



UNIVERSIDAD DE CHILE  
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS  
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO Y CONSTRUCCIÓN DE UN COMPENSADOR ESTÁTICO DE REACTIVOS PARA  
LABORATORIO

MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO CIVIL ELECTRICISTA

MARTÍN CORNEJO SAAVEDRA

PROFESOR GUÍA:  
PABLO MEDINA COFRE

MIEMBROS DE LA COMISIÓN:  
ARIEL VALDENEGRO ESPINOZA  
NELSON MORALES OSORIO

SANTIAGO DE CHILE  
AGOSTO 2014

## DISEÑO Y CONSTRUCCIÓN DE UN COMPENSADOR ESTÁTICO DE REACTIVOS PARA LABORATORIO

Los equipos FACTS cada día son más comunes en el mundo gracias al avance de la tecnología en electrónica y semiconductores, permitiendo así operar de manera más holgada los crecientes sistemas eléctricos. En especial, el Compensador Estático de Reactivos (CER), ha tenido gran utilidad en Chile debido a su eficaz control de voltaje en sistemas radiales. Ya se han instalado 5 de estos equipos, los 3 más recientes a partir del año 2000 cuyos nombres son SVC 'Puerto Montt', SVC 'Polpaico' y SVC 'Cerro Navia', propiedades de Transelec.

Bajo este contexto, el objetivo de este trabajo de título es instruir al lector sobre esta tecnología de compensación que se usa en el país explicando su funcionamiento teórico y práctico. Para eso se diseña y construye un CER a escala para laboratorio basándose en prototipos existentes en la literatura y utilizando un microprocesador vigente. Además el modelo a escala es capaz de operar en una red de comunicaciones bajo el protocolo Modbus.

Previo al diseño se revisa el estado de arte de los compensadores estáticos de reactivos tanto a escala como para aplicaciones de potencia, estudiando también su funcionamiento y principales componentes. Se revisa específicamente el modelo del microprocesador a utilizar en el modelo a escala junto con su programación y manejo.

El diseño empleado consiste en un banco trifásico de inductancias junto a dos bancos de condensadores, operados por tiristores y relés respectivamente, conectados a la red de baja tensión mediante un transformador trifásico y coordinados por un sistema de control electrónico.

Se realiza un algoritmo de adquisición de variables para la medición de la tensión en las líneas mediante conversión analógica digital, además de implementar un algoritmo de disparo en las inductancias para control de potencia.

De la construcción final se obtiene un banco de inductancias junto a las placas electrónicas de disparo, adquisición y alimentación que componen el CER a escala.

Como desarrollo futuro se propone programar un control PI en el microprocesador y mejorar la interfaz de adquisición de datos realizada en Matlab-Simulink.

*Dedicado a mi familia.*

# Agradecimientos

Agradezco a Dios por darme todo lo que tengo y he tenido en mi vida, gracias a lo cual ha nacido este trabajo de título como fruto de la trayectoria hasta hoy, como uno de muchos frutos que se vendrán a futuro.

Agradezco a mis padres por darme todo su apoyo incondicional en mi vida, escuchar mis vivencias y guiarme. También a la manu por hacerme reír con sus tonteras.

A los compañeros de universidad, por hacer esta estadía entretenida.

A don Villa por sus consejos y ayuda en la construcción y prueba de los inductores. A los ingenieros J. Muñoz y A. Vargas, cuyos trabajos de título fueron pilares fundamentales para el actual. A los memoristas C. Aedo, R. Rosati y al tesista I. Polanco por su ayuda, material y correcciones entregadas sobre microcontroladores y diseño de circuitos electrónicos.

Además quiero mencionar que Steve Jobs tenía razón sobre conectar los puntos. Distintos conocimientos puntuales aprendidos a lo largo de la estadía en la universidad permitieron que realizara esta memoria.

# Tabla de contenido

<b>1. Introducción</b>	<b>9</b>
1.1. Motivación . . . . .	9
1.2. Alcances y Objetivo General . . . . .	10
1.3. Objetivos Específicos . . . . .	10
1.4. Estructura de la memoria . . . . .	10
<b>2. Antecedentes</b>	<b>12</b>
2.1. Introducción . . . . .	12
2.2. Los tiristores . . . . .	13
2.3. Transferencia de potencia . . . . .	14
2.4. Introducción a los FACTS . . . . .	17
2.4.1. Compensadores en paralelo . . . . .	17
2.5. Topología y operación de un SVC . . . . .	19
2.5.1. Estructura básica . . . . .	19
2.5.2. Curvas de operación . . . . .	21
2.5.3. Variables de control . . . . .	21
2.5.4. Corriente en el TCR . . . . .	23
2.6. Filtros armónicos . . . . .	25
2.7. Simulaciones de un SVC . . . . .	26
2.8. Estado de arte de equipos FACTS . . . . .	29
2.8.1. FACTS a nivel internacional . . . . .	29

2.8.2. FACTS a nivel nacional . . . . .	31
2.8.3. FACTS a escala . . . . .	33
<b>3. Introducción a Modbus y DSP</b>	<b>35</b>
3.1. Introducción . . . . .	35
3.2. Protocolo de comunicaciones Modbus . . . . .	36
3.2.1. Origen . . . . .	36
3.2.2. Descripción del protocolo . . . . .	36
3.3. Especificaciones generales del microcontrolador . . . . .	39
3.4. Periféricos del microcontrolador . . . . .	39
3.4.1. Puertos digitales Entrada/Salida (GPIO) . . . . .	39
3.4.2. Interfaz de Interrupción Externa (XINTF) . . . . .	40
3.4.3. Conversor análogo digital (ADC) . . . . .	40
3.4.4. Puerto de comunicacion serial (SCI o UART) . . . . .	40
3.5. Estructura y funcionamiento de un programa . . . . .	40
3.6. Ejemplos de aplicación . . . . .	42
3.6.1. Parpadear un Led . . . . .	42
3.6.2. Enviar mensaje por puerto serial UART . . . . .	44
3.6.3. Implementación de Modbus en el DSP . . . . .	46
<b>4. Diseño de un CER para laboratorio</b>	<b>49</b>
4.1. Topología . . . . .	49
4.2. Dimensionamiento del CER . . . . .	50
4.3. Diseño de los inductores . . . . .	50
4.4. Transformador de bajada . . . . .	52
4.5. Controlador principal . . . . .	53
4.6. Dispositivos de medición . . . . .	53
4.6.1. Transductores . . . . .	53

4.6.2. Amplificador operacional . . . . .	53
4.6.3. Optoacoplador . . . . .	53
4.7. Dispositivos de accionamiento . . . . .	54
4.7.1. Tiristores . . . . .	54
4.7.2. Optoacoplador de salida triac . . . . .	54
4.7.3. Buffer XOR . . . . .	54
4.7.4. Relés . . . . .	54
4.7.5. Transistores . . . . .	54
4.8. Dispositivos de comunicación . . . . .	55
4.8.1. RS-232 . . . . .	55
4.8.2. RS-485 . . . . .	55
4.9. Diseño de placas . . . . .	55
4.9.1. Placa de control . . . . .	55
4.9.2. Placa de adquisición y disparo . . . . .	57
4.9.3. Placa de alimentación . . . . .	59
4.10. Algoritmo del CER . . . . .	60
4.10.1. Adquisición de variables . . . . .	60
4.10.2. Sincronizador de disparos . . . . .	60
<b>5. Construcción de un CER para laboratorio</b>	<b>62</b>
5.1. Construcción de los inductores . . . . .	62
5.2. Construcción de placas circuitales . . . . .	63
5.2.1. Placa de adquisición y disparo . . . . .	63
5.2.2. Placa de control . . . . .	64
5.2.3. Placa de alimentación . . . . .	65
5.2.4. Placa de ajuste . . . . .	65
<b>6. Conclusion</b>	<b>67</b>

6.1. Trabajo futuro y recomendaciones . . . . .	68
<b>Bibliografía</b>	<b>69</b>
<b>A. Esquemáticos de las placas</b>	<b>I</b>
<b>B. Código del programa</b>	<b>VI</b>
<b>C. Interfaz en Matlab-Simulink</b>	<b>XXX</b>



# Índice de tablas

2.1. SVC's instalados en el SIC . . . . .	31
3.1. Estructura de mensajes en Modbus . . . . .	36
3.2. Código de las funciones en Modbus . . . . .	37
3.3. Características técnicas de Modbus RTU . . . . .	37
3.4. Ejemplo de petición de Maestro en Modbus . . . . .	38
3.5. Ejemplo de respuesta de un esclavo en Modbus . . . . .	38
4.1. Parámetros del filtro de 5 <sup>a</sup> armónica . . . . .	50
4.2. Resultado del diseño del inductor . . . . .	52
5.1. Valores nominales del inductor . . . . .	62
5.2. Pruebas de energización del inductor . . . . .	62

# Índice de figuras

2.1. Estructura de un tiristor . . . . .	13
2.2. Tiristores conectados a una carga resistiva . . . . .	14
2.3. Voltaje controlado en una carga resistiva mediante tiristores . . . . .	14
2.4. Sistema de potencia simple . . . . .	15
2.5. Relación V-P o Curvas Nariz . . . . .	16
2.6. 1) TCR 2) TSC . . . . .	18
2.7. Topología básica del STATCOM . . . . .	19
2.8. Estructura básica de un SVC . . . . .	19
2.9. Curva de operación de un SVC . . . . .	21
2.10. Controlador proporcional integral . . . . .	22
2.11. Reglas lógicas en un controlador difuso para SVC [23] . . . . .	22
2.12. Forma de onda del voltaje en el TCR . . . . .	23
2.13. Porcentaje de armónicos respecto de la corriente fundamental en un TCR según el ángulo de disparo . . . . .	25
2.14. Topologías para filtros de armónicas . . . . .	26
2.15. Ejemplo de simulación en Simulink [16] . . . . .	27
2.16. Ejemplo del TCR en Simulink [16] . . . . .	27
2.17. Potencia reactiva del ejemplo en Simulink [16] . . . . .	28
2.18. Voltaje y corriente del ejemplo en Simulink [16] . . . . .	28
2.19. Imagen aérea del SVC de Black Oak [1] . . . . .	29
2.20. Unilineal del SVC de Black Oak [1] . . . . .	30

2.21. Imagen del SVC de Impala [18] . . . . .	30
2.22. Unilineal del SVC de Impala [18] . . . . .	31
2.23. Conexión de los SVC's Polpaico y Cerro Navia [3] . . . . .	32
2.24. Diagrama unilineal SVC Polpaico [3] . . . . .	32
2.25. Esquema lógico del funcionamiento del SVC de Tsinghua [22] . . . . .	33
2.26. Esquema del SVC del DIE [5] . . . . .	34
2.27. Interfaz del CER del DIE [5] . . . . .	34
3.1. Estructura del código . . . . .	40
3.2. Ciclo de Trabajo del DSP . . . . .	41
4.1. Diagrama unilineal del CER a construir . . . . .	49
4.2. Parámetros de diseño del inductor . . . . .	51
4.3. Diagrama de conexión del transformador . . . . .	52
4.4. Circuito de ajuste para la salida de los transductores LEM . . . . .	55
4.5. Capa inferior de la placa de control . . . . .	56
4.6. Esquemático de los relés . . . . .	57
4.7. PCB de lems de tensiones . . . . .	57
4.8. Circuito detector de cruces por cero . . . . .	58
4.9. PCB del circuito de detección de ceros . . . . .	58
4.10. Circuito de disparo de los tiristores . . . . .	58
4.11. PCB de la placa de disparo . . . . .	59
4.12. PCB de la placa de alimentación . . . . .	59
4.13. Diagrama de flujo del conversor ADC . . . . .	60
4.14. Señal de interrupción para sincronizador de disparos en la fase R . . . . .	61
4.15. Diagrama de flujo del sincronizador de disparos . . . . .	61
5.1. Inductor final . . . . .	63
5.2. Placa de adquisición y disparo . . . . .	63

5.3. Placa de Control . . . . .	64
5.4. Placa de alimentación . . . . .	65
5.5. Nueva placa de ajuste . . . . .	65
A.1. Esquemático de la placa de adquisición . . . . .	II
A.2. Esquemático de la placa de control . . . . .	III
A.3. Esquemático de la placa de alimentación . . . . .	IV
A.4. Esquemático de la placa de ajuste . . . . .	V
C.1. Interfaz para visualizar las variables del CER . . . . .	XXXI

# Capítulo 1

## Introducción

Debido a la geografía de Chile, su principal sistema eléctrico: Sistema Interconectado Central (SIC) presenta una topología radial. En este tipo de topología priman los problemas de estabilidad de voltaje, debido a las largas líneas de transmisión que presentan grandes impedancias oponiéndose principalmente al paso de la energía reactiva, la cual está directamente relacionada con la estabilidad de voltaje. Esto ha llevado a instalar en ciertos nodos de conexión críticos del sistema compensación de energía reactiva local, supliendo la necesidad de este tipo de energía en los momentos de mayor congestión o en contingencias.

