera esperando la señal de SoC. Cuando SoC se señala el ADC realiza cuatro conversiones sucesivas. Las tres primeras conversiones ceban el decimador. El resultado ADC es válida y disponible después de la cuarta conversión, momento en el cual se genera la señal de EoC.

3.1.5.4 *Continuos*

El modo de muestreo continuo se utiliza para tomar varias muestras sucesivas de una sola señal de entrada.

3.1.5.5 *Multimuestra*

El modo de Multimuestra es similar al modo continuo, excepto que el ADC se restablece entre las muestras. Este modo es útil cuando la entrada cambia entre múltiples señales.

3.1.5.6 *Multimuestra Turbo*

El modo multimuestra (turbo) opera idéntico al modo multimuestra para resoluciones de 8 a 16 bits.

3.1.5.7 *Inicio de entrada de conversión*

La señal de SoC se utiliza para iniciar una conversión ADC. Un reloj digital o salida UDB se pueden utilizar para conducir esta entrada. Se puede utilizar cuando el período de muestreo debe ser más largo que el tiempo de conversión ADC o cuando el ADC debe ser sincronizado con otro hardware.

3.1.5.8 *Fin de la salida de conversión*

La señal de EoC pasa a nivel alto al final de cada conversión ADC. Esta señal puede usarse para activar o bien una interrupción y solicitud de DMA.

3.1.6 Comparadores

Las familias de dispositivos CY8C58LP y CY8C38 contienen cuatro comparadores. Los comparadores tienen estas características:

- Entrada offset, a menos de 5mV.
- Rango de entrada-Rail-to-rail de modo común (VSSA a VDDA).
- Velocidad y potencia pueden ser negociados mediante el uso de uno de los tres modos: rápido, lento, o de bajo consumo de energía ultra.
- Salidas del comparador se pueden dirigir a las tablas de consulta para realizar funciones lógicas simples y luego también se pueden dirigir a los bloques digitales.

- La entrada positiva de los comparadores se puede pasar opcionalmente a través de un filtro de paso bajo. Se proporcionan dos filtros.
- Entradas del comparador pueden ser conexiones a GPIO, salidas del DAC y salidas del bloque SC.

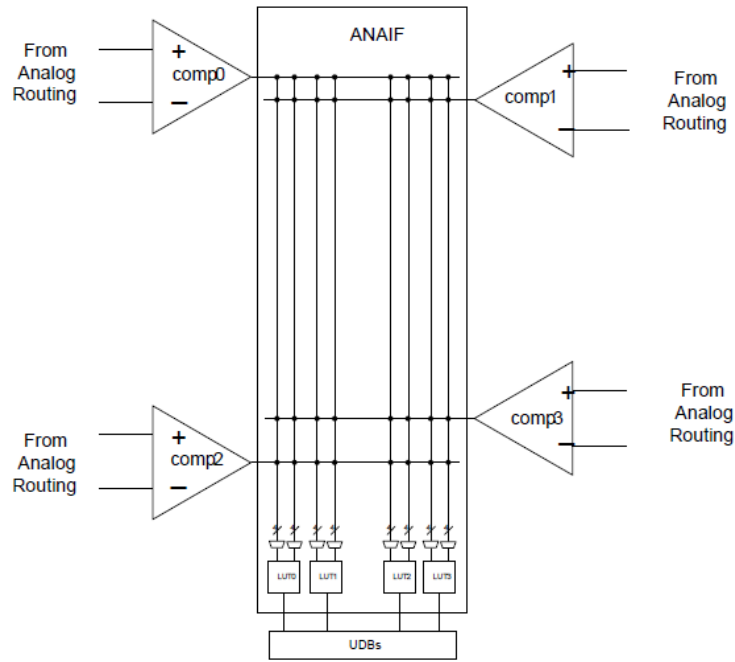


Figura 9. Comparador analógico. [5][7]

3.1.7 LUT

Las familias de dispositivos CY8C58LP y CY8C38 contienen cuatro tablas de búsqueda. El LUT es una tabla de búsqueda de dos entradas, una salida que está impulsado por cualquier, uno o dos de los comparadores en el chip. La salida de cualquier LUT se dirige a la interfaz del sistema digital de la matriz UDB.

3.1.8 Amplificadores operaciones

Las familias de dispositivos CY8C58LP y CY8C38 contienen cuatro amplificadores operacionales de propósito general.

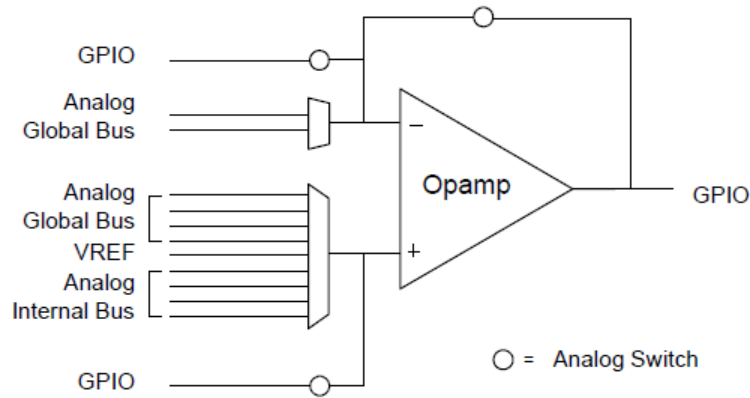


Figura 10. Amplificador Operacional [5][7]

El amplificador operacional es comprometido y se puede configurar como una etapa de ganancia o seguidor de tensión en las señales externas o internas.

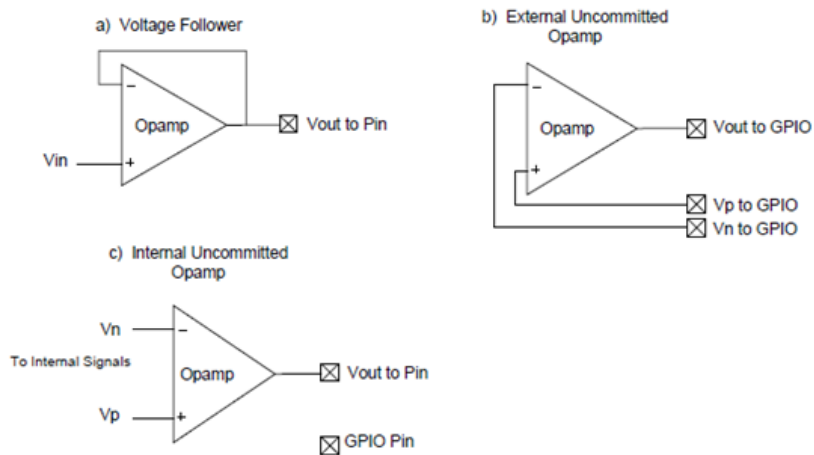


Figura 11. Configuraciones de Opamp [5][7]

El amplificador operacional tiene tres velocidades, lenta, media y rápida. El modo lento consume la menor cantidad de energía en reposo y el modo rápido consume la mayor cantidad de energía. Las entradas son capaces de pivotar rail-to-rail. La oscilación de salida es capaz de funcionar carril-a-carril a baja corriente de salida, a menos de 50 mV de los carriles. Al conducir cargas de corriente elevada (alrededor de 25 mA) la tensión de salida sólo se puede conseguir en el plazo de 500 mV de los carriles.

3.1.9 DAC

Las familias de dispositivos CY8C58LP y CY8C38 contienen cuatro convertidores de señal digital a analógica (DAC). Cada DAC es 8 bits y se puede configurar para cualquier salida de tensión o corriente. Los DAC soportan CapSense, la regulación de la fuente de alimentación, y la generación de formas de onda. Cada DAC tiene las siguientes características:

- Voltaje o corriente de salida ajustable en 255 pasos.
- Tamaño de paso programable (selección de rango).
- Ocho bits de calibración para corregir $\pm 25\%$ de error de ganancia
- Opción Fuente y sumidero de corriente de salida.
- Tasa de conversión 8 Msps para la salida de corriente
- Tasa de conversión 1 Msps para la salida de voltaje
- Monótona en la naturaleza
- Entradas de datos y estroboscópicas pueden ser proporcionados por la CPU o DMA o encaminan directamente desde la DSI
- Dedicado pin de salida de baja resistencia para el modo de alta corriente

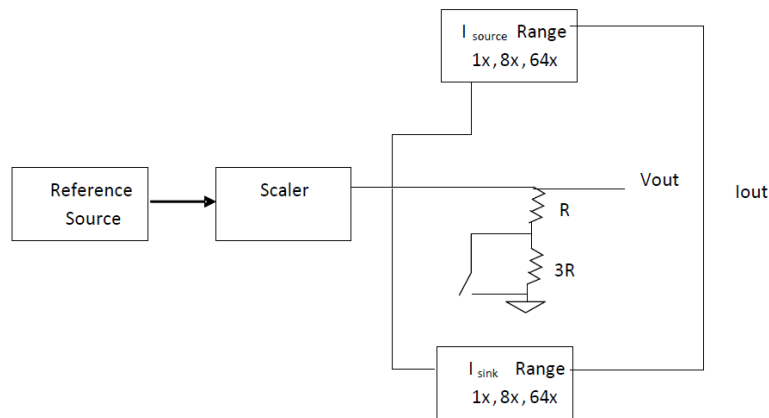


Figura 12. Diagrama en bloques del DAC. [5][7]

3.1.9.1 DAC de voltaje

Para el DAC de voltaje (VDAC), la corriente de salida del DAC se dirige a través de resistencias. Los dos rangos disponibles para la VDAC son 0 a 1,02 V y 0 a 4,08 V. En el modo de tensión de cualquier carga conectada a la salida de un DAC debe ser puramente capacitivo (la salida del VDAC no está tamponada).

3.2 MANUAL DE PRÁCTICAS

Desarrollar las guías de laboratorio enfocadas en la CPU, GPIO, interrupciones, contadores, temporizadores, amplificadores operacionales, VDAC, ADC delta sigma, PWM, UART, LUT disponibles en el entorno de desarrollo PSoC Creator.

3.2.1 Características de la CPU

3.2.1.1 Objetivos

- Conocer las características de la PSoC 3.
- Conocer las características de la PSoC 5LP.

3.2.1.2 PSoC 3

Los dispositivos CY8C38 utilizan una CPU 8051 de un solo ciclo, que es totalmente compatible con el conjunto de instrucciones original de MCS-51. La familia CY8C38 utiliza una arquitectura RISC para segmentar, ejecuta la mayoría de las instrucciones en 1 a 2 ciclos para proporcionar un rendimiento máximo de hasta 33 MIPS con un promedio de 2 ciclos por instrucción. El subsistema de la CPU 8051 incluye las siguientes características:

- Solo un ciclo 8051 de la CPU.
- Hasta 64 KB de memoria flash, hasta 2 KB de memoria EEPROM, y hasta 8 KB de SRAM.
- Caché de instrucciones de 512 bytes entre la CPU y el flash.
- Controlador de interrupciones programable vectorial anidado.
- Controlador DMA.
- HUB periférica (PHUB).
- Interfaz de memoria externa (EMIF).

3.2.1.3 PSoC 5LP

La familia de dispositivos CY8C58LP tienen una CPU ARM CORTEX-M3. El Cortex-M3 es de bajo consumo de 32 bits, de tres etapas para segmentar la CPU, Harvard-arquitectura que ofrece 1,25DMIPS/MHz. Está destinado a aplicaciones profundamente arraigados que requieren características de manejo de interrupción rápida.

- Controlador de interrupciones programable con vectores anidados (NVIC), estrechamente integrado con el núcleo de la CPU.
- Módulos de depuración y traza de funciones completo, estrechamente integrado con el núcleo de la CPU.
- Hasta 256 KB de memoria flash, 2 KB de memoria EEPROM, y 64KB de RAM.
- Controlador de memoria caché.
- Periféricos HUB (PHUB).
- Controlador DMA.
- Interfaz de memoria externa (EMIF).

3.2.2 GPIO

Los pines permiten que los recursos de hardware puedan conectarse a un puerto físico. Proporciona acceso a las señales externas a través de un pin físico debidamente configurado. También permite que las características eléctricas (por ejemplo, el modo de dispositivo) para ser elegido para uno o más pines; estas características son utilizados por PSoC Creator para colocar automáticamente y enviar las señales dentro del componente.

3.2.2.1 Pines de salida

3.2.2.1.1 Objetivo

- Conocer el funcionamiento de los pines de salida.

- Familiarizarse con el entorno PSoC Creator.
- Configurar dos pines de salida digital para que encienda el LED de la tarjeta y otro LED conectado a cualquier pin.

3.2.2.1.2 Guía paso a paso

Para crear un nuevo proyecto abrimos PSoS Creator, vamos a pestaña File, New, Project.

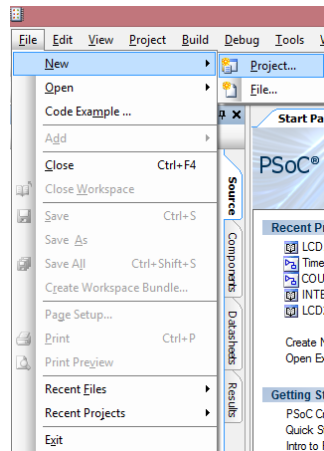


Figura 13. Crear un nuevo proyecto.

En la ventana que nos aparece escogemos la tarjeta que vamos a utilizar, es muy importante escoger la referencia del microcontrolador de nuestra tarjeta ya que si escogemos un modelo que no es el mismo de nuestra tarjeta no servirá el proyecto que realicemos. En este caso yo escogí la referencia de la PSoC 5LP. Si su modelo no aparece en los de la lista puede dar click al final de la lista en Launch Device Selector y ahí encontrará los demás modelos.

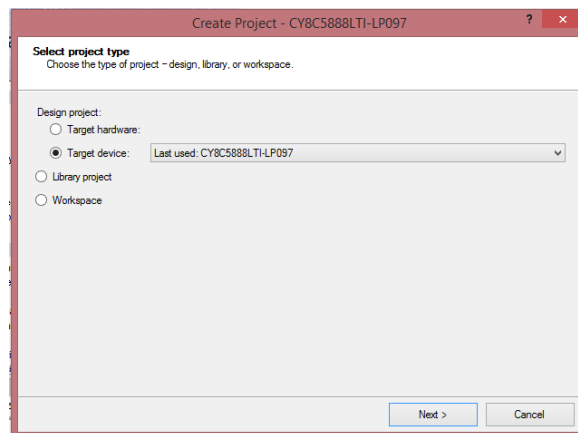


Figura 14. Escoger tarjeta de desarrollo

Le damos siguiente y escogemos un esquemático vacío o un código de ejemplo, según lo que usted desee. En este caso se escogerá un esquemático vacío.

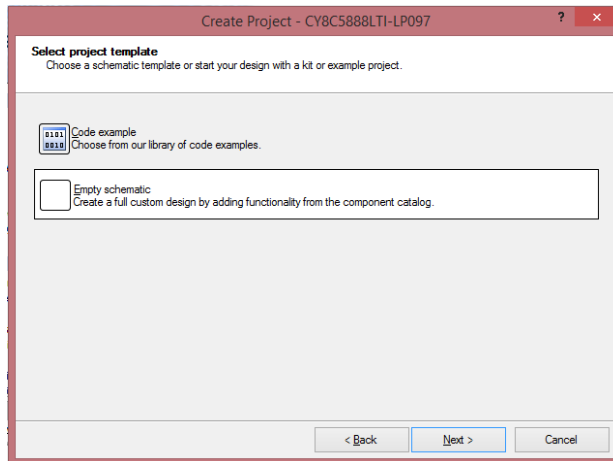


Figura 15. Escoger tipo de esquemático

Le damos siguiente, en este espacio debemos colocar el nombre del WorkSpace y del proyecto, es muy importante que tengan el mismo nombre y se encuentren en una carpeta ya que los proyectos de PSoC generan muchos archivos. Le damos finalizar.

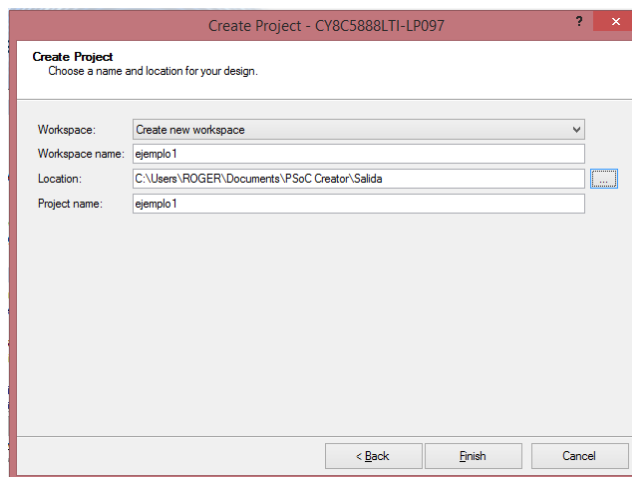


Figura 16. Escoger un nombre y localización para el proyecto.

3.2.2.1.2.1 Configuración de Hardware

En la ventana del WorkSpace, del lado izquierdo está el catálogo de componentes de Cypress. Buscamos Ports and Pins y escogemos una salida digital y se arrastra al WorkSpace.

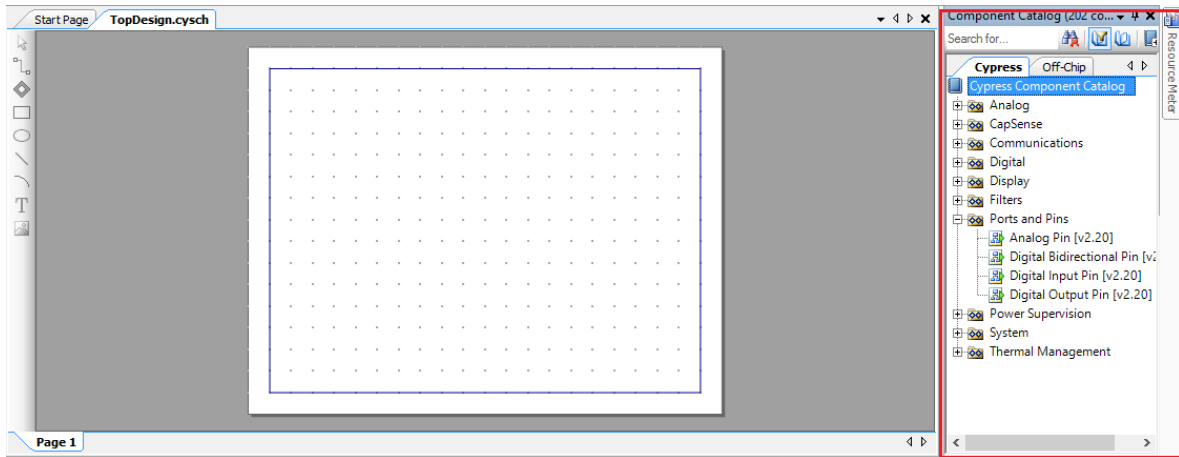


Figura 17. Ubicación de los componentes.

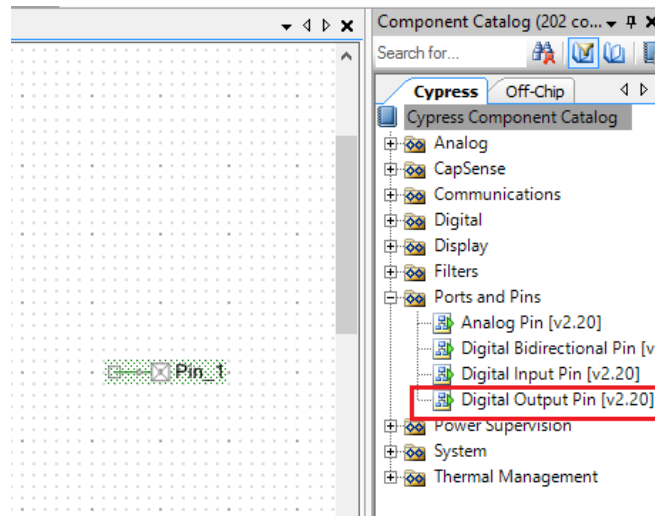


Figura 18. Ubicación de los pines digitales de salida

Hacemos doble click en el pin para configurar. En esta ventana se puede cambiar el nombre del pin, cambiar su modo de operación, su estado inicial, etc. En este caso, se deshabilitará la conexión externa (HW) porque utilizaremos el LED de la tarjeta, nombre del pin será cambiado por LED, y su estado inicial será '1'.

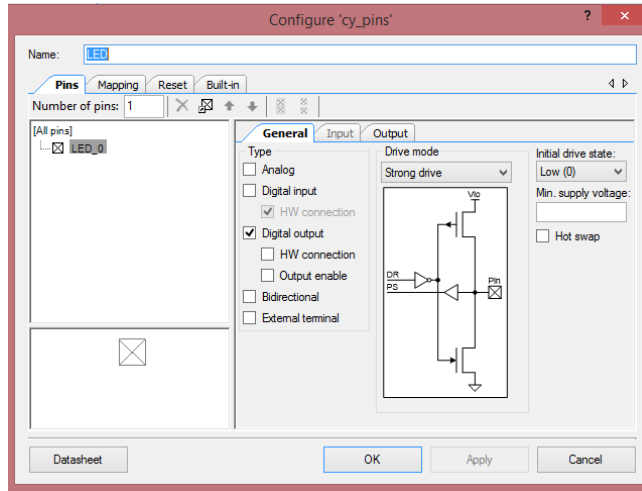


Figura 19. Configuración del pin de salida Pin_1 Como LED

Para utilizar otro pin de la tarjeta que no sea el LED interno, sacamos un pin más de salida y lo configuramos igual que el anterior, y le colocamos LED2.

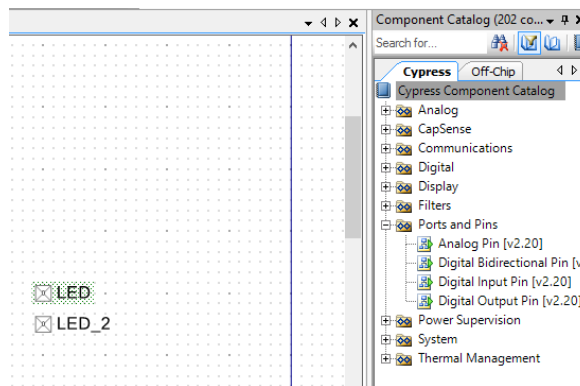


Figura 20. Componentes en el WorkSpace

Luego de esto es importante guardar (1) y compilar (2) para asegurarnos de que no exista ningún error en nuestro proyecto.

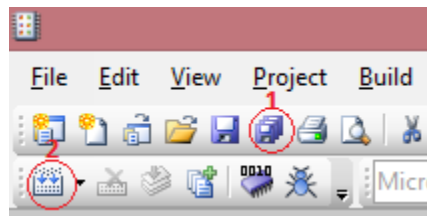


Figura 21. Guardar y Compilar

3.2.2.1.2.2 Asignación de Pines

Para asignar pines nos vamos al lado izquierdo de la pantalla y seleccionamos la pestaña .cydwr

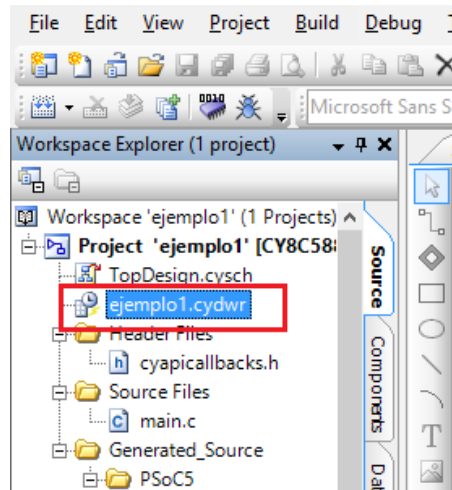


Figura 22. Archivo para asignar pines

Se abrirá una ventana con el diagrama de nuestro dispositivo. Para el caso de la PSoC 5LP los pines a utilizar son: el pin P2[1] que es el LED interno de la tarjeta y cualquier otro pin este caso escogí el pin P1[7].

Para PSoC 3 se pueden utilizar los pines P6[3] y P6[2] que son LEDs internos de la tarjeta.

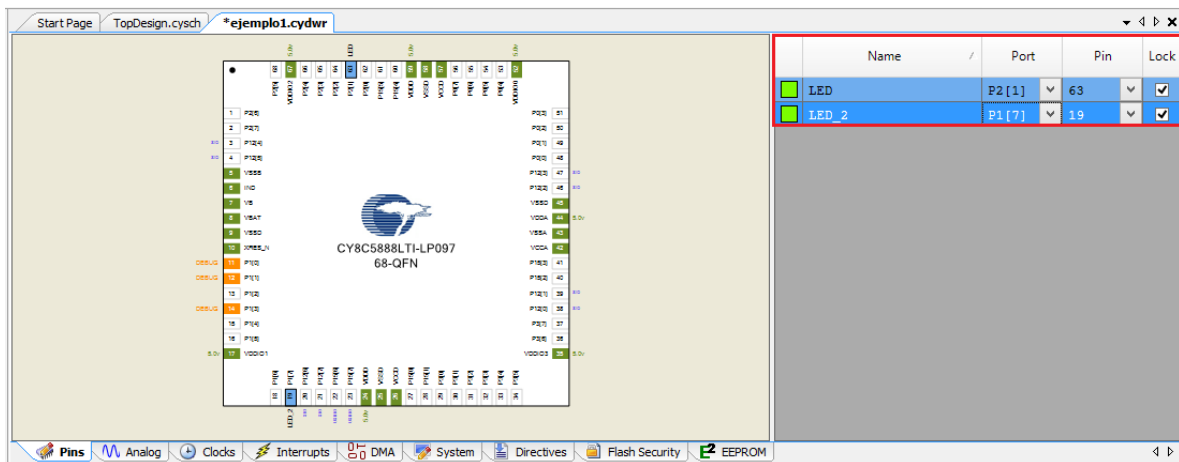


Figura 23. Pines asignados

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.2.1.2.3 Código del Main

En el lado izquierdo de la pantalla, nos vamos a la pestaña que dice main y damos doble click.

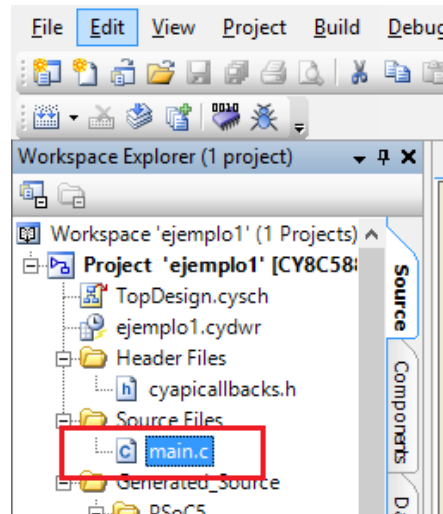


Figura 24. Archivo para ir al Main

En el main nos dirigimos a la parte donde se encuentra el for(;;) y dentro de él escribimos la instrucción Pin_1_Write(~Pin_1_Read()), esta instrucción sirve para escribir en el pin deseado el inverso del estado en que se encuentra. La palabra Pin_1 se cambia por el nombre asignado a cada pin.

```

#include <project.h>

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for (;;)
    {
        LED_Write(~LED_Read());
        CyDelay(500);
        LED_2_Write(~LED_2_Read());
        CyDelay(500);
    }
}
  
```

Figura 25. Código para del Main los pines de salida

Guardamos (1), compilamos (2), generamos aplicación (3) y programamos (4).



Figura 26. Guardar, Compilar, Generar Aplicación y Programar

3.2.2.2 Pines de entrada

3.2.2.2.1 Objetivos

- Conocer el funcionamiento de los pines de entrada.
- Familiarizarse con el entorno PSoC Creator.
- Configurar un pin de entrada para que encienda el LED de la tarjeta.

3.2.2.2.2 Guía paso a paso

Para crear un nuevo proyecto abrimos PSoC Creator, vamos a pestaña File, New, Project.

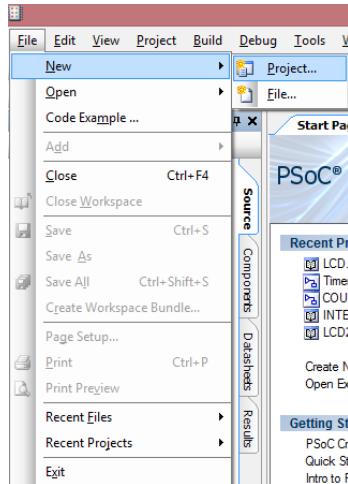


Figura 27. Crear un nuevo proyecto

En la ventana que nos aparece escogemos la tarjeta que vamos a utilizar, es muy importante escoger la referencia del microcontrolador de nuestra tarjeta ya que si escogemos un modelo que no es el mismo de nuestra tarjeta no servirá el proyecto que realicemos. En este caso yo escogí la referencia de la PSoC 5LP. Si su modelo no aparece en los de la lista puede dar click al final de la lista en Launch Device Selector y ahí encontrará los demás modelos.

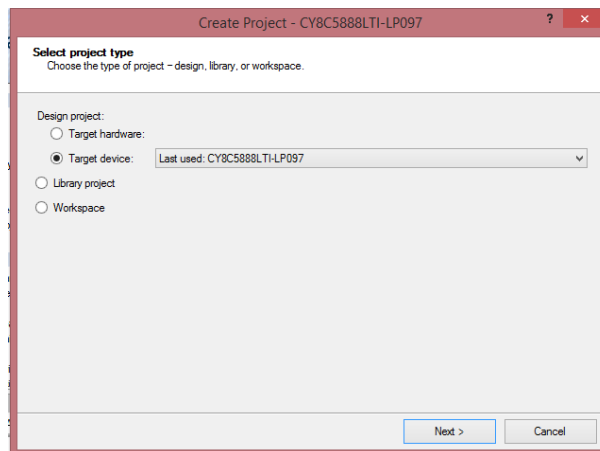


Figura 28. Escoger tarjeta de desarrollo

Le damos siguiente y escogemos un esquemático vacío o un código de ejemplo, según lo que usted desee. En este caso se escogerá un esquemático vacío.

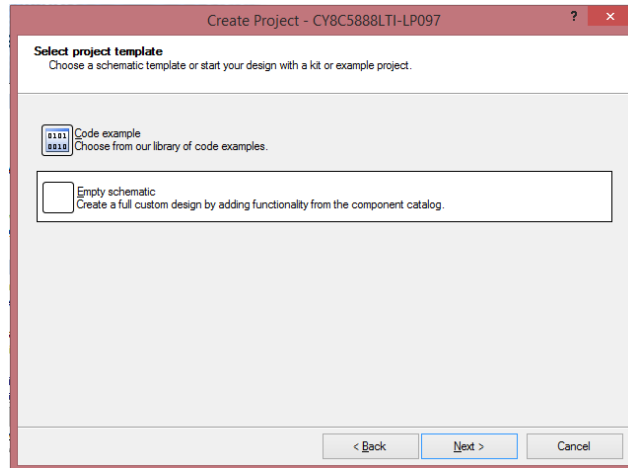


Figura 29. Escoger tipo de esquemático

Le damos siguiente, en este espacio debemos colocar el nombre del Workspace y del proyecto, es muy importante que tengan el mismo nombre y se encuentren en una carpeta ya que los proyectos de PSoC generan muchos archivos. Le damos finalizar.

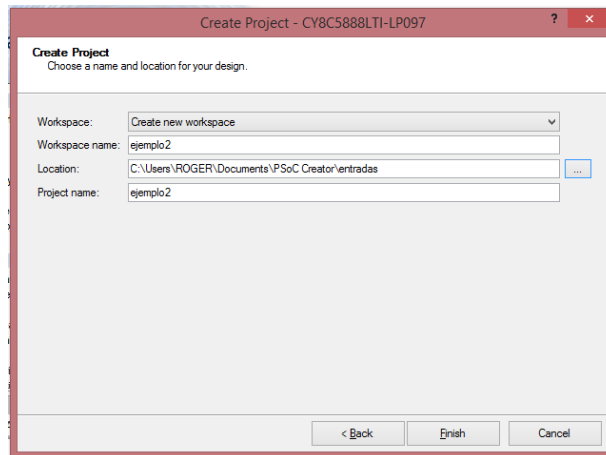


Figura 30. Escoger un nombre y localización para el proyecto

3.2.2.2.1 Configuración de Hardware

En la ventana del Workspace, del lado izquierdo está el catálogo de componentes de Cypress. Buscamos Ports and Pins y escogemos una entrada digital y se arrastra al Workspace.

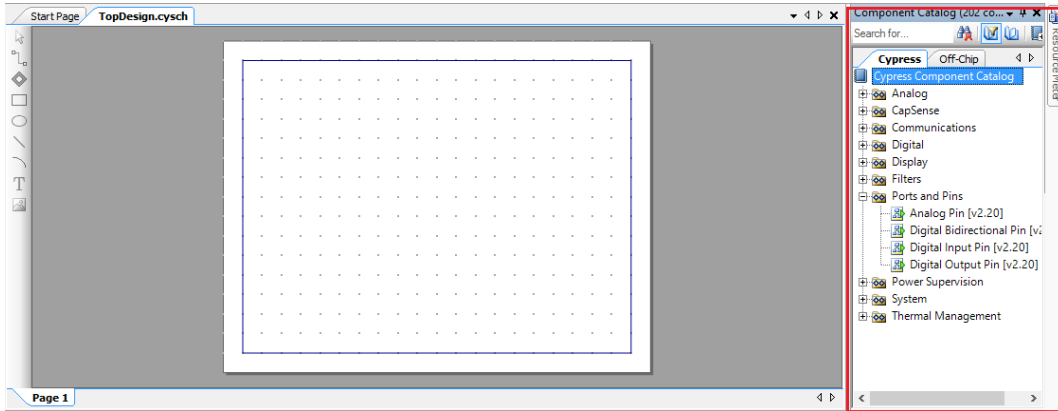


Figura 31. Ubicación de los componentes

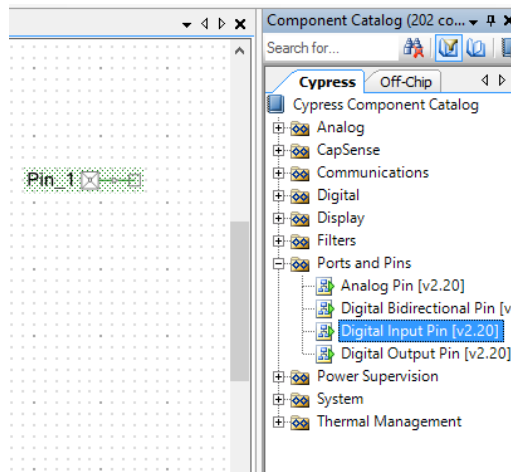


Figura 32. Ubicación de los pines digitales de entrada

Hacemos doble click sobre el pin para configurar, le quitamos la conexión HW y lo configuramos como Resistive pull up. Aplicar y aceptar.

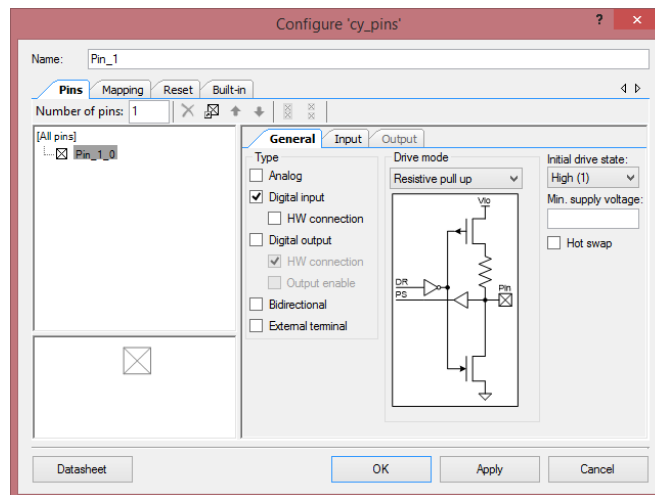


Figura 33. Configuración del pin de entrada Pin_1

Ahora necesitamos un pin de salida digital. Vamos a ports and pins y escogemos Digital Output Pin.

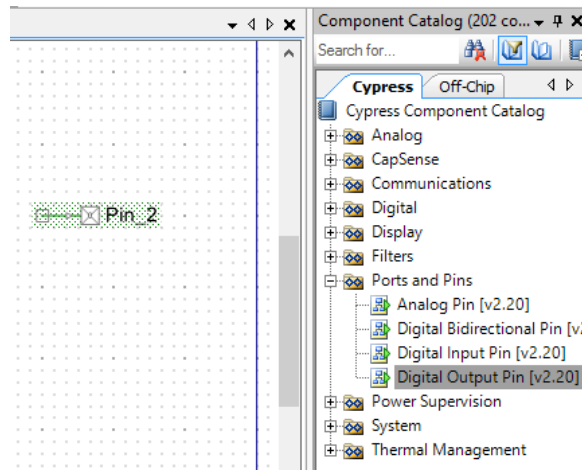


Figura 34. Ubicación de los pines digitales de salida

Doble click para configurar, se le quita la conexión HW y el nombre se le cambia por LED. Aplicar y aceptar.

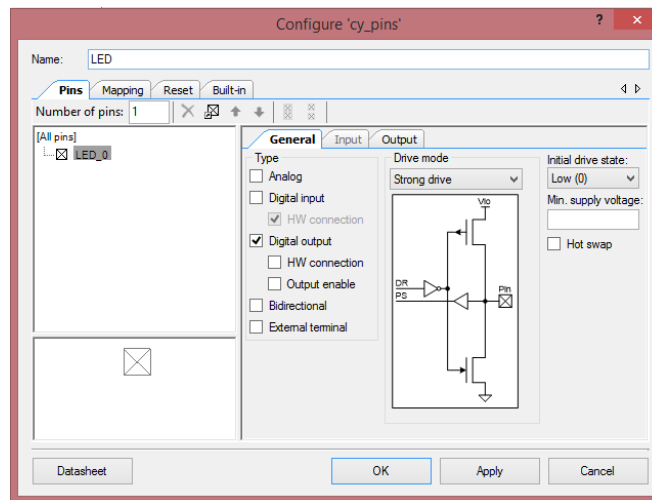


Figura 35. Configuración del pin de salida Pin_2 Como LED



Figura 36. Componentes en el WorkSpace

Luego de esto es importante guardar (1) y compilar (2) para asegurarnos de que no exista ningún error en nuestro proyecto.

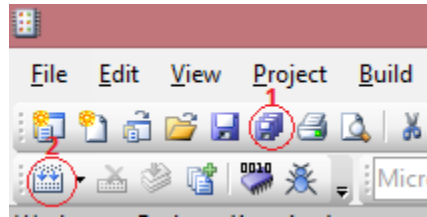


Figura 37. Guardar y Compilar

3.2.2.2.2 Asignación de Pines

Para asignar pines nos vamos al lado izquierdo de la pantalla y seleccionamos la pestaña .cydwr

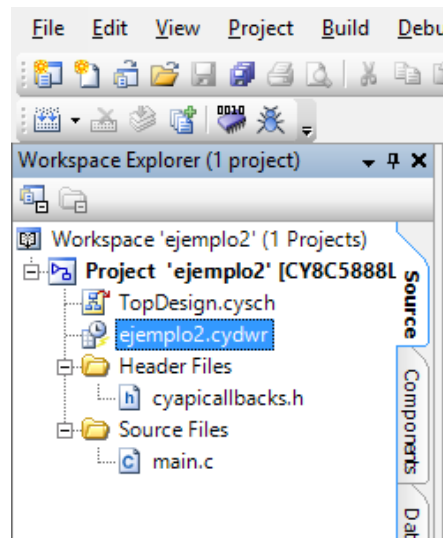


Figura 38. Archivo para asignar pines

Se abrirá una ventana con el diagrama de nuestro dispositivo. Para el caso de la PSoC 5LP los pines a utilizar son: el pin P2[1] que es el LED interno de la tarjeta y cualquier otro pin este caso escogí el pin P1[7] para el pin de entrada.

Para PSoC 3 se pueden utilizar los pines P6[3] o P6[2] que son LEDs internos de la tarjeta, y cualquier otro pin.

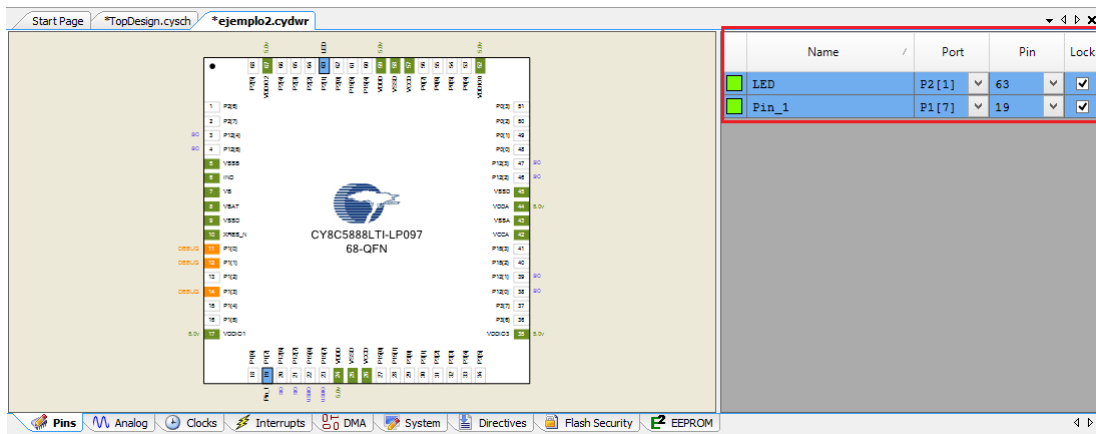


Figura 39. Pines asignados

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.2.2.3 Código del Main

En el lado izquierdo de la pantalla, nos vamos a la pestaña que dice main y damos doble click.

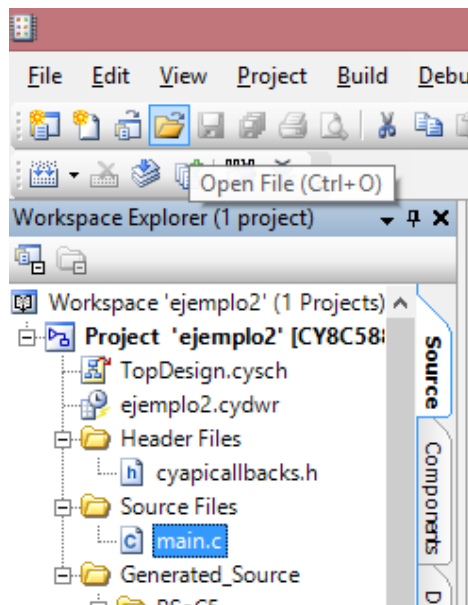


Figura 40. Archivo para ir al Main

En el main nos dirigimos a la parte donde se encuentra el for(;;) y dentro de él escribimos la instrucción `LED_Write(Pin_1_Read())`. Esta instrucción sirve para escribir en el pin LED el estado en que se encuentra el Pin_1. Es decir, si el pin_1 se encuentra el alto el LED estará prendido, y si está en bajo el Led estará apagado.

```

12 #include <project.h>
13
14 int main()
15 {
16     CyGlobalIntEnable; /* Enable global interrupts. */
17
18     /* Place your initialization/startup code here (e.g. MyInst_Start()) */
19
20     for(;;)
21     {
22         LED_Write(Pin_1_Read());
23     }
24 }
25
26 /* [] END OF FILE */
27

```

Figura 41. Código del Main para los pines de entrada

Guardamos (1), compilamos (2), generamos aplicación (3) y programamos (4).



Figura 42. Guardar, Compilar, Generar Aplicación y Programar

3.2.3 Manejo de interrupciones

Las interrupciones son un método del que disponen los dispositivos e incluso los procesos para hacer notar a la CPU la aparición de alguna circunstancia que requiera su intervención. De este modo, los dispositivos pueden provocar que la CPU deje por el momento la tarea que estaba realizando y atienda la interrupción. Una vez atendida, seguirá con su labor anterior.

3.2.3.1 Objetivos

- Conocer la funcionalidad de las interrupciones de PSoC Creator.
- Utilizar la interfaz de PSoC Creator para habilitar el LED de la tarjeta por medio de una interrupción.

3.2.3.2 Parámetros del componente

Arrastre el bloque de interrupción y haga doble click sobre el para que se abra la ventana de configuración.

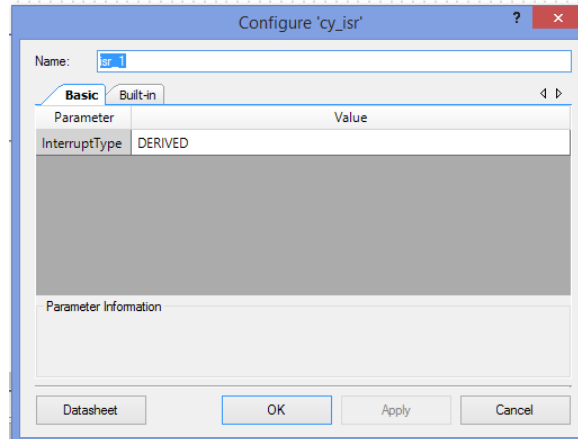


Figura 43. Parámetros de una interrupción

InterruptType: Este parámetro configura el tipo de forma de la onda que el componente procesará para disparar la interrupción. Hay tres posibles valores para este parámetro:

- **RISING_EDGE:** ha activado el flanco ascendente de la señal. Si se selecciona esta opción, un flanco ascendente en la entrada "int_signal" se convierte en un pulso de tiempo "bus_clk" y se envía al controlador de interrupciones.
- **LEVEL:** Selecciona la fuente conectada a la interrupción como una conexión sensible al nivel. Si se selecciona esta opción, la entrada "int_signal" se pasa directamente al controlador de interrupciones.
- **DERIVED:** Esta es la configuración predeterminada. Se inspecciona el conductor del "int_signal" y deriva el tipo de interrupción en base a lo que está conectado. Para la mayoría de los bloques de función fija, el tipo de interrupción es LEVEL. Para las fuentes de señal UDB, el tipo de interrupción es RISING_EDGE.

3.2.3.3 Guía paso a paso

Basados en lo dicho anteriormente, se realizará un proyecto utilizando una interrupción y activando el led de la tarjeta.

3.2.3.3.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Ir a Ports and Pins, escoger un pin de entrada digital y arrastrarlo al WorkSpace

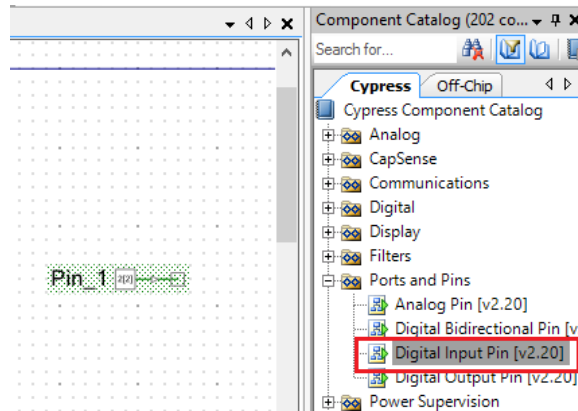


Figura 44. Ubicación de los pines digitales de entrada

Hacemos doble click sobre el elemento y se abre la ventana de configuración. En esta ventana vamos a la opción, Drive mode y escogemos la opción de Resistive pull up. En esta ventana también puede cambiar el nombre del pin si usted lo desea.

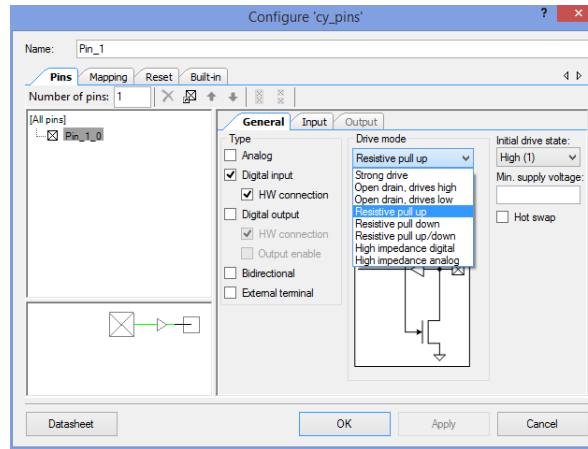


Figura 45. Configuración del Pin_1 como Pull Up

El siguiente paso es dirigirnos a la opción System, escoger Interrupt y arrastrar la interrupción al WorkSpace.

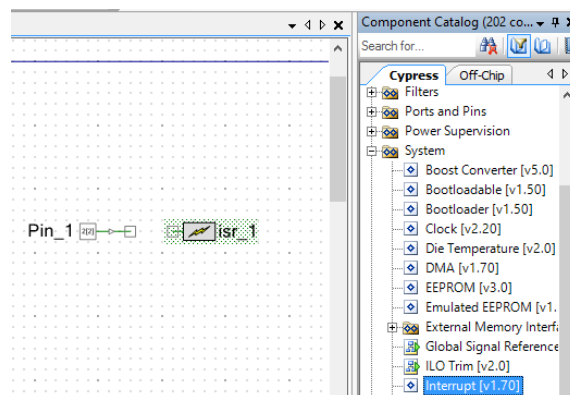


Figura 46. Ubicación de las interrupciones

Para terminar con la configuración del hardware, vamos a Ports and pins, y escogemos Digital Output pin, arrastramos.

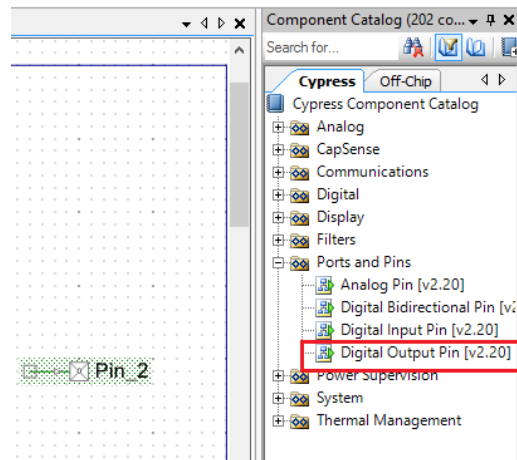


Figura 47. Ubicación de los pines digitales de salida

Hacemos doble click sobre el elemento para configurar. En este espacio, se cambiará el nombre del pin por "LED", se deshabilitará a opción "HW connection" ya que se utilizará el LED que trae la tarjeta y no tendrá ninguna conexión externa. En Drive mode se escoge Pull up. Le damos aplicar y aceptar.

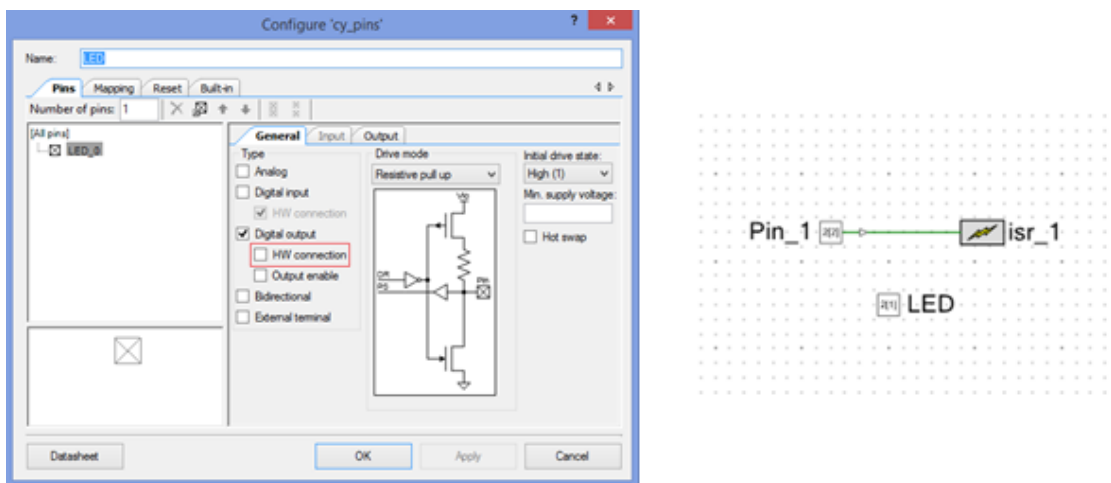


Figura 48. Configuración del LED y esquema de componentes

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.3.3.2 Código del Main

En el lado izquierdo de la pantalla, nos vamos a la pestaña que dice main y damos doble click.

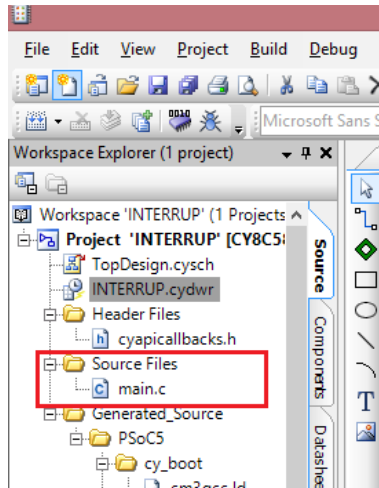


Figura 49. Archivo para ir al Main

Lo primero que se debe hacer es crear la función CY_ISR, para asignarle la tarea que va a realizar la PSoC al momento que ocurra la interrupción. Usted puede escoger cualquier nombre para la función, en este caso yo escogí el nombre “INTERRUP”. Lo que hará esta función es que, al momento de ocurrir una interrupción, el LED se pondrá en ‘1’ si su estado anterior era ‘0’, o viceversa. Esto se logra con la instrucción LED_Write(~ LED_Read()).

```
#include <project.h>

CY_ISR(INTERRUP) //Establecer la interrupción y la tarea a ejecutar
{
    LED_Write(~ LED_Read());
}
```

Figura 50. Código del Main para la función CY_ISR

En el main, se iniciará la interrupción con la instrucción isr_1_StartEx(nombre de la función CY). Es importante que antes de la expresión StartEx vaya el nombre que usted le asignó al bloque de interrupción.

```
int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    isr_1_StartEx(INTERRUP);

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        /* Place your application code here. */
    }
}
```

Figura 51. Código para iniciar la interrupción.

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.3.3 Asignación de pines

Para asignar pines nos vamos al lado izquierdo de la pantalla y seleccionamos la pestaña .cydwr

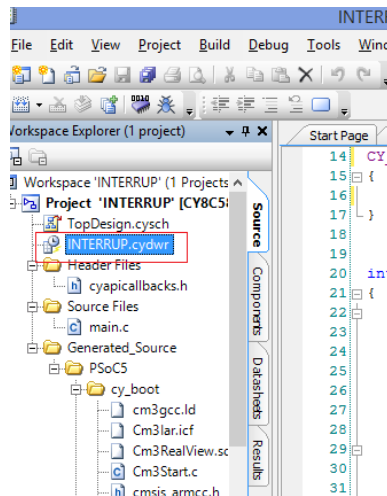


Figura 52. Archivo para asignar pines

Se abrirá una ventana con el diagrama de nuestro dispositivo. Para el caso de la PSoC 5LP los pines a utilizar son: para el Pin_1 que es la entrada digital se utilizará el pin P2[2], que es el pulsador que trae la tarjeta, y para el LED se utilizará el pin P2[1] que es el LED de la tarjeta.

Para la PSoC 3 los pines serían: P6[1] o P15[5], que son pulsadores de la tarjeta. Y para el LED se pueden utilizar los pines P6[3] y P6[2]

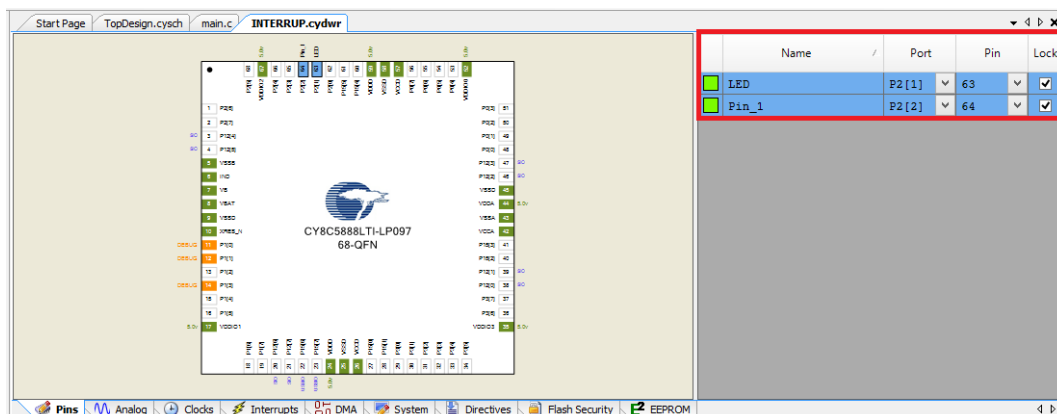


Figura 53. Pines Asignados

Guardamos (1), compilamos (2), generamos aplicación (3) y programamos (4).



Figura 54. Guardar, Compilar, Generar Aplicación y Programar

3.2.4 Contador

El componente contador proporciona un método para contar eventos. Para PSoC 3 y PSoC 5LP puede ser implementado utilizando bloques FF o UDB. Una aplicación UDB normalmente tiene más características que una implementación FF.

3.2.4.1 Objetivos

- Conocer la funcionalidad del contador de PSoC Creator.
- Utilizar la interfaz de PSoC Creator para realizar el conteo de cierto número de interrupciones.

3.2.4.2 Parámetros del componente

Arrastre un contador en su diseño y haga doble clic en él para abrir el cuadro de diálogo Configurar.

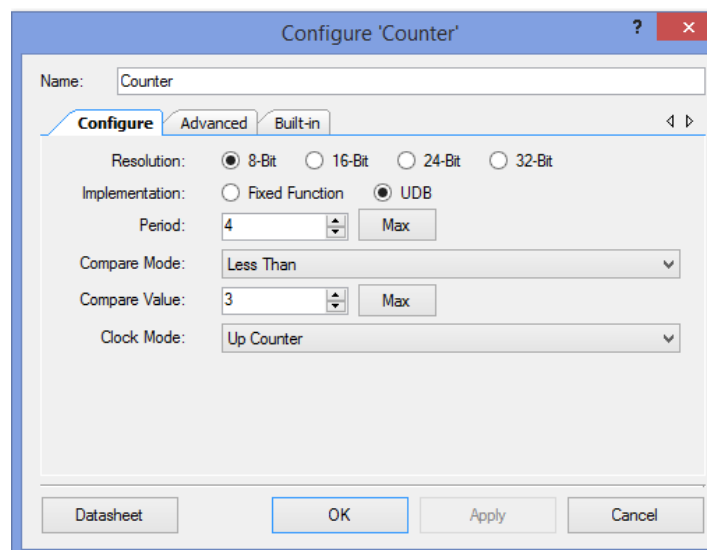


Figura 55. Parámetros de un Contador

Resolución: define la resolución de bits del contador. Este valor puede ser ajustado a 8, 16, 24, o 32 bits para los valores máximos de recuento de 255, 65 535, 16777215, y 4294967295, respectivamente.

Implementación: Permite elegir entre un bloque de función fija y un UDB. Si se selecciona la función fija, las funciones UDB están desactivados.

Periodo: Define el período máximo o valor de cuenta (o punto de vuelco) para el contador. Este parámetro define el valor inicial cargado en el registro de período, que el software puede cambiar en cualquier momento con la API Counter_WritePeriod ().

Compare Mode - Comparación de Modo (opción de software): Se configura el funcionamiento de la señal de salida. Esta señal es el estado de una comparación entre el parámetro de valor de comparación y el valor actual del contador. Este parámetro define la configuración inicial. Puede cambiarlo en cualquier momento para volver a configurar la operación de comparación del componente de contador.

Compare Mode se puede ajustar a cualquiera de los valores siguientes:

- **Less than:** El valor del contador es menor que el valor de comparación.
- **Less Than Or Equal:** El valor del contador es menor que o igual que el valor de comparación.
- **Equal To:** El valor del contador es igual al valor de comparación.
- **Greater Than:** El valor del contador es mayor que el valor de comparación.
- **Greater Than Or Equal:** El valor del contador es mayor que o igual al comparar valor.
- **Software Controlled:** El modo de comparación se puede establecer durante el tiempo de ejecución con él.

Compare Value (opción de software)

Define el valor inicial cargado en el registro de comparar el contador. Este valor se utiliza en conjunto con el parámetro de modo de comparación para definir el funcionamiento de la salida comparar.

Este valor puede ser cualquier número entero sin signo de 0 a $(2 \wedge \text{Resolución} - 1)$, pero debe ser menor o igual al valor del período.

Clock Mode: Configura cómo el contador contará. Este modo se puede ajustar a cualquiera de los valores siguientes:

- **Count Input and Direction:** Counter es un contador bidireccional. Se cuenta hacia arriba mientras que la entrada up_ndown es alta en cada flanco de subida del reloj de entrada y una cuenta atrás mientras up_ndown es baja en cada flanco de subida del reloj de entrada.
- **Clock With UpCnt & DwnCnt:** Counter es un contador bidireccional. Se incrementa el contador para cada flanco ascendente en la entrada upCnt y disminuye el contador para cada el flanco ascendente de la entrada dwnCnt, con respecto a la entrada de reloj.
- **Up Counter** - Counter es un contador ascendente solamente. Se incrementa en el flanco ascendente de la entrada de cuenta con respecto a la señal de reloj, mientras que el contador es habilitado.

- **Down Counter:** Contador es más que un contador regresivo. Se incrementa en el flanco ascendente de la entrada de cuenta con respecto a la señal de reloj, mientras que el contador es habilitado.

3.2.4.3 Guía paso a paso

Basados en lo dicho anteriormente, se realizará un proyecto en el cual se contarán dos interrupciones y se prenderá el LED de la tarjeta.

3.2.4.3.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos al catálogo de componentes en la parte derecha de la pantalla, Digital, Functions, y arrastramos un contador al WorkSpace.

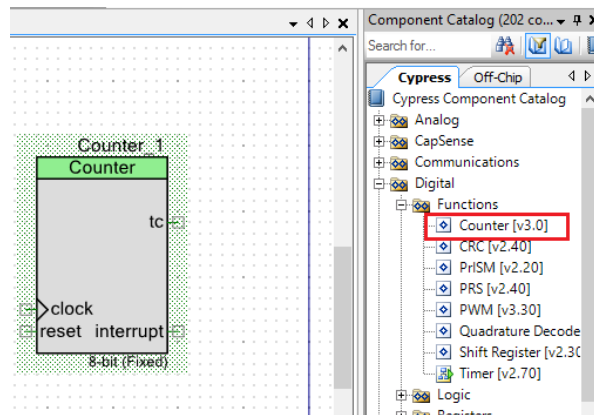


Figura 56. Ubicación de los contadores en el catálogo de Cypress

Hacemos doble click sobre el bloque para configurarlo. El nombre del contador se puede cambiar para comodidad de programación. El periodo se cambia a 2, porque es el número máximo que va a contar el contador, y el valor a comparar será 1.

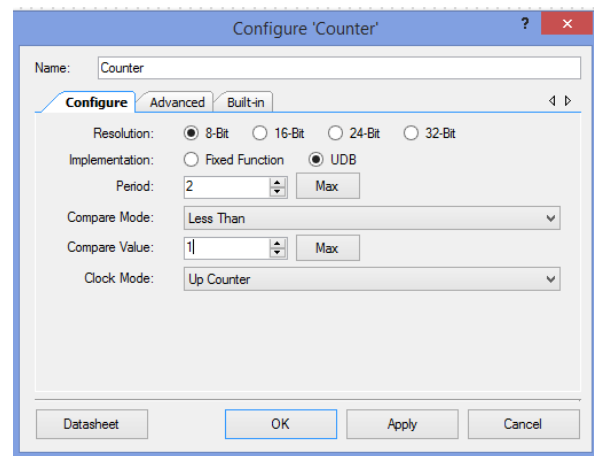


Figura 57. Configuración del contador

El siguiente paso es arrastrar un reloj para el contador, para esto nos vamos a la pestaña System, y escogemos Clock.

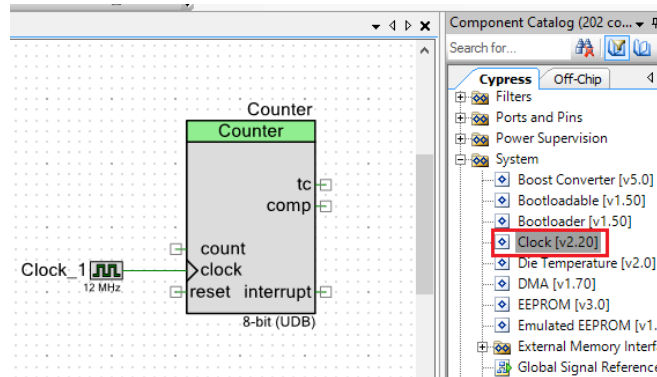


Figura 58. Ubicación de los relojes en el catálogo de Cypress

Lo siguientes colocar el pin de restet del contado a tierra, para eso nos vamos a Digital, Logic y escogemos Logic Low '0'.

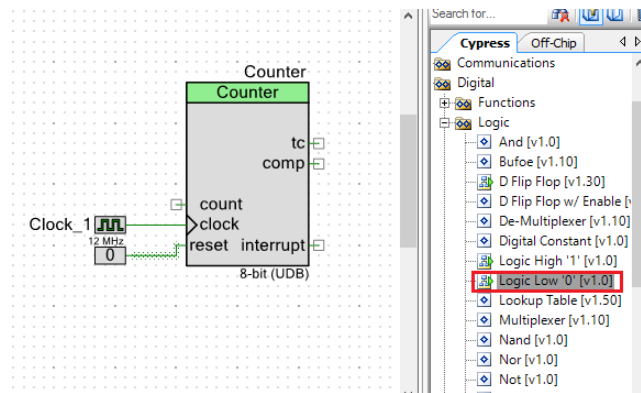


Figura 59. Ubicación y conexión de un '0' lógico

Ahora para iniciar el conteo se necesita una entrada, vamos a Ports and pins, y escogemos una entra digital. Y lo conectamos en el pin count del contador.

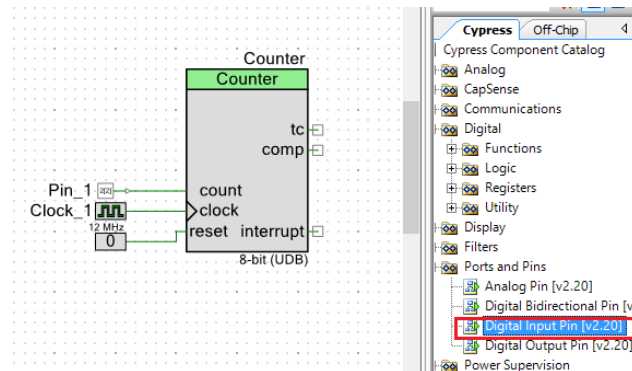


Figura 60. Ubicación y conexión del pin de entrada digital

Hacemos doble click sobre el pin de entrada y lo configuramos como Resistive pull up.

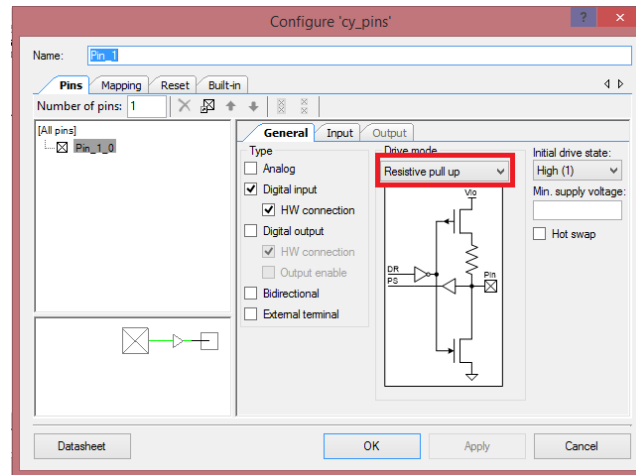


Figura 61. Configuración del Pin_1 como pull up

El siguiente paso es añadir una interrupción y conectarla al pin tc del contador.

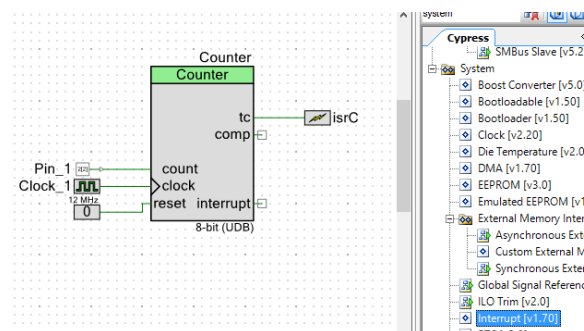


Figura 62. Ubicación y conexión de una interrupción

En la configuración de la interrupción, cambiamos el tipo de interrupción por RISING_EDGE para que cuente cada vez que existan flancos de subida.

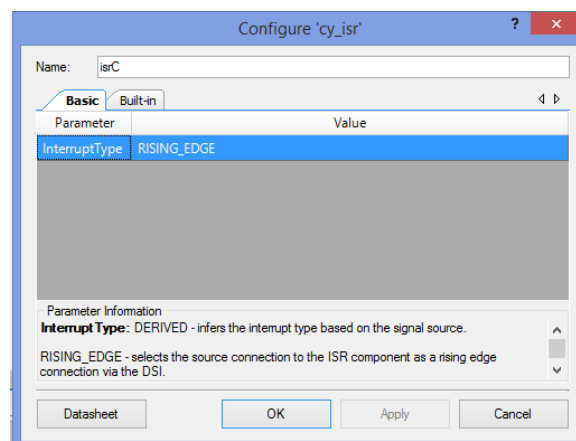


Figura 63. Configuración de la interrupción

Por último, arrastramos un pin digital de salida, y lo configuramos como un LED en modo Strong drive.

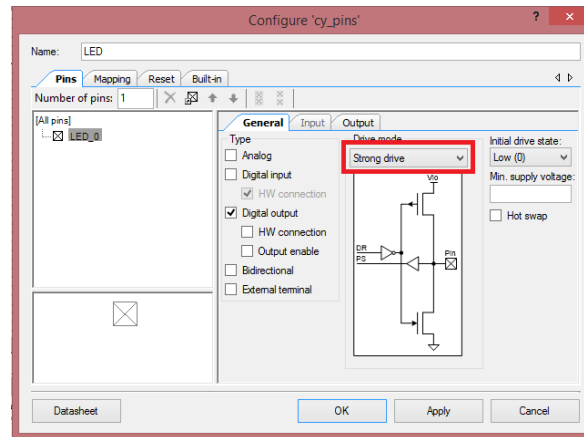


Figura 64. Configuración del pin de salida como LED en modo Strong drive

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.4.3.2 Código del Main

En el lado izquierdo de la pantalla, nos vamos a la pestaña que dice main y damos doble click.

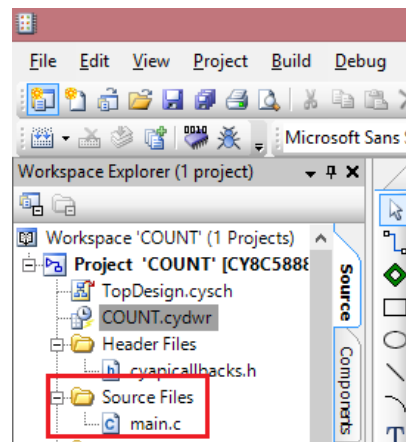


Figura 65. Archivo para ir al Main

Igual que en el proyecto anterior lo primero que se debe hacer es crear la función CY_ISR para la interrupción. Dentro de esta función debe ir la tarea que va a realizar el LED al momento de que el conteo llegue al número deseado, parar el contador y reiniciar el mismo. En este caso, el nombre de la función CY_ISR es isrC1. Para detener el contador se utiliza la instrucción `Counter_Stop()`, para desbordar el contador `Counter_WritePeriod(ValorPeriodo)`, y para volver a iniciar el contador se utiliza `Counter_Start()`.

```

#include <project.h>

CY_ISR(isrC1)
{
    LED_Write( ~ LED_Read() );

    Counter_Stop();

    Counter_WritePeriod(2);

    Counter_Start();
}

```

Figura 66. Código para la Función CY_ISR

En el Main, debemos iniciar, el contador, el reloj del contador y la interrupción.

```

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    Clock_1_Start();
    Counter_Start();
    isrC_StartEx (isrC1);
}

```

Figura 67. Código del main para iniciar componentes

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.4.3.3 Asignación de pines

Para asignar pines nos vamos al lado izquierdo de la pantalla, en el WorkspaceExplorer y seleccionamos la pestaña .cydwr

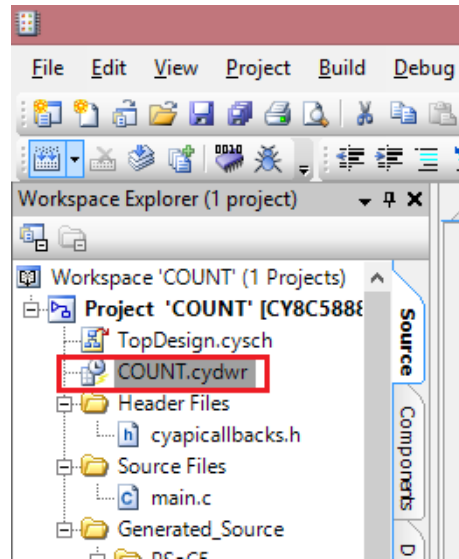


Figura 68. Archivo para asignar pines

Para el caso de la PSoC 5LP los pines a utilizar son: para el Pin_1 que es la entrada digital se utilizará el pin P2[2], que es el pulsador que trae la tarjeta, y para el LED se utilizará el pin P2[1] que es el LED de la tarjeta. Para la PSoC 3 los pines serían: P6[1] o P15[5], que son pulsadores de la tarjeta. Y para el LED se pueden utilizar los pines P6[3] y P6[2]

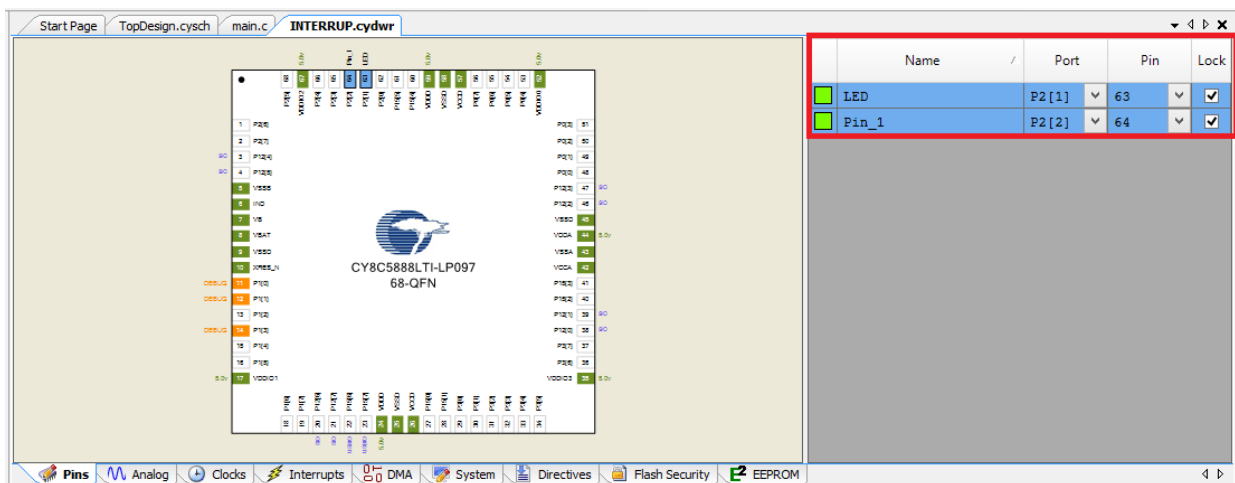


Figura 69. Pines asignados

Guardamos (1), compilamos (2), generamos aplicación (3) y programamos (4).



Figura 70. Guardar, Compilar, Generar Aplicación y Programar

3.2.5 Timer

El Timer proporciona un método para medir los intervalos. Se puede implementar una función básica temporizador y ofrece características avanzadas tales como la captura con el contador de captura e interrupción/generación de DMA. Para PSoC 3 y PSoC 5LP puede ser implementado utilizando bloques FF o UDB. Una aplicación UDB normalmente tiene más características que una implementación FF.

3.2.5.1 Objetivos

- Conocer el funcionamiento del bloque Timer de PSoC Creator.
- Utilizar la interfaz de PSoC Creator para que por que el Timer se active por medio de una interrupción.

3.2.5.2 Parámetros del componente

Arrastre un timer en su diseño y haga doble clic en él para abrir el cuadro de diálogo Configurar.

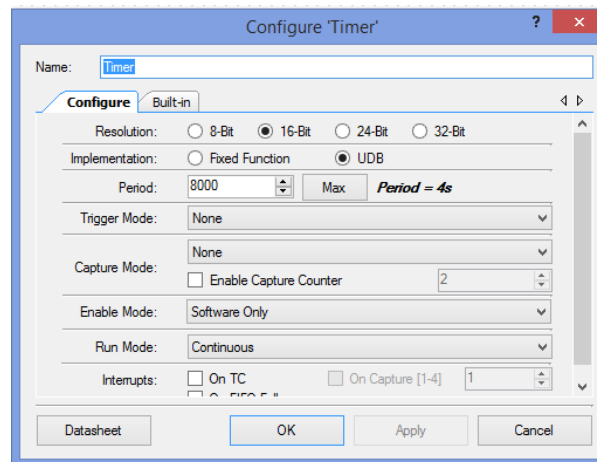


Figura 71. Parámetros del Timer

Resolución: define la resolución de bits del contador. Este valor puede ser ajustado a 8, 16, 24, o 32 bits para los valores máximos de recuento de 255, 65 535, 16777215, y 4294967295, respectivamente.

Implementación: permite elegir entre un bloque de función fija y un UDB. Si se selecciona la función fija, las funciones UDB están desactivados.

Periodo: define el período máximo o valor de cuenta (o punto de vuelco). Este parámetro define el valor inicial cargado en el registro de período, que el software puede cambiar en cualquier momento con la API `Timer_WritePeriod()`.

Trigger Mode (Opción de software): configura la aplicación de la entrada de disparo. Este parámetro sólo se activa cuando la implementación se establece en UDB.

Capture Mode (Opción de software):

Capture Mode: configura cuando una captura se lleva a cabo. La entrada de captura se muestrea en el flanco ascendente de la entrada de reloj.

Enable Capture Counter: se utiliza para definir el número de eventos de captura ocurren antes de que el contador es realmente capturado.

Capture Count: establece el número inicial de eventos de captura que se producen antes de que el contador está en realidad capturado. Se puede ajustar a un valor de 2 a 127. El valor de recuento de captura puede ser modificado en tiempo de ejecución llamando a la función API `Timer_SetCaptureCount ()`.

Enable Mode: Configura la aplicación de habilitación del temporizador. La entrada de habilitación es muestreada en el flanco ascendente de la entrada de reloj.