Gracias al desarrollo de los semiconductores y la electrónica, se han creado aplicaciones electrónicas para el control de potencia en alta tensión, rama de estudio conocida como *electrónica de potencia*. De esta rama han surgido diversas tecnologías para compensación de reactivos automatizadas, tales como el Compensador Estático de Reactivos (CER), cuya principal herramienta de accionamiento es el tiristor (dispositivo a base de semiconductores creado en los '50).

En el SIC existen 5 equipos CER operando conectados al sistema troncal de 220 kV a lo largo del país, propiedad de Transelec e instalados por ABB. El más grande hasta la fecha es el CER de Cerro Navia con una capacidad de Absorción de 65 MVar y una capacidad de entrega de 140 MVar.

### 1.1. Motivación

Así como la electrónica ha dado grandes avances para crear nuevos dispositivos de accionamiento basados en semiconductores (tiristores, IGBT, TRIACS) también ha nacido otro tipo de dispositivos electrónicos de pequeño tamaño y fácil programación, capaces de automatizar tareas, realizar cálculos y controlar periféricos, comúnmente llamados microcontroladores. En la actualidad los microcontroladores están al alcance de todos y su documentación abunda en la web, por lo que es posible realizar aplicaciones de este tipo en casi cualquier laboratorio con implementos electrónicos básicos. Uniendo esto a lo planteado en la introducción, se puede crear un dispositivo que controle

de manera automática la compensación de reactivos en un nodo de conexión a la red eléctrica de distribución (380 V), es decir, un CER a escala. El resultado de esta motivación es el presente trabajo de título.

## 1.2. Alcances y Objetivo General

El objetivo general del trabajo de título es diseñar y construir un compensador estático de reactivos a escala para laboratorio. El uso de este equipo permitirá mostrar de forma práctica conceptos sobre regulación de voltaje y compensación de reactivos en los laboratorios docentes del Departamento de Ingeniería Eléctrica de la Universidad de Chile.

## 1.3. Objetivos Específicos

Dentro de los objetivos específicos se encuentran:

- Revisar el estado de arte de los compensadores estáticos de reactivos, tanto a escala como para aplicaciones de potencia, entendiendo también su funcionamiento y principales componentes.
- Diseñar y construir un SVC para laboratorio, basándose en los prototipos existentes en la literatura y para aplicaciones de potencia, utilizando la capacidad de procesamiento de los DSP actuales.
- Mostrar en forma práctica la teoría del control de voltaje y la compensación de reactivos, además de estudiar los armónicos generados por la electrónica de potencia del CER y diseñar filtros para el modelo a escala.
- Integrar el modelo a escala a una red de comunicaciones utilizando el protocolo Modbus, con el fin de monitorear las variables de operación del CER en una interfaz remota y coordinar su operación junto a otros equipos existentes en la red.

## 1.4. Estructura de la memoria

La estructura utilizada en este documento para exponer el trabajo realizado es la siguiente:

- **Capítulo 1. Introducción:** Corresponde a la descripción del tema, la motivación de éste y los alcances y objetivos del trabajo realizado.
- **Capítulo 2. Antecedentes:** Corresponde a la revisión bibliográfica o antecedentes. En este capítulo se explican los conceptos necesarios para la comprensión y contextualización del trabajo.

- **Capítulo 3. Introducción a Modbus y DSP:** Se explican los conceptos básicos para entender el protocolo Modbus. Se describen las características y funcionalidades del microcontrolador a utilizar.
- **Capítulo 4. Diseño de un CER para laboratorio:** En este capítulo se muestran detalladamente las distintas partes del CER, tanto físicas como electrónicas y la manera en que se construirán.
- **Capítulo 5. Construcción de un CER para laboratorio:** Se muestran los resultados del diseño obtenido en el capítulo 4.
- **Capítulo 6. Conclusión:** Se enumeran las conclusiones del trabajo realizado y se proponen trabajos a realizar en el futuro.

## Capítulo 2

# Antecedentes

### 2.1. Introducción

Los Sistemas Eléctricos de Potencia tienen como objetivo suplir la demanda de energía de los consumos a partir de la energía generada en las centrales eléctricas. Para el transporte de la energía se construyen sistemas de transmisión en Alta Tensión que permiten el flujo de grandes bloques de energía desde las centrales hasta los centros industriales o urbanos.

Los sistemas de transmisión deben estar siempre dentro de un rango de operación que permita circular el flujo de energía sin saturar las líneas y manteniendo niveles de voltaje permitidos según la norma. Los parámetros limitantes del flujo por una línea de transmisión cualquiera son: el voltaje en las barras de la línea, la reactancia serie de la línea y el ángulo de desfase entre las barras de la línea (sección 2.3). El consumo eléctrico, ya sea industrial o urbano, varía según transcurre el día. Debido a esto las líneas de transmisión presentan distintos niveles de carga. Sin embargo, en momentos donde se concentra la mayor demanda las líneas pueden presentar saturación y caídas de voltaje, siendo más o menos crítica la situación dependiendo de la capacidad de cada línea. Por esto es preciso contemplar algún tipo de compensación en la línea para poder permitir el flujo de energía en los momentos de mayor estrés del sistema.

Debido a lo anterior nacieron los **Equipos de Compensación Flexibles (FACTS)** (sección 2.4). La función de estos equipos es entregar de manera rápida y controlada a la línea la compensación necesaria tal que ésta pueda operar de manera óptima en los momentos de mayor demanda. Gracias a la electrónica de potencia actual estos equipos actúan en tiempos del orden de los milisegundos. Dentro de los equipos FACTS se encuentra el **Compensador Estático de Reactivos (CER o SVC)**. Este equipo puede entregar o absorber potencia reactiva a la línea, controlando así los niveles de voltaje dentro del rango seguro en la línea (sección 2.5). Existen SVC's de diversos tamaños, ya sea a nivel de transmisión en alta potencia o a escala de laboratorio (sección 2.8).



## 2.2. Los tiristores

Los tiristores son dispositivos compuestos por tres junturas p-n (cuatro capas intercaladas de material tipo p y tipo n). Poseen tres terminales: la compuerta **G**, el ánodo **A** y el cátodo **K**. Desde la compuerta se controla el estado de conducción del tiristor, mientras el ánodo y el cátodo se conectan en serie con la carga cuya potencia se desea controlar. En pocas palabras, el tiristor se comporta como un interruptor de potencia de dos estados, idealmente cerrado e idealmente abierto, cuya velocidad de conmutación es del orden de los milisegundos. Típicamente son usados en la alta potencia debido a que son capaces de funcionar con voltajes superiores a 1 kV y corrientes mayores a 100 A [15].

La característica principal de los tiristores es su propiedad llamada *latching*. Cuando el tiristor pasa del estado abierto al estado cerrado o de conducción, el tiristor seguirá en el estado de conducción hasta que la corriente en el mismo vuelva a cero, aunque desaparezca la señal de activación en la compuerta.

Existen 4 tipos importantes de tiristores: Rectificador controlado de silicio (SCR), Tiristor apagado por compuerta (GTO), Tiristor controlado por MOS (MCT) y Tiristor estático de inducción (SITh). Los tiristores más comunes son el SCR y el GTO. El SCR es el tiristor común recién explicado. El GTO, además de poseer el *latching*, puede ser apagado en cualquier momento mediante un pulso de corriente negativa en la compuerta. En adelante cuando se hable de tiristores se estará refiriendo al tipo SCR.

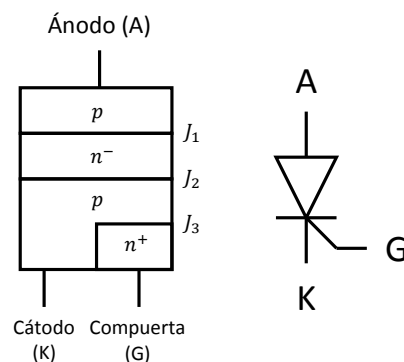


Figura 2.1: Estructura de un tiristor

En particular, los tiristores usados en corriente alterna (50 o 60 Hz), son llamados "dispositivos de control de fase" ya que son encendidos por la compuerta después de cierto ángulo de desfase con respecto a la forma sinusoidal del voltaje de la red, típicamente se usa el cruce por cero como referencia del ángulo de disparo. De esta manera, con un tiristor conectado en serie, se puede cambiar la potencia consumida por una carga resistiva o inductiva variando la corriente entre su valor nominal y cero.

En la figura 2.2 se muestra un esquema básico de control de potencia en una carga resistiva. En la figura 2.3 se muestra como varía el voltaje en la carga resistiva mediante dos tiristores conectados en configuración antiparalela.

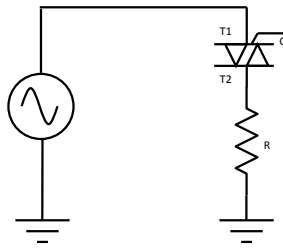


Figura 2.2: Tiristores conectados a una carga resistiva

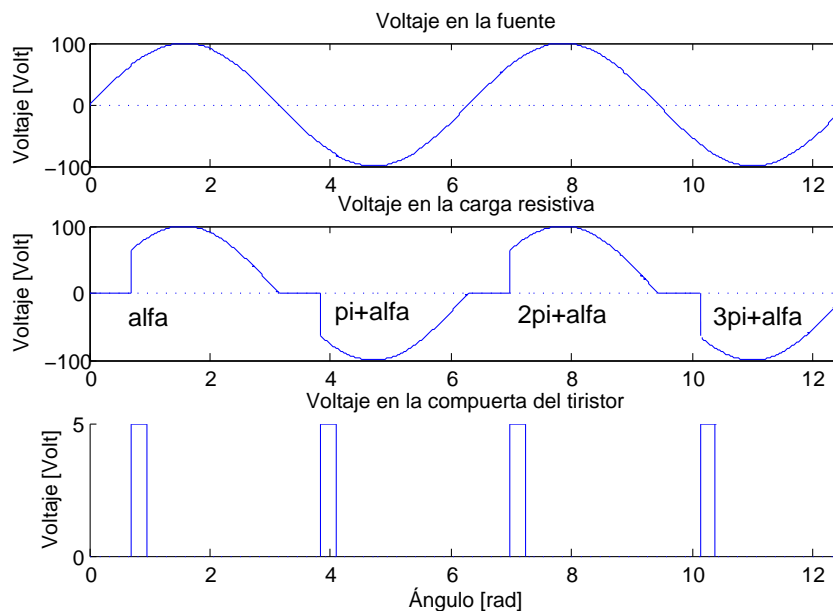


Figura 2.3: Voltaje controlado en una carga resistiva mediante tiristores

El ejemplo de la figura 2.3 muestra una fuente de voltaje sinusoidal de 100 V de amplitud, y los tiristores tienen un voltaje de activación por compuerta de 5 V. Para cargas resistivas el ángulo de disparo  $\alpha$  puede variar entre  $0^\circ$  y  $180^\circ$ . El ángulo de extinción  $\beta$  corresponde al momento en que la corriente pasa por cero y el tiristor se desactiva. Para cargas resistivas  $\beta = 180^\circ$ . Se observa también que solo basta un pulso de voltaje en la compuerta del tiristor para activar el estado de conducción. En este caso se tienen dos tiristores en configuración antiparalela, por lo tanto se debe dar un pulso de voltaje a cada tiristor por separado, según sea el ciclo positivo o negativo de la onda de voltaje.

### 2.3. Transferencia de potencia

Para entender los flujos de potencia se suele estudiar su caso más reducido, que consiste en un sistema de dos barras, una transmisora y una receptora, conectadas por una línea de transmisión formando así una topología radial (figura 2.4).

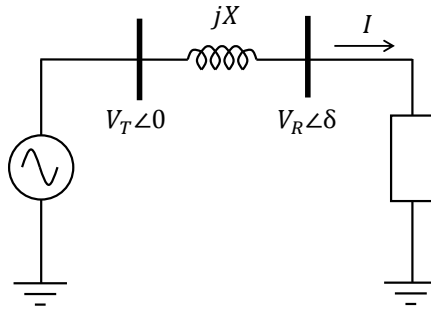


Figura 2.4: Sistema de potencia simple

La barra transmisora está conectada a un generador sincrónico y la receptora a una carga. Las ecuaciones para las potencias activa y reactiva se muestran a continuación:

$$\begin{aligned}
 P_T &= \frac{V_T V_R \sin(\delta)}{X} = P_R \\
 Q_T &= -\frac{V_T V_R \cos(\delta)}{X} + \frac{V_T^2}{X} \\
 Q_R &= \frac{V_T V_R \cos(\delta)}{X} - \frac{V_R^2}{X}
 \end{aligned} \tag{2.1}$$

Según las ecuaciones 2.1, considerando  $Q = Q_R$  y  $P = P_T = P_R$ , al eliminar la variable del ángulo se obtiene:

$$(V^2)^2 + (2QX - E^2)V^2 + X^2(P^2 + Q^2) = 0 \tag{2.2}$$

Según la relación 2.2, al aumentar la carga en el extremo receptor el voltaje cae dependiendo del factor de potencia del sistema (o nivel de compensación reactiva).