Run Mode: se utiliza para configurar el componente temporizador para funcionar de forma continua o en un modo de una sola vez.

Interrupt: se utilizan para configurar las fuentes de interrupción iniciales. Una interrupción se genera cuando uno o más de los siguientes eventos seleccionados ocurrir.

3.2.5.3 Interfaz de programación

De forma predeterminada, PSoC Creator asigna el nombre de "timer_1" al primer timer en un diseño dado. Puede cambiar el nombre a cualquier valor único que sigue las reglas sintácticas de los identificadores.

3.2.5.4 Guía paso a paso

Esta práctica pretende mantener encendido el LED de la tarjeta por 4 segundos después que se presione un pulsador. A continuación, podrá encontrar una guía donde se describe paso por paso el procedimiento para realizar dicha práctica.

3.2.5.4.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Ir a los componentes del catálogo, digital, functions, y arrastramos un Timer.

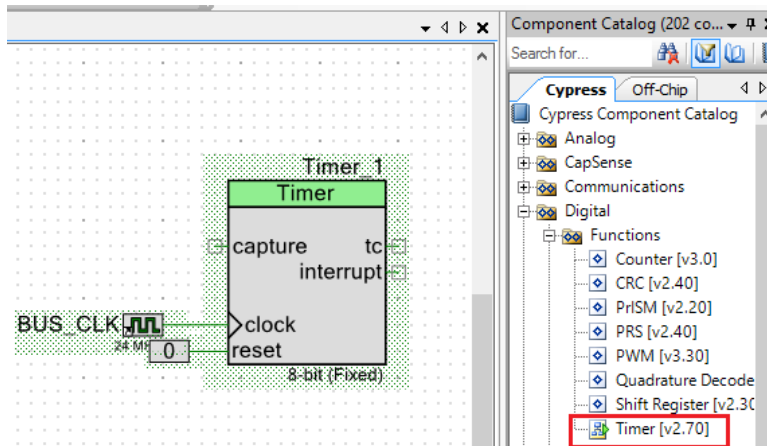


Figura 72. Ubicación del Timer en el catálogo de Cypress

Antes de configurar el Timer, daremos doble click sobre el Clock del timer (BUS_CLK), y lo configuraremos, el nombre se le cambia por timer_clock. Como se quiere un tiempo de 4 segundos el valor del reloj se cambiará a 2kHz. Para ello entramos a la configuración del reloj, en clock type escogemos new, en las especificaciones colocamos 2kHz. Aplicar y aceptar.

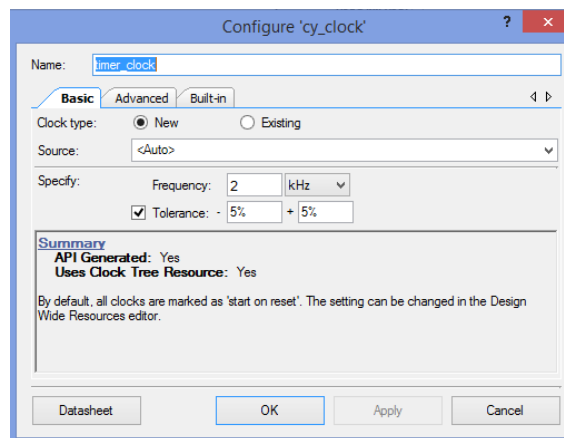


Figura 73. Configuración de un nuevo reloj

Ahora damos doble click en el Timer para configurar. Seleccionamos una resolución de 16 bits y la opción UDB para tener acceso a mas parámetros del timer si lo deseamos.

Como el tiempo deseado es 4 segundos, el periodo se calcula de la siguiente manera.

$$Periodo = \frac{Frecuencia\ de\ reloj}{inverso\ del\ tiempo\ deseado} = \frac{2000Hz}{1/4s} = 8000$$

Este valor de 8000 será el periodo. Los demás parámetros del timer se dejan como vienen por defecto. Aplicar y aceptar.

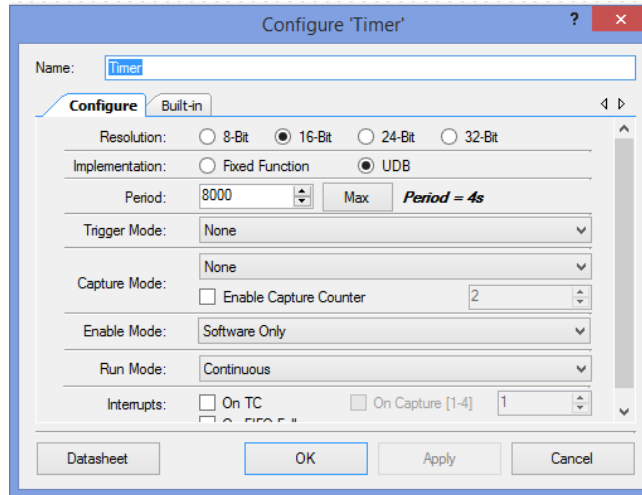


Figura 74. Configuración del timer

Luego vamos a system, arrastramos una interrupción y se une al pin tc del timer. Y se cambia su nombre por isrT.

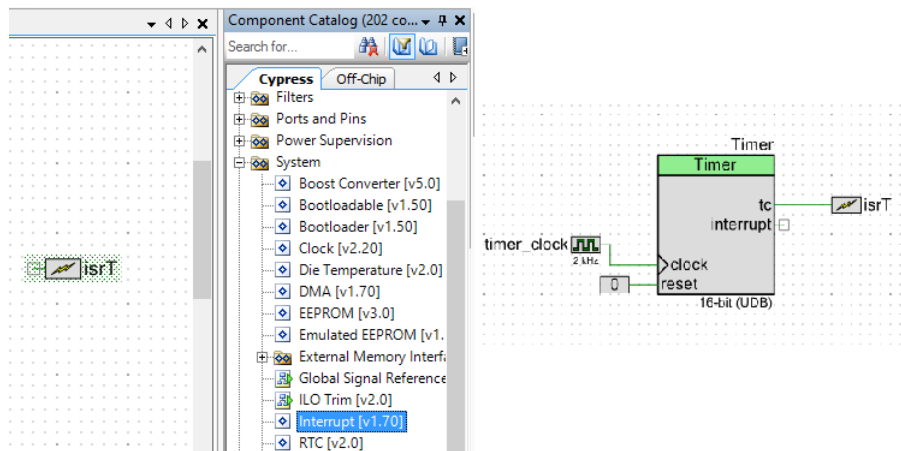


Figura 75. Ubicación y conexión de la interrupción

Vamos a Ports and Pins, arrastramos una entrada digital, y también necesitamos otra interrupción. Se une el pin de entrada con la interrupción, la interrupción llevará el nombre de isrI.

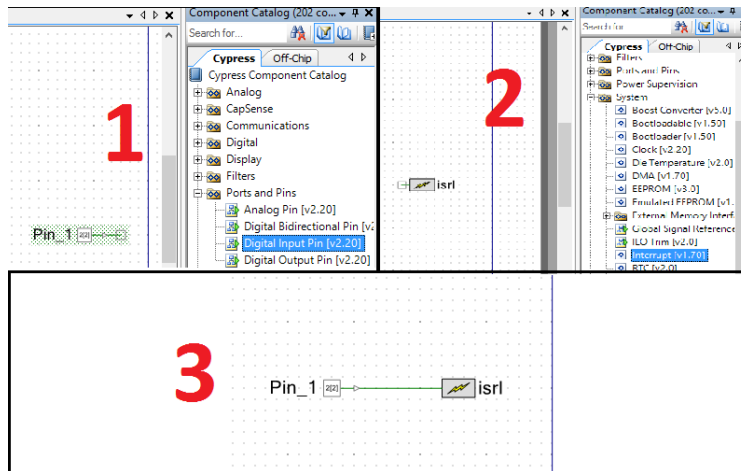


Figura 76. Ubicación y conexión del pin de entrada y la interrupción

El pin_1 se hace doble click para configurarlo como Resistive pull up.

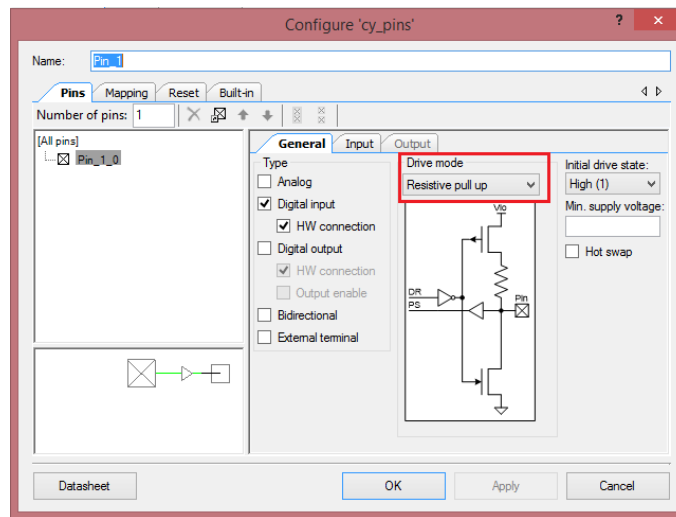


Figura 77. Configuración del pin de entrada como Pull up

Por último, vamos a Ports and Pins, y escogemos un pin da salida digital, se le cambia el nombre por LED, se configura como pull down y se le quita la conexión HW. Aplicar y aceptar.

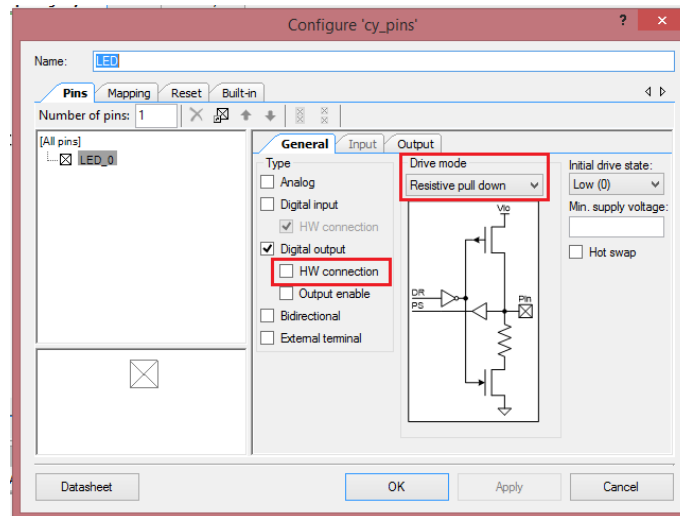


Figura 78. Configuración del pin de salida como LED

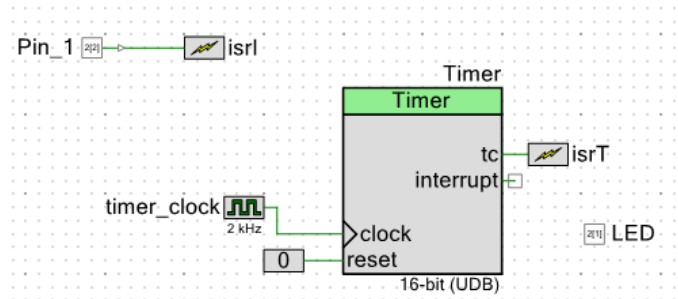


Figura 79. Componentes en el Workspace

Luego de esto es importante guardar y compilar.

3.2.5.4.2 Código del Main.

En el lado izquierdo de la pantalla, nos vamos a la pestaña que dice main y damos doble click.

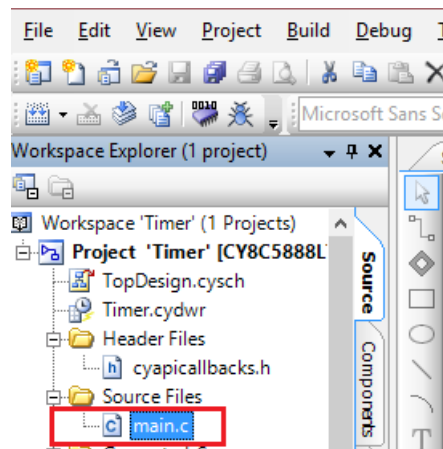


Figura 80. Archivo para ir al Main

El primer paso es configurar las dos interrupciones que tenemos, una es para el pulso de entrada y la otra es la que me enlazará el timer con el pulsador.

Primero se declara la función CY_ISR para el pin de entrada. En esta función se encenderá el LED y se activará el timer. La segunda función me apagará el LED, detendrá el timer y lo reiniciará. La función CY_ISR de la interrupción llevará el nombre del InterruptI y la del timer InterruptT.

```
#include <project.h>

CY_ISR(InterruptI)
] {
    LED_Write(1);
    Timer_Start();
- }

CY_ISR(InterruptT)
] {
    LED_Write(0);
    Timer_Stop();

    Timer_WritePeriod(8000);
- }
```

Figura 81. Código para las funciones de las interrupciones.

En el Main, simplemente se iniciarán: el reloj del timer, y las dos funciones de las interrupciones.

```
int main()
] {
]   CyGlobalIntEnable; /* Enable global interrupts. */
   timer_clock_Start();
   isrI_StartEx(InterruptI);
   isrT_StartEx(InterruptT);

]   /* Place your initialization/startup code here (e.g. MyInst_Start()) */

   for(;;)
]   {
]     /* Place your application code here. */
-   }
- }

] /* [] END OF FILE */
```

Figura 82. Código del main para iniciar los componentes

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.5.4.3 Asignación de pines

Para asignar pines nos vamos al lado izquierdo de la pantalla y seleccionamos

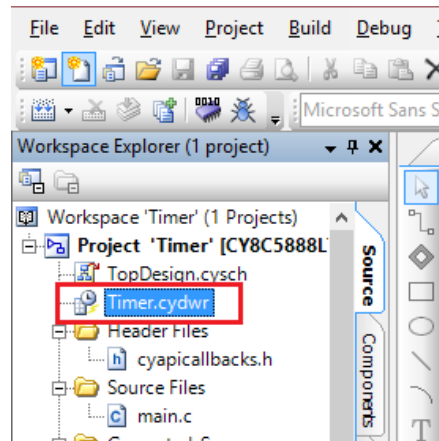


Figura 83. Archivo para asignar pines

Para la PSoC 5LP: el Pin_1 que es la entrada digital se utilizará el pin P2[2], que es el pulsador que trae la tarjeta, y para el LED se utilizará el pin P2[1] que es el LED de la tarjeta. Para la PSoC 3 los pines serían: P6[1] o P15[5], que son pulsadores de la tarjeta. Y para el LED se pueden utilizar los pines P6[3] y P6[2].

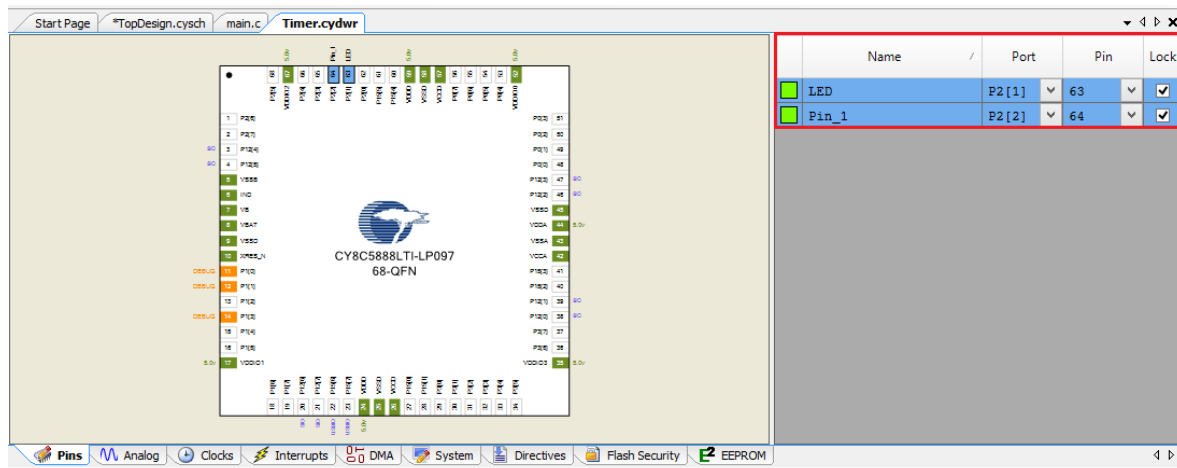


Figura 84. Pines utilizados

Después de esto, guardamos, compilamos, generamos aplicación y programamos.

3.2.6 Contador y Timer

Esta práctica trata de contar cierto número de interrupciones (4) y activar el LED de la tarjeta por cierto tiempo (2 s).

3.2.6.1 Objetivos

- Combinar la funcionalidad de las interrupciones, el contador y el timer, utilizando la interfaz de PSoC Creator para una aplicación.

3.2.6.2 Guía paso a paso

3.2.6.2.1 Configuración del Hardware

Primero vamos al catálogo, Digital, Functions y escogemos un contador.

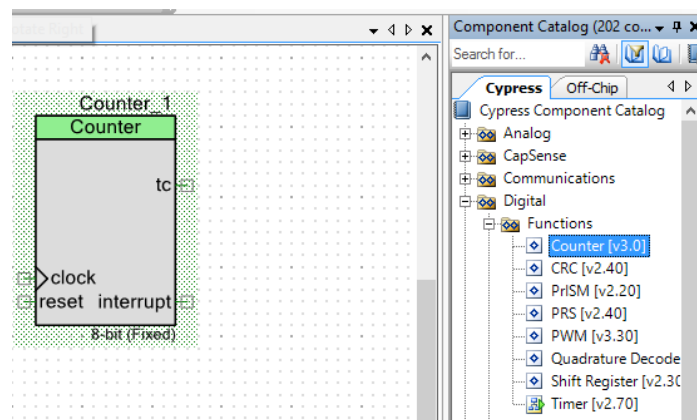


Figura 85. Ubicación del Timer

Doble click en contador para configurar. Cambiamos el nombre por Counter, como queremos que el número de interrupciones sea 4, en el periodo colocamos 4 y valor a comparar 3. Como lo muestra la imagen. Aplicar y aceptar.

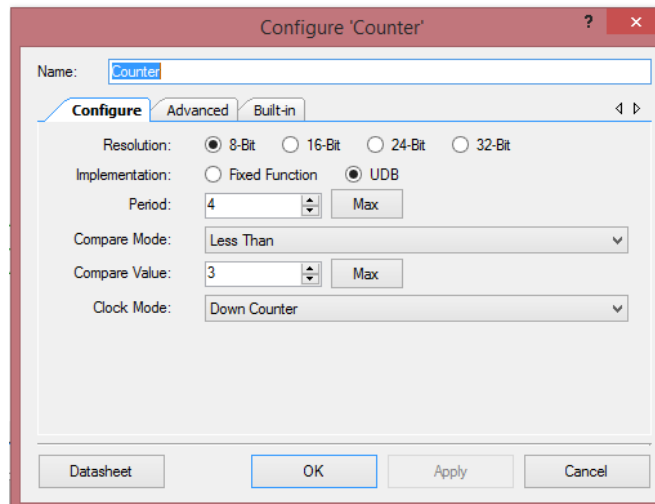


Figura 86. Configuración del Contador

Para seguir configurando el contador, necesitamos un reloj, para eso vamos a system, arrastramos un clock y lo unimos al pin clock del contador.

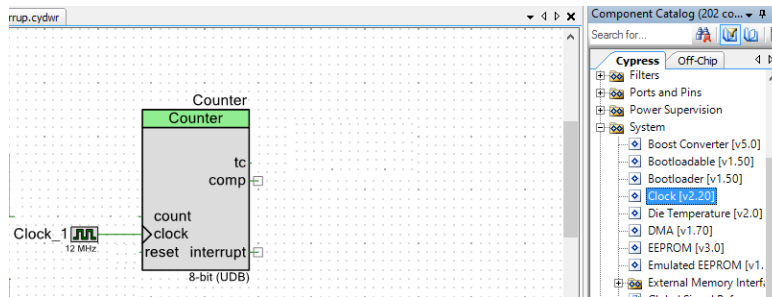


Figura 87. Ubicación y conexión del Clock

También se necesita un '0' lógico, para eso voy a Digital, Logic y arrastro el cero lógico y lo uno al reset del contador.

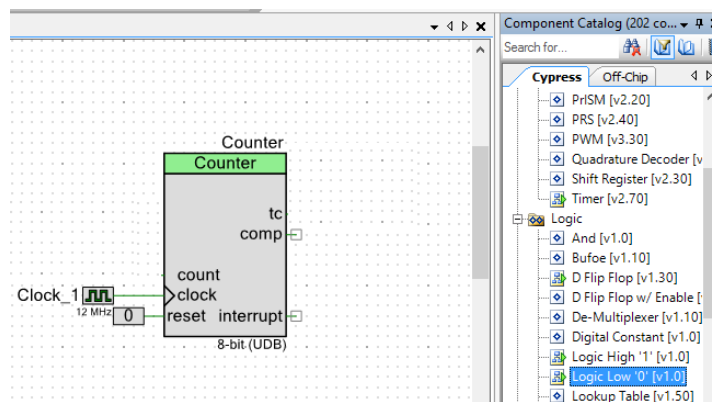


Figura 88. Ubicación y conexión del '0' lógico

También necesito una interrupción y un pin de entrada digital, para la interrupción me voy a system, y para la entrada digital al ports and pins. Y se conectan según la imagen.

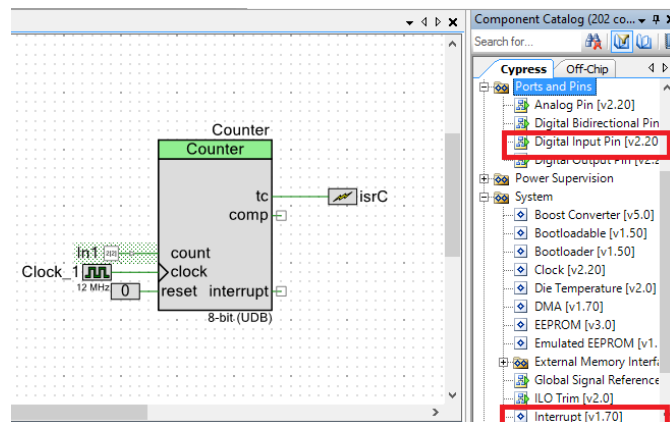


Figura 89. Ubicación y conexión del pin de entrada y de la interrupción

La interrupción tendrá el nombre de isrC y será de tipo RISING_EDGE.

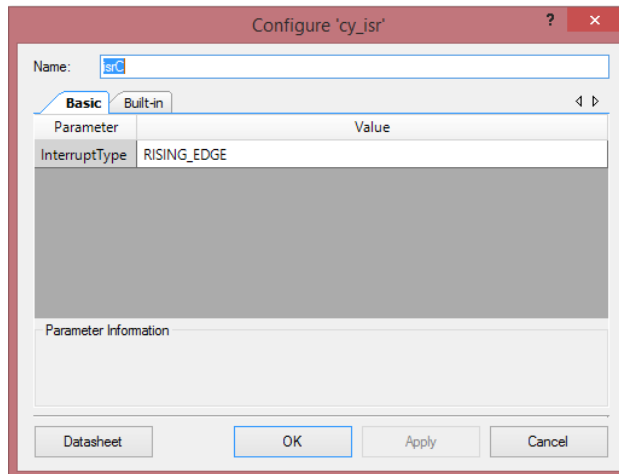


Figura 90. Configuración de la interrupción del contador isrC

El pin de entrada se llamará In1 y estará en modo resistive pull up.

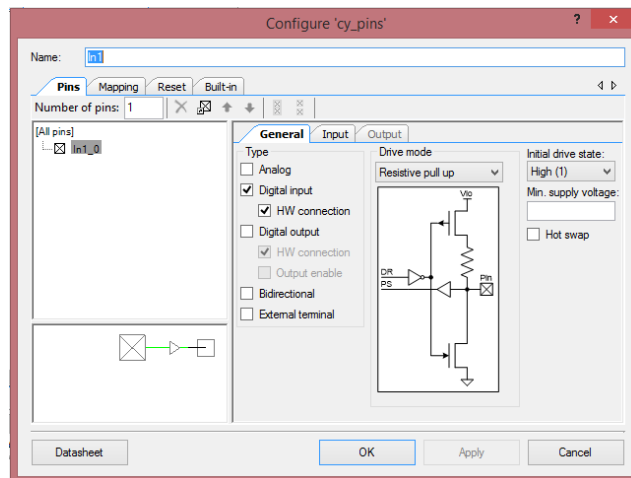


Figura 91. Configuración del pin de entrada como pull up

El siguiente paso es añadir un Timer.

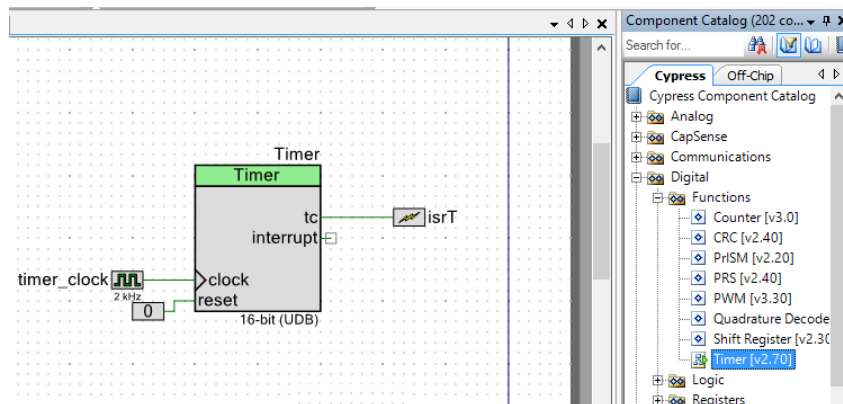


Figura 92. Ubicación y conexión del timer

El reloj del timer será de 2kHz, para configurarlo hacemos doble click sobre el reloj, en clock type damos new y le damos el valor del reloj. Aplicar y aceptar.

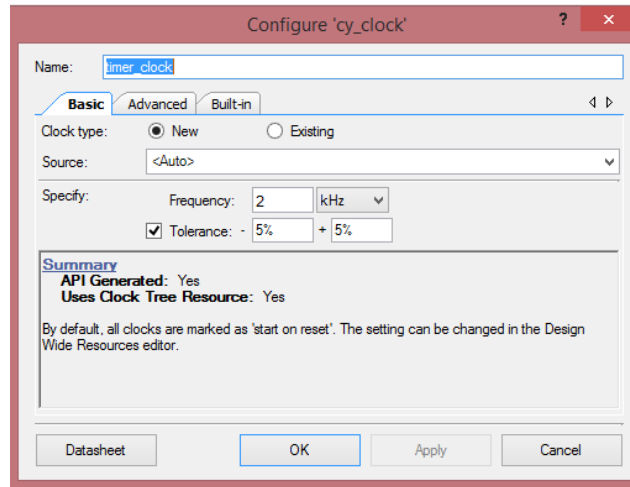


Figura 93. Configuración del reloj del timer

También necesitamos una interrupción que irá conectada al pin tc del timer. Se llamará isrT y será de tipo derive. La configuración del timer será de la siguiente manera.

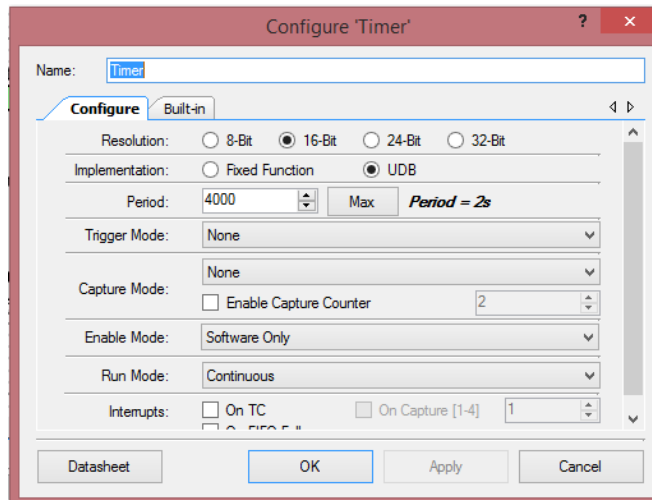


Figura 94. Configuración del timer

Por último, necesitamos un pin de salida digital que será nuestro LED. Se le deshabilita la conexión HW y será de modo strong drive.

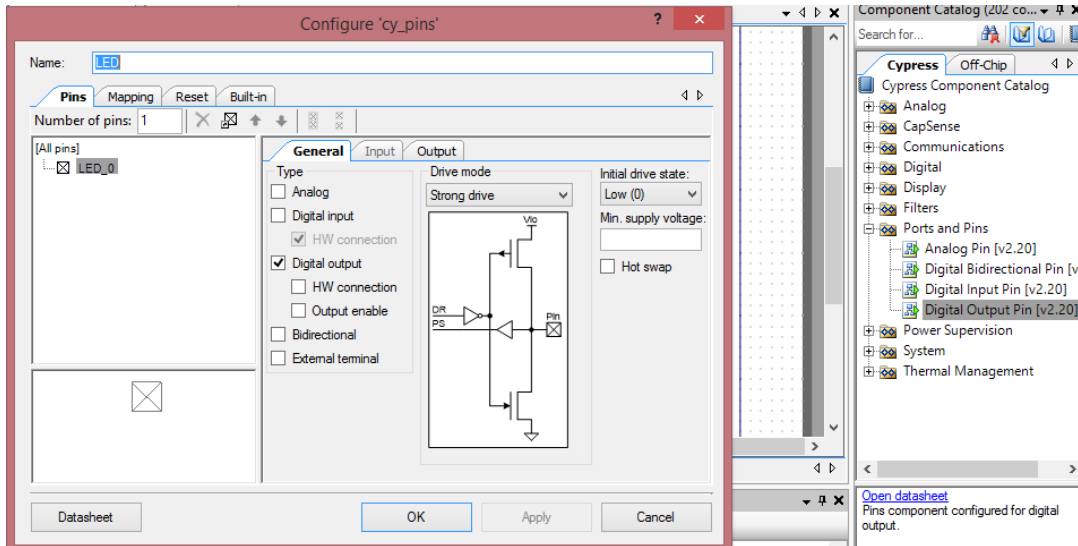


Figura 95. Ubicación y configuración del pin digital de salida

Guardamos y compilamos.

3.2.6.2.2 Código del Main

Lo primero es declarar las dos funciones para las interrupciones. En la interrupción del contador se encenderá el LED y se activará el timer. Y en la interrupción del timer, se apagará el LED, se detiene el contador y el timer, se escriben los periodos de ambos y se arranca de nuevo el contador.

```

#include <project.h>
CY_ISR(InterruptC)
{
    LED_Write(1);
    Timer_Start();
}

CY_ISR(InterruptT)
{
    LED_Write(0);
    Counter_Stop();
    Timer_Stop();

    Counter_WritePeriod(4);
    Timer_WritePeriod(4000);

    Counter_Start();
}

```

Figura 96. Código de las funciones de las interrupciones

En el main, se inician todos los componentes, como son el contador, reloj del timer, reloj del contador, y las dos funciones de las interrupciones.

```

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    Clock_1_Start();
    timer_clock_Start();

    Counter_Start();

    isrC_StartEx(InterruptC);
    isrT_StartEx(InterruptT);
}

```

Figura 97. Código del main para iniciar componentes

Guardamos y compilamos.

3.2.6.2.3 Asignación de pines

Para la PSoC 5LP: el Pin_1 que es la entrada digital se utilizará el pin P2[2], que es el pulsador que trae la tarjeta, y para el LED se utilizará el pin P2[1] que es el LED de la tarjeta. Para la PSoC 3 los pines serían: P6[1] o P15[5], que son pulsadores de la tarjeta. Y para el LED se pueden utilizar los pines P6[3] y P6[2]

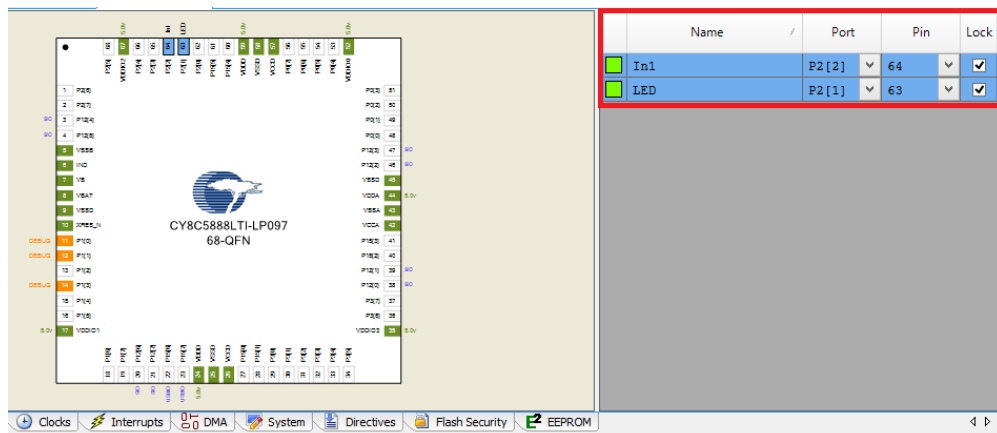


Figura 98. Pines asignados

Después de esto, guardamos, compilamos, generamos aplicación y programamos.

3.2.7 Manejo de LCD

El componente de LCD contiene un conjunto de rutinas de biblioteca que permiten un uso simple de módulos LCD uno, dos, o cuatro líneas que siguen a la interfaz de 4 bits estándar Hitachi 44780. El componente proporciona APIs para aplicar gráficos de barras horizontales y verticales, o se puede crear y mostrar sus propios caracteres.

3.2.7.1 Objetivos

- Conocer la funcionalidad del bloque LCD que trae PSoC Creator.

- Configurar el bloque LCD para imprimir un mensaje, parpadear la pantalla, como barra de progreso y con caracteres creados por el usuario.

3.2.7.2 Parámetros del componente

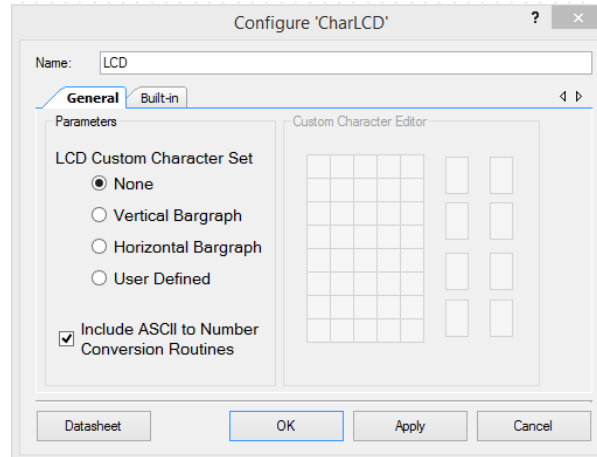


Figura 99. Parámetros de la LCD

LCD conjunto de caracteres personalizado: Este parámetro permite la selección de las siguientes opciones:

- Ninguno (predeterminado): No hacer nada con caracteres personalizados.
- Gráfico de barra vertical: Genera caracteres personalizados para manipular un gráfico de barras verticales.
- Gráfico de barra horizontal: Generar caracteres personalizados para manipular un gráfico de barras horizontal.
- Caracteres definidos por usuario: Crear caracteres para manipularlos.

3.2.7.3 Guía paso a paso

En el siguiente proyecto, se utilizarán diferentes configuraciones de la LCD, como imprimir un mensaje, dormir-despertar y una barra de progreso.

3.2.7.3.1 Configuración de Hardware.

- Abrir PSoC Creator y crear un nuevo proyecto.
- Ir a los componentes del catálogo, Display, Character LCD y arrastramos hasta el WorkSpace.

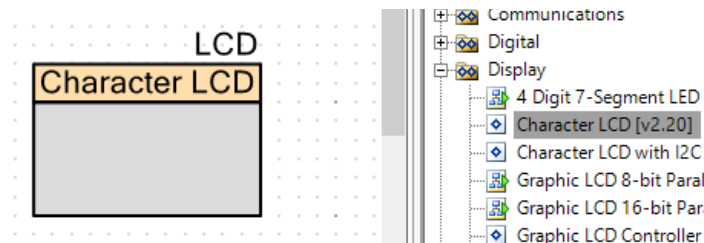


Figura 100. Ubicación de la LCD

Luego de esto es importante guardar y compilar para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.7.3.2 Asignación de pines

Antes de empezar con el código del main, primero hay que asignar los pines.

Al agregar el bloque de la LCD, Creator automáticamente asigna un puerto que es donde iría conectada la LCD. Este puerto puede modificarse si uno lo desea. Para este caso yo escogí el puerto dos (P2) para manejar la LCD.

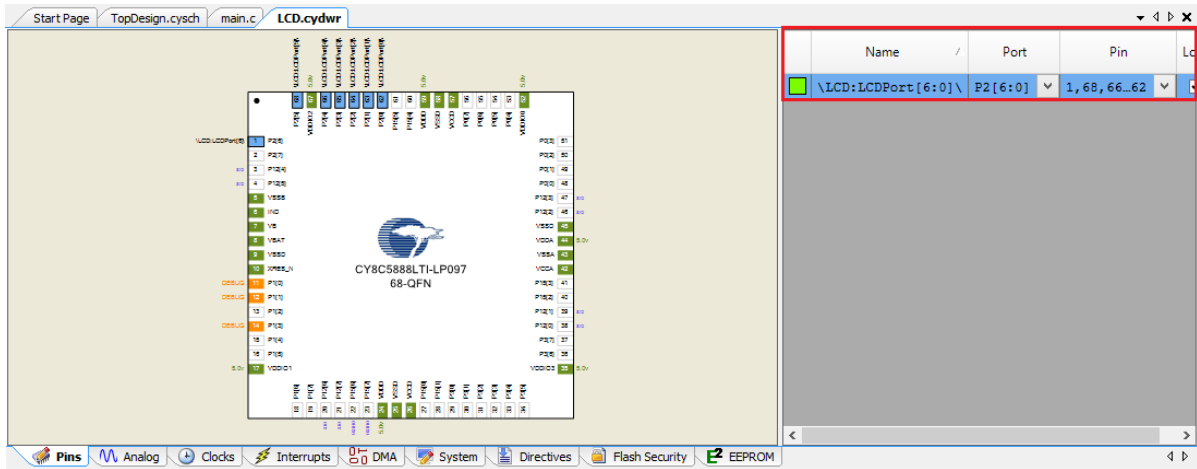


Figura 101. Pines utilizados

Los pines irían distribuidos de la siguiente manera:

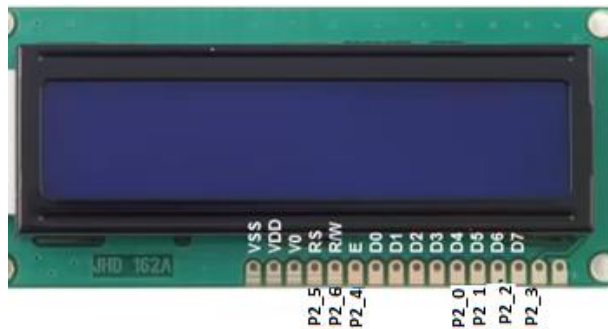


Figura 102. Distribución de pines para la LCD

Luego de esto es importante guardar y compilar.

3.2.7.3.3 Código del Main

En el main lo primero que haremos será iniciar al LCD, con la instrucción LCD_Start().

- **Imprimir un mensaje:** Para imprimir un mensaje en la LCD, lo primer es posicionar la LCD con la instrucción LCD_Position(fila,columna), y para imprimir el mensaje usar la instrucción LCD_PrintString("Mensaje")

```

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    LCD_Start();
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    //MENSAJE LCD
    LCD_Position(0,0);
    LCD_PrintString("HOLA");
    LCD_Position(1,0);
    LCD_PrintString("MUNDO");
}

```

Figura 103. Código para imprimir un mensaje

Guardamos, compilamos, generamos aplicación y programamos.