En la figura 2.5 se observa la relación entre el voltaje y la potencia transferidas para factores de potencia inductivos y capacitivos, considerando el voltaje unitario  $\frac{V}{E}$  y la potencia unitaria  $\frac{PX}{E^2}$ . Para factores de potencia cercanos a 1 ( $\tan(\phi) \sim 0$ ) el voltaje cae gradualmente a medida que aumenta la potencia transferida. Sin embargo, a medida que  $\tan(\phi)$  es inductivo y aumenta, el voltaje cae bruscamente a medida que aumenta la transferencia de potencia. En estos casos es necesario aumentar la potencia transferida mediante algún método de compensación, mencionados a continuación.

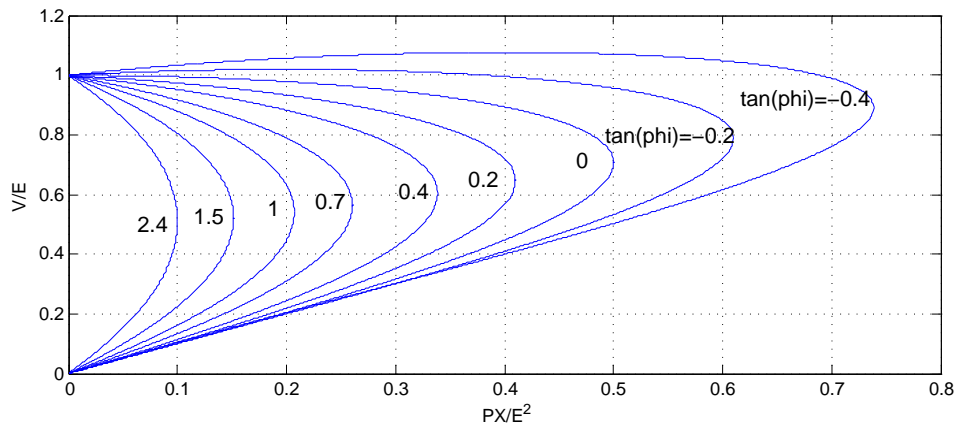


Figura 2.5: Relación V-P o Curvas Nariz

La ecuación de transferencia de potencia mostrada en la ecuación 2.1 puede ser generalizada para una transferencia de potencia activa  $P_{ij}$  que va desde una barra  $i$  a una barra  $j$  a través de una reactancia  $X_{ij}$  y una diferencia angular  $\delta_{ij} = \delta_i - \delta_j$  (ecuación 2.3).

$$P_{ij} = \frac{V_i V_j \sin(\delta_{ij})}{X_{ij}} \quad (2.3)$$

Según la ecuación 2.3 existen tres maneras de aumentar la potencia activa transferida. Una manera es elevar  $\delta = \delta_{ij}$ , sin embargo existe un límite para  $\delta = \pi/2$ . Otra forma es elevar las tensiones, sin embargo éstas también tienen límite de operación para el buen funcionamiento del sistema. Al reducir la reactancia  $X = X_{ij}$  también se aumenta la potencia activa transferida, pero solo es posible mediante condensadores en serie. Esto también tiene un límite cuando  $x_C \sim x_L$ , ya que en ese punto el sistema es sensible a los cambios en las condiciones de transmisión [20]. Estas 3 variables ( $V, X, \delta$ ) pueden ser controladas por los equipos mencionados en la sección 2.4.

## 2.4. Introducción a los FACTS

Por lo mencionado en 2.3 y en la figura 2.5, con tal de aumentar la potencia transferida en las redes alternas de alta tensión y evitar caídas importantes de voltaje, han surgido los **Equipos de Compensación Flexibles (FACTS)**, los cuales se dividen en 3 categorías: Compensadores en paralelo ( $V$ ), Compensadores en serie ( $X$ ) y Compensadores por ángulo de fase ( $\delta$ ). El uso de estos equipos FACTS tiene los siguientes beneficios: disminución de las pérdidas de potencia activa, mejoras en la estabilidad de la red en cuanto a voltaje y menores costos de producción de energía [20]. Se profundizará solo sobre los compensadores en paralelo ya que son el tema de interés del presente trabajo de título.

### 2.4.1. Compensadores en paralelo

La compensación en paralelo consiste en entregar potencia reactiva capacitiva o inductiva a la línea para mantener el voltaje dentro de un rango definido, o para mantener un factor de potencia fijo en el punto de conexión. Es sabido que existe un fuerte acoplamiento entre las inyecciones de potencia reactiva y las tensiones en las barras de inyección [20]. Dado esto, según la ecuación 2.3, al aumentar el voltaje en una de las barras es posible aumentar la transferencia de potencia. Es por esto que los principales beneficios de usar compensación en paralelo son mantener el voltaje de barra en un rango de operación seguro y mantener las transmisiones de potencia. A continuación se muestran los equipos FACTS de compensación en paralelo:

- **Reactor Controlado por Tiristores (TCR):** Este equipo consiste en un reactor en derivación, conectado en serie con tiristores que actúan como interruptor bidireccional. La corriente que circula por el reactor se puede modificar variando el ángulo de disparo de los tiristores, obteniendo así una reactancia de compensación variable.
- **Condensador Conmutado por Tiristores (TSC):** Es un condensador en derivación conectado en serie con tiristores y a menudo se conecta en serie con una inductancia pequeña para limitar sobrecorrientes. Los tiristores tienen la función de conectar o desconectar el condensador, funcionando como un interruptor de dos estados. De esta manera el tiempo de conexión/desconexión es menor que si se usaran relés convencionales. Una alternativa a este módulo es el **MSC (Mechanically Switched Capacitor)**, el cual utiliza relés mecánicos en vez de tiristores, por lo que su respuesta dinámica es bastante más lenta.

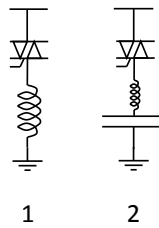


Figura 2.6: 1) TCR 2) TSC

- **Compensador Estático de Reactivos (SVC o CER):** Es la combinación de un TCR y un TSC. Estos módulos se deben coordinar mediante un sistema de control. La estrategia de control más común es la de mantener una consigna de voltaje. De esta manera se puede realizar compensación capacitiva o inductiva ya sea por sobrecarga o por baja demanda (sobrevoltaje). Otra estrategia de control menos común es la de mantener una consigna de factor de potencia en la barra de conexión.
- **Compensador Estático Síncrono (STATCOM):** Este equipo es distinto a los anteriores, ya que su compensación es un condensador en corriente continua conectado a la red principal mediante un inversor controlado. Los inversores actuales están compuestos por válvulas IGBT, las cuales mediante modulación por ancho de pulso (PWM) convierten corriente continua en corriente alterna. Así, controlando la magnitud relativa entre la tensión de la red y la salida del inversor, el STATCOM puede proporcionar compensación capacitiva o inductiva. La ventaja de este equipo con respecto al SVC es que su calidad de compensación no depende del voltaje en la red, ya que la magnitud de la corriente inyectada depende netamente del inversor. En cambio, en el SVC la corriente de compensación depende del voltaje en la barra, ya que los equipos de compensación están conectados directamente a la red, y es un hecho que la compensación en paralelo es proporcional al voltaje de conexión. Debido a esto el STATCOM tiene un mejor comportamiento que el SVC frente a contingencias de caídas de tensión.

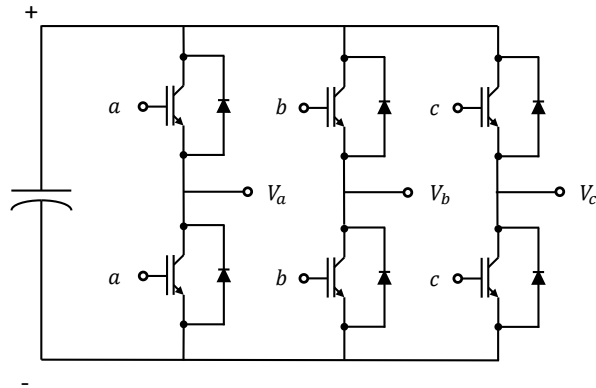


Figura 2.7: Topología básica del STATCOM

## 2.5. Topología y operación de un SVC

### 2.5.1. Estructura básica

Como se mencionó en 2.4.1, el SVC es una combinación de un TCR y un TSC, los cuales trabajan de manera coordinada obteniendo así compensación de reactivos inductiva o capacitiva según sea la necesidad en la barra de conexión. Además de los TCR y TSC están los sistemas de medición, el controlador principal, el transformador de bajada (no es indispensable) y el sincronizador de disparos (PLL). Un esquema típico de un SVC se muestra en la figura 2.8

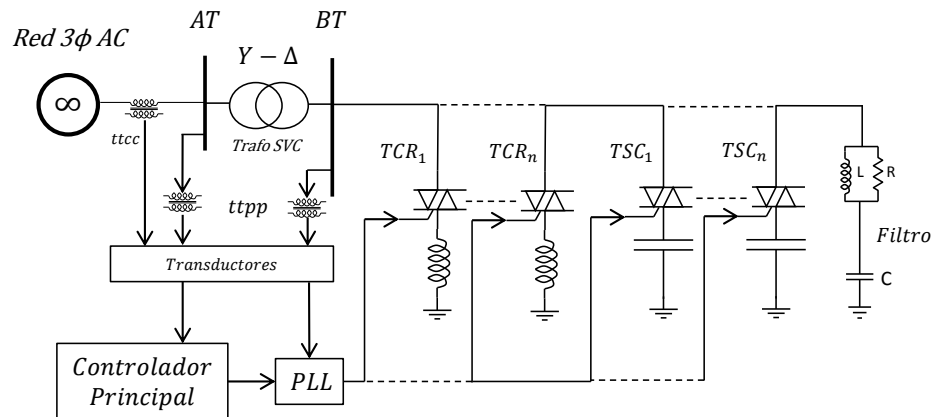


Figura 2.8: Estructura básica de un SVC

A continuación se explica la función de cada componente del diagrama.

- **TCR:** Los módulos TCR (Thyristor Controlled Reactor) consisten en reactores conectados en serie con tiristores. Los tiristores conducen o interrumpen la corriente circulante, obteniendo así una inductancia variable. Los tiempos de conducción de los tiristores vienen dados por el ángulo de disparo calculado por el Controlador Principal, y la orden de disparo es sincronizada por el

PLL. La cantidad de módulos conectados depende del diseño del SVC, siendo generalmente 1 o 2 la cantidad de TCR's utilizados.

- **TSC:** Los módulos TSC (Thyristor Switched Capacitor) consisten en condensadores conectados en serie con tiristores. Los tiristores sólo tienen dos estados, abierto o cerrado. La orden de apertura o cierre de los relés es ejecutada por el Controlador Principal y sincronizada por el PLL. El número de TSC's conectados es generalmente 1 unidad más que la cantidad de TCR's conectados.
- **TTPP:** Los transformadores de potencial *tpp* se encargan de reducir el voltaje en AT y BT con el fin de obtener señales aptas para ser procesadas por los componentes electrónicos del controlador principal.
- **TTCC:** El transformador de corriente *tcc* reduce la corriente obtenida en AT para tener una señal de corriente apta para los medidores del controlador.
- **Transductores:** Entre los transformadores de medida *tpp-tcc* y el controlador principal se utilizan transductores que filtran y transforman las señales alternas en ondas digitales aptas para ser procesadas. Generalmente los transductores consisten en filtros pasa bajas para eliminar el ruido y rectificadores de onda completa para que los procesadores no tengan que lidiar con señales de voltaje negativas.
- **Controlador Principal:** Es el procesador principal del SVC. Este controlador lee la señal de voltaje proveniente de la barra AT o BT, luego la compara con el voltaje de referencia y realiza la rutina de control programada. La orden de salida del controlador es el ángulo de disparo de los tiristores y el número de bancos de condensadores conectados, de esta manera controla los reactivos inyectados a la red.
- **PLL:** El bloque PLL (Phase Locked Loop) o "bloque sincronizador de disparos" realiza el encendido de los tiristores según el ángulo calculado por el controlador principal. Para esto se sincroniza con los cruces por cero del voltaje de la red mediante la medida obtenida por los *tpp*.
- **Transformador SVC:** La función de este transformador es reducir el voltaje de la red de alta tensión a una tensión apta para el buen funcionamiento de los tiristores, reactores y condensadores. La configuración de la conexión del enrollado de baja tensión generalmente es tipo delta, ya que así se elimina la tercera armónica de las líneas (sección 2.5.4).
- **Filtros:** Debido a la rápida conmutación de los tiristores y a las resonancias entre los módulos TCR y TSC es común la generación de armónicas en los equipos SVC. Dejando de lado la 3<sup>a</sup> armónica, las armónicas más influyentes en este caso son la 5<sup>a</sup> y la 7<sup>a</sup> armónica, por lo tanto es necesario sintonizar filtros que eliminen ambas armónicas (capítulo 2.6).