- **Dormir-Despertar:** Esta opción lo que hace es que la pantalla esté intermitente durante el tiempo que queramos. Para ellos se utiliza las funciones LCD_DisplayOff() y LCD_DisplayOn().

```

//DORMIR-DESPERTAR
LCD_PrintString("HOLA MUNDO");
for(i=0;i<5;i++)
{
    LCD_DisplayOff();
    CyDelay(200);
    LCD_DisplayOn();
    CyDelay(200);
}

```

Figura 104. Código para parpadear la pantalla

Guardamos, compilamos, generamos aplicación y programamos.

- **Barra de progreso:** Se imprimirá un mensaje, mientras se llena una barra de progreso. Como será una barra horizontal es importante que el bloque de la LCD tenga seleccionada la opción Horizontal Bargraph, para ello vamos al bloque de la LCD y hacemos doble click para configurar.

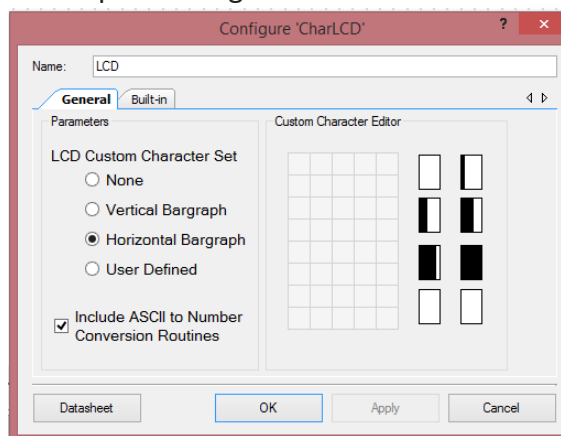


Figura 105. Configuración de la LCD como un gráfico de barras horizontal

Para la barra de progreso se utilizará la función LCD_DrawHorizontalBG(fila, Columna, máximo de caracteres, máximo de pixeles).

```
//BARRA DE PROGRESO
LCD_PrintString("HOLA MUNDO");
for(i=0;i<80;i++)
{
    LCD_DrawHorizontalBG(1,0,16,i);
    CyDelay(100);
}
```

Figura 106. Código como barra de progreso

Guardamos, compilamos, generamos aplicación y programamos.

- **Crear caracteres:** para crear un carácter, nos dirigimos a la configuración de la LCD y seleccionamos, User define. En ese espacio podemos crear los caracteres que queramos. Aquí tendremos 8 espacios para crear nuestros caracteres.

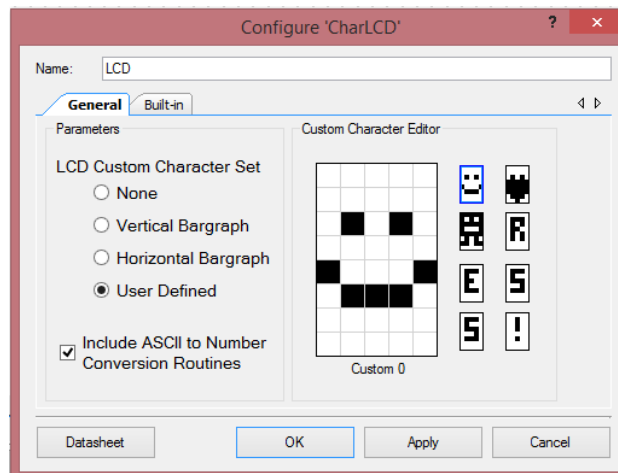


Figura 107. Configuración de la LCD para caracteres especiales

Guardamos y compilamos.

Después de crear los caracteres, la manera indicada de llamarlos en el main es utilizando la instrucción LCD_LoadCustomFonts(LCD_customFonts). Y para imprimir, se utiliza la instrucción LCD_PutChar(LCD_CUSTOM_X).

```
//FIGURAS
LCD_LoadCustomFonts(LCD_customFonts);
LCD_WriteControl(LCD_CLEAR_DISPLAY);
LCD_Position(0,0);
LCD_PutChar(LCD_CUSTOM_0);
LCD_PutChar(LCD_CUSTOM_1);
LCD_PutChar(LCD_CUSTOM_2);
```

Figura 108. Código para imprimir los caracteres

Guardamos, compilamos, generamos aplicación y programamos.

3.2.7.4 Segundo método

También hay otra manera de manejar la LCD, asignando los pines que el usuario quiera. Pero para esto se necesita una librería.

3.2.7.4.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Dirigirse a la pestaña Project, Dependencies, buscamos la sección user dependencies

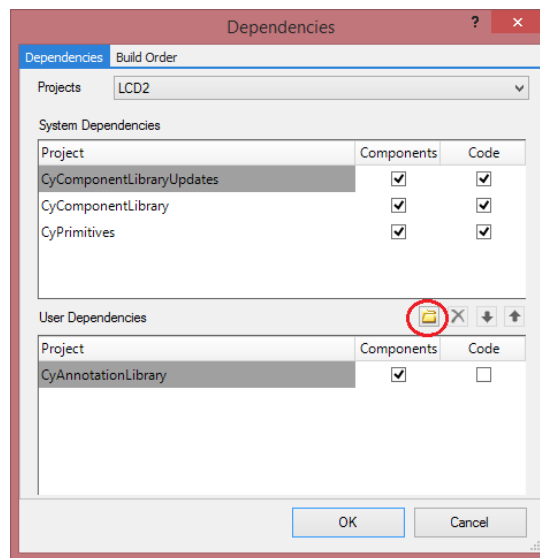


Figura 109. Agregar librería

Busca donde guardó la carpeta, CharLCDmp_Demo3, abrimos escogemos la carpeta CharLCDmpLib.cylib, y escogemos el archivo CharLCDmpLib.

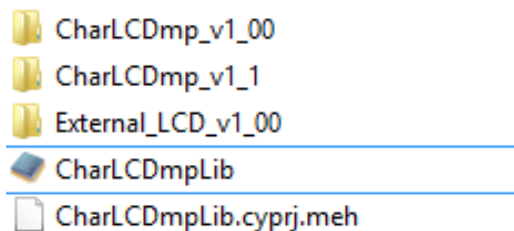


Figura 110. Nombre de librería

Después de agregar la librería, vamos a pestaña comunidad, al lado del catálogo de componentes y ahí aparecerá la LCD.

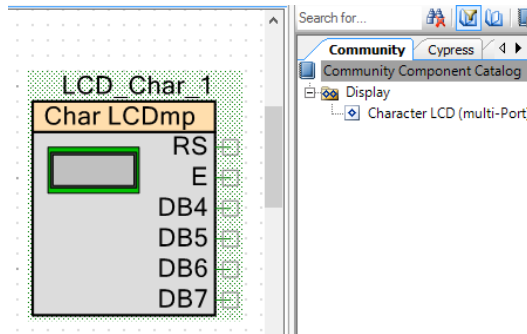


Figura 111. Ubicación del bloque LCD

Lo que sigue es asignar los pines, para eso utilizamos pines de salida digital y le cambiamos los nombres para la comodidad a la hora de asignar los pines.

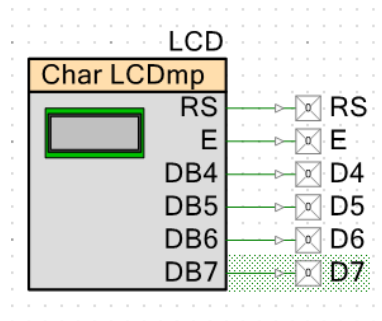


Figura 112. Conexión de pines con la LCD

La conexión sería de la siguiente manera.

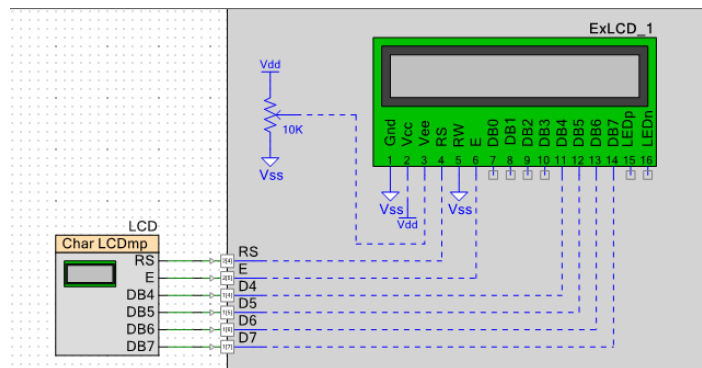


Figura 113. Distribución de pines

Guardamos y compilamos.

3.2.7.4.2 Asignación de pines

El usuario puede escoger los pines que desee, para este caso se configuraron de la siguiente manera:

	Name ▲	Port	Pin	Lock
<input type="checkbox"/>	D4	P2[6]	1	<input checked="" type="checkbox"/>
<input type="checkbox"/>	D5	P2[7]	2	<input checked="" type="checkbox"/>
<input type="checkbox"/>	D6	P12[7]	21	<input checked="" type="checkbox"/>
<input type="checkbox"/>	D7	P12[6]	20	<input checked="" type="checkbox"/>
<input type="checkbox"/>	E	P2[5]	68	<input checked="" type="checkbox"/>
<input type="checkbox"/>	RS	P2[4]	66	<input checked="" type="checkbox"/>

Figura 114. Pines asignados

Guardamos y compilamos

3.2.7.4.3 Código del Main

Para configurar esta LCD se utilizan las mismas librerías como si fuera la LCD del ejemplo anterior. En este solo se va a imprimir un mensaje para cerciorarse de que el dispositivo funcione correctamente.

```
int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    LCD_Start();
    LCD_Position(0,0);
    LCD_PrintString("Hola");
    LCD_Position(1,0);
    LCD_PrintString("Mundo");
}
```

Figura 115. Código para imprimir mensaje

Guardamos, compilamos, generamos aplicación y programamos.

3.2.8 ADC

El convertor de análogo a digital delta sigma (ADC_DelSig) proporciona un bajo consumo de energía, bajo ruido para aplicaciones de medición de precisión. Cuando se procesa la información de audio, el ADC_DelSig se utiliza en un modo de funcionamiento continuo. Cuando se utiliza para la exploración de múltiples sensores, el ADC_DelSig se utiliza en uno de los modos de múltiples muestras. Cuando se utiliza para mediciones de alta resolución de un solo punto, el ADC_DelSig se utiliza en el modo de una sola muestra.

3.2.8.1 Objetivos

- Conocer la funcionalidad del bloque ADC Sigma Delta del PSoC Creator.
- Manejar el ADC Delta Sigma del PSoC 3 y PSoC 5 LP utilizando la interfaz de PSoC Creator.

3.2.8.2 Parámetros del componente

En la ventana de configuración del ADC podemos encontrar diferentes opciones de configuración:

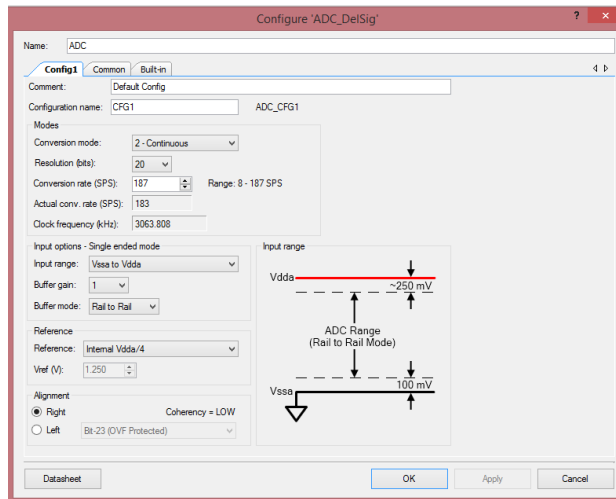


Figura 116. Parámetros del ADC

Conversion mode

- **0 - Single Sample:** El ADC produce una muestra por la conversión de inicio.
- **1 - Multi Sample:** Modo multi-muestra, captura muestras individuales espalda con espalda, al restablecer en sí y el modulador entre cada muestra automáticamente.
- **2 - Continuos:** el modo de muestreo continuo funciona como un convertidor sigma-delta normal.
- **3 - Multi Sample (Turbo):** El modo de muestras múltiples (Turbo) funciona de forma idéntica al modo de muestras múltiples para resoluciones de 8 a 16 bits. Para resoluciones de 17 a 20 bits, el rendimiento de este modo es aproximadamente cuatro veces más rápido que el modo de muestras múltiples.

Resolución: La resolución del ADC_DeISig se introduce como un valor entero, limitado entre 8 y 20 bits. La resolución predeterminada es de 16 bits.

Tasa de conversión: Se introduce como un valor decimal entero en muestras por segundo (SPS). El reloj mínimo para todas las resoluciones es de 128 kHz. El reloj máximo para resoluciones de entre 16 y 20 bits es 3.027 MHz.

Rango: Este campo es un área de sólo lectura (siempre disponible) que muestra el mínimo y el índice de conversión máxima disponible para la configuración actual.

Tasa de conversión actual: Este campo de sólo lectura muestra un tipo de conversión recalculado real basado en la frecuencia del reloj maestro tomado de los Recursos de diseño-ancha (DWR) Editor de reloj y calcula divisor entero.

Frecuencia de reloj: Este cuadro de texto es una lectura única área (siempre disponible) que muestra la tasa requerida de reloj para las condiciones operativas seleccionadas: el modo de conversión, resolución, tasa de conversión, rango de entrada, y la ganancia de amortiguación.

3.2.8.3 Interfaz de programación de aplicaciones

Interfaz de programación de aplicaciones (API) de rutinas le permiten configurar el componente utilizando el software. La siguiente tabla muestra y describe la interfaz para cada función. Las secciones siguientes cubren cada función con más detalle.

3.2.8.4 Guía paso a paso.

En este proyecto, utilizaremos un ADC y la pantalla LCD. El conversor ADC se utilizará para leer una entrada análoga y ver en la LCD el resultado de esa conversión a digital.

3.2.8.4.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos analog, ADC, y escogemos Delta Sigma ADC

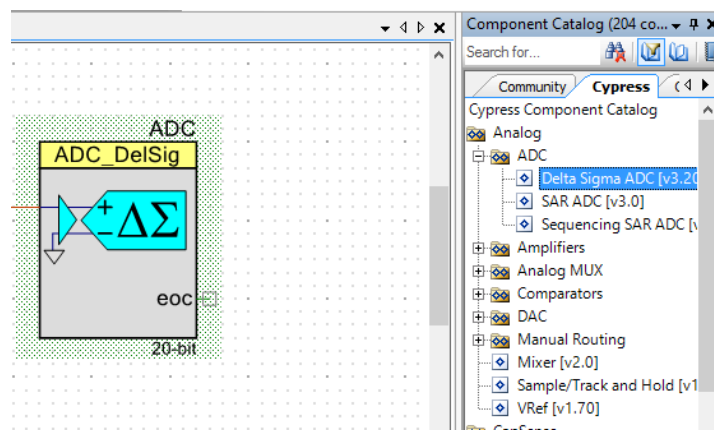


Figura 117. Ubicación del ADC delta sigma en el catálogo de componentes

Hacemos doble click para configurar, lo primero que debemos hacer es ir a la pestaña common y en el número de configuraciones escogemos 1 y en input mode escogemos single ended, le damos aplicar.

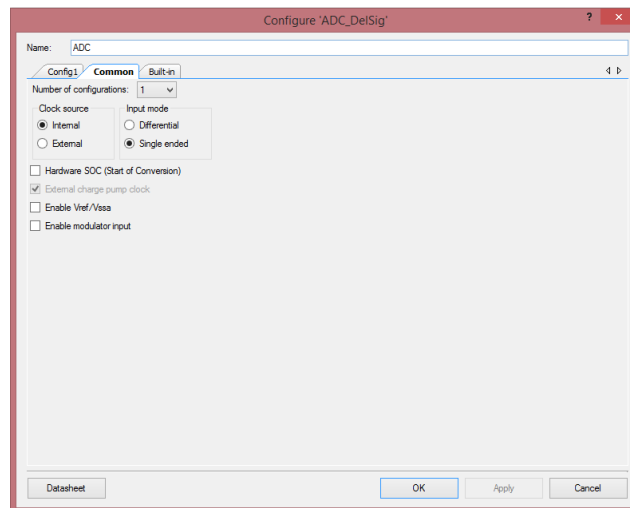


Figura 118. Configuración del ADC

En la pestaña config, el nombre lo cambiamos por ADC para comodidad de programación, el modo de conversión será continuo ya que queremos que el conversor funcione de manera normal, la resolución de 20 bits. El rango de entrada será de V_{SSA} a V_{DDA} , ya que la entrada será emulada por un potenciómetro irá de 0V a 5V, ganancia del buffer 1. Y en referencia internal $V_{DDA}/4$.

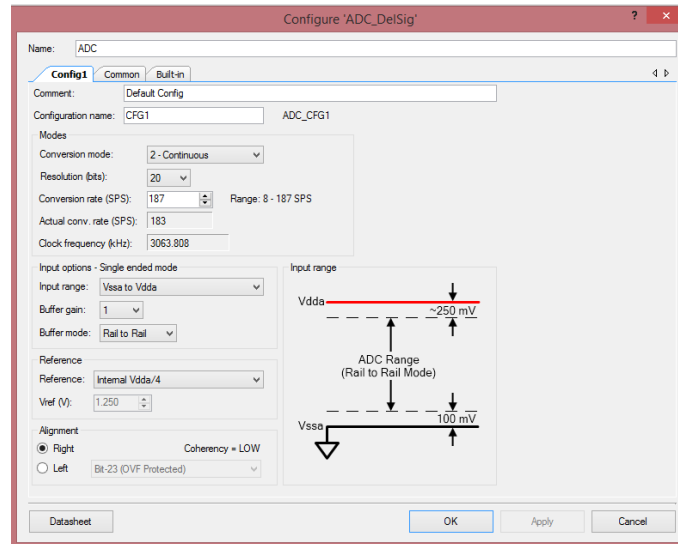


Figura 119. Configuración del ADC

Ahora necesitamos un pin análogo, para ello vamos a ports and pins y escogemos analog.

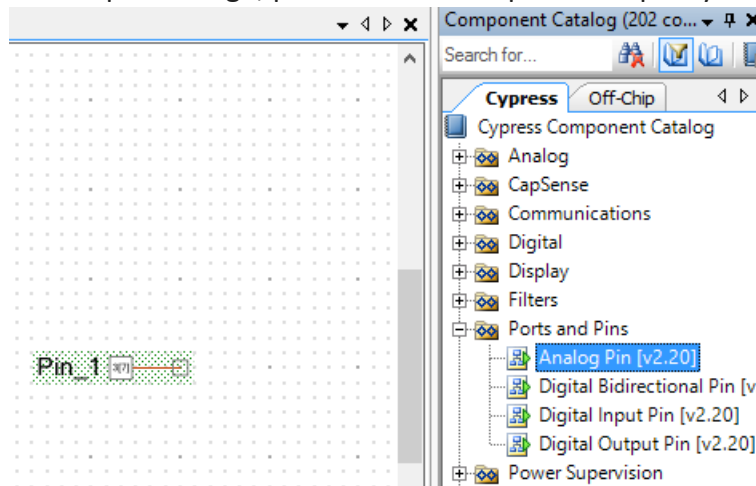


Figura 120. Ubicación del pin análogo

Este pin llevará por nombre InAnalog, y se conectará a la entrada del VDAC.

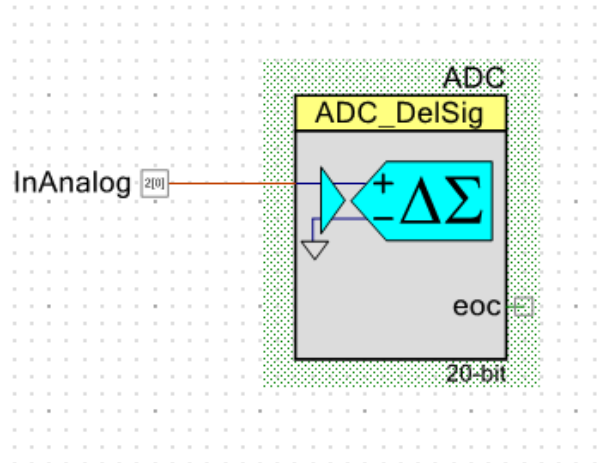


Figura 121. Conexión del pin analógico con el ADC

Lo que sigue es añadir la LCD, en este caso voy a utilizar la LCD que me permite asignar los pines que yo quiera.

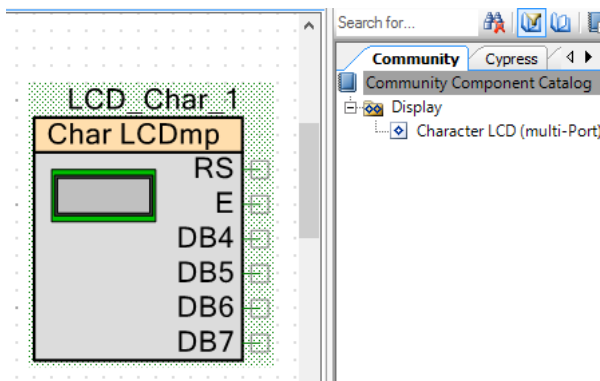


Figura 122. Ubicación de la LCD

Lo que sigue es asignar los pines, para eso utilizamos pines de salida digital y le cambiamos los nombres para la comodidad a la hora de asignar los pines.

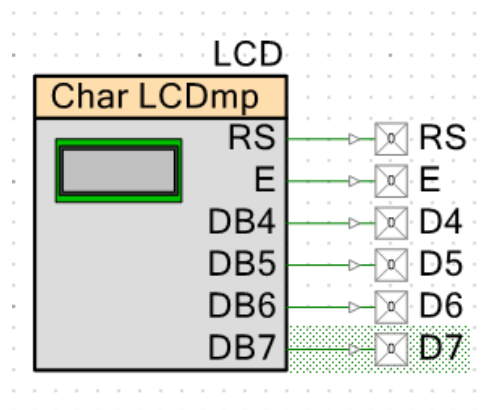


Figura 123. Conexión de pines con la LCD

Guardamos y compilamos.

Vamos al archivo de asignar pines. A system, y en Hype Size (bytes) se coloca el valor de 0x200.

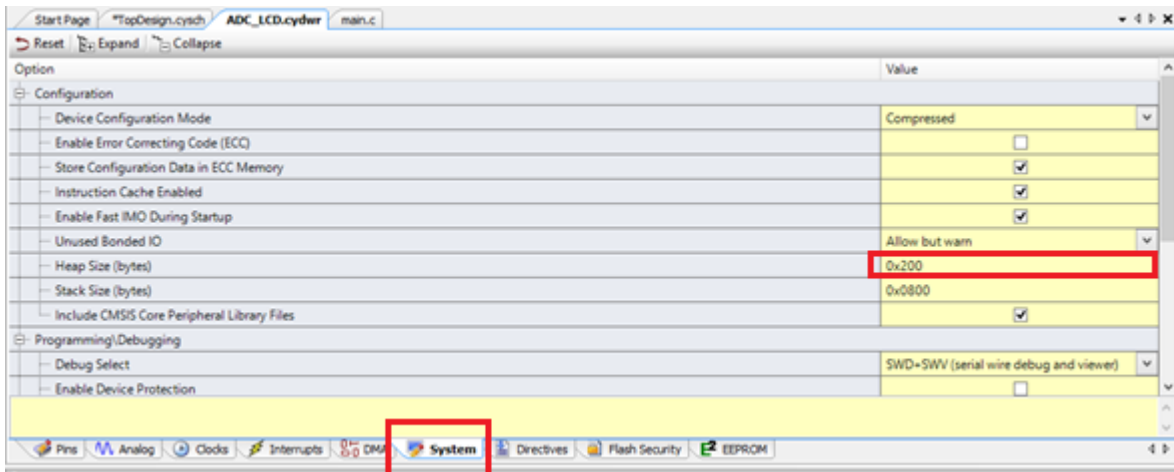


Figura 124. Configuración de la memoria para la LCD

Lo que se hace aquí es aumentar la memoria, pero para que funcione bien la LCD nos vamos al Workspace Explorer, damos click derecho en Project y escogemos Built Settings.

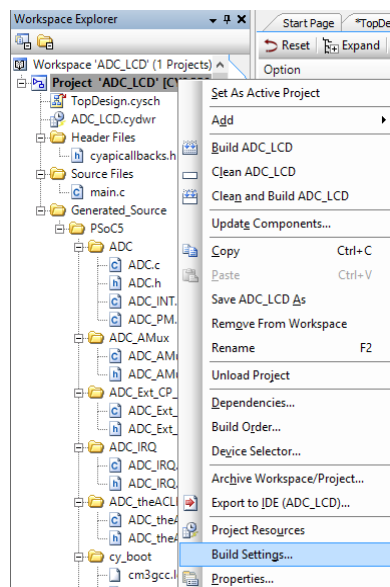


Figura 125. Configuración para el buen funcionamiento de la LCD

En la ventana que se abrió, vamos a Linker, Command Line, y en Custom Flags escribimos "-u_printf_float". Esto quiere decir que se va a habilitar la función para imprimir las variables de tipo float.

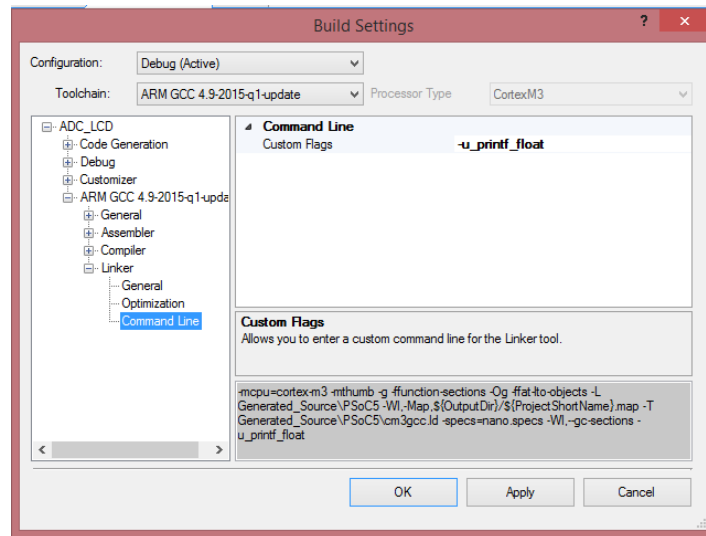


Figura 126. Comando para la LCD

Guardamos y compilamos

3.2.8.4.2 Código del Main

Antes de empezar se debe agregar la librería `#include <stdio.h>` porque esta librería me permite manejar la instrucción `sprintf`, esta función me permite convertir cualquier variable a carácter. Luego de agregar la librería en el main creamos una variable `int32` que se llamará `output` que será la equivalencia de binario a entero de lo que lee el ADC, una variable tipo `char8` que se llamará `str[12]` de doce posiciones, también habrá una variable tipo `float` que llevará la conversión en voltaje del potenciómetro.

```

12 | #include <project.h>
13 | #include <stdio.h>
14 |
15 |
16 | int main()
17 | {
18 |     int32 output;
19 |     char8 str[32];
20 |     float ADCVoltaje;
21 |

```

Figura 127. Código para declarar variables y agregar librería.

Lo siguiente es iniciar la LCD, posicionarla e imprimir un mensaje que diga "Voltaje:"

```

24 | CyGlobalIntEnable; /* Enable global interrupts. */
25 | LCD_Start();
26 | LCD_Position(0,0);
27 | LCD_PrintString("VOLTAJE:");
28 |

```

Figura 128. Código para iniciar LCD.

Ahora iniciamos el conversor, iniciamos la conversión y la instrucción ADC_IsEndConversion(ADC_WAIT_FOR_RESULT), que comprueba la existencia de final de la conversión y no retorna hasta que la conversión ADC es completa.

```

29 |         ADC_Start ();
30 |         ADC_StartConvert ();
31 |         ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);

```

Figura 129. Código para iniciar ADC

En el For infinito, vamos a leer el ADC, pasar de la representación entera de lo que lee el ADC a float que represente ese valor en voltaje. Para ello utilizamos la variable ADCVoltaje que será igual al Voltaje (5.000 V) sobre la resolución en bits ($2^{20}= 1048576$) y se multiplica sobre la variable que lleva la cuenta binaria. Posicionamos la LCD y llamamos a la función sprintf(variable a asignar la conversión, que tipo de variable voy a convertir, nombre), y luego imprimimos el valor.

```

for(;;)
{
    output=ADC_GetResult32(); //Output = a leer el adc
    ADCVoltaje=(5.000/1048576) * output;
    LCD_Position(1,0);
    sprintf(str,"%0.2f",ADCVoltaje);
    LCD_PrintString(str);
}

```

Figura 130. Código para leer e imprimir resultado del ADC

Guardamos y compilamos

3.2.8.4.3 Asignación de Pines

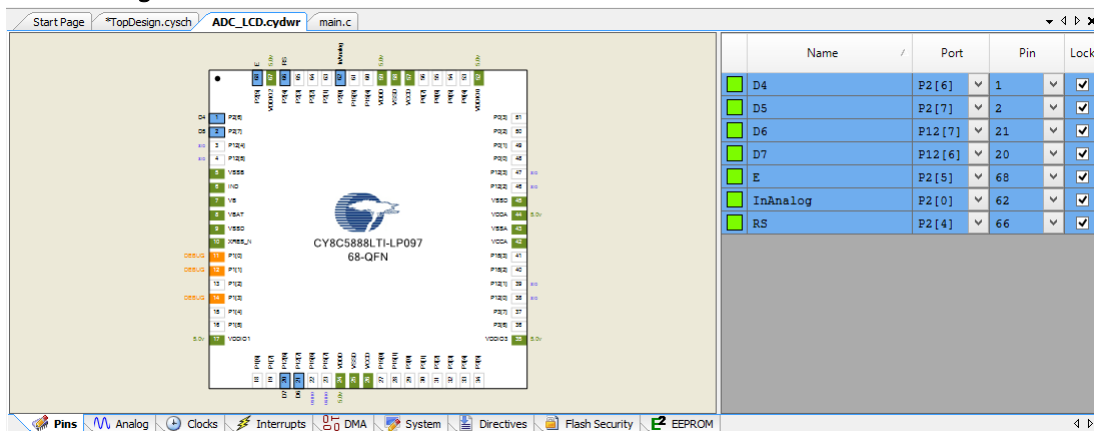


Figura 131. Pines Asignados

Guardamos, compilamos, generamos aplicación y programamos.

3.2.9 VDAC

El componente VDAC8 es una salida de voltaje de 8 bits del convertidor de digital a analógico (DAC). El rango de salida puede ser desde 0 hasta 1,020 V (4 mV / bit) o 0 a 4,08 V (16 mV / bit).

3.2.9.1 Objetivos

- Conocer la funcionalidad del DAC de voltaje de PSoC Creator.
- Utilizar la interfaz de PSoC Creator para configurar un DAC de voltaje.

3.2.9.2 Parámetros del componente.

Arrastre un componente VC8 en su diseño y haga doble clic para abrir el cuadro de diálogo Configurar.

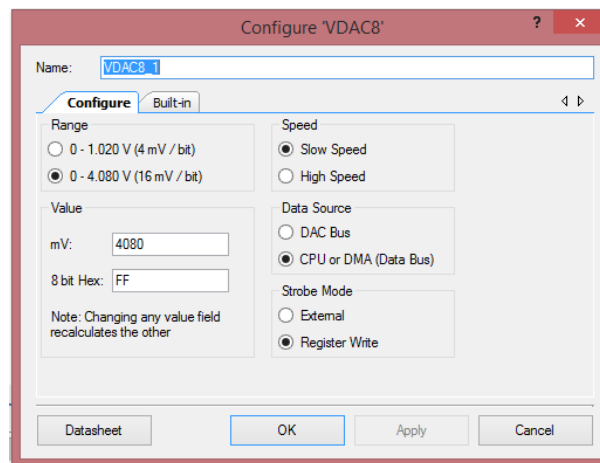


Figura 132. Parámetros del VDAC.

Rango: Este parámetro permite configurar una de las dos gamas de tensión como el valor predeterminado.

Ecuación de salida:

- Rango 1-V: $V_{OUT} = (\text{Valor}/256) \times 1.024 \text{ V}$
- Rango 4-V: $V_{OUT} = (\text{Valor}/256) \times 4.096 \text{ V}$

Nota El término "valor" es un número entre 0 y 255.

Valor: Este es el valor inicial del VDAC8 después de ejecutar el comando `VDAC8_Start ()`. La función `VDAC8_SetValue ()` o una escritura directa al registro DAC tienen prioridad sobre el valor por defecto en cualquier momento.

Velocidad: Este parámetro proporciona dos ajustes: slow y high. En el modo slow, el tiempo de establecimiento es más lento, pero consume menos corriente de funcionamiento. En el

modo High, la tensión se instala mucho más rápido, pero a un costo de más corriente de funcionamiento.

Fuente de datos: Este parámetro selecciona la fuente de los datos a escribir en el registro de DAC. Selección de la CPU o la opción DMA (bus de datos), se seleccionará la CPU (firmware) o la DMA para escribir datos en el VDAC8.

Modo Strobe: Este parámetro selecciona si los datos se escriben inmediatamente al DAC cuando los datos se escriben en el registro de datos VDAC8. Este modo se activa cuando se selecciona la opción Write Registro.

3.2.9.3 Guía paso a paso

Se realizará un proyecto sencillo, utilizando el componente análogo VDAC y una salida análoga. Esto para familiarizarnos con el DAC de voltaje.

3.2.9.3.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos al catálogo de Cypress, en la pestaña analog, escogemos un DAC de voltaje.

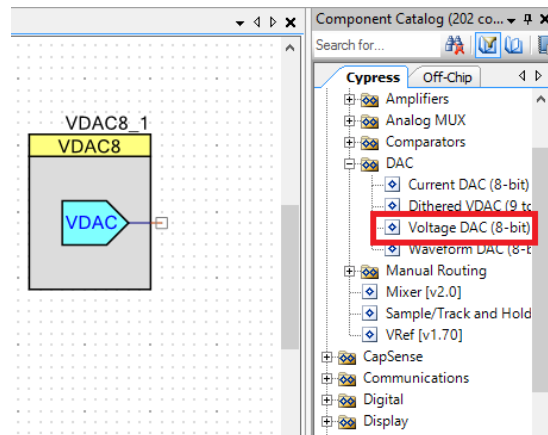


Figura 133. Ubicación del VDAC en el catálogo

Hacemos doble click para configurar, en el rango escogemos 0 – 4.080V, y en el valor escogemos el máximo que sería FF en hexadecimal, esto quiere decir que el voltaje de salida del VDAC será 4,080V. Aplicar y aceptar.

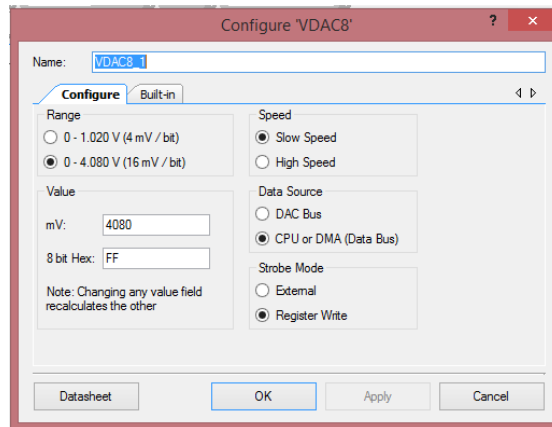


Figura 134. Configuración del VDAC

Lo siguiente es añadir un pin, análogo, para ello vamos a ports and pins y escogemos analog.

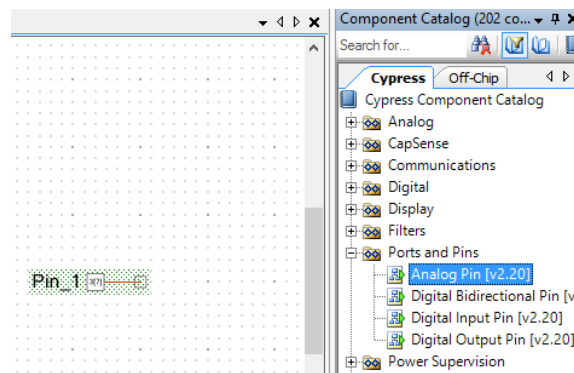


Figura 135. Ubicación del pin análogo

Para configurarlo como salida, hacemos click derecho sobre el pin, shape, Flip Horizontal. Después de eso unimos el pin a la salida del DAC.

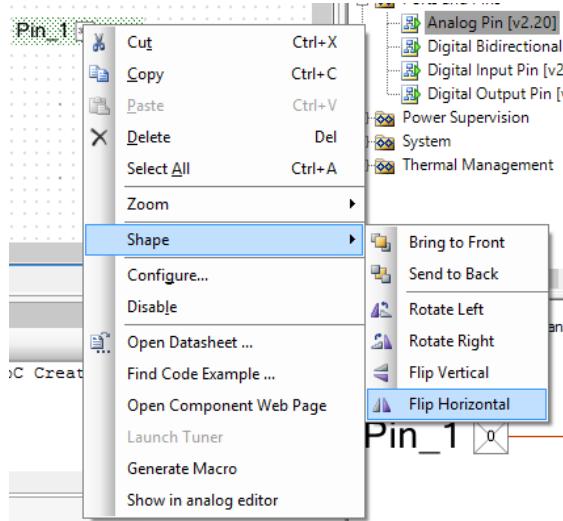


Figura 136. Pin análogo como salida

Luego de esto es importante guardar y compilar.

3.2.9.3.2 Código del Main

En el main, simplemente iniciaremos el DAC con la instrucción VDAC8_1_Start().

```
#include <project.h>

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    VDAC8_1_Start();

    for(;;)
    {
        /* Place your application code here. */
    }
}
```

Figura 137. Código para iniciar el VDAC

Luego de esto es importante guardar y compilar

3.2.9.3.3 Asignación de pines

En este caso como se están manejan pines análogos, Cypress hace una recomendación sobre los mejores pines análogos para las PSoC 3 y PSoC 5LP. Estos pines son P0[7:0], P3[7:0] y P4[7:4].

Al momento de asignar los pines se escogió el pin P3[7]

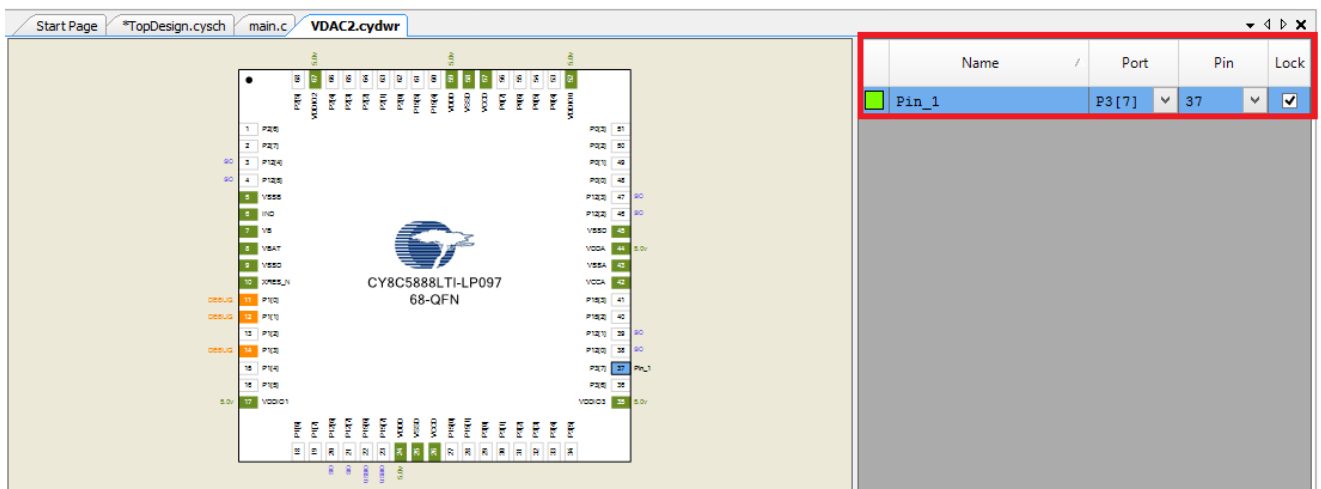


Figura 138. Pines asignados

Guardamos, compilamos, generamos aplicación y programamos.