## 2.5.2. Curvas de operación

La susceptancia  $B_{SVC}$  del SVC es:

$$B_{SVC} = B_{TCR_1} + \dots + B_{TCR_n} + B_{TSC_1} + \dots + B_{TSC_n} \quad (2.4)$$

Se debe tener en cuenta que las susceptancias  $B_{TCR_i}$  varían según el ángulo de disparo y que las susceptancias  $B_{TSC_i}$  pueden estar o no estar conectadas. De esto se desprende que el SVC presenta una susceptancia variable que cambia según la demanda de reactivos en la barra de conexión. En la figura 2.9 se muestra como cambia el voltaje según la corriente que circula en el SVC.

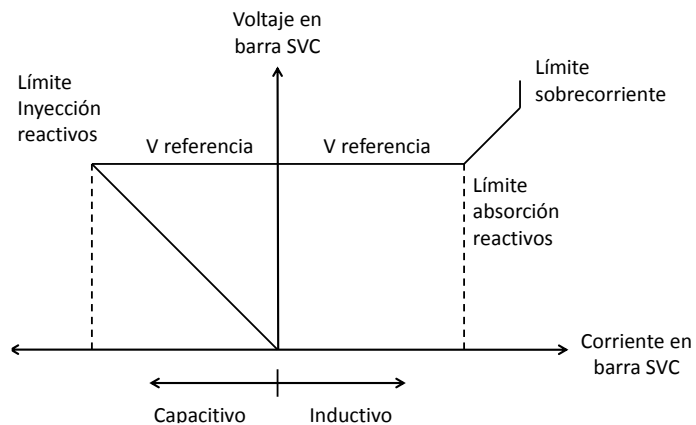


Figura 2.9: Curva de operación de un SVC

Dentro de los límites de inyección y absorción de reactivos el SVC es capaz de mantener el voltaje de la barra según el voltaje de referencia. Si se alcanza el límite de inyección de reactivos, a medida que aumenta la demanda de reactivos, el voltaje cae gradualmente junto con la compensación de reactivos ya que los reactivos entregados por los condensadores del TSC son proporcionales al voltaje de conexión.

Al contrario, si se alcanza el límite de absorción de reactivos, al aumentar los reactivos circulantes el voltaje de conexión sube gradualmente aumentando también la corriente reactiva entregada por los módulos TCR. Estos módulos poseen un límite de sobrecorriente, al alcanzar este límite el control del CER reacciona controlando el ángulo de disparo para disminuir la corriente por el TCR [5].

## 2.5.3. Variables de control

La lógica de control del SVC dependerá del objetivo a controlar en la red. La estrategia de control más común es tener una referencia de voltaje, y mediante algoritmos de control obtener en la red un voltaje cercano o igual al de la referencia. El control retroalimentado o lazo cerrado es el más común en estos equipos ya que permite actuar al SVC frente a contingencias en la red. El controlador más común es el proporcional integral (PI) [5] [16] como el de la Figura 2.10.

La variable manipulada es el ángulo de disparo de los tiristores del módulo TCR, mientras que la variable controlada es el voltaje en la barra del SVC. Es primordial poseer una medición de calidad sin ruido, para esto los transformadores de potencial deben ser robustos ante subidas de potencial y evitar saturaciones.

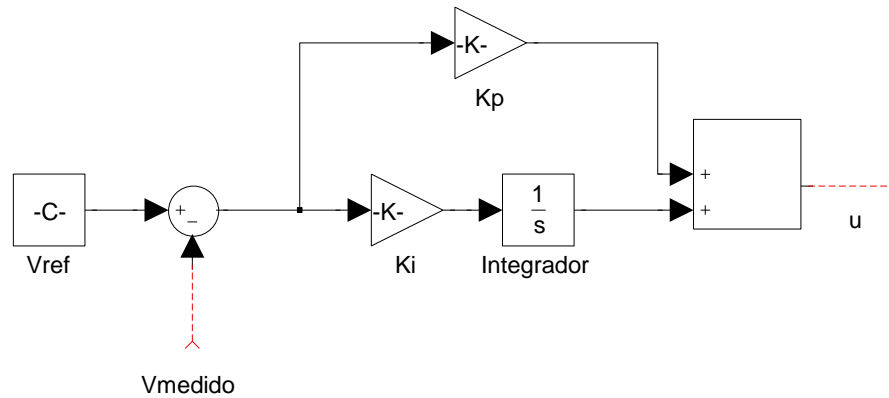


Figura 2.10: Controlador proporcional integral

Otra forma de control, menos común pero no menos interesante, es el control difuso (*fuzzy control*). En vez de seguir una función de transferencia PI, el control difuso toma decisiones en base a reglas lógicas para actuar en la variable manipulada. En la Figura 2.11 se ve un ejemplo de las reglas lógicas implementadas para un controlador de SVC. La ventaja de este controlador es que se puede implementar sin necesariamente saber la función de transferencia del fenómeno físico que ocurre en el TCR.

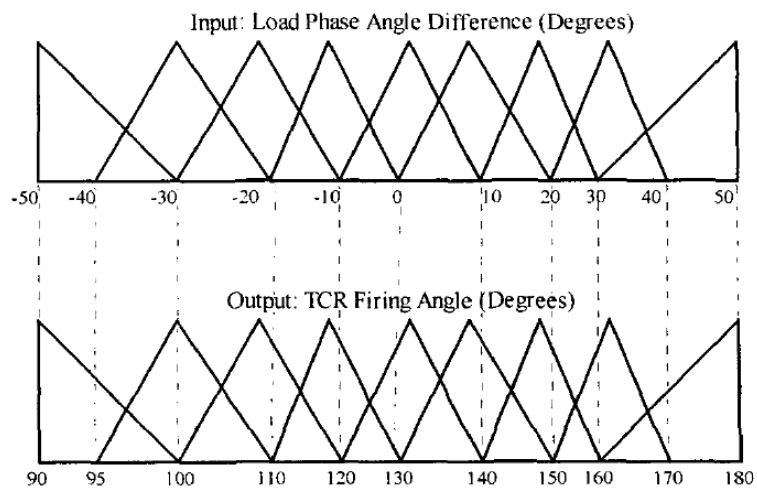


Figura 2.11: Reglas lógicas en un controlador difuso para SVC [23]

### 2.5.4. Corriente en el TCR

Ya se vió en la sección 2.2 el comportamiento del voltaje en una carga resistiva controlada por tiristores. Sin embargo el TCR presenta una carga resistiva-inductiva, prácticamente inductiva pura (reactores reales), por lo tanto la forma de onda de la corriente en los reactores del TCR presentarán un desfase practicamente de  $90^\circ$  respecto al voltaje. Además se debe estudiar el espectro de armónicas generado según el ángulo de disparo en los tiristores.

Se puede usar un análisis de las series de Fourier para obtener el voltaje y la corriente en una carga resistiva-inductiva en función del ángulo de disparo  $\alpha$  y un ángulo de extinción  $\beta$  [14]. Luego, para una carga inductivo-resistiva se tiene el siguiente desarrollo:

$$v_0(t) = V_{cd} + \sum_{n=1,2,\dots}^{\infty} a_n \cos(n\omega t) + \sum_{n=1,2,\dots}^{\infty} b_n \sen(n\omega t) \quad (2.5)$$

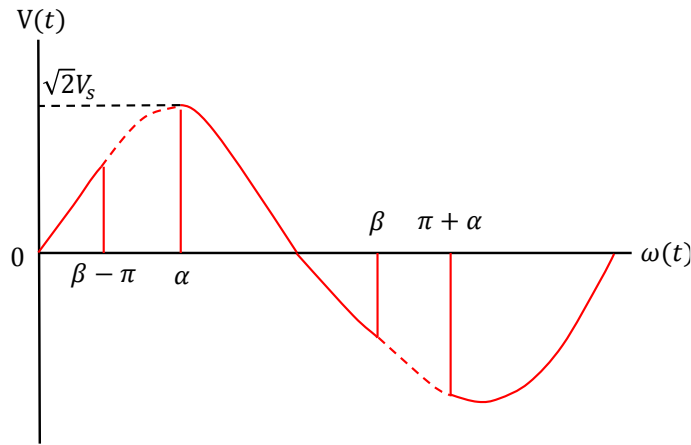


Figura 2.12: Forma de onda del voltaje en el TCR

donde

$$\begin{aligned} V_{cd} &= \frac{1}{2\pi} \int_0^{2\pi} V_m \sen(\omega t) d(\omega t) = 0 \\ a_n &= \frac{1}{\pi} \left[ \int_{\alpha}^{\beta} \sqrt{2}V_s \sen(\omega t) \cos(n\omega t) d(\omega t) + \int_{\pi+\alpha}^{\pi+\beta} \sqrt{2}V_s \sen(\omega t) \cos(n\omega t) d(\omega t) \right] \\ &= \frac{\sqrt{2}V_s}{2\pi} \left[ \frac{\cos(1-n)\alpha - \cos(1-n)\beta + \cos(1-n)(\pi+\alpha) - \cos(1+n)(\pi+\beta)}{1-n} \right. \\ &\quad \left. + \frac{\cos(1+n)\alpha - \cos(1+n)\beta + \cos(1+n)(\pi+\alpha) - \cos(1+n)(\pi+\beta)}{1+n} \right] \end{aligned} \quad (2.6)$$

para  $n = 3, 5, \dots$  y  $a_n = 0$  para  $n = 2, 4, \dots$

$$\begin{aligned}
b_n &= \frac{1}{\pi} \left[ \int_{\alpha}^{\beta} \sqrt{2}V_s \text{sen}(\omega t) \sin(n\omega t) d(\omega t) + \int_{\pi+\alpha}^{\pi+\beta} \sqrt{2}V_s \text{sen}(\omega t) \text{sen}(n\omega t) d(\omega t) \right] \\
&= \frac{\sqrt{2}V_s}{2\pi} \left[ \frac{\text{sen}(1-n)\beta - \text{sen}(1-n)\alpha + \text{sen}(1-n)(\pi+\beta) - \text{sen}(1-n)(\pi+\alpha)}{1-n} \right. \\
&\quad \left. - \frac{\text{sen}(1+n)\beta - \text{sen}(1+n)\alpha + \text{sen}(1+n)(\pi+\beta) - \text{sen}(1+n)(\pi+\alpha)}{1+n} \right]
\end{aligned} \tag{2.7}$$

para  $n = 3, 5, \dots$  y  $b_n = 0$  para  $n = 2, 4, \dots$

$$\begin{aligned}
a_1 &= \frac{1}{\pi} \left[ \int_{\alpha}^{\beta} \sqrt{2}V_s \text{sen}(\omega t) \cos(\omega t) d(\omega t) + \int_{\pi+\alpha}^{\pi+\beta} \sqrt{2}V_s \text{sen}(\omega t) \cos(\omega t) d(\omega t) \right] \\
&= \frac{\sqrt{2}V_s}{2\pi} [\text{sen}^2\beta - \text{sen}^2\alpha + \text{sen}^2(\pi+\beta) - \text{sen}^2(\pi+\alpha)]
\end{aligned} \tag{2.8}$$

$$\begin{aligned}
b_1 &= \frac{1}{\pi} \left[ \int_{\alpha}^{\beta} \sqrt{2}V_s \text{sen}^2(\omega t) d(\omega t) + \int_{\pi+\alpha}^{\pi+\beta} \sqrt{2}V_s \text{sen}^2(\omega t) d(\omega t) \right] \\
&= \frac{\sqrt{2}V_s}{2\pi} \left[ 2(\beta - \alpha) - \frac{\text{sen}(2\beta) - \text{sen}(2\alpha) + \text{sen}2(\pi+\beta) - \text{sen}2(\pi+\alpha)}{2} \right]
\end{aligned} \tag{2.9}$$

La impedancia de la carga es

$$Z = R + j(n\omega L) = [R^2 + (n\omega L)^2]^{\frac{1}{2}} \angle \theta_n$$

Dividiendo la ecuación 2.5 por la expresión de impedancia de la carga se obtiene:

$$i_0(t) = \sum_{n=1,3,5,\dots}^{\infty} \sqrt{2}I_n \text{sen}(n\omega t - \theta_n + \phi_n) \tag{2.10}$$

donde  $\theta_n = \tan^{-1}(n\omega \frac{L}{R})$ ,  $\phi_n = \tan^{-1}(\frac{a_n}{b_n})$  y el valor RMS de la corriente es

$$I_n = \frac{1}{\sqrt{2}} \frac{(a_n^2 + b_n^2)^{\frac{1}{2}}}{[R^2 + (n\omega L)^2]^{\frac{1}{2}}} \tag{2.11}$$

De la ecuación 2.11 se desprende que los armónicos en la corriente dependerán no solo de los componentes  $R$  y  $L$  de la carga, sino también de los ángulos de disparo y extinción:  $\alpha$  y  $\beta$ . Como se vio en la figura 2.2 que el ángulo de extinción para una carga resistiva es  $\beta = 180^\circ$ . Sin embargo, para una carga inductiva pura, debido al desfase entre corriente y voltaje, el ángulo de extinción puede alcanzar ángulos de  $\beta = 270^\circ$ . Por lo tanto el ángulo de disparo queda acotado ente  $90^\circ$  y  $180^\circ$ .

A continuación se muestra cómo varían las distintas armónicas según el ángulo de disparo en los tiristores en un TCR (Figura 2.13).

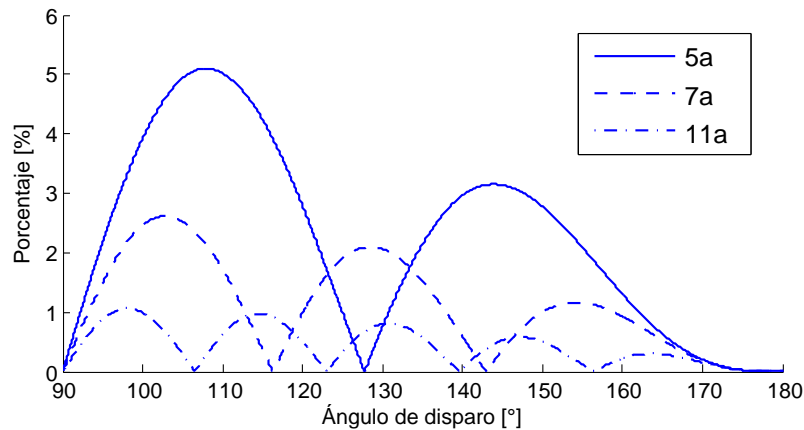


Figura 2.13: Porcentaje de armónicos respecto de la corriente fundamental en un TCR según el ángulo de disparo

En la figura 2.13 no se muestra la 3ª armónica ya que debido a la configuración delta del lado de baja tensión del transformador del SVC (figura 2.8) las componentes armónicas múltiplos de 3 (3ª, 9ª, etc..) permanecen circulando alrededor de la delta y no aparecen en la línea [14]. Después de la 3ª las armónicas más relevantes son la 5ª y la 7ª, alcanzando alrededor de un 5% y un 2,5% de la amplitud de la corriente fundamental respectivamente. Es necesario eliminar o reducir en lo posible el efecto de las armónicas debido a las normativas vigentes, por eso se debe estudiar el diseño de filtros de armónicas para casos como este, lo cual es abarcado con detalle en la sección 2.6.

## 2.6. Filtros armónicos

Es común ver en la electrónica de potencia la generación de armónicos no deseados en la red. Esto se debe a la alta frecuencia de conmutación de los tiristores o debido a la resonancia de los elementos de la red. Para ello es necesario filtrar los armónicos en el momento en que son generados para impedir su ingreso a la red y evitar dañar la calidad del suministro. Los filtros más comunes son los de 5ª y 7ª armónica. El trabajo del filtro es presentar una impedancia baja a una corriente de una frecuencia determinada, con el fin de que ésta quede circulando en el filtro cuando entra en resonancia [11]. La 3ª armónica puede ser eliminada del equipo de compensación si sus elementos se conectan en delta. Las configuraciones para filtros más usadas se muestran en la Figura 2.14.

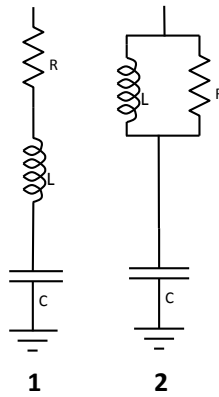


Figura 2.14: Topologías para filtros de armónicas

Para la configuración 1 se tiene la siguiente relación:

$$\begin{aligned}
 X_{cap} &= -\frac{1}{\omega C} \\
 X_{reac} &= \frac{X_{cap}}{h^2} \\
 R &= \frac{X_{reac}(f_{res})}{K} \\
 20 < K < 30
 \end{aligned}
 \tag{2.12}$$

$h$  es la armónica a la cuál se sintoniza el filtro. La impedancia del filtro serie se calcula con la siguiente expresión:

$$\begin{aligned}
 Z &= R + j\left(\omega L - \frac{1}{\omega C}\right) \\
 Z &= R + jX
 \end{aligned}
 \tag{2.13}$$

Si  $X > 0$  entonces el filtro es inductivo, en cambio si  $X < 0$  el filtro es capacitivo.

## 2.7. Simulaciones de un SVC

La simulación es una herramienta poderosa para abordar un problema de diseño sin tener que construir físicamente. Es útil para darse una idea de las dimensiones del producto, y para estudiar el comportamiento de un modelo frente a distintos escenarios. Existen diversos softwares de simulación de sistemas eléctricos, entre los cuales destacan *PSCAD*, *MATLAB Simulink* y *DIGSILENT*.

En la literatura son variados los ejemplos de simulaciones de SVC's [6] [16] [12]. Un ejemplo de simulación es el de la Universidad de AnHui (China) [16]. En este caso el software utilizado es MATLAB *Simulink* para estudiar el comportamiento de un SVC con una capacidad de reactivos de  $\pm 300$  KVar. En la Figura 2.15 se muestra la red simulada. La carga es resistiva-inductiva (R-L) y está conectada a la red mediante un rectificador trifásico. De esta manera se logra crear una carga trifásica equilibrada en el software.

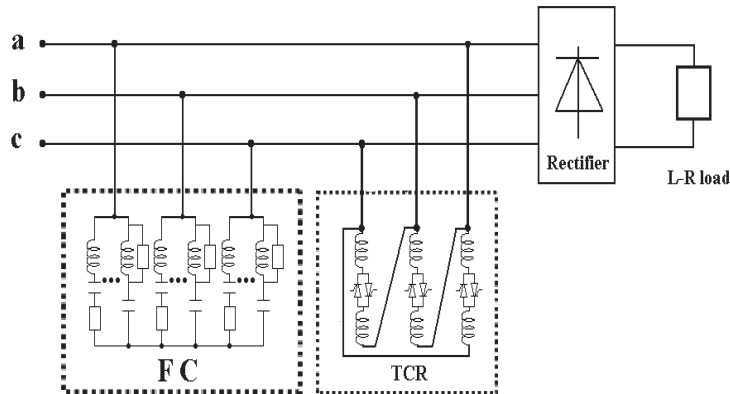


Figura 2.15: Ejemplo de simulación en Simulink [16]

El módulo TCR del SVC se muestra en la Figura 2.16. Consiste en reactancias conectadas a los tiristores y un bloque disparador de pulsos.

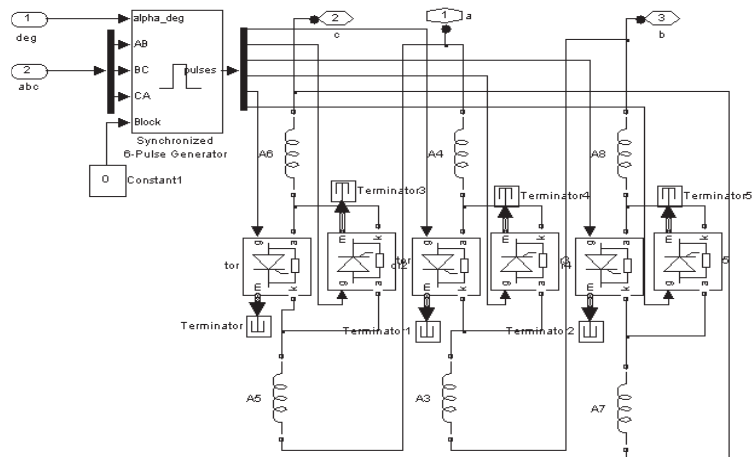


Figura 2.16: Ejemplo del TCR en Simulink [16]

Los resultados se muestran en la Figura 2.17. El SVC es conectado a la red en el segundo 0.1 y la carga es conectada en el segundo 0.2. Se aprecia que el voltaje (2.18, en rojo) permanece constante.

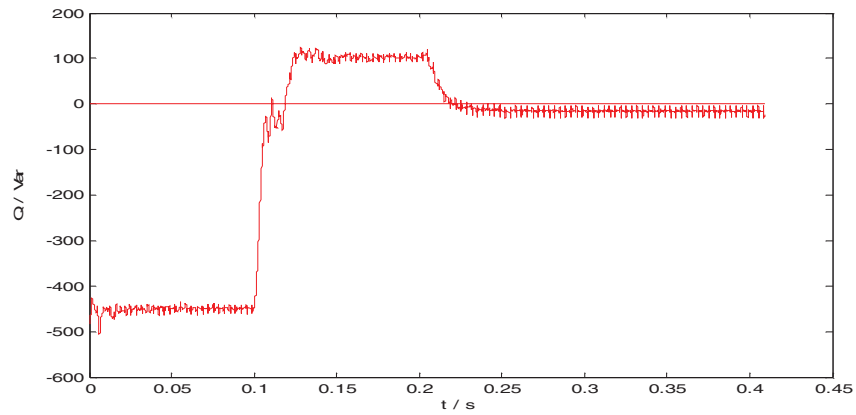


Figura 2.17: Potencia reactiva del ejemplo en Simulink [16]

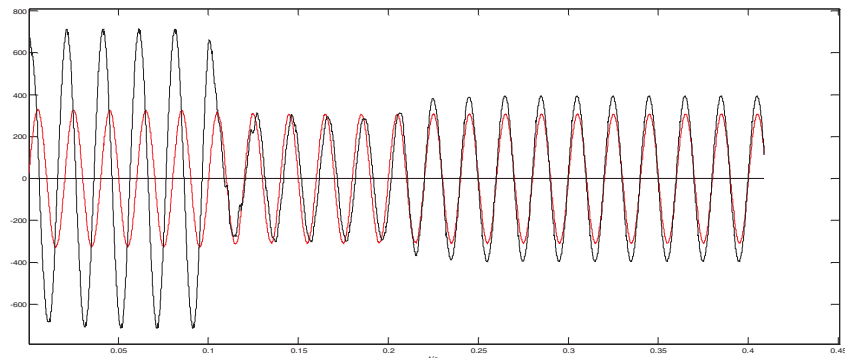


Figura 2.18: Voltaje y corriente del ejemplo en Simulink [16]



## 2.8. Estado de arte de equipos FACTS

### 2.8.1. FACTS a nivel internacional

Los SVC en alta tensión están presentes en gran parte del mundo, siendo sus mayores exponentes las empresas **ABB** y **SIEMENS** [2] [18].

En la subestación Allegheny Power's Black Oak, cerca de Rawlings, Maryland en EE.UU, existe un SVC instalado por ABB el 2007 [1]. El equipo está conectado a la red de 500 kV del sistema PJM (Pennsylvania, Jersey, Maryland) en uno de los puntos más cargados de la red. La capacidad del SVC es de  $-145/+575$  MVar. La instalación tardó 14 meses, un gran logro considerando el tamaño de la unidad y la complejidad de la conexión.



Figura 2.19: Imagen aérea del SVC de Black Oak [1]

En la figura 2.20 se muestra el diagrama unilineal del SVC de Black Oak. Está compuesto por dos módulos TCR de 144 MVar cada uno y de tres módulos TSC también de 144 MVar. Además posee dos filtros para las armónicas 5ª y 7ª. En el lado de alta tensión se tienen dos módulos MSC que se utilizan cuando la respuesta del SVC no es urgente (debido a que son dinámicamente más lentos que los TSC).