3.2.10 VDAC y ADC

Esta práctica trata de tener un valor digital de 8 bits y convertirlo a un valor análogo en voltaje, para ellos utilizaremos un VDAC y un ADC.

3.2.10.1 Objetivo

- Combinar las funcionalidades del VDAC y el ADC delta sigma del PSoC Creator para ser utilizados en una aplicación.

3.2.10.2 Guía paso a paso

Con el pulsador que se tiene disponible en la tarjeta, vamos a ir aumentando de 10 en 10 el valor de 8 bits que tiene el VDAC, para ver su voltaje correspondiente se utilizará el DAC. Y el valor se mostrará en la LCD.

3.2.10.2.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos analog, ADC, y escogemos Delta Sigma ADC

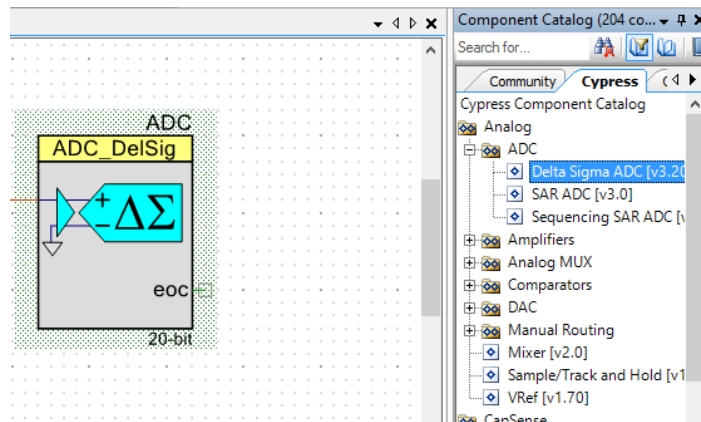


Figura 139. Ubicación del ADC en el catálogo

También necesitamos un DAC. En la pestaña analog, escogemos un DAC de voltaje.

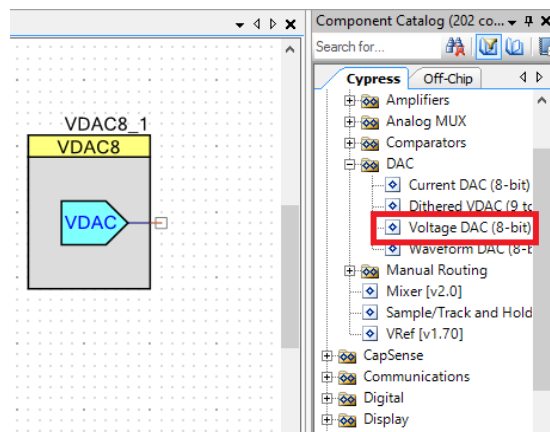


Figura 140. Ubicación del DAC en el catálogo

Hacemos doble click para configurar, lo primero que debemos hacer es ir a la pestaña common y en el número de configuraciones escogemos 1 y en input mode escogemos single ended, le damos aplicar.

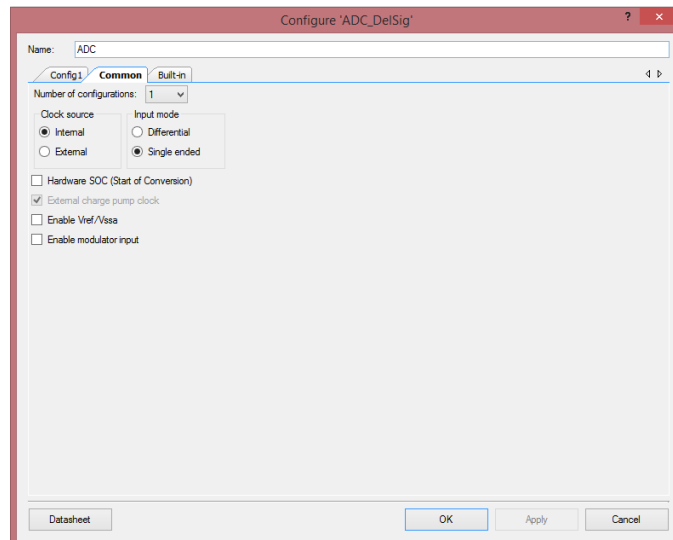


Figura 141. Configuración del ADC

En la pestaña config, el nombre lo cambiamos por ADC para comodidad de programación, el modo de conversión será continuo y le damos las siguientes indicaciones.

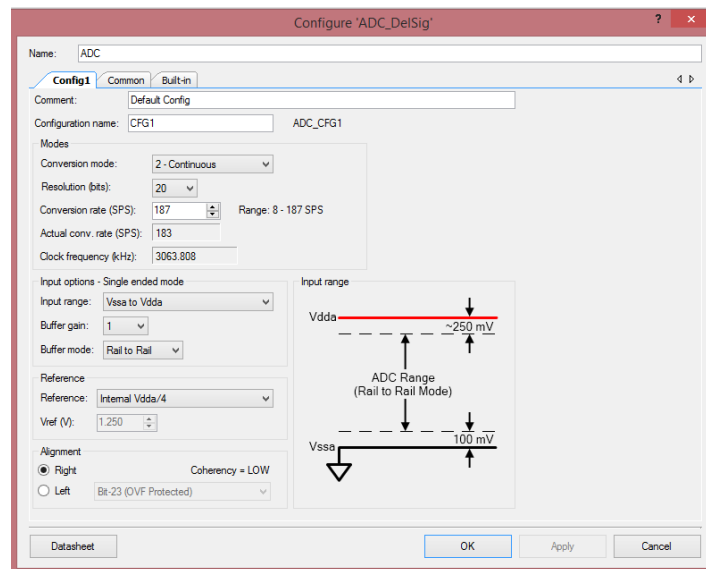


Figura 142. Configuración del ADC

Ahora voy a ports and pins y arrastro un pin análogo que irá conectado a la entrada del ADC y otro que irá a la salida del VDAC.

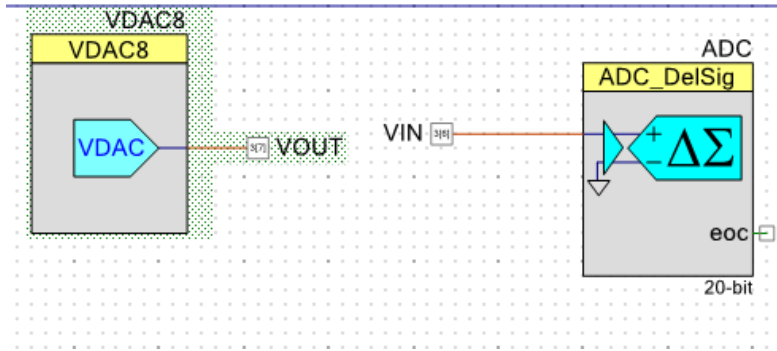


Figura 143. Conexión de componentes

Ahora vamos a configurar el DAC como lo indica la imagen.

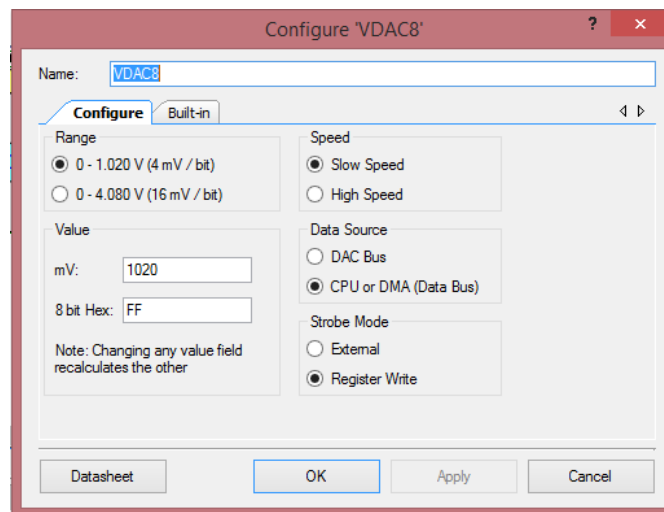


Figura 144. Configuración del VDAC

El siguiente paso es añadir la LCD.

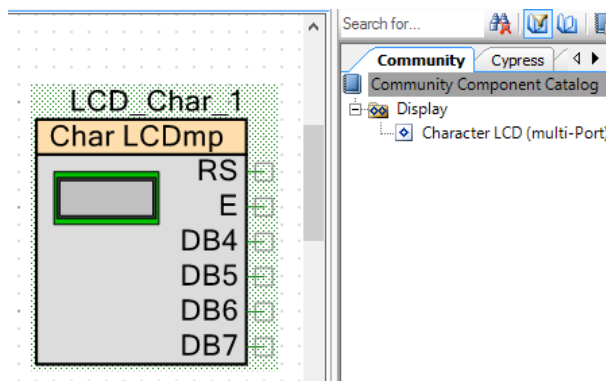


Figura 145. Ubicación de la LCD.

Lo que sigue es asignar los pines, para eso utilizamos pines de salida digital y le cambiamos los nombres para la comodidad a la hora de asignar los pines.

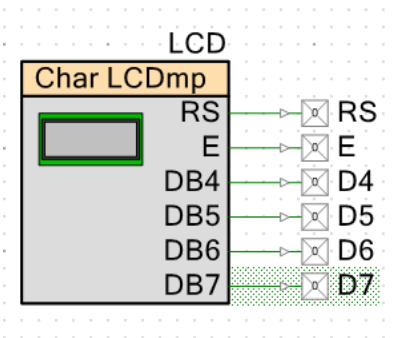


Figura 146. Conexión de pines con la LCD.

Ahora añadimos un pin digital de entrada y una interrupción

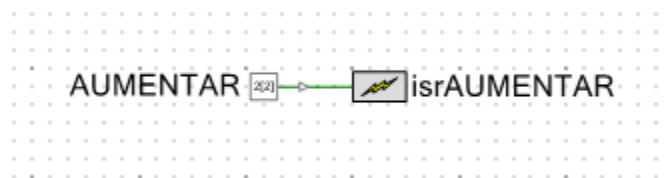


Figura 147. Conexión del pin de entrada con la interrupción

El pin de entrada se llamará aumentar, y estará configurado como pull up resistivo. Y la interrupción se llamará isrAUMENTAR y será de tipo RISING_EDGE.

Guardamos y compilamos.

3.2.10.2.2 Código del Main

Lo primero que hay que hacer es incluir la librería `#include <stdio.h>` que nos permite hacer uso de la función `sprintf`. Creamos una variable `int VDAC=0` que se irá cambiando de 0 – 255 para los 8 bits del ADC. `uint32_t ADCin` entero de entrada para el ADC. `float32 vin` variable a convertir, `string` para poder ver el voltaje `char str[12]`.

```
#include <project.h>
#include <stdio.h>

int VDAC=0;
uint32_t ADCin;
float32 vin;
char str[12];
```

Figura 148. Código para declarar variables

En el main iniciaremos la LCD, el ADC y la función de la interrupción.

```

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    isrAUMENTAR_StartEx(AumentarV);
    LCD_Start();

    VDAC8_Start();
    ADC_Start();
    ADC_StartConvert();
    ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
}

```

Figura 149. Código para iniciar componentes

En el For infinito, leemos el ADC, ese valor que lee el ADC lo vamos a convertir a voltaje, para eso utilizamos la instrucción `ADC_CountsTo_Volts(variable)`. Se utiliza la función `LCD_PrintNumber(VDAC)` para imprimir un número.

```

for(;;)
{
    ADCin=ADC_GetResult32();
    vin = ADC_CountsTo_Volts(ADCin);
    LCD_Position(0,0);
    LCD_PrintString("DAC 8bit: ");
    LCD_PrintNumber(VDAC);
    LCD_PrintString(" "); //Borrar numero anterior
    LCD_Position(1,0);
    LCD_PrintString("ADC_INPUT:");
    LCD_Position(1,10);
    sprintf(str,"%3fV",vin);
    LCD_PrintString(str);
}

```

Figura 150. Código para leer ADC y mostrar resultado en LCD

Lo que nos falta es declarar la función de la interrupción, la llamaremos `AumentarV`. Aquí aumentamos la variable `VDAC` de a 10 en 10, hasta que su valor sea mayor igual a 255. Cuando alcance este valor `VDAC` será igual a 0. Y luego colocar la instrucción que haga cambiar el valor del `VDAC` que será `VDAC8_SetValue(VDAC)`.

```

CY_ISR(AumentarV)
{
    VDAC=VDAC+10;
    if (VDAC>=255)
    {
        VDAC=0;
    }
    VDAC8_SetValue(VDAC);
}

```

Figura 151. Código para función de la interrupción

Guardamos y compilamos.

3.2.10.2.3 Asignación de Pines

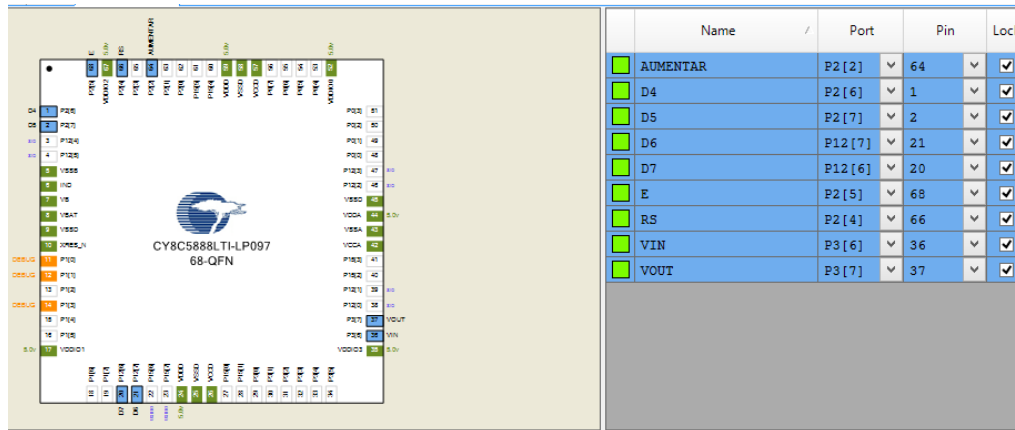


Figura 152. Pines asignados.

Los pines VIN y VOUT se unen (puente).

Guardamos, compilamos, generamos aplicación y programamos.

3.2.11 OpAmp

Proporciona un bajo voltaje, amplificador operacional de baja potencia y puede ser conectado internamente como un seguidor de tensión. El Opamp es adecuado para la interfaz con sensores de alta impedancia, amortiguando la salida del DAC de tensión.

3.2.11.1 Objetivos

- Conocer la funcionalidad de un amplificador operacional PSoC.
- Utilizar la interfaz de PSoC Creator para configurar un amplificador operacional en modo seguidor, con la ayuda de un VDAC.

3.2.11.2 Parámetros del componente.

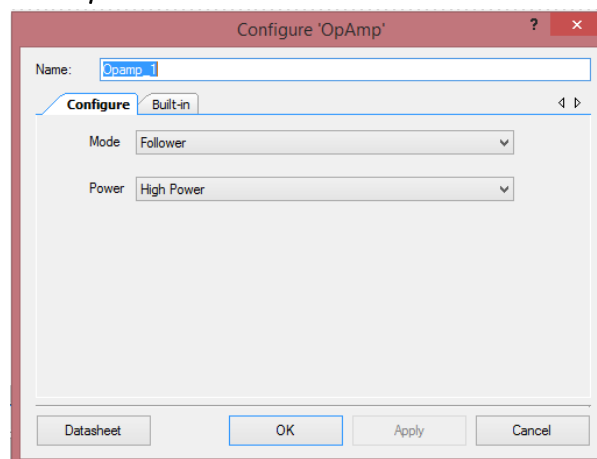


Figura 153. Parámetros del Amplificador Operacional

Modo

Este parámetro le permite seleccionar entre dos configuraciones: OpAmp y seguidor. OpAmp es la configuración por defecto.

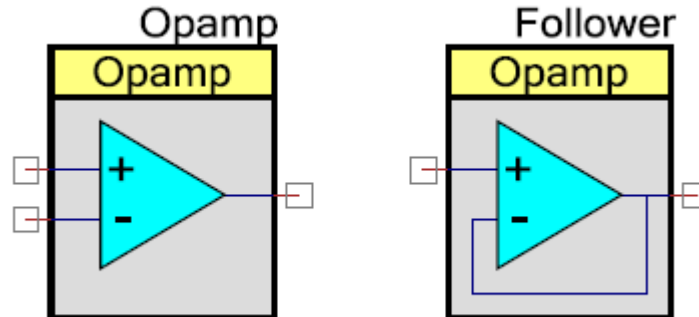


Figura 154. Configuraciones del AmpOp

POWER: El Opamp funciona sobre una amplia gama de corrientes de funcionamiento. La corriente de funcionamiento más alta aumenta el ancho de banda del Opamp.

3.2.11.3 Guía paso a paso

El siguiente proyecto pretende verificar el funcionamiento de un amplificador operacional en la configuración de seguidor de tensión. Para ello necesitaremos, además del amplificador operacional, un VDAC para fijar un voltaje, este voltaje irá aumentando progresivamente en el for(;;).

3.2.11.3.1 Configuración del Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos al catálogo de Cypress, Analog, Amplifiers, Opamp y arrastramos.

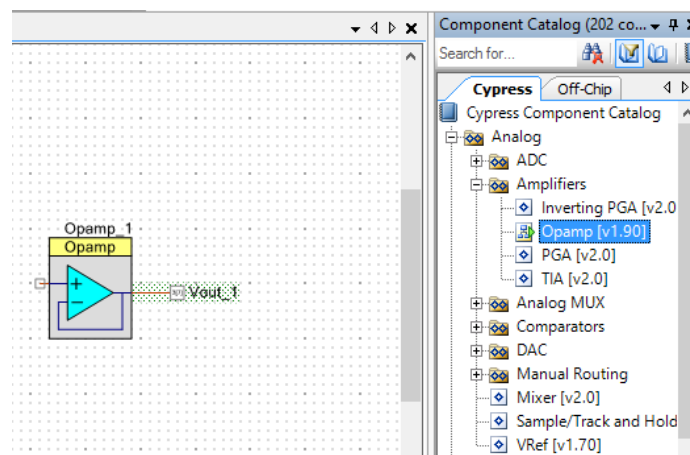


Figura 155. Ubicación del AmpOp en el catálogo

Doble click en el componente para configurar, escogemos el modo seguidor (Follower), y High Power. Aplicar y aceptar.

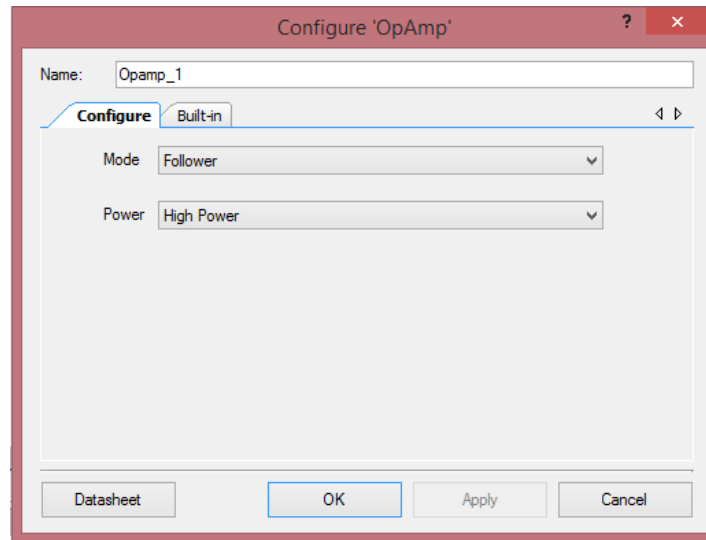


Figura 156. Configuración del Amplificador como seguidor

Ahora necesitamos un VDAC, para ello vamos a Analog, DAC, y escogemos el DAC de voltaje.

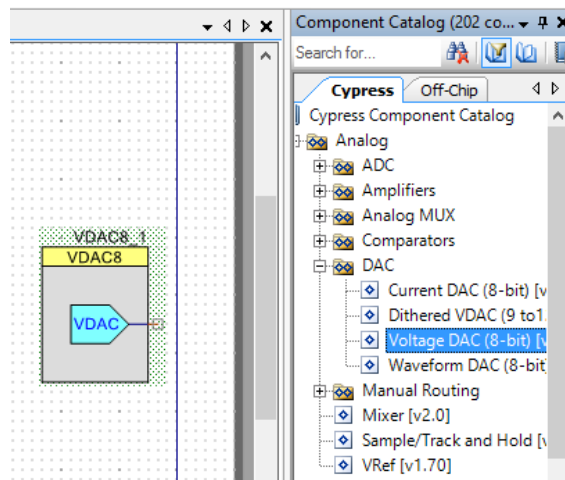


Figura 157. Ubicación del VDAC en el catálogo

Hacemos doble click para configurar, escogemos el rango de 0 - 4.080V, en Value le damos el máximo valor en hexadecimal que es FF. Aceptar y aplicar.

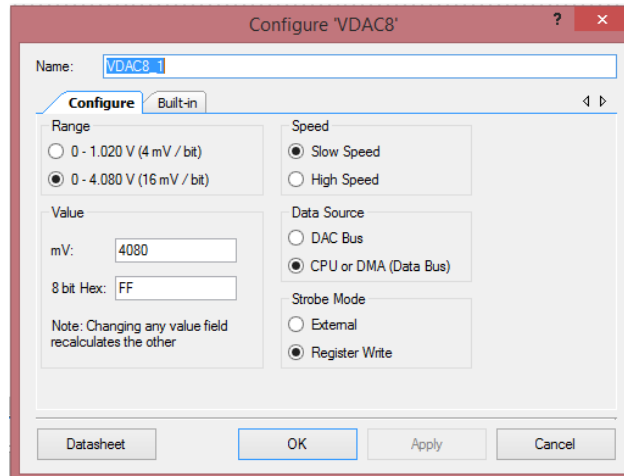


Figura 158. Configuración del VDAC

Unimos la salida del VDAC a la entrada no inversora del amplificador.

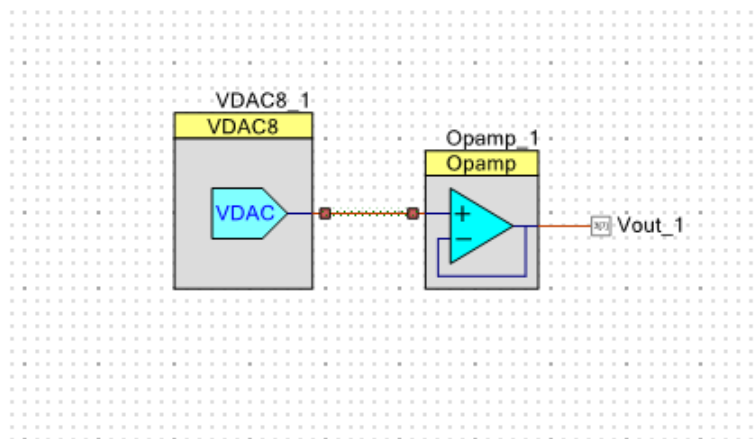


Figura 159. Conexión de los componentes

Luego de esto es importante guardar y compilar.

3.2.11.3.2 Código del Main

En el main, simplemente iniciaremos el DAC con la instrucción `VDAC8_1_Start()`, iniciaremos el amplificador con la instrucción `Opamp_1_Start()`. Además de esto necesitamos una variable `uint8` para ir aumentando el valor del voltaje de referencia esta variable puede ser `i`. La solución a la práctica sería de la siguiente manera.


```

#include <project.h>

int main()
{
    uint8 i;
    CyGlobalIntEnable; /* Enable global interrupts. */
    VDAC8_1_Start();
    Opamp_1_Start();

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        VDAC8_1_SetValue(i++);
        CyDelay(10);
    }
}

```

Figura 160. Código para iniciar variables y componentes

Luego de esto es importante guardar y compilar.

3.2.11.3.3 Asignación de pines

Cypress recomienda ciertos pines para cada amplificador, como en este caso utilizamos en Opamp_0 el pin indicado seria el P0[1].

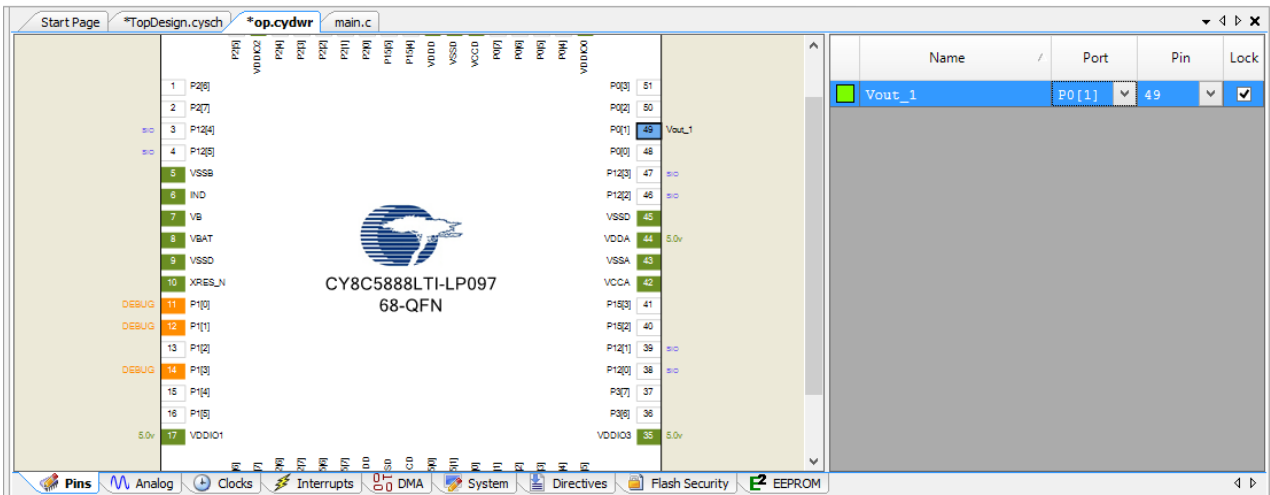


Figura 161. Pines asignados

Guardamos, compilamos, generamos aplicación y programamos.

3.2.12 Comparadores

El Comparador (Comp) proporciona una solución de hardware para comparar dos tensiones de entrada analógicas.

3.2.12.1 Objetivos

- Conocer la funcionalidad de los comparadores de PSoC Creator.

- Configurar los cuatro comparadores que vienen disponibles en los dispositivos PSoC 3 y PSoC 5LP, con diferentes niveles de voltaje para probar su funcionamiento.

3.2.12.2 Parámetros del componente

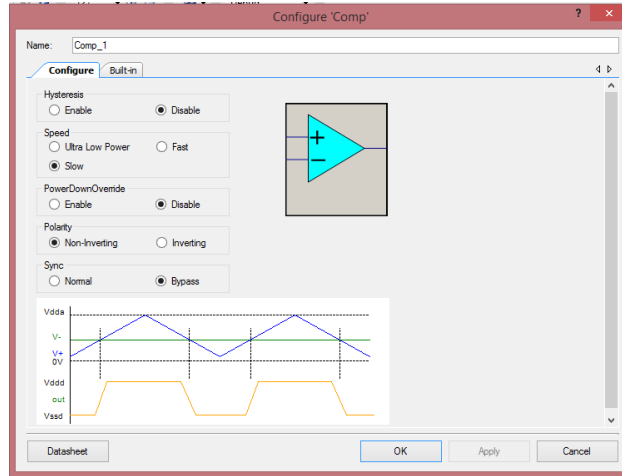


Figura 162. Parámetros del Comparador

Histéresis: Este parámetro le permite añadir aproximadamente 10 mV de histéresis al comparador.

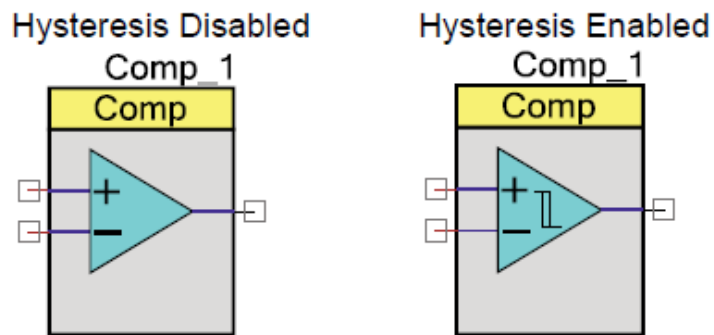


Figura 163. Configuraciones del comparador

Velocidad: Este parámetro proporciona una manera para que el usuario pueda optimizar la velocidad de consumo de energía.

PowerDownOverride: Aprovechando el abajo consumo, el parámetro override hace que el comparador pueda mantenerse activo durante el modo de suspensión.

Nota No utilice la salida invertida en este modo.

Polaridad: Este parámetro le permite invertir la salida del comparador.

Sync: Este parámetro selecciona entre la sincronización de la salida con un reloj. Cuando se selecciona Normal, la salida cambiará en el flanco ascendente de la entrada de reloj. Cuando

se selecciona Bypass, la salida será establecida inmediatamente y se restablece en el flanco ascendente del reloj.

3.2.12.3 Interfaz de programación de aplicaciones

Interfaz de programación de aplicaciones (API) de rutinas le permite configurar el componente utilizando el software. La siguiente tabla muestra la interfaz para cada función.

3.2.12.4 Guía paso a paso

En este proyecto trabajaremos con dos comparadores para probar su correcto funcionamiento, la actividad consiste en tener 2 niveles de voltaje diferentes, para eso se utilizarán 2 VDAC, y otro voltaje que irá cambiando que se representará con un potenciómetro.

3.2.12.4.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos al catálogo de Cypress, Analog, comparators y arrastramos un comparador.

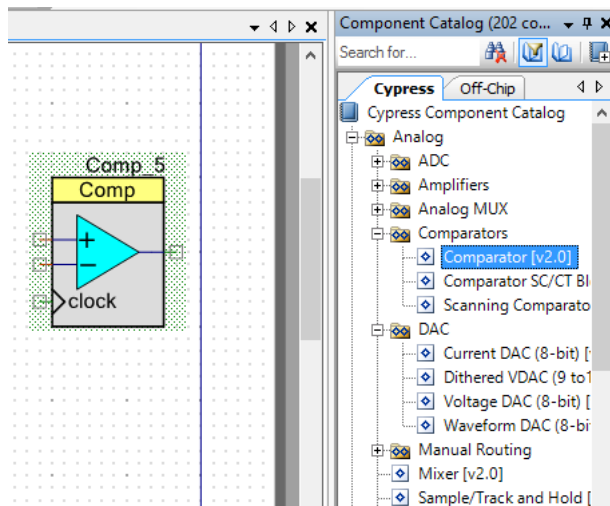


Figura 164. Ubicación del Comparador en el catálogo

Hacemos doble click sobre el componente y lo configuramos de la siguiente manera

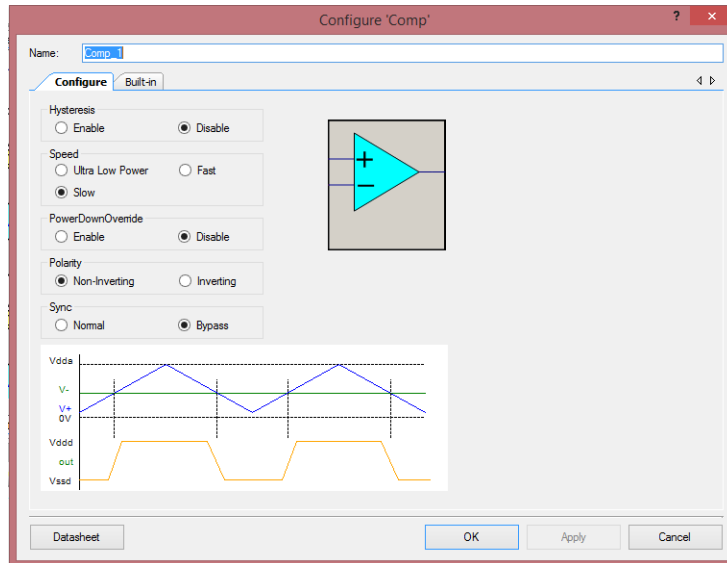


Figura 165. Configuración del Comparador

Ahora necesitamos un VDAC, vamos a analog, DAC y escogemos un VDAC

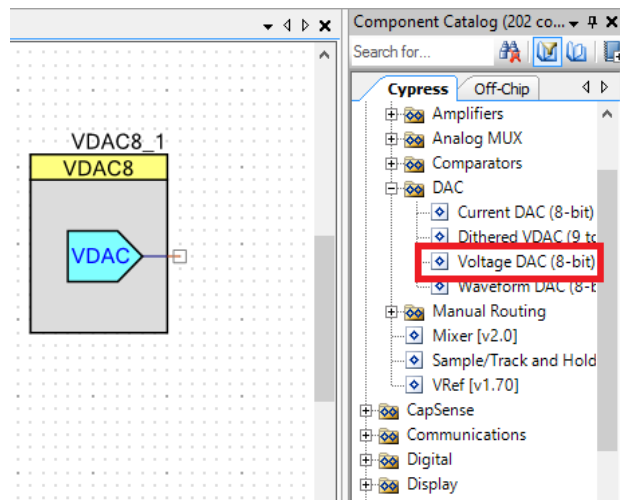


Figura 166. Ubicación del VDAC en el catálogo

Y lo configuramos de la siguiente manera, para el primer comparador pondremos una referencia de 4V.

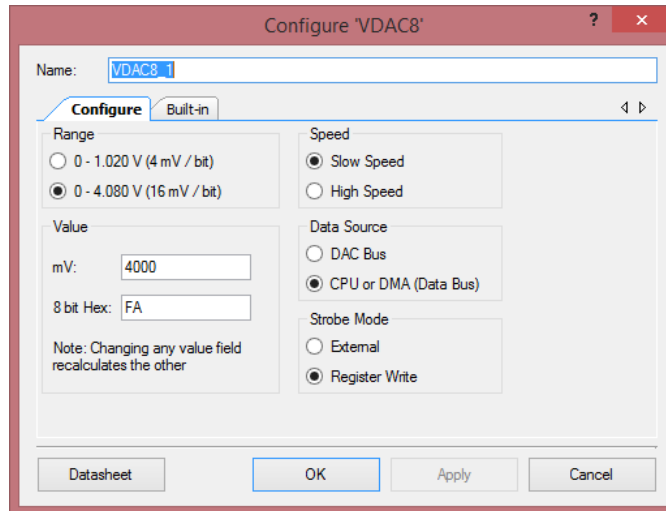


Figura 167. Configuración del VDAC

Ahora necesitamos un pin analógico y le cambiamos el nombre por POT.

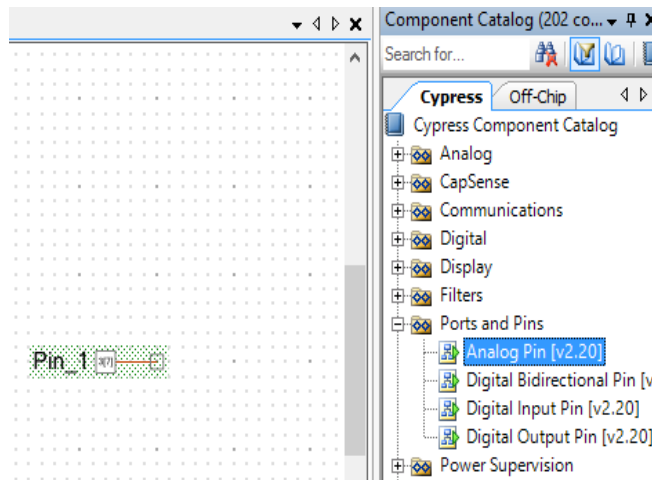


Figura 168. Ubicación de los pines analógicos en el catálogo

Ahora necesitamos una salida digital para representar nuestro LED. Los componentes quedarán configurados de la siguiente manera.

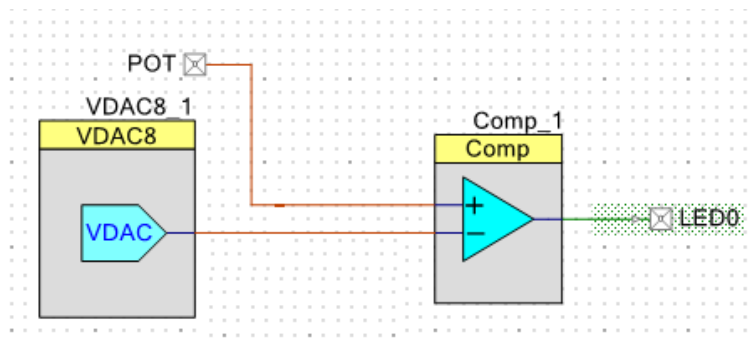


Figura 169. Conexión del VDAC y el comparador

Para el comparador 2 se repite el mismo procedimiento, con el voltaje de referencia 2 V, el esquema debe ser el siguiente:

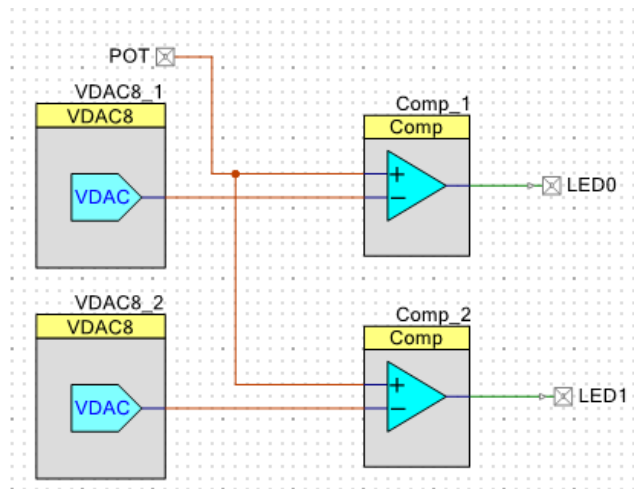


Figura 170. Conexión de todos los componentes

Luego de esto es importante guardar (1) y compilar (2) para asegurarnos de que no exista ningún error en nuestro proyecto.

3.2.12.4.2 Código del Main

En el lado izquierdo de la pantalla, nos vamos a la pestaña que dice main y damos doble click.

En el main iniciaremos todos los elementos que configuramos anteriormente, es decir, los 2 comparadores y los 2 VDAC.

```

#include <project.h>

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    Comp_1_Start();
    Comp_2_Start();
    VDAC8_1_Start();
    VDAC8_2_Start();
    for(;;)
    {
        /* Place your application code here. */
    }
}

```

Figura 171. Código para iniciar componentes

Luego de esto es importante guardar y compilar.