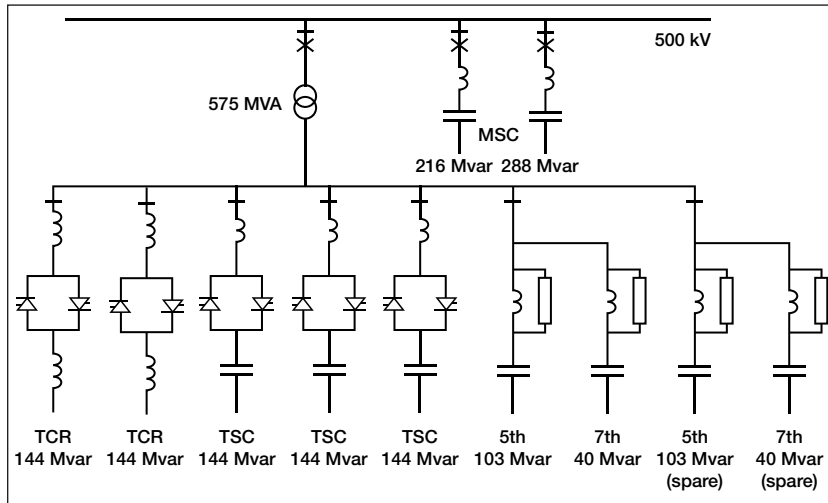


Figura 2.20: Unilineal del SVC de Black Oak [1]

En las subestaciones Impala (275 kV), Illovo (275 kV) y Athene (400 kV), en Richards Bay, Sudáfrica, existen SVC instalado por SIEMENS desde el año 1995. La capacidad de cada unidad es de -300/+100 MVar.



Figura 2.21: Imagen del SVC de Impala [18]

Este diagrama unilineal (figura 2.22) es diferente al de Black Oak. Consiste en dos módulos TCR de 150 MVar y posee tres filtros de armónicas, que a su vez realizan compensación de reactivos permanente (no son conmutables).

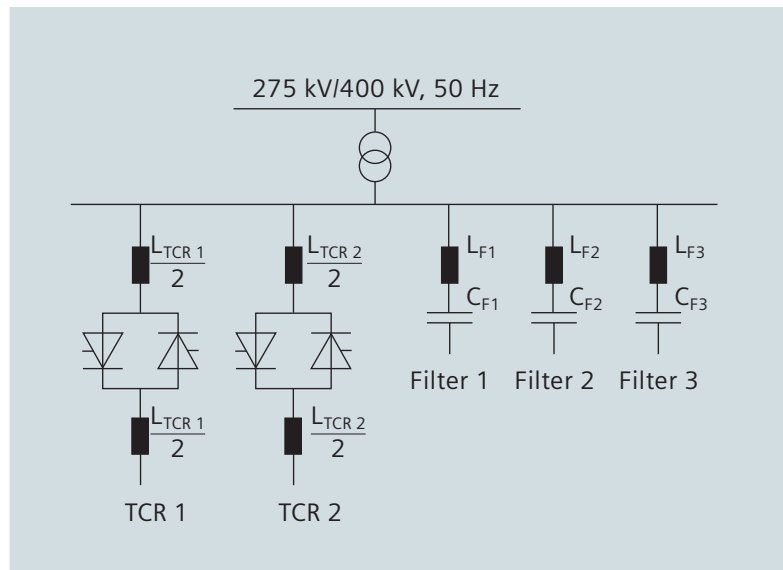


Figura 2.22: Unilineal del SVC de Impala [18]

### 2.8.2. FACTS a nivel nacional

Debido a la geografía en Chile la topología en el SIC es principalmente radial. Es sabido que en las topologías radiales priman los problemas de voltaje, debido a las largas líneas que se traducen en mayores caídas de voltaje. Es por eso que el SIC posee 5 equipos SVC's funcionando a lo largo del país (propiedad de Transelec y fabricados por ABB [3]) conectados al sistema troncal de transmisión (Tabla 2.1). Los más nuevos (año 2000 en adelante) son los SVC's Puerto Montt, Polpaico y Cerro Navia.

Unidad	Tensión de conexión	Absorción reactivos	Entrega reactivos
SVC 'Pan de Azúcar'	220 kV	80 MVar	48 MVar
SVC 'Maitencillo'	220 kV	40 MVar	24 MVar
SVC 'Puerto Montt'	220 kV	40 MVar	70 MVar
SVC 'Polpaico'	220 kV	65 MVar	100 MVar
SVC light 'Cerro Navia'	220 kV	65 MVar	140 MVar

Tabla 2.1: SVC's instalados en el SIC

Ambas subestaciones Polpaico y Cerro Navia están cercanas a la capital Santiago de Chile, donde se concentra la mayor parte de la carga del SIC. Gracias a ambos equipos FACTS la capacidad de transmisión aumento desde 1.400 MW hasta 1.600 MW en la línea de 500 kV entre Ancoa y Alto Jahuel [3]. En la Figura 2.24 se observa que el SVC de Polpaico contiene un módulo TCR y un módulo FC, que además contiene filtros de armónicas.

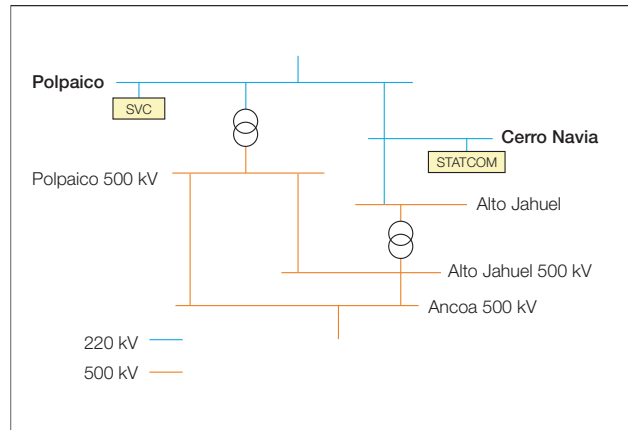


Figura 2.23: Conexión de los SVC's Polpaico y Cerro Navia [3]

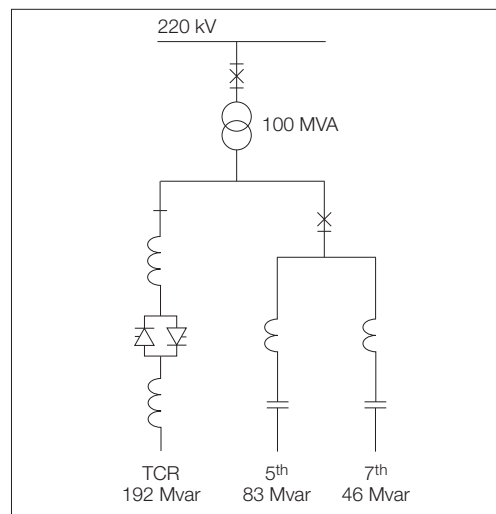


Figura 2.24: Diagrama unilineal SVC Polpaico [3]

En el diagrama unilineal de la figura 2.24 se muestra el módulo TCR de 192 MVar y los filtros de armónicas ( $5^a$  y  $7^a$ ) que a su vez realizan la compensación reactiva. Es importante notar que los filtros no son conmutables, por lo tanto la compensación reactiva capacitiva es fija y debe ser coordinada con el TCR para evitar sobrevoltajes en la barra. Si bien el equipo FACTS de Cerro Navia recibe el nombre de 'SVC Light', este equipo técnicamente corresponde a un STATCOM debido a su tecnología empleada [4]. Los reactores son controlados por un VSC (Voltage Source Converter), el cual está compuesto por IGBT's controlados por pulsos PWM. Esto permite crear una onda de voltaje, similar a un generador, y generar reactivos independientemente del voltaje de la red, obteniendo así mayor confiabilidad en el sistema.

### 2.8.3. FACTS a escala

En diversos laboratorios de universidades en el mundo se han desarrollado equipos FACTS a escala con fines pedagógicos o de investigación [5] [8] [10] [19] [22] .

En la universidad de Tsinghua (China) se implementó un SVC a escala que consiste en un módulo TCR y un módulo TSC [22]. Éstos son controlados por un microcontrolador TMS320C32 (DSP) el cual se comunica con un computador para mostrar los datos en una interfaz de usuario.

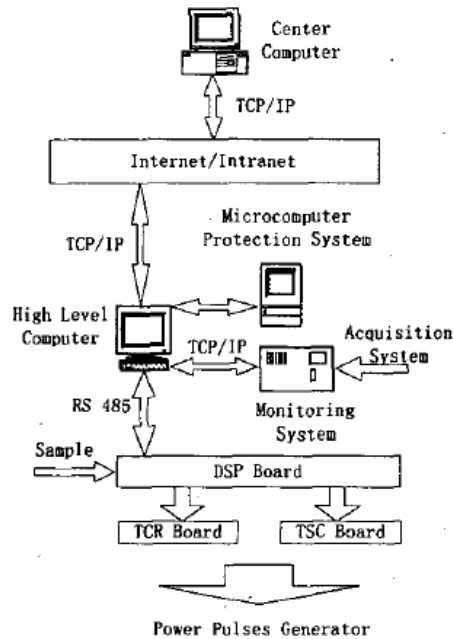


Figura 2.25: Esquema lógico del funcionamiento del SVC de Tsinghua [22]

El computador se comunica mediante RS485 con el microcontrolador, esto permite visualizar las variables de monitoreo y cambiar si se desea los algoritmos de control del DSP. También en la Universidad de Chile se implementó un SVC en el laboratorio de energía del Departamento de Ingeniería Eléctrica [10] [5]. Este equipo está compuesto por un módulo TCR y dos módulos FC, los cuales tienen una capacidad de abosrción de 660 VAR y una entrega de 740 VAR, conectado a la red de distribución de  $380 V_{ff}$ . El esquema de control de este SVC se muestra en la figura 2.26.

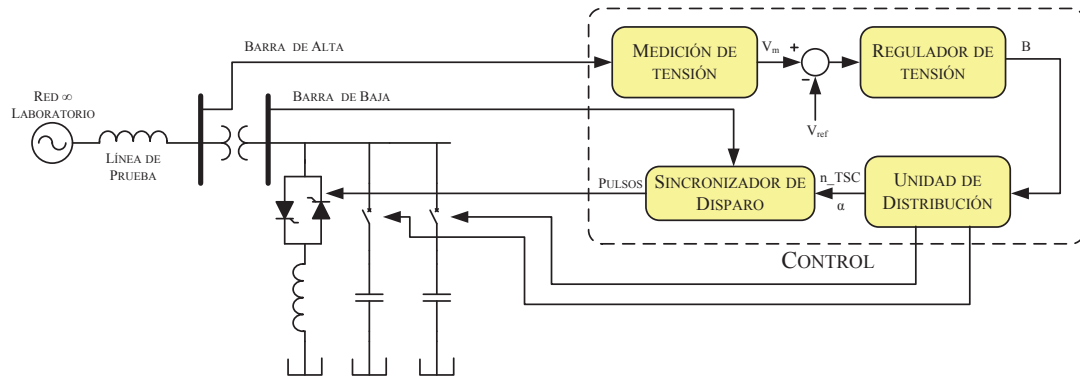


Figura 2.26: Esquema del SVC del DIE [5]

El control principal es realizado por un PIC 16F877A y el disparo de los tiristores es sincronizado por el PIC 18F242. El monitoreo se puede realizar mediante un LCD instalado en el SVC o por un computador conectado mediante RS232 al PIC principal, usando el software MATLAB *Simulink* (Figura 2.27).

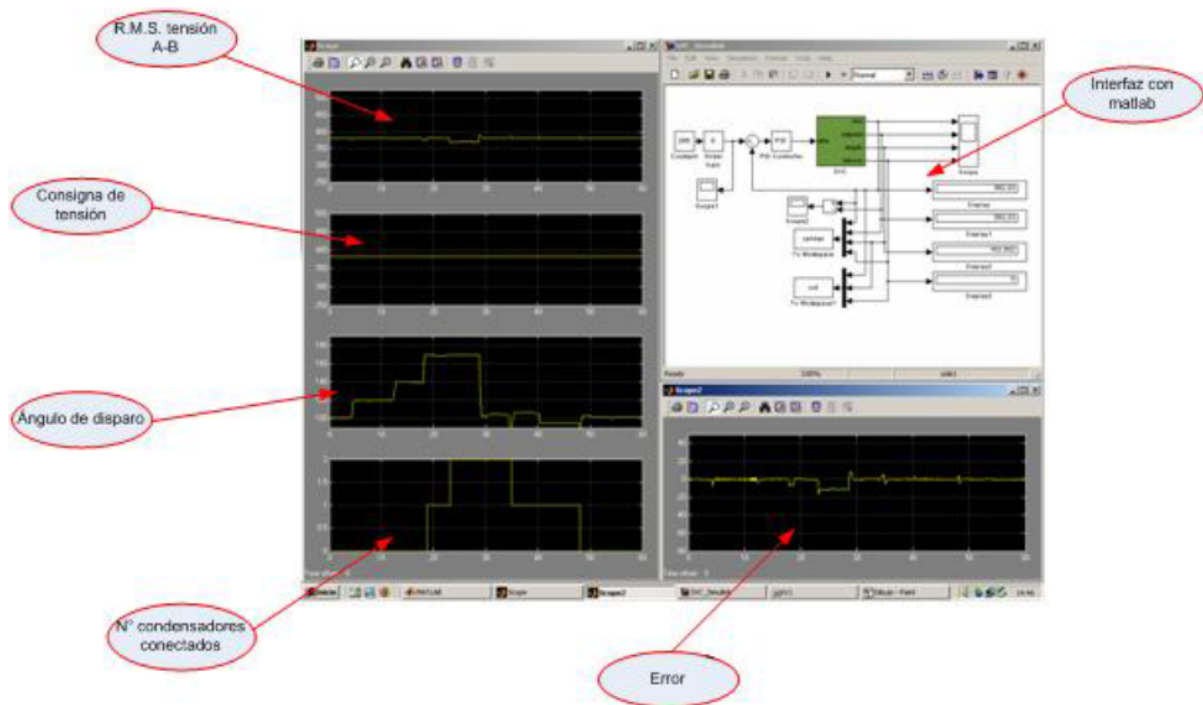


Figura 2.27: Interfaz del CER del DIE [5]

## Capítulo 3

# Introducción a Modbus y DSP

### 3.1. Introducción

Con el auge de los semiconductores y los microprocesadores se han automatizado o controlado diversos procesos que antes eran difíciles o imposibles de controlar manualmente. En particular, para el monitoreo de la red eléctrica, la frecuencia de la red es de 50 Hz por lo que se requiere de una tecnología veloz que alcance a reaccionar frente a contingencias en la red en el lapsus de un par de ciclos, del orden de los milisegundos. Actualmente en el mercado se tiene acceso a microprocesadores capaces de realizar una acción en el orden de los nanosegundos. Gracias al fácil acceso a estos controladores es posible emular prototipos a escala de tecnologías de equipos FACTS que se utilizan actualmente en la industria.

La comunicación de los microprocesadores es un tema no menor, debido a la gran cantidad de equipos que puede haber en una misma micro-red. En este caso puede resultar complicado recolectar información de cada equipo si no se dispone de un protocolo de comunicación. En particular se tiene el protocolo Modbus, un estándar común a nivel industrial por su fácil implementación y entendimiento.

## 3.2. Protocolo de comunicaciones Modbus

### 3.2.1. Origen

Modbus es un protocolo creado para comunicar diferentes dispositivos electrónicos. Se basa en la comunicación maestro-esclavo. Tiene su origen en 1978 desarrollado por la empresa *Modicon* con el fin de comunicar distintos controladores lógicos programables (PLC). Con el paso del tiempo el protocolo fue adoptado por distintas empresas dedicadas al control automático, convirtiéndose en el primer estándar de comunicación de la industria de la automatización [17].

### 3.2.2. Descripción del protocolo

El protocolo Modbus define una estructura de mensajes que es independiente de la interfaz física de comunicación. En la tabla 3.1 se muestran en orden los 4 componentes que definen un mensaje.

Campo	Descripción
<b>Dirección</b>	Identificador del receptor
<b>Función</b>	Código que define el tipo de mensaje
<b>Datos</b>	Bloque de datos o información
<b>Chequeo de error</b>	Función que comprueba errores de comunicación

Tabla 3.1: Estructura de mensajes en Modbus

Las conversaciones son iniciadas por el (los) maestro de la red Modbus. El esclavo responde según el contenido del mensaje. La dirección del mensaje define cual esclavo es el que debe responder.

Existen 4 tipos de datos en Modbus, descritos a continuación:

- **Holding Registers:** Registro de 16 bits. Se puede usar como escritura/lectura. Numerado como registros 4xxxx. (40001, 40002, etc..).
- **Input Registers:** Registro de 16 bits. Sólo puede ser usado como lectura. Generalmente usado para las entradas análogas. Numerado como 3xxxx.
- **Inputs:** Registro de lectura binario. Usado como lectura de encendido/apagado. Numerado como 1xxxx.



- **Coils:** Registro binario de escritura/lectura. Usado para el encendido/apagado de relés, luces, entre otros. Numerado como 0xxxx.

Las funciones en Modbus también tienen un estándar. En la tabla 3.2 se muestran las más usadas en la industria:

Función	Descripción
<b>01</b>	Leer estado Coil
<b>02</b>	Leer estado Input
<b>03</b>	Leer registro Holding
<b>04</b>	Leer registro Input
<b>05</b>	Escribir estado Coil
<b>06</b>	Escribir registro Holding
<b>15</b>	Escribir múltiples Coils
<b>16</b>	Escribir múltiples registros Holding

Tabla 3.2: Código de las funciones en Modbus

El protocolo Modbus tiene 3 formatos de implementación: Modbus RTU, Modbus ASCII y Modbus TCP. En el presente trabajo se implementó el primer formato, cuyas características técnicas se muestran en la tabla 3.3.

	Modbus RTU
<b>Caracteres</b>	Binario: 0...255
<b>Chequeo Error</b>	CRC: Comprobación redundante cíclica
<b>Inicio</b>	3.5 caracteres de silencio
<b>Término</b>	3.5 caracteres de silencio
<b>Bits</b>	8
<b>Paridad</b>	Par/Impar/Ninguna
<b>Stop Bits</b>	1 ó 2

Tabla 3.3: Características técnicas de Modbus RTU

Para aclarar los conceptos previos se ilustrará a continuación un ejemplo de Modbus, que consiste en la petición de un maestro y la respuesta del dispositivo esclavo. En la tabla 3.4 se ve que el maestro le pide al dispositivo número 5 la información contenida en los 2 primeros registros Inputs.

Código	Descripción
<b>05</b>	Dirección esclavo
<b>04</b>	Función (leer registro Input)
<b>00</b>	Comienzo de los registros (alto)
<b>00</b>	Comienzo de los registros (bajo)
<b>00</b>	Número de registros (alto)
<b>02</b>	Número de registros (bajo)
<b>B1</b>	CRC (alto)
<b>C8</b>	CRC (bajo)

Tabla 3.4: Ejemplo de petición de Maestro en Modbus

En este caso el comienzo de los registros es 0 (0x0000 en hexadecimal), esto quiere decir que el esclavo comenzará la lectura desde el registro 30001. En este caso se requiere la información de los dos primeros registros, los cuales son el 30001 y el 30002. En la tabla 3.5 se muestra la respuesta del dispositivo número 5. Siempre que la petición del maestro haya sido correctamente leída el esclavo responde el eco de su dirección y la función pedida por el maestro. En este caso el esclavo devuelve como eco 05 (su dirección) y 04 (la función pedida). Luego envía el número de bytes leídos. Cabe recordar que los Input registers tienen un tamaño de 16 bits, por lo tanto su información es guardada en 2 bytes de 8 bits cada uno. En este caso se leyeron 2 registros lo que corresponde a 4 bytes (04). Después el esclavo muestra el contenido en cada registro, comenzando por el byte más significativo. En este ejemplo, en el registro 30001 está guardado el número 161 y en el registro 30002 el número 494. Finalmente envía la respuesta correspondiente al CRC.

Código	Descripción
<b>05</b>	Dirección esclavo
<b>04</b>	Función (eco de la función pedida)
<b>04</b>	Número de bytes leídos (2*2 registros)
<b>00</b>	Contenido del primer registro (alto)
<b>A1</b>	Contenido del primer registro (bajo) (A1=161 en decimal)
<b>01</b>	Contenido del segundo registro (alto)
<b>FF</b>	Contenido del segundo registro (bajo)(FF=255)
<b>C2</b>	CRC (alto)
<b>E4</b>	CRC (bajo)

Tabla 3.5: Ejemplo de respuesta de un esclavo en Modbus

El protocolo Modbus se puede implementar en distintos canales físicos de comunicación, entre los cuales destacan: RS-232, RS-485 y Ethernet.

### 3.3. Especificaciones generales del microcontrolador

En el presente Trabajo de Título se utilizó el microcontrolador TMS320F28335 fabricado por *Texas Instrument*. Se eligió este microcontrolador por su gran velocidad de procesamiento necesaria para poder medir, procesar y enviar información en el transcurso de un ciclo de la red eléctrica (20 ms). Las características más importantes se muestran a continuación:

- 150 MHz de velocidad de procesamiento (6.67-ns por ciclo de trabajo).
- Operación en punto flotante.
- 18 salidas con Modulación por Ancho de Pulso (PWM).
- 3 relojes de 32 Bits.
- Puertos de comunicación: SCI (UART), SPI, CAN, I2C.
- 16 Canales de conversores análogo/digital (ADC).
- 88 puertos entrada/salida (GPIO).
- 8 pines de Interrupción Externa (XINTF)
- 256K x 16 en memoria Flash, 34K x 16 en memoria RAM.

La programación del microcontrolador se realiza en lenguaje C/C++ en la interfaz *Code Composer Studio*.

### 3.4. Periféricos del microcontrolador

De los periféricos mencionados anteriormente se describen a continuación sólo los utilizados en la rutina de control del SVC.

#### 3.4.1. Puertos digitales Entrada/Salida (GPIO)

Estos puertos son básicamente pines digitales que se pueden implementar como entradas o salidas, en donde el estado "Alto" es 3.3 V y el estado "Bajo" es 0 V. El modo salida es útil para encender/apagar leds de operación y transistores, los cuales permiten energizar equipos de potencias mayores. El modo entrada se utiliza generalmente para leer pulsadores o entradas paralelas.

### 3.4.2. Interfaz de Interrupción Externa (XINTF)

El DSP permite hasta 8 pines configurados para interrupción externa, los cuales deben estar entre GPIO00 y GPIO63. El pin puede ser configurado para detectar flanco ascendente, flanco descendente o ambos. Por ejemplo, si el pin se configura para flanco ascendente y luego se somete a un cambio de 0 a 3 V, gatillará enseguida a la rutina de interrupción asociada.

### 3.4.3. Conversor análogo digital (ADC)

Este puerto permite leer voltajes entre 0 y 3 V de manera digital, con una precisión de 12 bits. Es útil para leer distintos tipos de sensores (temperatura, humedad, fotoresistencias, entre otros) y para leer corrientes y voltajes de una red eléctrica. Es necesaria una etapa de reducción de voltaje si se desea medir dicha variable en la red eléctrica de distribución (220 V).

### 3.4.4. Puerto de comunicación serial (SCI o UART)

El puerto de comunicación serial está formado por 3 pines: Transmisión (Tx), Recepción (Rx) y Tierra (Gnd). Este puerto puede conectarse a la interfaz RS-232 para comunicarse con un computador o también a la interfaz RS-485 para conectarse a una red Modbus. Para la conexión física es necesario un transceptor debido a la diferencia de voltajes existente entre el DSP y las redes de comunicación.

## 3.5. Estructura y funcionamiento de un programa

Las rutinas de programación del DSP se pueden dividir en 6 partes, mostradas en orden en la figura 3.1.

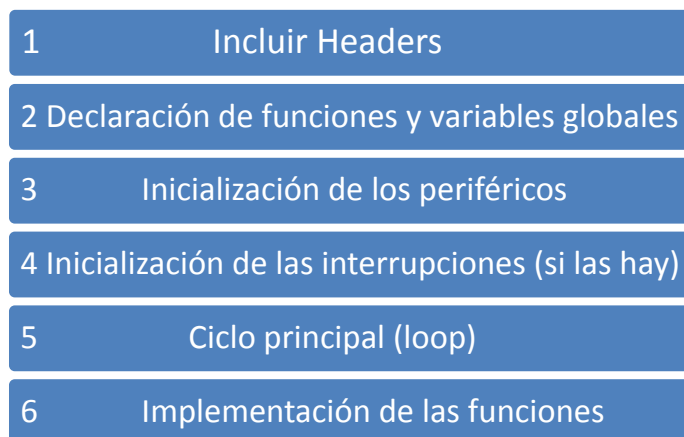


Figura 3.1: Estructura del código

La primera parte consiste en incluir los *Headers*, los cuales son archivos con las funciones básicas para configurar el microcontrolador (los headers varían según el modelo). La segunda parte del código contiene las declaraciones de las funciones utilizadas en el programa y las variables globales que se utilizarán en el ciclo principal o en las funciones. La tercera parte corresponde a la configuración y puesta en marcha de los periféricos que se utilizarán en la rutina del DSP. Luego viene la configuración de las interrupciones. Existen diversos tipos de interrupciones las cuales se dividen en dos categorías, internas y externas. Las más comunes son: por reloj (interna), por UART (externa), por XINTF (externa), por ADC (interna). No siempre se utilizan las interrupciones, sin embargo su uso permite abordar distintas tareas en paralelo de manera eficiente. La quinta parte consiste en el ciclo principal del código, que generalmente es un ciclo *while true*, en donde se realizará la tarea implementada de manera periódica. La parte final del código contiene la implementación de las funciones declaradas al comienzo de la rutina.