3.2.12.4.3 Asignación de pines

Los pines a utilizar son

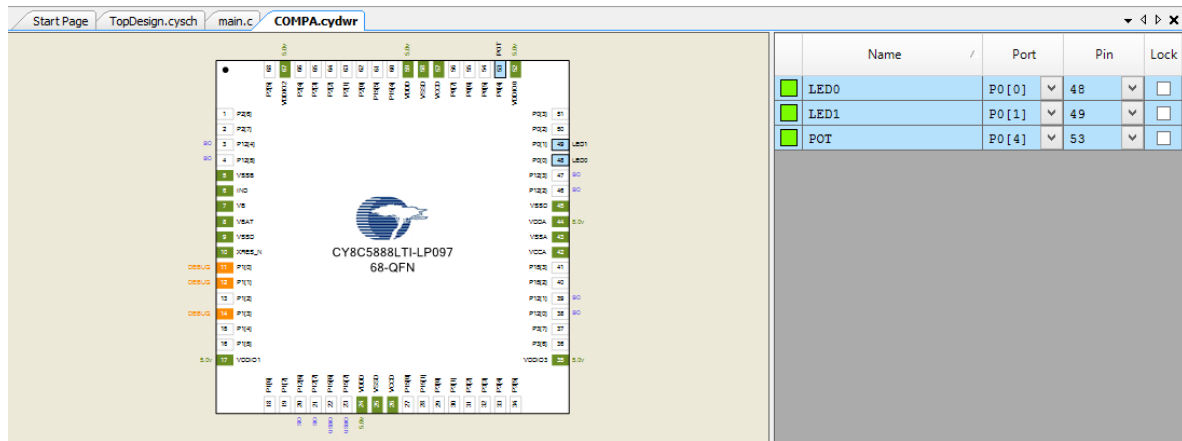


Figura 172. Pines asignados

Después de esto, guardamos, compilamos, generamos aplicación. y programamos.

3.2.13 PWM

El componente PWM ofrece comparación de salidas para generar señales únicas o continuas de temporización y control en hardware. Las características PWM se pueden combinar con otros componentes analógicos y digitales para crear periféricos personalizados.

3.2.13.1 Parámetros del componente

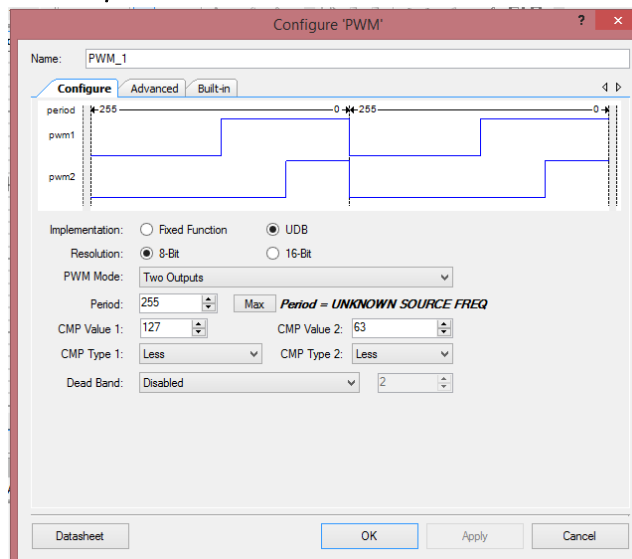


Figura 173. Parámetros del PWM

Implementación: Este parámetro le permite elegir entre una función fija y una implementación UDB del PWM.

Resolución: Define la resolución de bits.

Modo PWM: Define la funcionalidad global del PWM.

- **Una salida:** Sólo una única salida PWM. En este modo la salida PWM es visible
- **Dos de salida:** Dos salidas PWM configurables individualmente.
- **borde doble:** Una sola salida de doble filo creado por AND juntas las señales PWM1 y PWM2.
- **Centro Alinear:** Una sola salida alineado al centro creado por tener el contador de contar hasta el valor del período incrementa y volver a bajar a cero, mientras que la creación de un ancho de pulso alineado al centro basado en el valor de comparación.
- **Dither:** Una sola salida seleccionada de las dos señales internas PWM (PWM1 y PWM2) por una máquina de estados de hardware incluidos en la implementación de hardware PWM.
- **Hardware Select:** Una sola salida seleccionada de las dos señales PWM internos por un cmp_sel hardware pin de entrada.

Periodo (Software): Define el valor de partida inicial del contador y el valor cada vez que se alcanza la cuenta terminal y el modo PWM permite la recarga del contador de tiempo. El PWM se implementa como un contador descendente a contar desde el valor del período a cero.

CMP Value 1 / CMP Value2 (Software): Compara los valores que definen la funcionalidad de comparación de salida junto con el hardware comparar opciones de Tipo.

Banda muerta: El parámetro de banda muerta activa o desactiva la funcionalidad de banda muerta de la PWM.

Interfaz de programación de aplicaciones

Interfaz de programación de aplicaciones (API) de rutinas le permiten configurar el componente utilizando el software. La siguiente tabla muestra y describe la interfaz para cada función.

3.2.13.2 Objetivos

- Conocer la funcionalidad del bloque PWM de PSoC Creator.
- Configurar el bloque PWM de un canal y ver su variación en el LED de la tarjeta utilizando la interfaz de PSoC Creator.

3.2.13.3 Guía paso a paso

En esta práctica, utilizaremos un PWM para ver su variación en el LED de la tarjeta.

3.2.13.3.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos al catálogo de Cypress, digital, functions y escogemos PWM

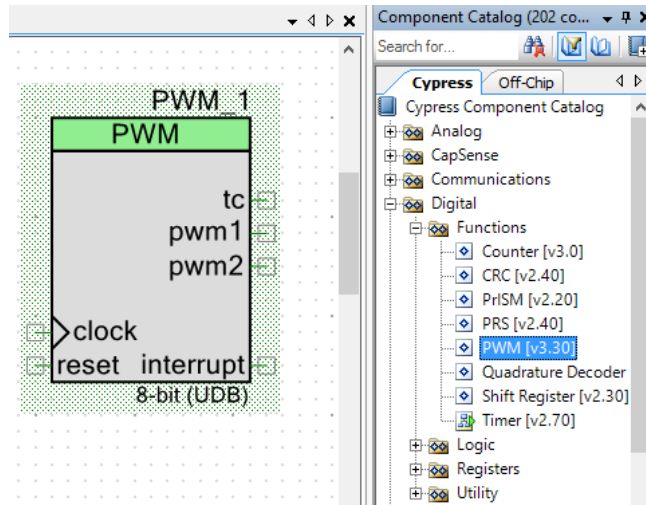


Figura 174. Ubicación del PWM en el catálogo

Hacemos doble click para configurar. Cambiamos el nombre por PWM, UDB de 16 bits, en el modo escogemos una sola salida, periodo 1000, y en el valor 1 colocamos 500.

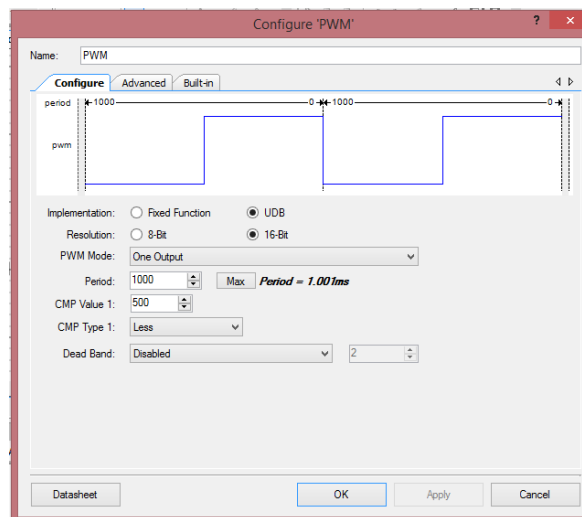


Figura 175. Configuración del PWM

Ahora necesitamos un reloj, vamos a system, arrastramos un clock y lo conectamos al pin clock del PWM. Hacemos doble click sobre el reloj y configuramos como un reloj de 1MHz.

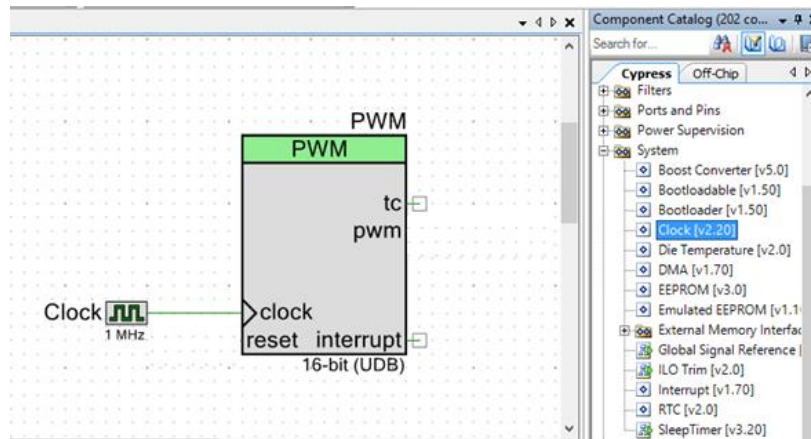


Figura 176. Ubicación y conexión del clock para el PWM

En el pin de reset conectamos un '0' lógico, para eso vamos, digital, logic y escogemos el elemento.

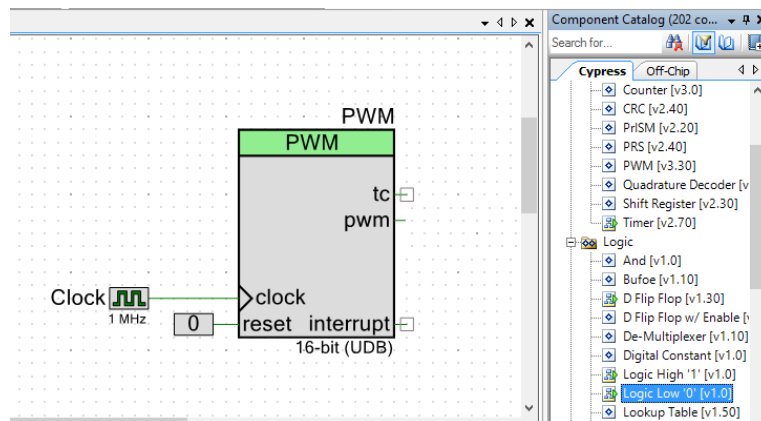


Figura 177. Ubicación y conexión del '0' lógico para el PWM

Para terminar, necesitamos una salida digital que será el LED e irá conectado al pin pwm.

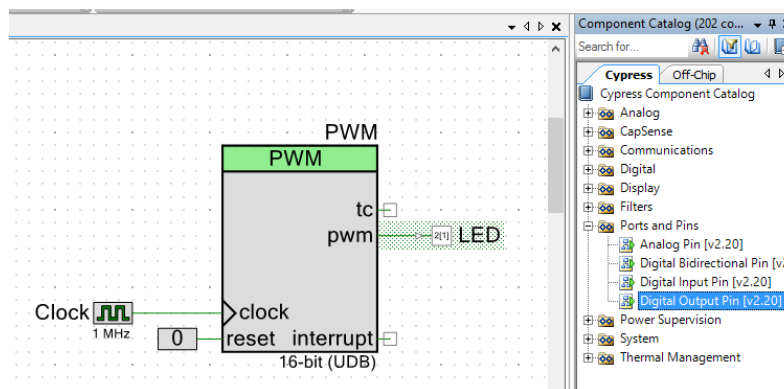


Figura 178. Ubicación y conexión de un pin digital de salida al PWM

Luego de esto es importante guardar y compilar.

3.2.13.3.2 Código del Main

En el main, iniciaremos el PWM, y en el for infinito escribiremos valor desde 100 hasta 900 con la instrucción PWM_WriteCompare(valor), y esperaremos por cada valor un tiempo de 1 segundo.

```
int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    PWM_Start();

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    for(;;)
    {
        PWM_WriteCompare(100);
        CyDelay(1000);
        PWM_WriteCompare(300);
        CyDelay(1000);
        PWM_WriteCompare(500);
        CyDelay(1000);
        PWM_WriteCompare(700);
        CyDelay(1000);
        PWM_WriteCompare(900);
        CyDelay(1000);
    }
}
```

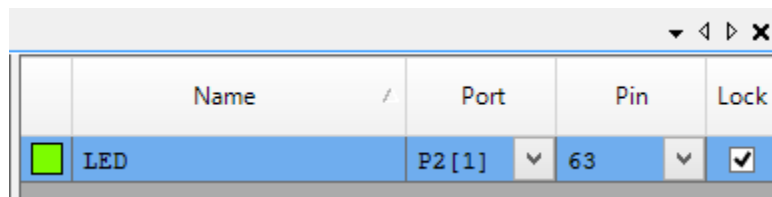
Figura 179. Código para establecer valor del PWM

Luego de esto es importante guardar y compilar.

3.2.13.3.3 Asignación de pines

La asignación de pines es sencilla ya que solo debemos declarar el pin del LED.

El pin que se escogió fue



	Name	Port	Pin	Lock
<input checked="" type="checkbox"/>	LED	P2[1]	63	<input checked="" type="checkbox"/>

Figura 180. Pines adignados.

Después de esto, guardamos, compilamos, generamos aplicación y programamos

3.2.14 Control de velocidad de un motor DC con PWM

En esta práctica vamos a variar la velocidad de un motor DC con un PWM. Para eso vamos a utilizar las instrucciones CompareValue para variar el voltaje promedio que le llega al motor, de esa manera se puede variar la velocidad del motor.

3.2.14.1 Objetivos

- Controlar la velocidad de un motor DC usando el bloque PWM de PSoC PSoC Creator y ver su variación en una pantalla LCD.

3.2.14.2 Guía paso a paso

3.2.14.2.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Para empezar, debemos añadir la LCD.

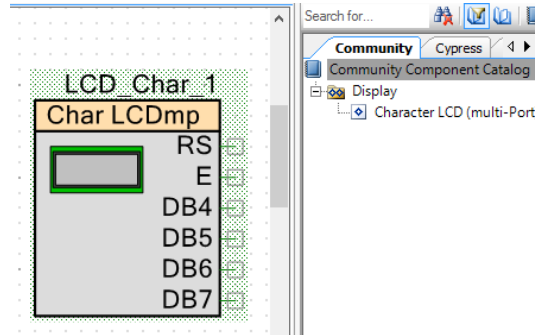


Figura 181. Ubicación del bloque LCD

Lo que sigue es asignar los pines, para eso utilizamos pines de salida digital y le cambiamos los nombres para la comodidad a la hora de asignar los pines.

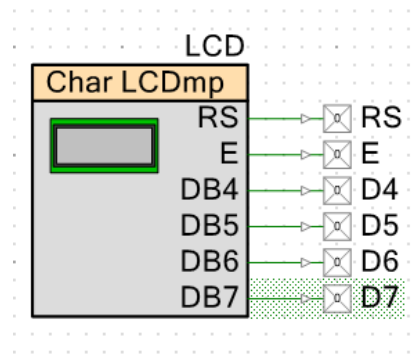


Figura 182. Conexión de pines con la LCD

Ahora necesitamos añadir un PWM. Vamos al catálogo, Digital, functions y arrastramos el PWM.

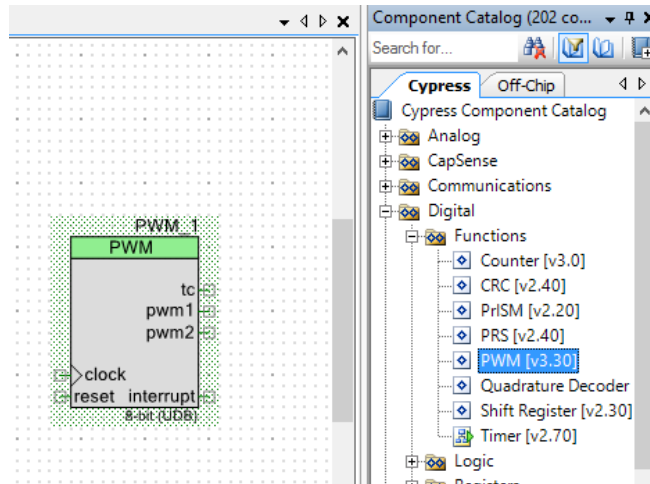


Figura 183. Ubicación del PWM en catálogo

Doble click para configurar, cambiamos el nombre por PWM y en el modo escogemos solo una salida. En el compare Value 1 colocamos 22, esto nos da un periodo de casi 25.6mS. Esto quiere decir que desde el momento en que empieza el ciclo hasta que termina hay 25.6mS y esto lo de tiempo al motor de que quede un tiempo parado y un tiempo andando.

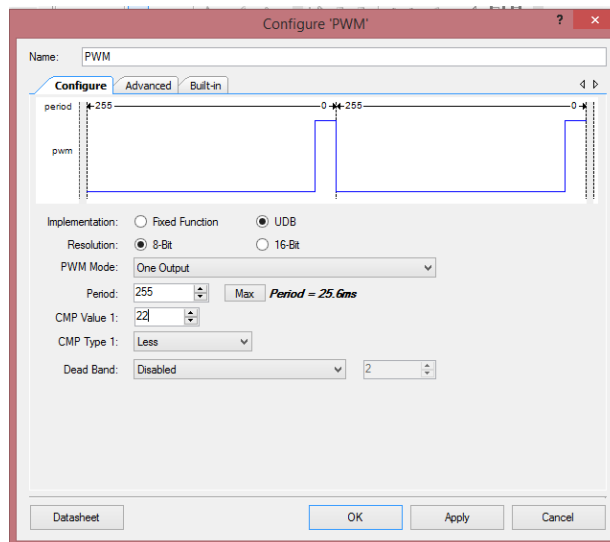


Figura 184. Configuración del PWM

En este caso es necesario reiniciar nuestro PWM, ya que el ciclo útil del PWM estará cambiando. Para ello voy a conectar un pin de entrada digital al reset. Y será configurado como pull down resistivo. Llevará por nombre Resetpin.

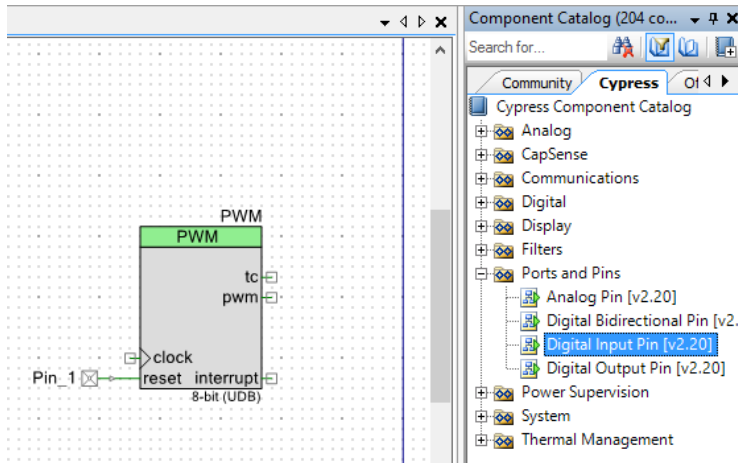


Figura 185. Ubicación y conexión del pin digital de entrada al PWM

Como no se puede escribir directamente en un pin digital de entrada, lo que va a hacer es sacar una salida digital, se cambia el nombre por ResetPWM, quitarle la conexión HW. Y estos dos pines se van a puentear externamente (Resetpin y ResetPWM).

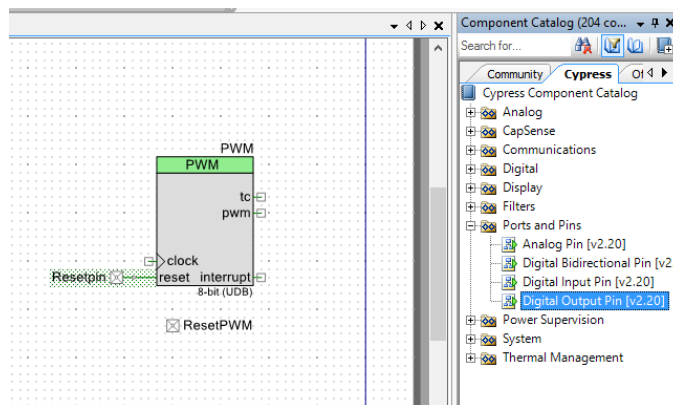


Figura 186. Ubicación de un pin digital de salida

Ahora necesitamos un reloj, vamos a system y arrastramos el clock.

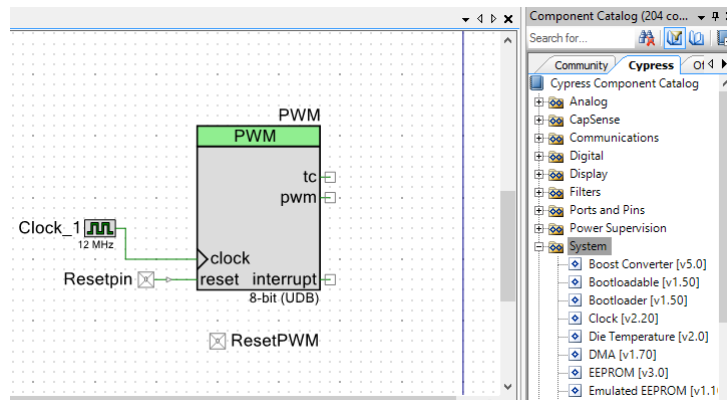


Figura 187. Ubicación y conexión del reloj para el PWM

Hacemos doble click para configurar, lo colocamos de 10kHz.

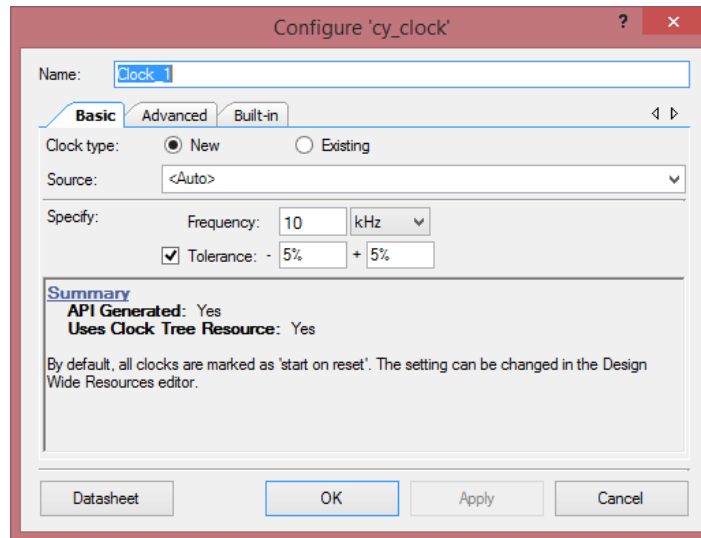


Figura 188. Configuración del reloj del PWM

Lo que sigue es colocar pin de salida digital en el pin pwm de nuestro PWM para poder observar las variaciones. Y se llamará motor.

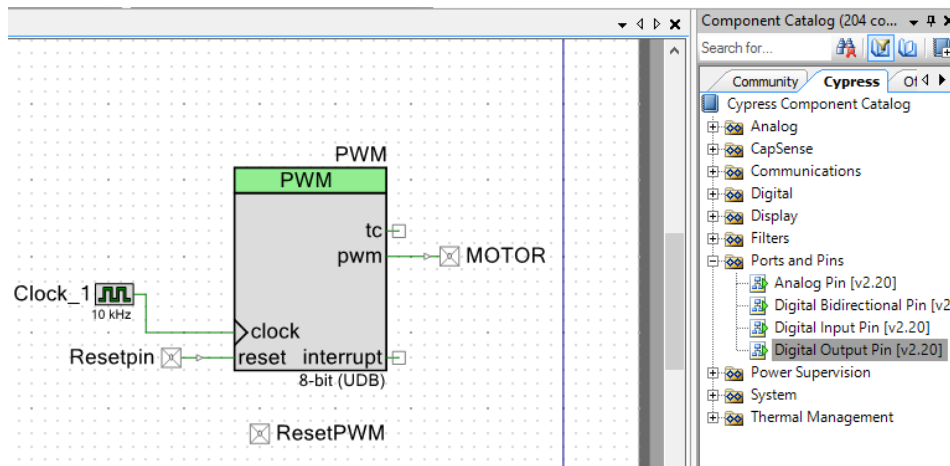


Figura 189. Ubicación y conexión de un pin de salida para visualizar el PWM

Ahora necesitamos dos botones para controlar la velocidad del motor, un botón será el pulsador de la tarjeta y otro pulsador externo. Para ellos necesitamos dos pines de entrada digital, llevará el nombre de AUMENTAR y será un pull up resistivo y el otro llevará el nombre de DISMINUIR y será un pull down resistivo.

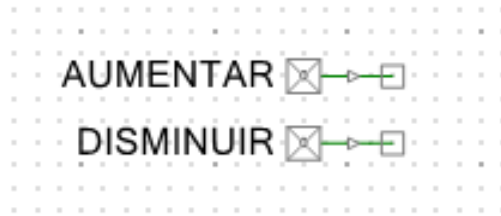


Figura 190. Pines de entrada para cambiar el valor del PWM

Ahora para leer estos pines necesitamos las interrupciones. Vamos a system y sacamos las dos interrupciones, los nombres serán isrA e isrD, ambos en modo RISING_EDGE. Como el pin de la tarea tiene un pull up y la interrupción esta por flanco de subida, hay una contradicción y no se dará lo que esperamos. Para eso utilizamos una compuerta not para que la lógica sea normal.

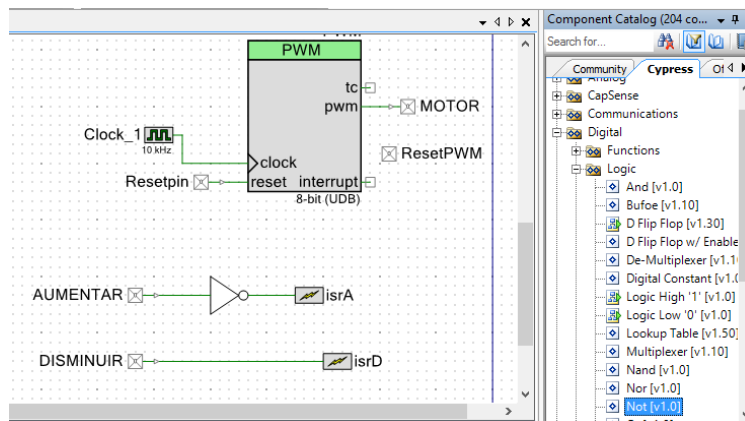


Figura 191. Conexión de los pines de entradas con las interrupciones.

Ahora necesitamos utilizar el LED de la tarjeta. Vamos a ports and pins, sacamos una salida digital y en la configuración le quitamos la conexión HW. el LED será un indicador para saber que si se están ejecutando las interrupciones.

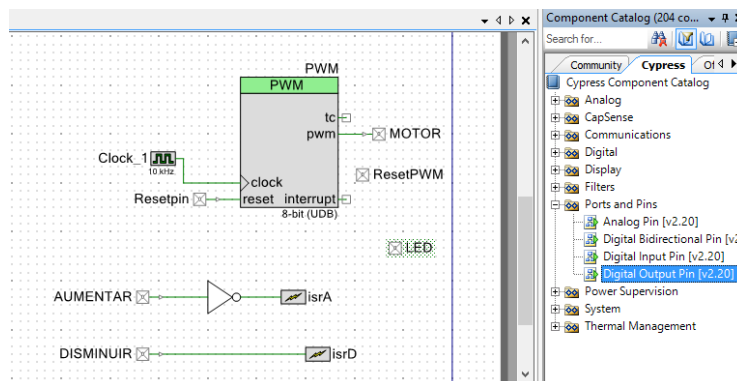


Figura 192. Pin de salida (LED).

Guardamos y compilamos.

3.2.14.2.2 Asignación de Pines

Vamos al icono para asignar pines

Los pines que yo utilicé son los siguientes, ud puede utilizar los que desee siempre y cuando use los indicados para el pulsador y el LED.

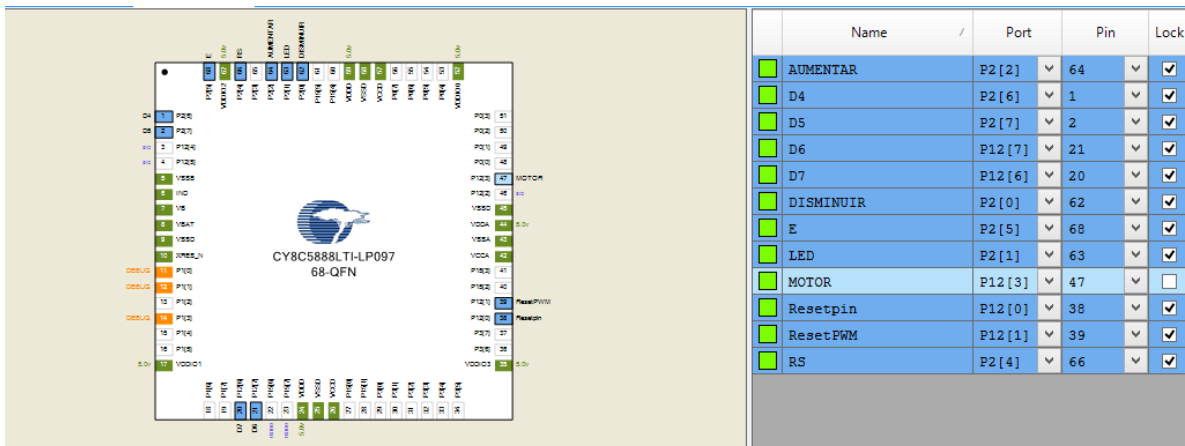


Figura 193. Pines asignados.

3.2.14.2.3 Código del Main

Primero que todo vamos a declarar las funciones de las interrupciones. En la función de aumentar debemos tener una variable que es la que se va aumentar para variar la velocidad del motor. Esta misma variable es la que se va a disminuir en la otra función, por lo que debe ser global.

En la primera función, se apaga el LED, e iremos aumentando la velocidad del motor. Como el periodo de 255 es mi 100%, iremos aumentando nuestra variable de 1 en 1 hasta llegar a 10, y el aumento de cada unidad representará el aumento del 10% en el periodo del motor. Ejemplo: variable=1, periodo=10% (25). El porcentaje de la velocidad se podrá ver en la LCD.

```
#include <project.h>

int CONTROL=1;

CY_ISR(Aumentar)
{
    LED_Write(0);
    if (CONTROL<=10)
    {
        CONTROL=CONTROL+1;
        if (CONTROL==1)
        {
            PWM_WriteCompare(25); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("10%");
        }
    }
    if (CONTROL==2)
    {
```

```

        PWM_WriteCompare(51); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("20%");
    }
    if(CONTROL==3)
    {
        PWM_WriteCompare(76); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("30%");
    }
    if(CONTROL==4)
    {
        PWM_WriteCompare(102); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("40%");
    }
    if(CONTROL==5)
    {
        PWM_WriteCompare(127); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("50%");
    }
    if(CONTROL==6)
    {
        PWM_WriteCompare(152); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("60%");
    }
    if(CONTROL==7)
    {
        PWM_WriteCompare(178); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("70%");
    }
    if(CONTROL==8)
    {
        PWM_WriteCompare(204); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("80%");
    }
    if(CONTROL==9)
    {
        PWM_WriteCompare(229); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("90%");
    }
    if(CONTROL==10)
    {
        PWM_WriteCompare(255); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("100%");
    }
    ResetPWM_Write(1);
    ResetPWM_Write(0);
}
}

```

En la segunda función se prende el LED e iremos disminuyendo la velocidad.

```

CY_ISR(Disminuir)
{
    LED_Write(1);
    if(CONTROL>1)
    {
        CONTROL=CONTROL-1;
        if(CONTROL==1)
        {
            PWM_WriteCompare(25); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("10%");
        }
        if(CONTROL==2)
        {
            PWM_WriteCompare(51); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("20%");
        }
        if(CONTROL==3)
        {
            PWM_WriteCompare(76); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("30%");
        }
        if(CONTROL==4)
        {
            PWM_WriteCompare(102); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("40%");
        }
        if(CONTROL==5)
        {
            PWM_WriteCompare(127); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("50%");
        }
        if(CONTROL==6)
        {
            PWM_WriteCompare(152); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("60%");
        }
        if(CONTROL==7)
        {
            PWM_WriteCompare(178); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("70%");
        }
        if(CONTROL==8)
        {
            PWM_WriteCompare(204); //CICLO UTIL DEL PWM
            LCD_Position(1,5);
            LCD_PrintString("80%");
        }
        if(CONTROL==9)
        {
            PWM_WriteCompare(229); //CICLO UTIL DEL PWM

```

```

        LCD_Position(1,5);
        LCD_PrintString("90%");
    }
    if(CONTROL==10)
    {
        PWM_WriteCompare(255); //CICLO UTIL DEL PWM
        LCD_Position(1,5);
        LCD_PrintString("100%");
    }
    ResetPWM_Write(1);
    ResetPWM_Write(0);
}
}

```

En el MAIN simplemente inicializamos variables.

```

int main()
{
    PWM_Start();
    LCD_Start();
    Clock_1_Start();

    LCD_Position(0,0);
    LCD_PrintString("VELOCIDAD MOTOR");

    CyGlobalIntEnable; /* Enable global interrupts. */
    isrA_StartEx(Aumentar);
    isrD_StartEx(Disminuir);

    for(;;)
    {
        /* Place your application code here. */
    }
}

```

Guardamos, compilamos, generamos aplicación y programamos.

3.2.15 UART

El UART proporciona comunicaciones asíncronas comúnmente como RS232 o RS485. El componente UART puede ser configurado para dúplex, semidúplex, solamente RX, TX o sólo versiones.

3.2.15.1 Objetivos

- Conocer la funcionalidad del bloque UART de PSoC Creator.
- Establecer comunicación serial entre la PSoC y el PC utilizando el programa 232Analyzer.

3.2.15.2 Parámetros del componente

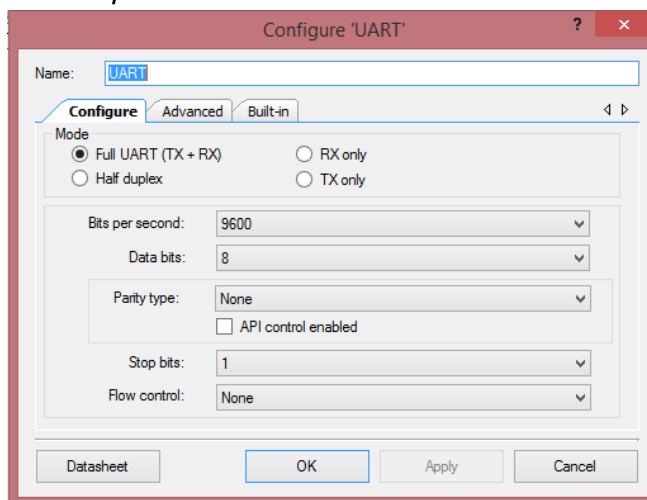


Figura 194. Parámetros de UART

Modo: Este parámetro define los componentes funcionales que desee incluir en el UART. Esto puede ser configurado para ser un UART bidireccional (TX + RX) (por defecto), Half Duplex (utiliza la mitad de los recursos), RS232 Receptor (Sólo recepción) o transmisor (TX solamente).

Bits por segundo: Este parámetro define la configuración de velocidad de transmisión o anchura de bit del hardware para la generación de reloj. El valor predeterminado es 57600.

Bits de datos: Este parámetro define el número de bits de datos transmitidos entre el inicio y la parada de una sola transacción UART.

- Ocho bits de datos es la configuración por defecto, el envío de un byte por transferencia.
- El modo de 9 bits no transmite 9 bits de datos; el noveno bit toma el lugar del bit de paridad como un indicador de dirección utilizando la marca de paridad/Espacio. Marcar de paridad/espacio se debe seleccionar si se utiliza el modo de 9 bits de datos.

Tipo de paridad: Este parámetro define la funcionalidad de la ubicación bit de paridad en la transferencia.

Control API habilitada: Esta casilla de verificación se utiliza para cambiar la paridad mediante el registro de control y la función `UART_WriteControlRegister()`.

Bits de parada: Este parámetro define el número de bits de parada implementados en el transmisor. Este parámetro se puede ajustar a 1 (por defecto) o 2 bits de datos.

Control de flujo: Este parámetro le permite elegir entre Hardware o Ninguno (predeterminado).

3.2.15.3 Interfaz de programación de aplicaciones

Interfaz de programación de aplicaciones (API) de rutinas le permiten configurar el componente utilizando el software. La siguiente tabla muestra y describe la interfaz para cada función.

3.2.15.4 Guía paso a paso

Para esta práctica utilizaremos el módulo UART que trae la PSoC, para hacer la comunicación serial se utilizará el programa 232Analyzer que permite hacer uso de los puertos seriales. Lo haremos será: cuando el PC envíe un '1' y la PSoC lo reciba, se prenderá el LED y la PSoC enviará la letra R. Cuando el PC envíe un '0' y la PSoC lo reciba, el LED se apagará.

3.2.15.4.1 Configuración de Hardware

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos a communications, SPI y seleccionamos UART.

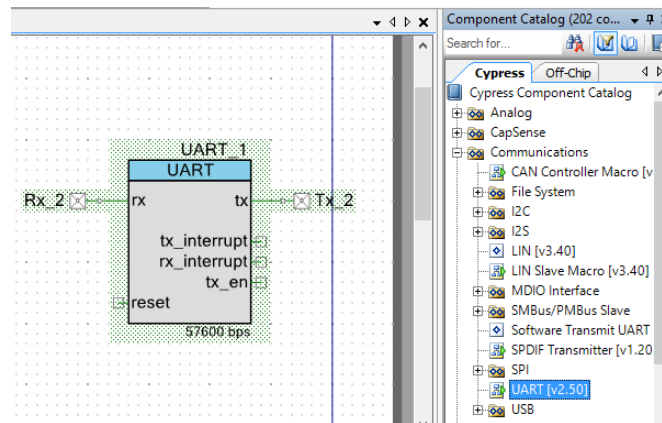


Figura 195. Ubicación del componente UART

Hacemos doble click para configurar. Cambiaremos el nombre por UART, el modo en que trabajará será Full UART, a una velocidad de 9600. Aplicar y aceptar.

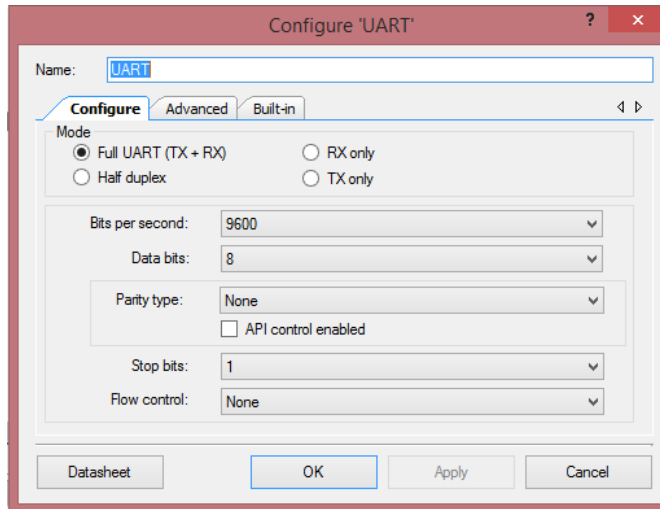


Figura 196. Configuración del UART

Es importante que en la pestaña advanced esté seleccionada la casilla RX-On Byte Received. Ya que va a tener la interrupción por byte recibido.

Lo siguiente es añadir un cero lógico al reset del UART. Para eso vamos a digital, logic y escogemos el elemento deseado.

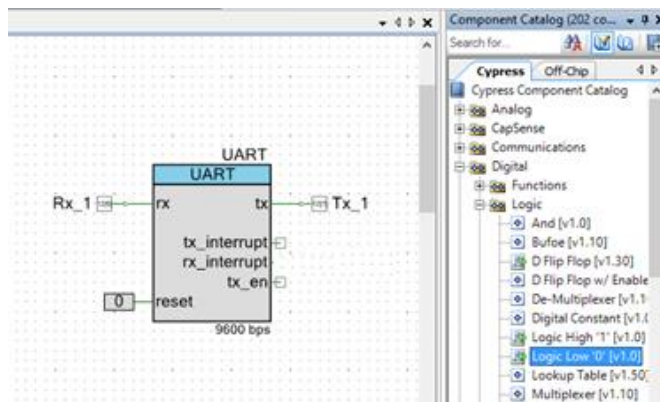


Figura 197. Ubicación y conexión de un '0' lógico

Ahora necesitamos una interrupción, vamos a system y la arrastramos. Esta interrupción llevará el nombre de isrRx e irá en el pin rx_interrupt del UART. La interrupción se usa para recibir datos, ya que cuando haya un bit de entrada de dato se vaya hacia una interrupción que será la que me recibirá los datos.

Para terminar con la configuración del hardware, necesitamos un pin de salida digital que será nuestro LED.

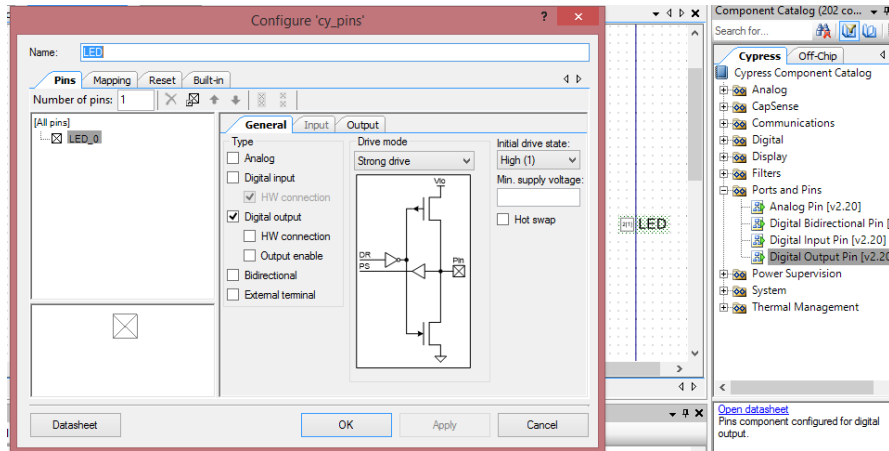


Figura 198. Configuración y ubicación del pin digital de salida

Luego de esto es importante guardar y compilar.

3.2.15.4.2 Código del Main

En el lado izquierdo de la pantalla, nos vamos a la pestaña que dice main y damos doble click.

Lo primero que se debe hacer es definir la función de la interrupción (CY_ISR). Aquí irá toda la rutina de recepción de datos, para ello necesitamos una variable tipo char, ya que todo lo que envío por el pc lo voy a enviar como un carácter en código ascii. Para recibir datos se utiliza la instrucción UART_GetChar(). Y nuestra función quedaría de la siguiente manera.

```

#include <project.h>

char datorecibido;

CY_ISR (INTERRUPTRX)
{
    datorecibido=UART_GetChar(); //Recibir dato
}

```

Figura 199. Código para la función CY_ISR

Ahora en el main, se va a iniciar el UART y la función de la interrupción.

```

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    isrRX_StartEx(INTERRUPTRX);

    UART_Start();
}

```

Figura 200. Código para iniciar componentes

En el for infinito, va nuestra condición, si se recibe un '1', se prende el LED de la tarjeta y se envía la letra R. Si se recibe un '0' se apaga el LED.

```

for(;;)
{
    if(datorecibido=='1')
    {
        LED_Write(1);
        UART_PutChar('R'); //Enviar dato
    }
    if(datorecibido=='0')
    {
        LED_Write(0);
    }
}

```

Figura 201. Código para establecer comunicación serial mediante el bloque UART

Luego de esto es importante guardar y compilar

3.2.15.4.3 Asignación de pines

Al momento de asignar los pines es importante utilizar los pines indicados por la PSoC para Rx t Tx

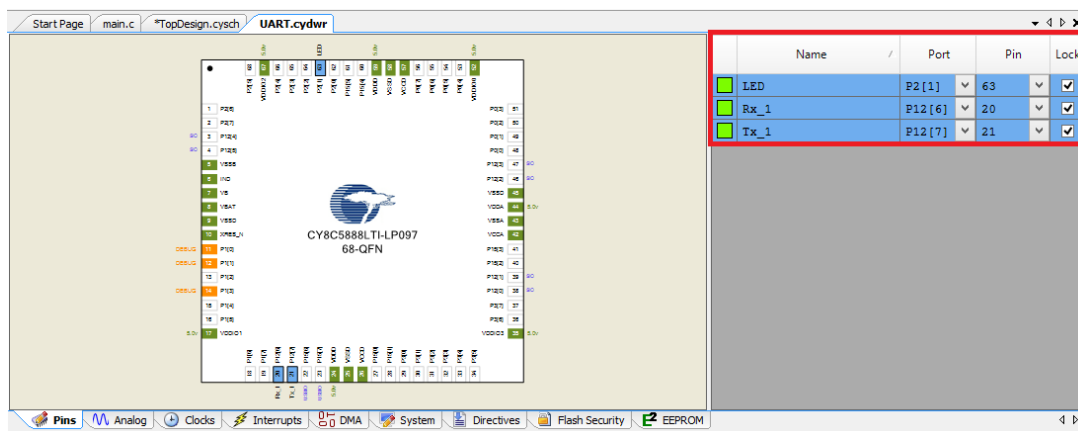


Figura 202. Pines asignados

Después de esto, guardamos, compilamos, generamos aplicación y programamos.

3.2.16 LUT

Puede configurar el componente de tabla de búsqueda (LUT) para realizar cualquier función lógica con un máximo de cinco entradas y ocho salidas.

3.2.16.1 Objetivos

- Conocer la funcionalidad del bloque Lookup Table (LUT) de PSoC Creator.
- Resolver un ejercicio utilizando el bloque LUT.

3.2.16.2 Parámetros del componente

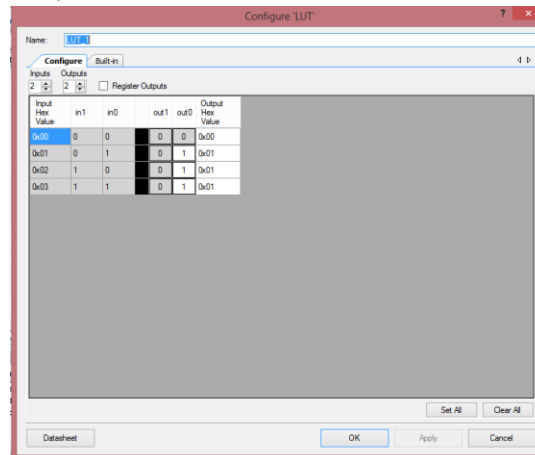


Figura 203. Parámetros del LUT

Opciones de configuración de hardware: La LUT puede configurar todas sus salidas para todas las posibles combinaciones de entrada.

Opciones de configuración de software: La LUT es un bloque de hardware único y por lo tanto no tiene ninguna opción de configuración de software.

Configuración por defecto: La primera vez que instancian, la LUT por defecto está configurado con dos entradas y dos salidas.

3.2.16.3 Guía a paso a paso

Para probar el funcionamiento de la LUT e irnos familiarizando con ella, desarrollaremos un problema sencillo de lógica digital. El problema trata de realizar un multiplicador, que solo permita multiplicar, máximos dos números binarios que sean de dos dígitos máximo.

3.2.16.3.1 Configuración de Hardware.

- Abrir PSoC Creator y crear un nuevo proyecto.
- Vamos al catálogo de Cypress, Digital, Logic y escogemos Lookup Table.

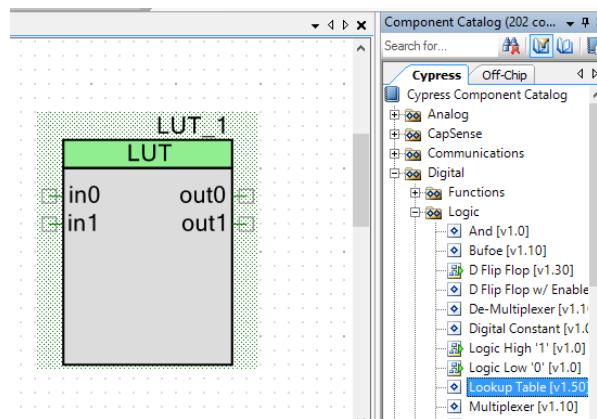


Figura 204. Ubicación del componente LUT

Hacemos doble click para configurar las entradas, salidas y nuestra tabla de verdad. Aplicar y aceptar.

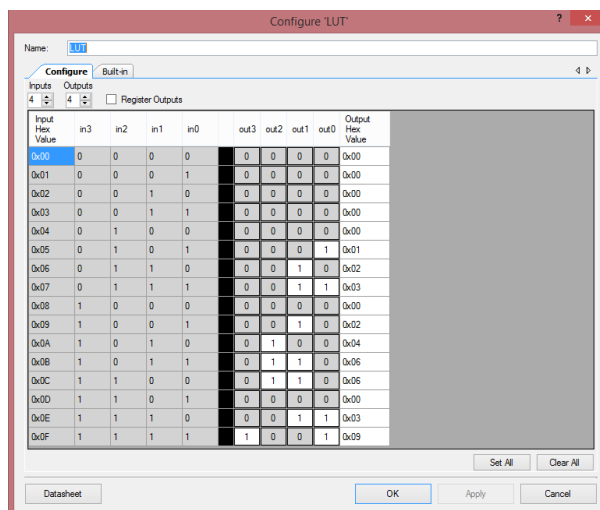


Figura 205. Configuración de la LUT

Lo que sigue es conectar los pines digitales de entrada y salida.

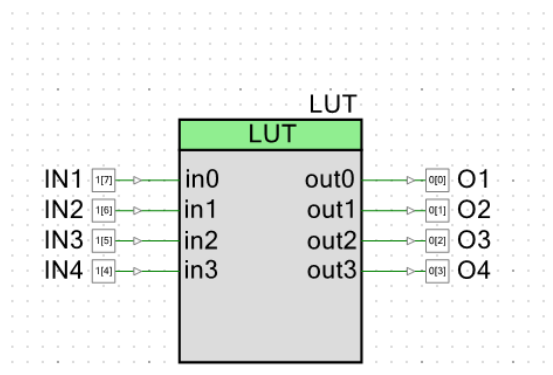


Figura 206. Conexión de pines de entrada y salida con el bloque LUT

Los pines de salida solo se les cambia el nombre y se deja la configuración que traen por defecto. A los pines de entrada se les cambia el nombre y se configuran como pull up resistivo.

3.2.16.3.2 Asignación de pines

Los pines que se utilizaron fueron los siguientes.

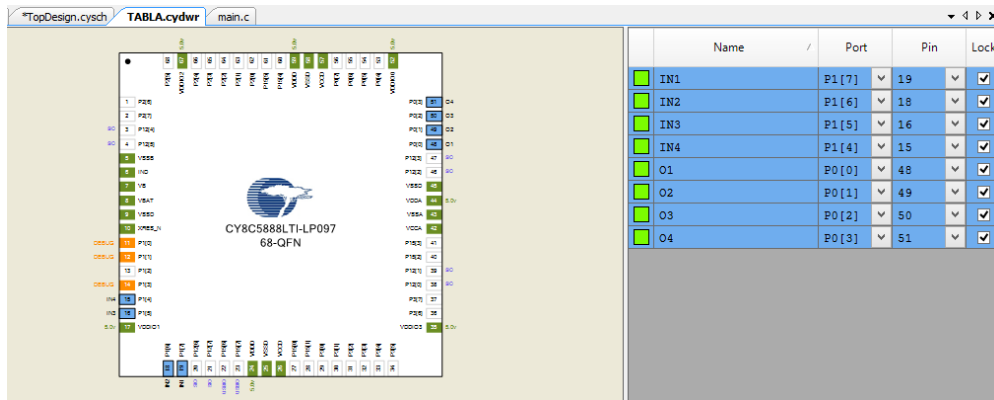


Figura 207. Pines asignados

Después de esto, guardamos, compilamos, generamos aplicación y programamos.

3.3 DISPOSITIVO HARDWARE PARA PSoC 5LP

Desarrollar un dispositivo hardware para las prácticas de laboratorio.

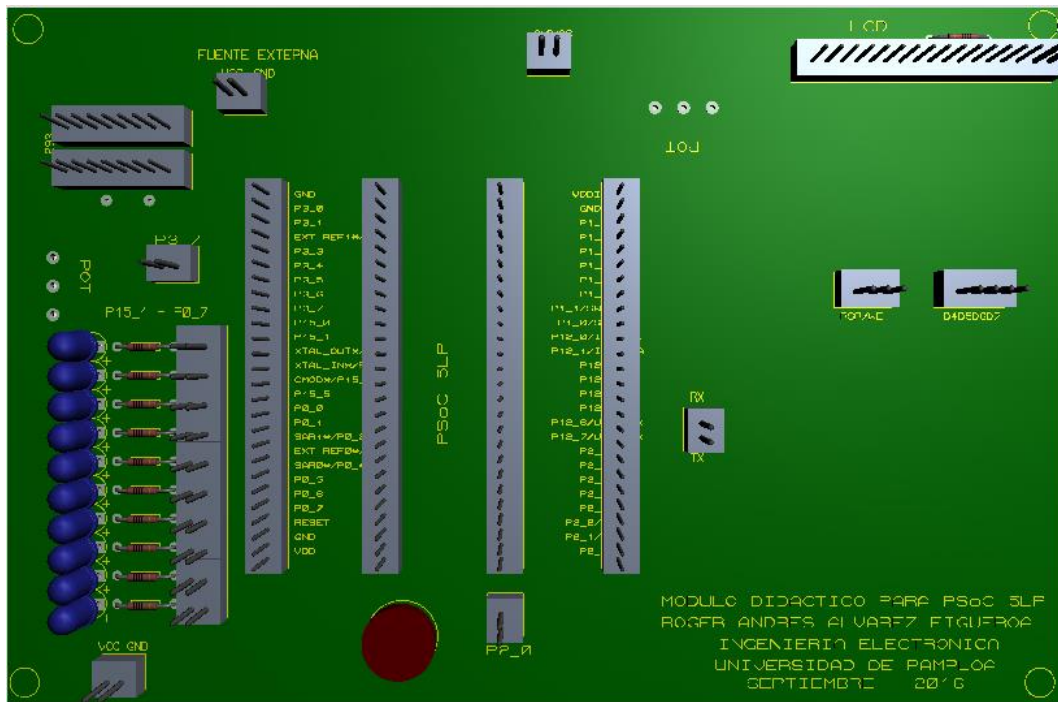


Figura 208. Esquema de componentes

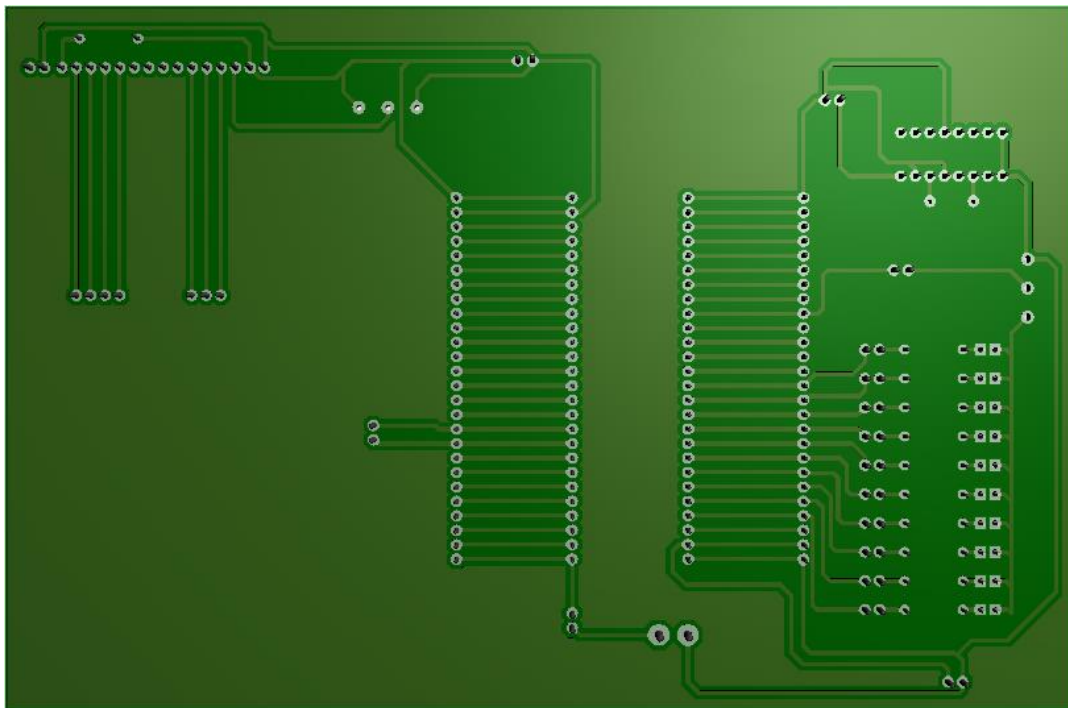


Figura 209. Esquema de conexiones

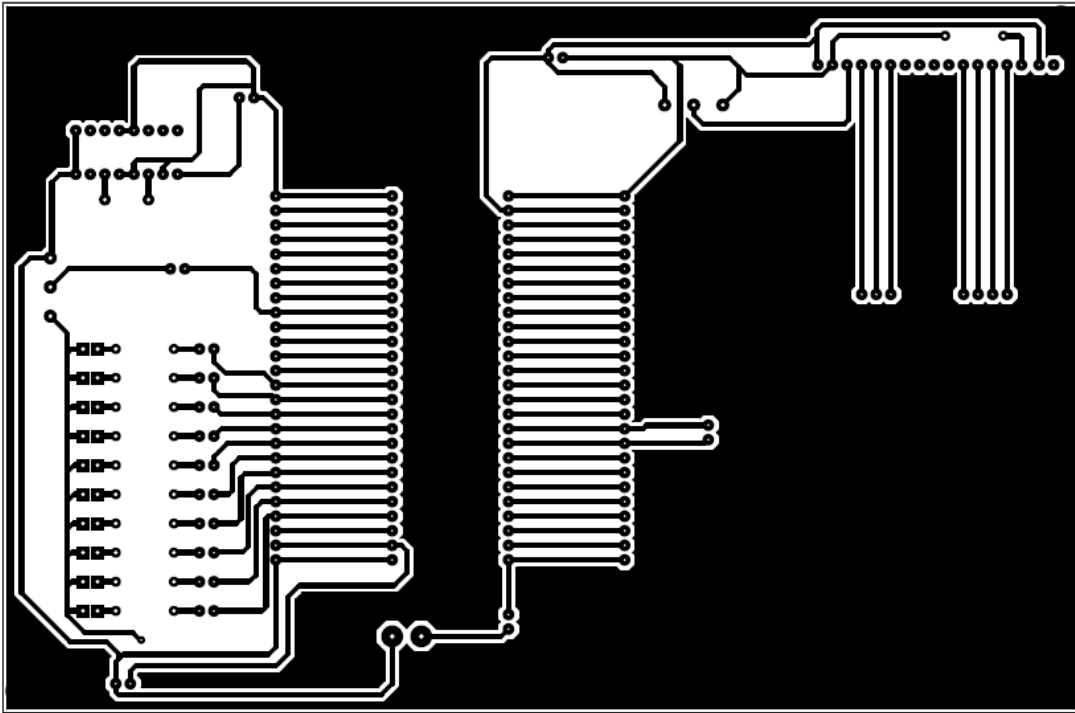


Figura 210. Diseño del dispositivo hardware

4 RESULTADOS

4 Resultados	128
4.1 Pines digitales de salida	129
4.2 Pines digitales de entrada	130
4.3 Manejo de interrupciones	130
4.4 Contador	131
4.5 Timer	132
4.6 Contador y Timer	132
4.7 Manejo de LCD	133
4.8 ADC Delta Sigma	135
4.9 DAC de Voltaje	135
4.10 ADC Delta Sigma y VDAC.....	136
4.11 Amplificador Operacional	136
4.12 Comparadores.....	137
4.13 PWM	137
4.14 Control de Velocidad de un motor usando PWM.....	138
4.15 UART	139
4.16 LUT	140

4.1 PINES DIGITALES DE SALIDA

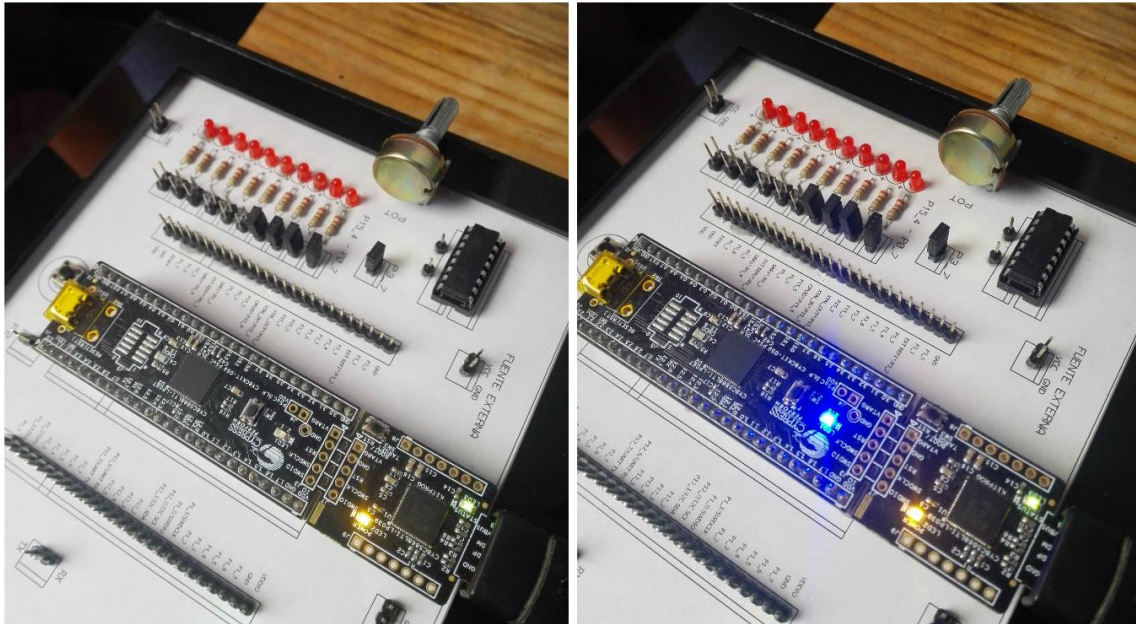


Figura 211. Diferentes estados de las salidas

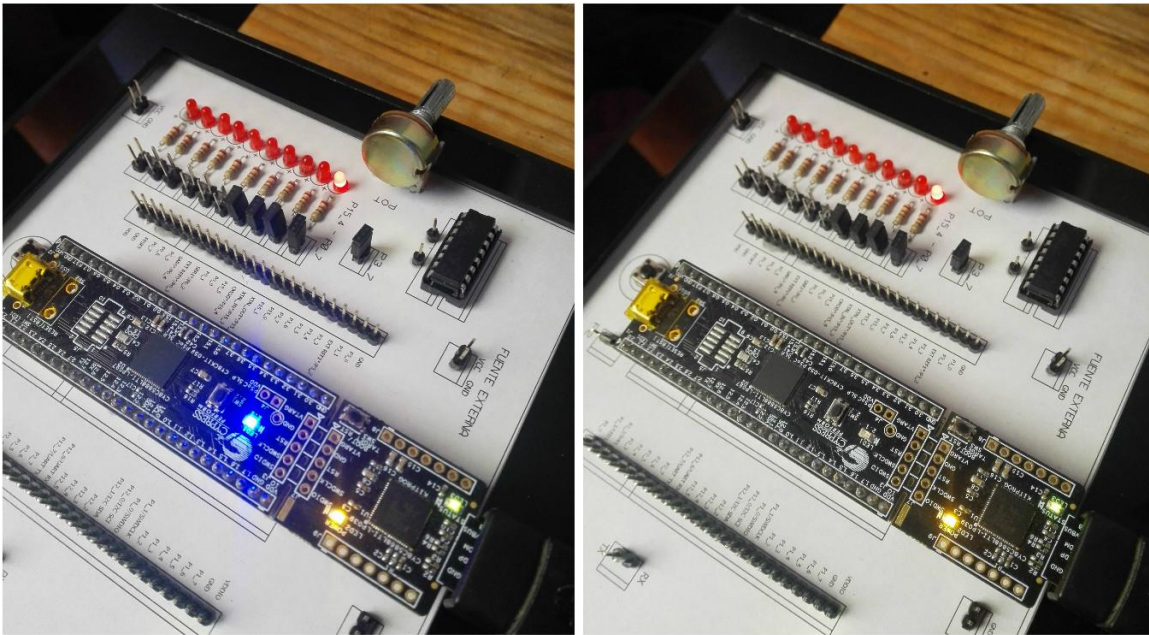


Figura 212. Diferentes estados de las salidas

4.2 PINES DIGITALES DE ENTRADA

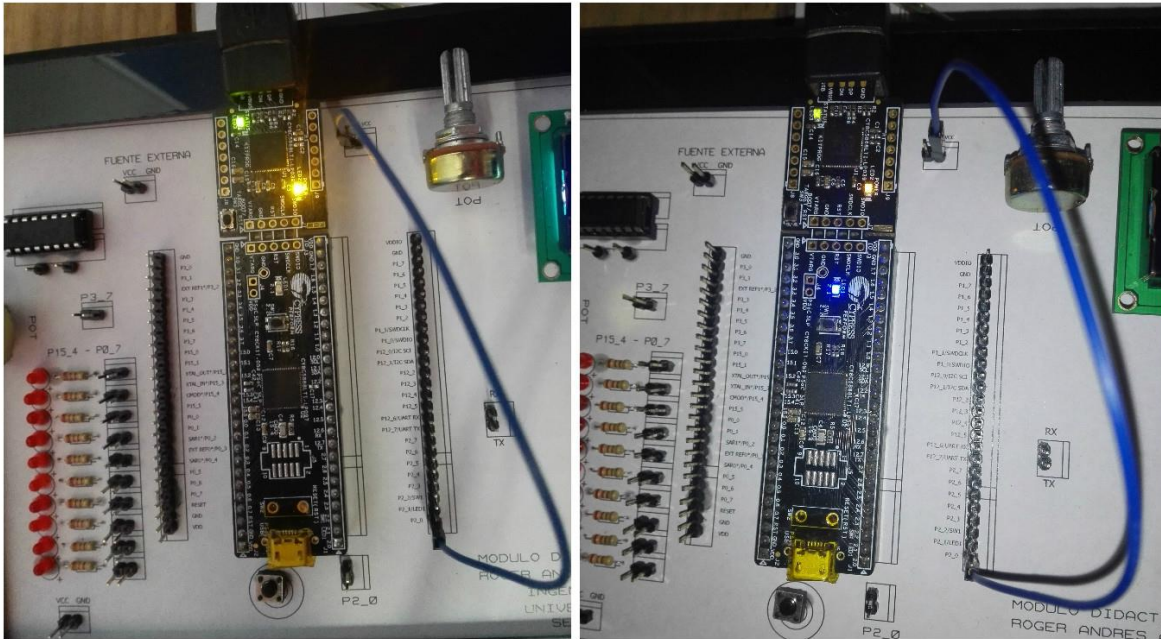


Figura 213. Pines digitales de entrada

4.3 MANEJO DE INTERRUPCIONES

Al pulsar el botón de la tarjeta se encenderá el LED.

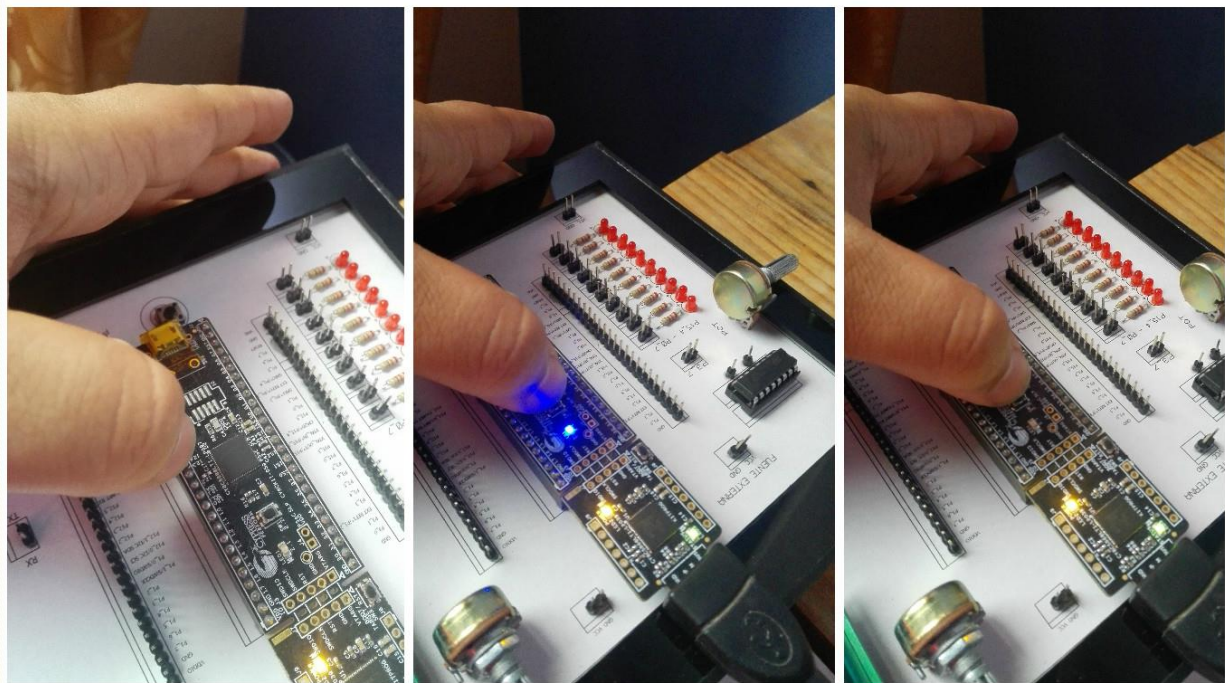


Figura 214. Manejo de interrupciones

4.4 CONTADOR

Después de dos interrupciones prende el LED de la tarjeta, y para apagarlo habrá que pulsar nuevamente en dos ocasiones.

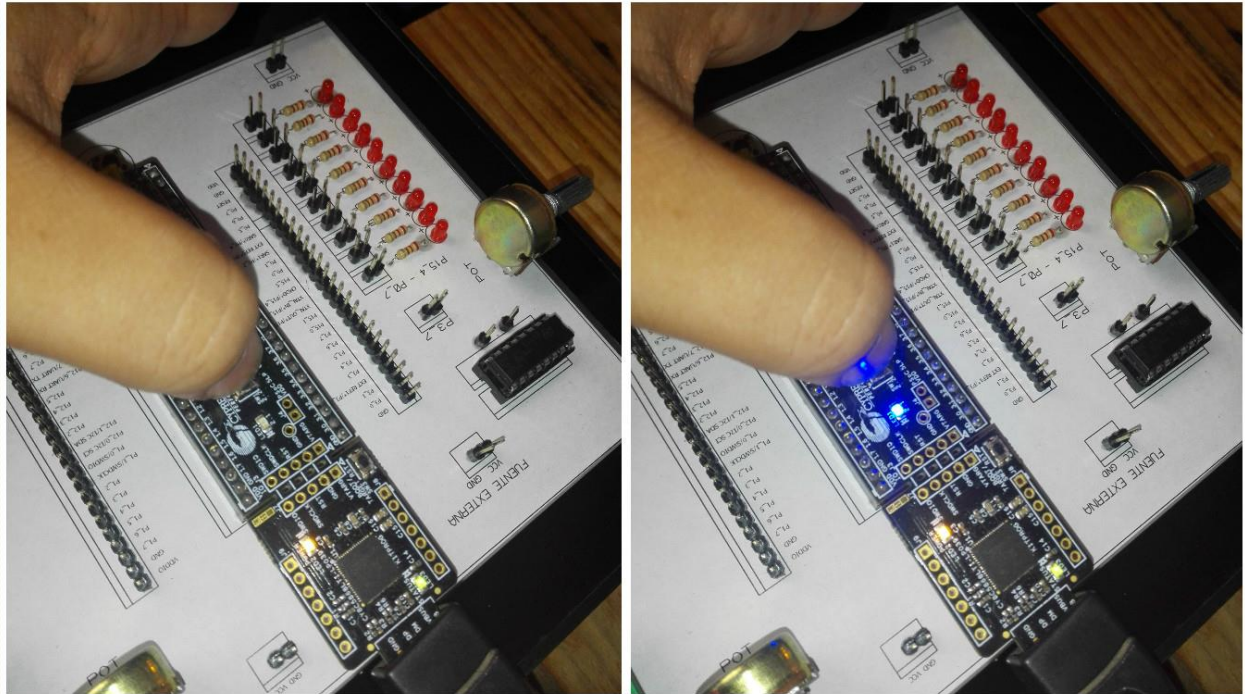


Figura 215. Funcionamiento del Contador

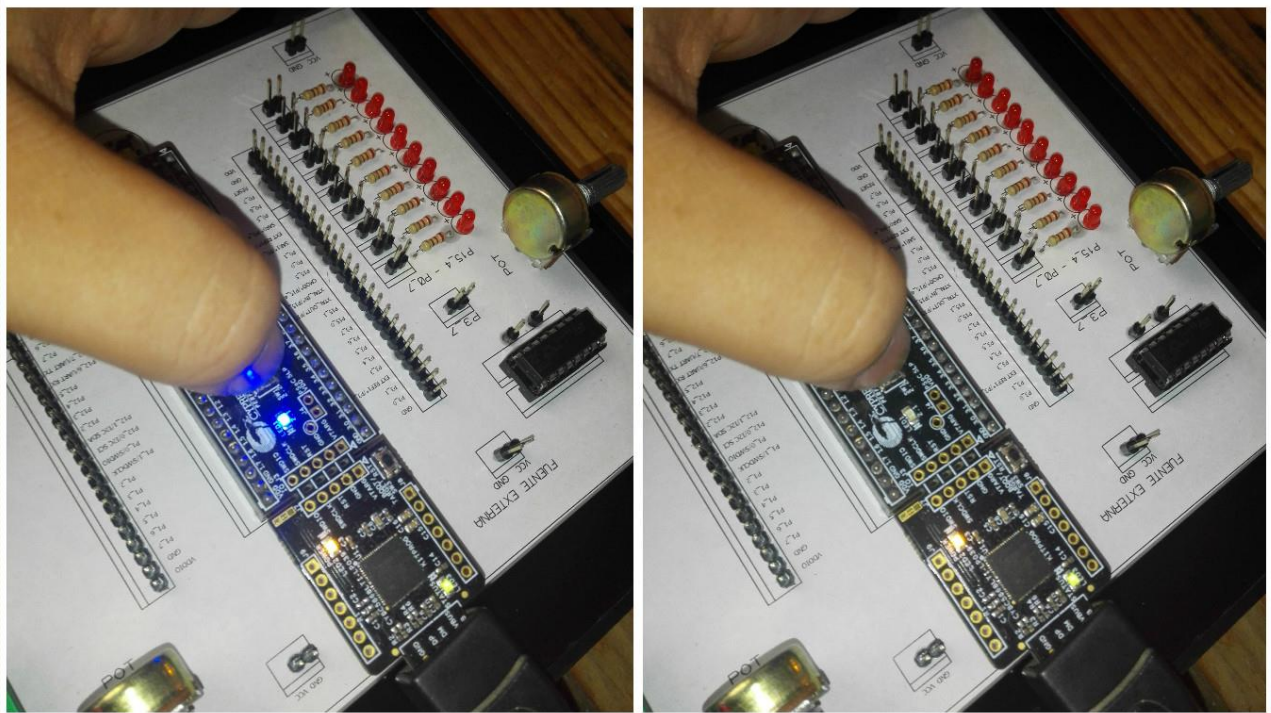


Figura 216. Funcionamiento del Contador

4.5 TIMER

El LED prende por un tiempo de 4 segundos

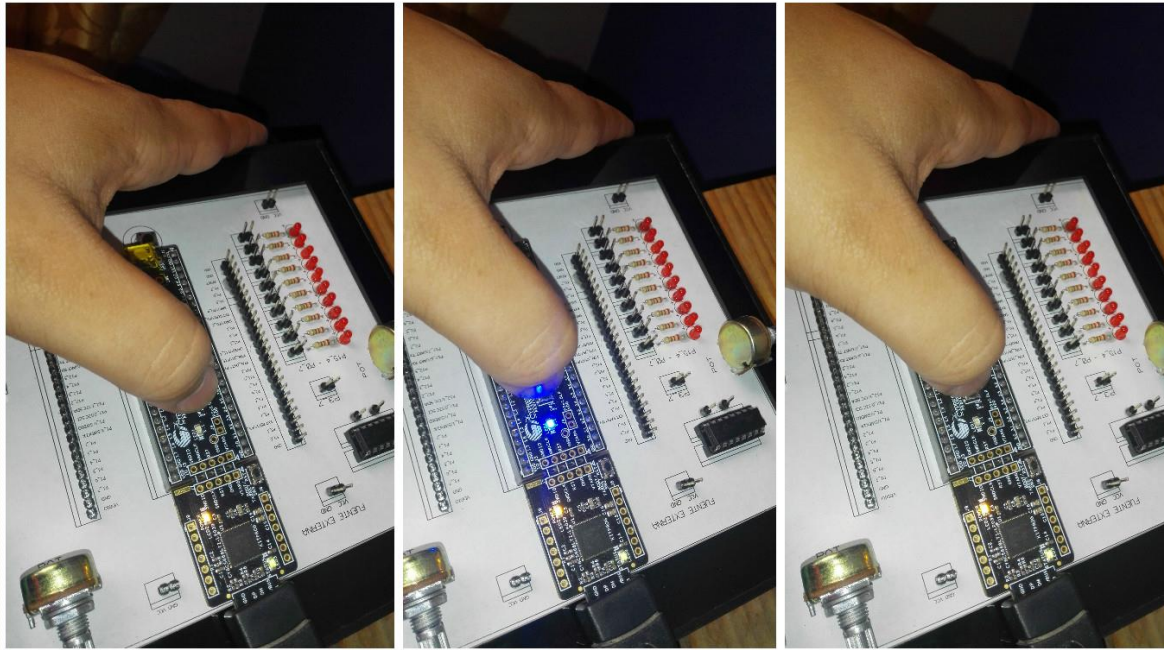


Figura 217. Funcionamiento del Timer

4.6 CONTADOR Y TIMER

Luego de 4 interrupciones el LED prende por 2 segundos

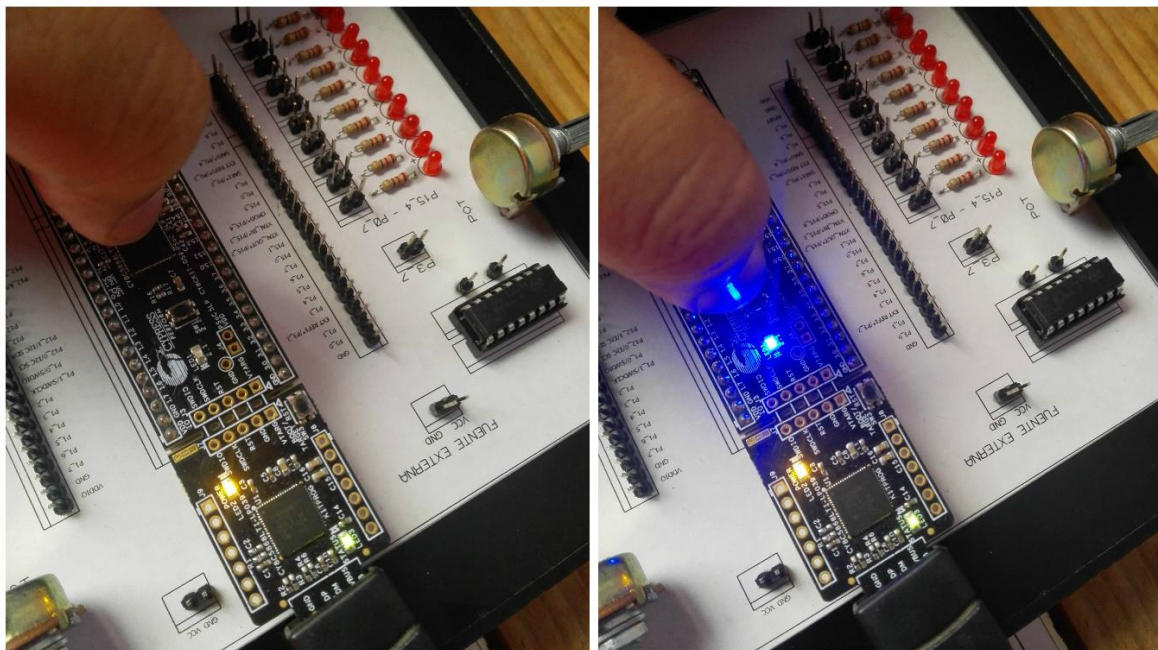


Figura 218. Contador y Timer

4.7 MANEJO DE LCD

- Imprimir un mensaje



- Pantalla intermitente

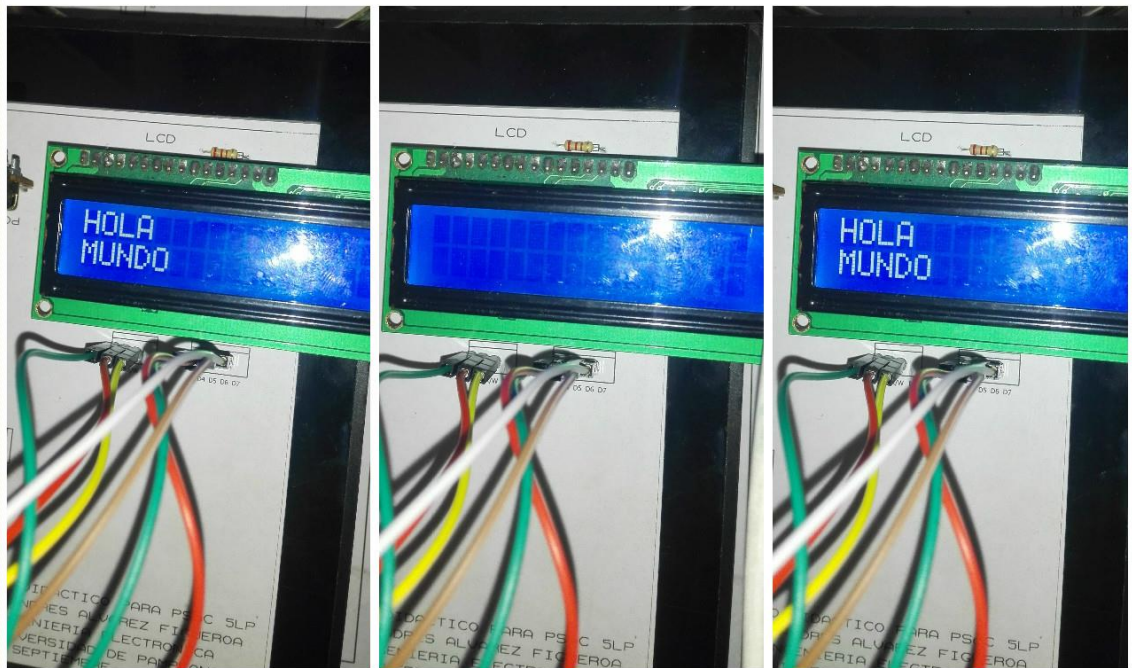


Figura 219. Pantalla Intermitente

- Barra de progreso



Figura 220. Barra de progreso

- Caracteres especiales



Figura 221. Caracteres especiales

4.8 ADC DELTA SIGMA

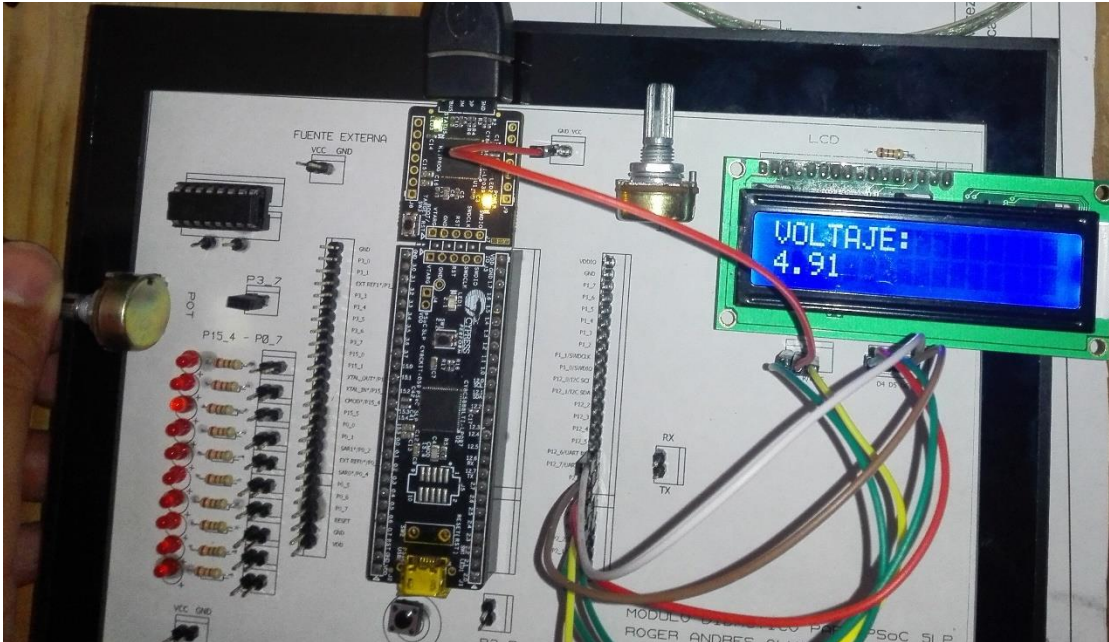


Figura 222. ADC Delta Sigma

4.9 DAC DE VOLTAJE

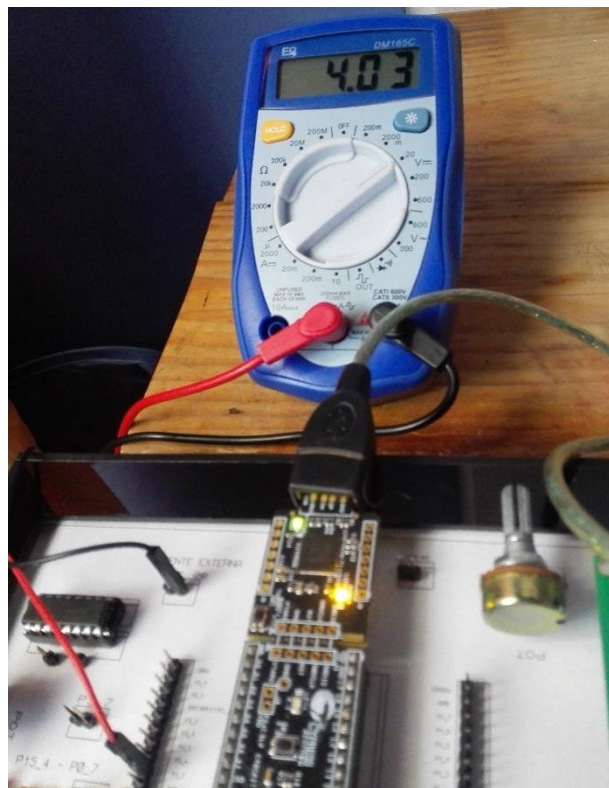


Figura 223. Funcionamiento del VDAC

4.10 ADC DELTA SIGMA Y VDAC

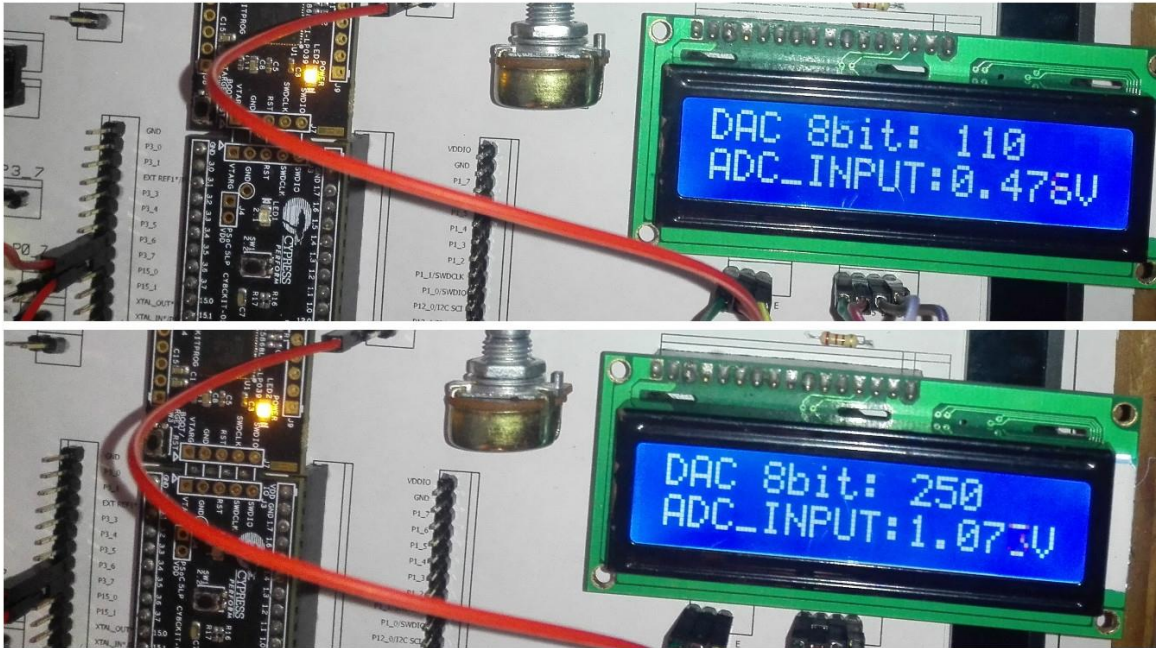


Figura 224. ADC y VDAC

4.11 AMPLIFICADOR OPERACIONAL

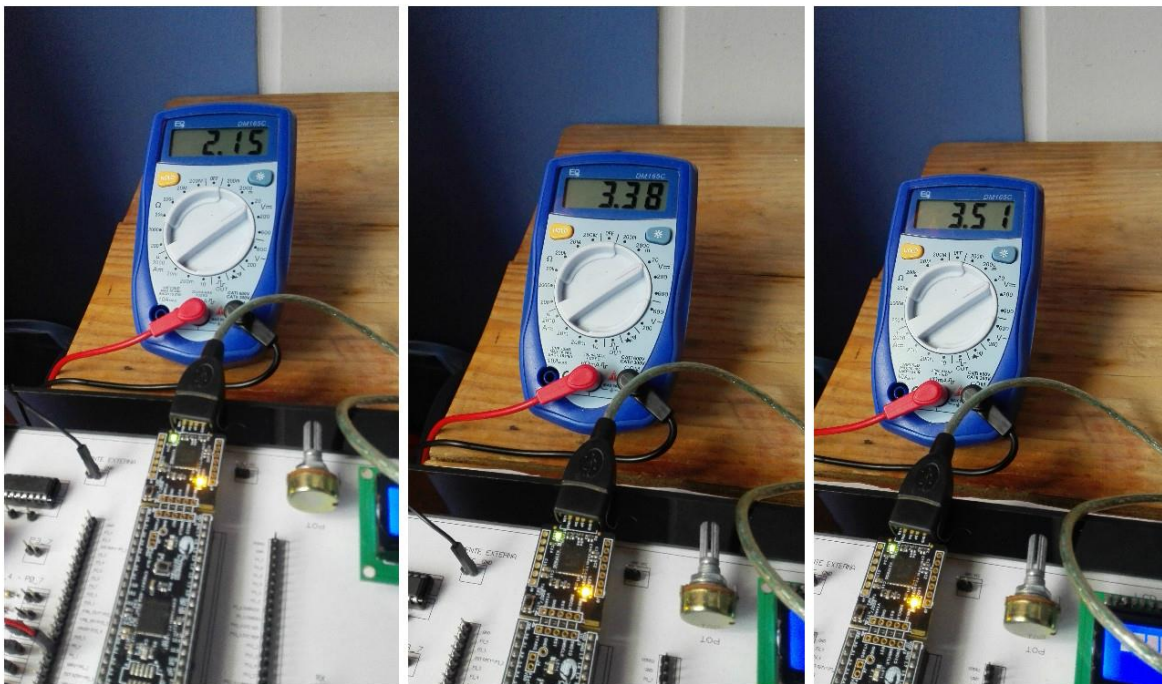


Figura 225. Amplificador como seguidor de voltaje

4.12 COMPARADORES

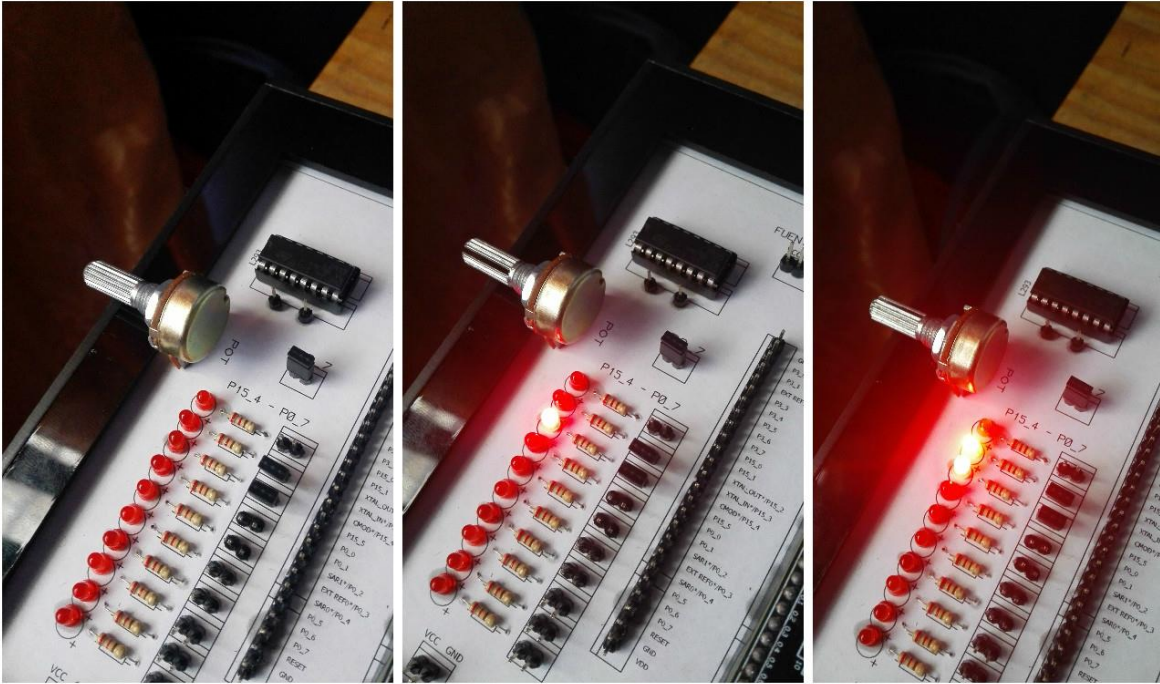


Figura 226. Comparadores

4.13 PWM

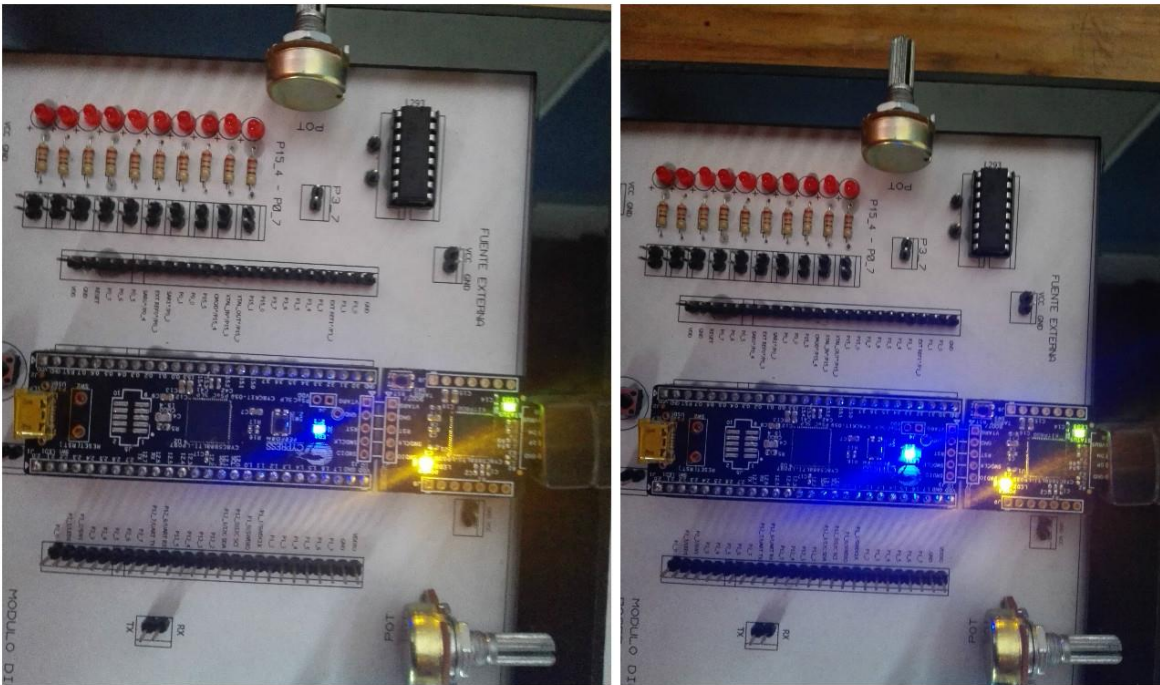


Figura 227. PWM

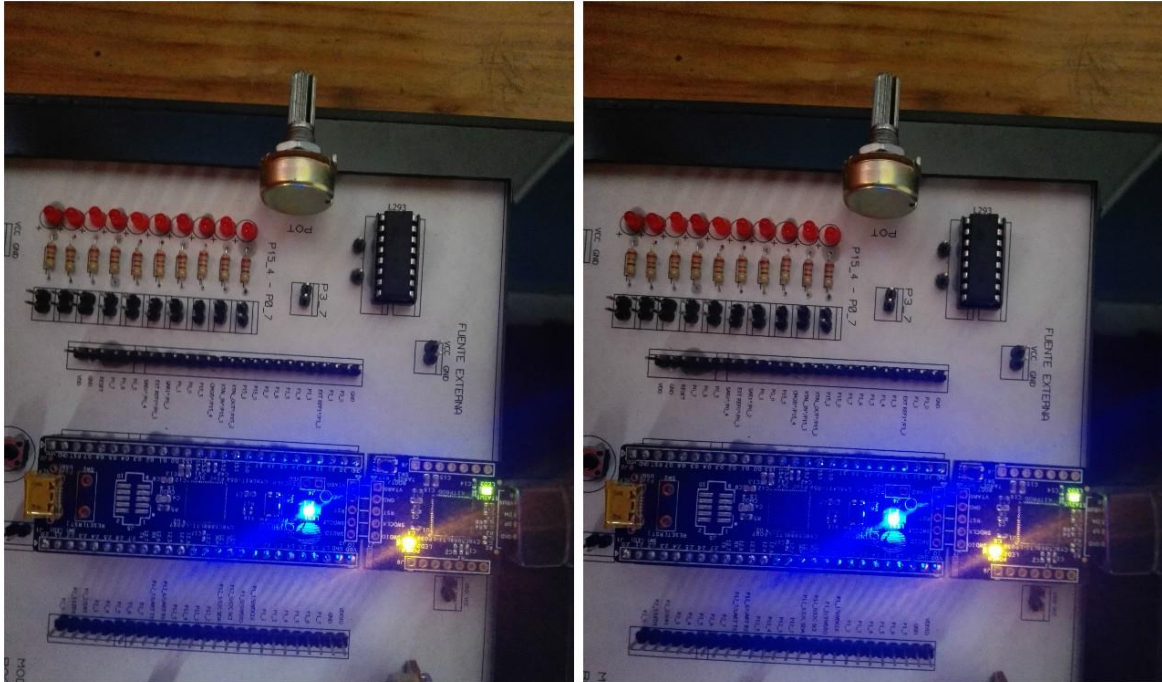


Figura 228. PWM

4.14 CONTROL DE VELOCIDAD DE UN MOTOR USANDO PWM

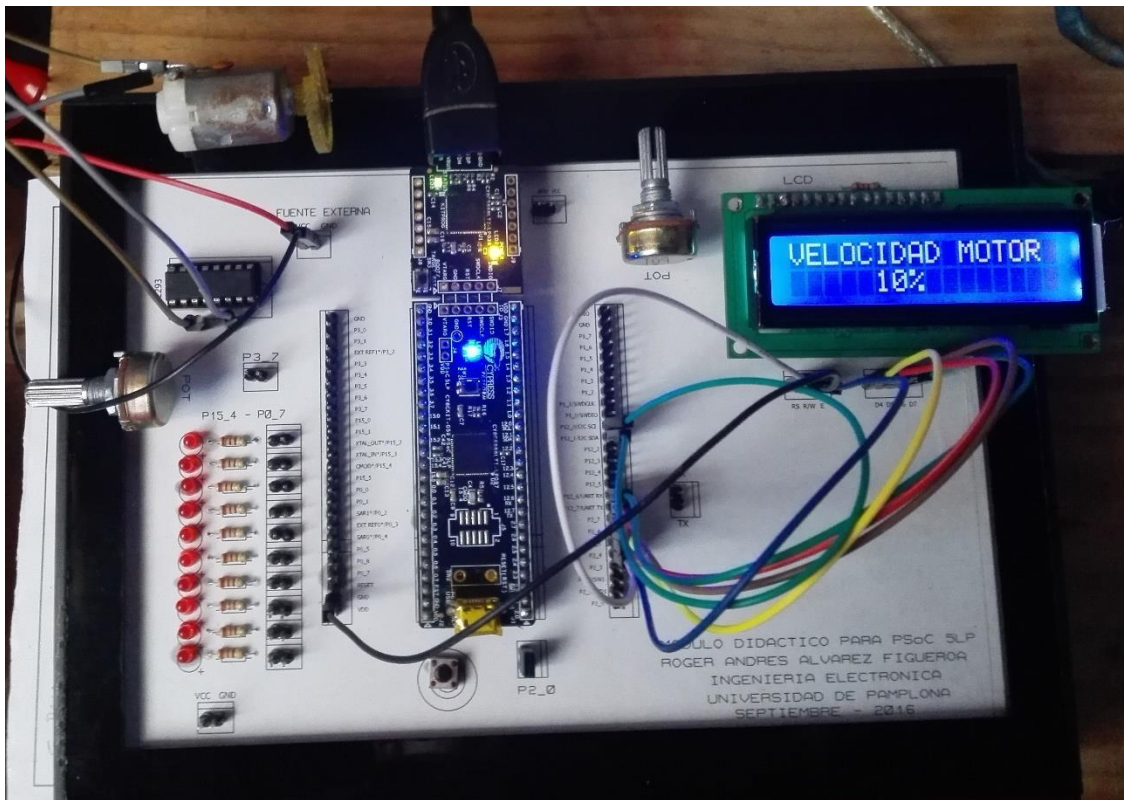


Figura 229. Velocidad de motor 10%

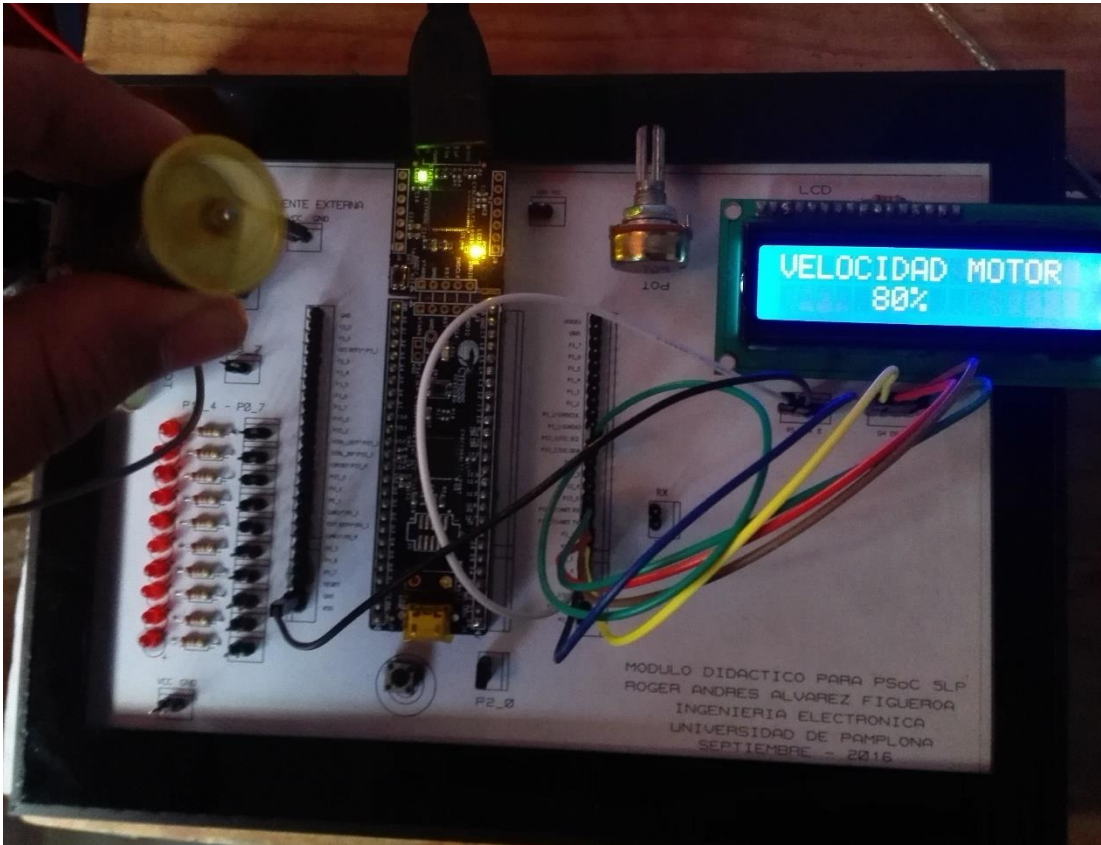


Figura 230. Velocidad de motor 80%

4.15 UART

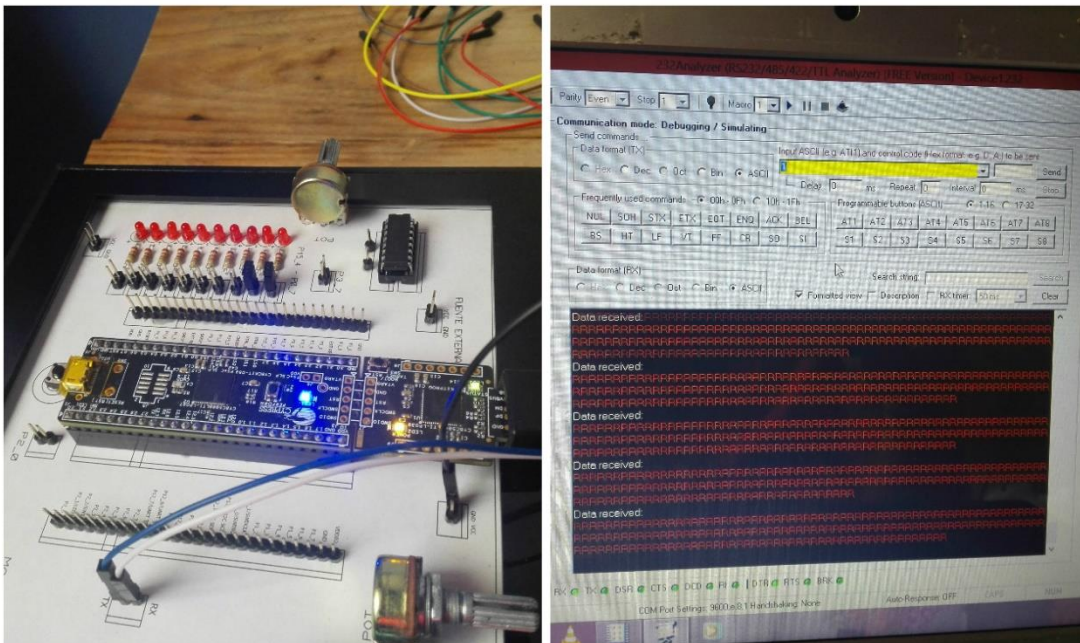


Figura 231. Comunicación serial UART

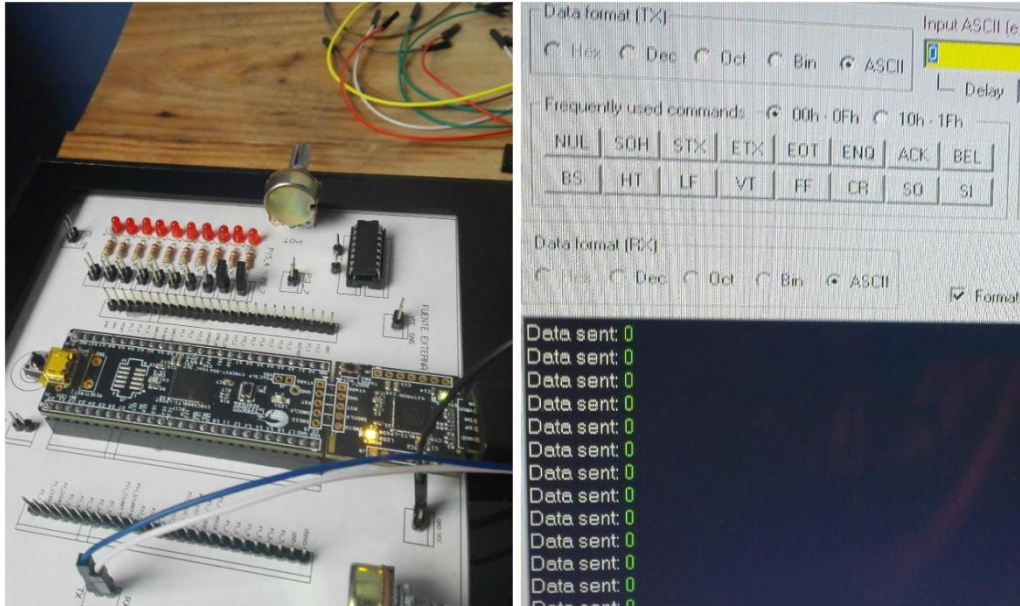


Figura 232. Comunicación serial UART

4.16 LUT

En este caso se multiplican los números 11 y 11 y como resultado se obtiene un 1001

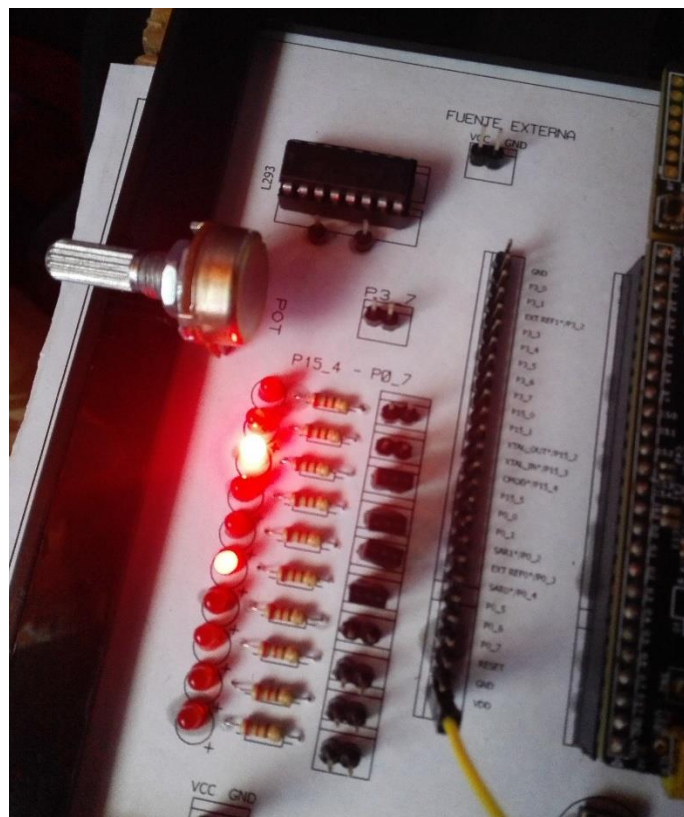


Figura 233. LUT

5 CONCLUSIONES

5 Conclusiones.....	141
----------------------------	------------

Trabajar con el entorno de PSoC Creator de Cypress Semiconductor es sencillo, ya que su interfaz nos permite acceder de forma rápida a todos los componentes que nos brinda el catálogo de Cypress y de esta manera conocer cómo se debe programar tanto el hardware como el software. Conocer el entorno de PSoC Creator nos da la facilidad de seguir indagando y conocer más acerca de los dispositivos PSoS.

Durante la elaboración de las prácticas se presentaron varios inconvenientes ya que no se tenía mucha información de cómo era la manera correcta de configurar los diferentes dispositivos que nos brinda el catálogo de Cypress. Si bien se sabe, el programa de Ingeniería Electrónica de la Universidad de Pamplona adquirió unos Kits de desarrollo a través del convenio University Union en el año 2010, pero estos no se habían utilizado por la escasa información. Los kits de desarrollo para PSoC 3 y PSoC 5 traían unos CD's de instalación del programa PSoC Creator, pero con una versión muy vieja y muy poca información acerca de cómo trabajar estas tarjetas. Cuando se comenzó el trabajo el profesor Julio Cesar Ospino quien es mi director de tesis, contaba con los conocimientos básicos de lo que es un dispositivo PSoC y como trabajarlo, a partir de ese momento tocó indagar sobre que trabajos se habían realizado utilizando la tecnología PSoC en Latinoamérica y en Colombia, y si existían algunas guías que nos ayudaran a trabajar con los elementos que traen los dispositivos PSoC desarrollados por Cypress Semiconductor.

A medida que se fueron realizando cada una de las practicas se fueron presentando diferentes inconvenientes. Por ejemplo, cuando se quiso utilizar la tarjeta de desarrollo PSoC 5, se notó que PSoC Creator ya no cuenta con la referencia de este chip dentro de los dispositivos que son compatibles con este software. Por eso, se decidió trabajar con una PSoC 5LP. Otro inconveniente que se presentó es que Cypress brinda una licencia gratuita por un tiempo limitado para trabajar con los kits de desarrollo, esta licencia genera inconvenientes y no deja siquiera compilar los proyectos, esto puede un aviso de que la PSoC 3 también será descontinuada.

Pero también a medida que se iba avanzando se fueron aclarando muchas dudas y solucionando los inconvenientes que se presentaban. Es muy importante que a la hora de elegir el dispositivo de desarrollo que se va a utilizar, se escoja la referencia del microcontrolador de la tarjeta con la que se va a trabajar, ya que PSoC Creator no brinda la opción de emigrar a otros dispositivos diferentes al que se escogió a la hora de crear el proyecto.

Uno de los motivos más fuertes para comenzar con este trabajo, fue suplir la necesidad de ciertos componentes que no se encuentran disponibles en Pamplona, además de que los sistemas diseñados en PSoC Creator pueden ser modificados por el usuario según las necesidades de su interés.

Entre otras cosas, con este trabajo, se puede ofrecer a los docentes y estudiantes del departamento de Ingeniería Eléctrica, Electrónica, Sistemas y Telecomunicaciones, una

herramienta nueva y muy eficiente en el trabajo con microcontroladores y sistemas embebidos.

6 BIBLIOGRAFÍA

- [1] <http://www.cypress.com/documentation/ceo-articles/cypress-startups-historytheory-funding-lessons>
- [2] <http://www.cypress.com/products/psoc-1>
- [3] L. B. Diagram, PSoC[®] Programmable System-on-Chip TM CY8C24533. 2015.
- [4] <http://www.cypress.com/products/psoc-3>
- [5] G. Description, “Programmable System-on-Chip (PSoC 3),” 2015.
- [6] G. Description, “Programmable System-on-Chip (PSoC 4),” 2016.
- [7] G. Description, “Programmable System-on-Chip (PSoC 5LP),” 2015.
- [8] P. Pin, A. Designs, and A. M. Hastings, “PSoC[®] 3 and PSoC 5LP – Pin Selection for Analog Designs,” no. 001, pp. 1–12.
- [9] D. R. C, C. Court, and S. Jose, “CY8CKIT-059 PSoC[®] 5LP Prototyping Kit Guide Cypress Semiconductor,” pp. 1–42, 1810.
- [10] C. S. Corporation, “PSoC[®] 3 Development Kit Guide Cypress Semiconductor,” 2012.
- [11] <http://www.cypress.com/documentation/development-kitsboards/cy8ckit-030-psoc-3-development-kit>
- [12] C. S. Corporation, “THE PSoC 3 LABBOOK,” 2012.
- [13] D. P. Creator, C. Support, and R. Notes, “PSoC[®] Creator TM Quick Start Guide,” pp. 1–7.
- [14] <http://www.psoclatinoamerica.com/>
- [15] J. A. A. M, M. A. R. Q, and J. S. R. L, “PSoC-based embedded system for the acquisition of EMG signals with Android mobile device display Sistemas embebidos basados en PSoC para la adquisición de señales EMG con un dispositivo móvil Android,” vol. 10, no. 18, pp. 14–19, 2015.
- [16] W. F. Urbina Rojas and F. Martínez Santa, “Implementación de un electromiógrafo con interfaz USB,” Rev. Tecnura, vol. 16, no. 33, p. 117, 2012.
- [17] K. Y. B. L, K. F. N. O, A. A. A. A and O. C. F. A, “SISTEMA ELECTRÓNICO BASADO EN TECNOLOGÍA PSoC[®] PARA LA CARACTERIZACIÓN ELECTROQUÍMICA DE MATERIALES MEDIANTE LA TÉCNICA DE VOLTAMETRÍA CÍCLICA”, vol 1, no. 27, 2016
- [18] Monrroy D. Frangel. “METODOLOGIA PARA EL DISEÑO DE APLICACIONES BASADAS EN PSoC (PROGRAMABLE SYSTEM ON CHIP) DE CYPRESS SEMICONDUCTOR”. 2010.