Una vez compilado el código, el DSP realiza el ciclo de trabajo de acuerdo a la rutina programada. El flujo del ciclo de trabajo se ve en la figura 3.2.

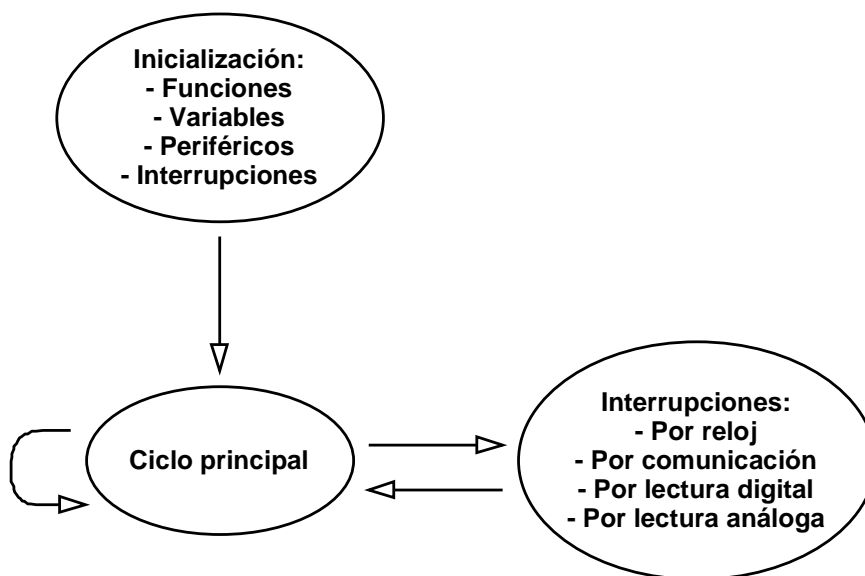


Figura 3.2: Ciclo de Trabajo del DSP

La inicialización se realiza al comienzo de la rutina y este paso ocurre sólo una vez. Por esto hay que tener en cuenta la configuración de los periféricos, ya que es difícil volver a configurarlos estando en el ciclo principal. El ciclo principal ocurre periódicamente, la frecuencia del ciclo depende de la cantidad de tareas que se hayan programado y de los retardos (*delays*) incorporados. Cada vez que ocurre una interrupción, el DSP deja de hacer lo que estaba haciendo en el ciclo principal y atiende a la rutina de interrupción.

Una vez finalizada la rutina de interrupción, el microcontrolador retoma la tarea que estaba realizando. De esta manera se pueden realizar tareas en paralelo, siempre y cuando la duración del ciclo de trabajo de la rutina de interrupción sea pequeña en comparación a la duración del ciclo de trabajo del loop principal.

Las rutinas de interrupción (ISR: *Interrupt Service Routine*) corresponden a una función como cualquier otra que es llamada cada vez que se detecta la interrupción específica. Estas funciones son declaradas al comienzo el código, y luego son implementadas al final del código, al igual que las otras funciones. Es posible implementar una rutina sólo con interrupciones sin ciclo principal, ya que dentro de las funciones ISR pueden ir implementadas las tareas a realizar.

## 3.6. Ejemplos de aplicación

A continuación se muestran dos ejemplos de programación en el DSP que permiten ver las sentencias típicas de inicialización y configuración. Para los usuarios de un DSP que quieran implementar alguno de estos ejemplos se les recomienda primero leer *SystemFrameworkOverview*, tutorial disponible en *controlSUITE* que explica como crear un proyecto en *Code Composer Studio*.

### 3.6.1. Parpadear un Led

Existen diversas maneras de hacer parpadear un Led con un microcontrolador. En este ejemplo se utiliza el método del retardo dentro del ciclo de trabajo principal. Para prender y apagar el Led se utiliza el pin GPIO34 como salida. Es importante destacar que los pines del DSP entregan una corriente máxima de 4 mA, en caso de exceder se puede dañar el pin. Se explicará el código siguiendo el esquema de la figura 3.1.

La primera parte del código es así:

```
#include "DSP28x_Project.h" // Header del microcontrolador
```

Esta sentencia incluye el header del microcontrolador a utilizar. En este caso el número 28x hace referencia al TMS320F**28335**. Luego viene la declaración de funciones y variables globales.

```
//-----Declaración de funciones y variables globales-----  
void retardo(void);  
int contador = 0; //Contador de retardos
```

Se observa la declaración de la función **retardo**, la implementación de la función se deja para el final del código. Se creó un contador de retardos para ilustrar el manejo de variables. Una vez hechas las declaraciones se procede al tercer paso, la inicialización del DSP:

```

void main(void)
{

//-----Inicialización de los periféricos-----

InitSysCtrl();          // Inicia el PLL y los relojes internos
                        // Se encuentra en DSP2833x_SysCtrl.c

EALLOW;                //Iniciar configuración de los GPIO
    GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0; // 0=GPIO, 1=ECAP1, 2=Resv, 3=Resv
    GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; // 1=OUTput, 0=INput
    GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1; //Inicia en 0 [V]
EDIS;                  //Finalizar configuración de los GPIO

```

La función **InitSysCtrl()** viene por defecto en el compilador CCS y es vital para el funcionamiento del DSP en todas las rutinas. En palabras prácticas es la función que inicia los relojes internos del microcontrolador, los cuales son imprescindibles para cualquier instrucción. Luego viene la configuración de los puertos digitales GPIO. Los registros de estos puertos están protegidos, por eso es necesario el comando **EALLOW**, el cual permite modificar registros protegidos. Para finalizar la escritura de registros bloqueados se termina con el comando **EDIS**. Una vez finalizada la configuración del DSP se procede al ciclo principal.

```

for(;;) //análogo a while (true)
{
    GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;          //Pone en 0[V] el GPIO34
    retardo();
    GpioDataRegs.GPBSET.bit.GPIO34 = 1;           //Pone en 3.3[V] el GPIO34
    retardo();
}
} //Fin de void main(void)

```

El comando **GPBCLEAR** corresponde al apagado del pin, mientras que el **GPBSET** enciende el pin. Cada vez que el pin cambia de estado se llama a la función **retardo()**, de lo contrario el cambio de estado del pin sería tan veloz que no se apreciaría por el ojo humano. Finalmente se procede a implementar las funciones utilizadas en la rutina.

```

void retardo()
{
    ++contador;
    volatile long i;
    for (i = 0; i < 5000000; i++) {} //retardo de 500 [ms]
}
//-----FIN-----

```

### 3.6.2. Enviar mensaje por puerto serial UART

El uso del puerto UART es indispensable para enviar información desde el DSP a un computador u otros microcontroladores. El DSP posee 3 puertos de comunicación serial (SCI-A, SCI-B y SCI-C), en este caso se utilizará el primero, compuesto por los pines GPIO28 y GPIO29. El ejemplo más sencillo para visualizar esta función es mandar el texto "Hola Mundo" desde el DSP y recibirlo en el computador. Para eso se implementó el siguiente código:

```
#include "DSP28x_Project.h"    // Header del microcontrolador

//-----Declaración de funciones y variables globales-----

void scia_configurar(void);
void scia_fifo_iniciar(void);
void scia_emitir(int a);
void scia_msg(char *msg);
void retardo(void);

Uint16 LoopCount; //Contador de mensajes emitidos
```

Se observan 4 nuevas funciones, cuya implementación se muestra al final del código. Las dos primeras funciones tienen como objetivo la configuración de los parámetros técnicos del SCI-A, como la elección del *baudrate*, la paridad y la cantidad de bits a enviar. Las funciones **scia\_emitir** y **scia\_msg** son útiles para el envío de *strings* y *chars*. A continuación se muestra el *main*, que contiene la inicialización de los periféricos y el ciclo principal.

```
void main(void)
{
    char *msg; //Variable auxiliar
              //Arreglo de chars

    InitSysCtrl(); //DSP2833x_SysCtrl.c
    InitSciaGpio(); //Inicia el puerto serial SCI-A
                  //DSP2833x_Sci.c
    scia_fifo_iniciar(); // Inicia el FIFO
    scia_configurar(); // Configura el SCI-A

    LoopCount = 0;
//-----Ciclo principal-----
    for(;;)
    {
        msg = "\r\nHola Mundo!\0";
        scia_msg(msg);
        ++LoopCount;
        retardo();
    }
}
```



```

    }
} //Fin del main

```

La función **InitSciaGpio** también viene por defecto, su función es inicializar los pines GPIO28 y GPIO29 como pines de recepción (Rx) y transmisión (Tx) respectivamente. Luego se llama a las funciones previamente declaradas para la configuración del SCI-A y se procede con el ciclo principal. Se utiliza la variable **msg** para almacenar el string que se desea enviar. Luego el string es enviado con la función **scia\_msg(msg)**.

```

//-----Implementación de las funciones declaradas---
void retardo()
{
    volatile long i;
    for (i = 0; i < 10000000; i++) {}
}
void scia_configurar()
{
    SciaRegs.SCICCR.all =0x0007; // 1 stop bit, No loopback
                                // Sin paridad,8 char bits,
                                // async mode, idle-line protocol
    SciaRegs.SCICTL1.all =0x0003; // Habilitar TX, RX, internal SCICLK,
                                // Disable RX ERR, SLEEP, TXWAKE

    SciaRegs.SCICTL2.all =0x0003;
    SciaRegs.SCICTL2.bit.TXINTENA =1;
    SciaRegs.SCICTL2.bit.RXBKINTENA =1;
    #if (CPU_FRQ_150MHZ)
        SciaRegs.SCIHBAUD    =0x0001; // 9600 baud @LSPCLK = 37.5MHz.
        SciaRegs.SCILBAUD    =0x00E7;
    #endif
    #if (CPU_FRQ_100MHZ)
        SciaRegs.SCIHBAUD    =0x0001; // 9600 baud @LSPCLK = 20MHz.
        SciaRegs.SCILBAUD    =0x0044;
    #endif
    SciaRegs.SCICTL1.all =0x0023; // Relinquish SCI from Reset
}
void scia_emitir(int a) //Transmite un caracter
{
    while (SciaRegs.SCIFFTX.bit.TXFFST != 0) {}
    SciaRegs.SCITXBUF=(int)a;
}
void scia_msg(char * msg) //Transmite un arreglo de caracteres
{
    int i;
    i = 0;
    while(msg[i] != '\0')
    {

```

```

        scia_emitir(msg[i]);
        i++;
    }
}
void scia_fifo_iniciar()
{
    SciaRegs.SCIFFTX.all=0xE040;
    SciaRegs.SCIFFRX.all=0x204f;
    SciaRegs.SCIFFCT.all=0x0;
}
//-----FIN-----

```

Para la configuración del UART se utilizó un *baudrate* de 9600, con transmisión de 8 bits sin paridad como se observa en la configuración de **scia\_configurar**. Esta configuración depende de la velocidad del DSP, que puede ser 100 o 150 MHz en este caso. Para una correcta visualización del mensaje se recomienda usar el software *Hyperterminal*, si no se puede utilizar el monitor serial que viene en el software de *Arduino*. En ambos casos se debe seleccionar el puerto COM al cual está conectado el puerto SCI-A del DSP.

### 3.6.3. Implementación de Modbus en el DSP

Modbus es un protocolo que puede implementarse en cualquier microcontrolador con puerto de comunicación UART. Este puerto puede conectarse a RS-232, RS-485 o Ethernet, dependiendo de la necesidad. En este ejemplo se puede utilizar tanto RS-232 como RS-485. Cabe destacar que para conectar el puerto UART del DSP al puerto RS-232 de un computador se debe usar el transceptor *max3232*, el cual eleva la señal de 3.3 V a 12 V. En caso de no tener puerto RS-232 el computador se debe utilizar además un conversor RS-232 a puerto USB. Si se desea conectar el puerto UART a una red Modbus con RS-485 se debe usar el transceptor *max3485*.

El código para la implementación de Modbus RTU en el TMS320F28335 es extenso en comparación a los ejemplos anteriores. La rutina completa se puede observar en la sección Anexos. A continuación se explicarán solo los conceptos más importantes del código.

```

#include "DSP28x_Project.h"

//-----Declaración de funciones y variables globales-----

void scia_configurar(void);
void EnviarDatosModbus();
void InicializarRegistros();
void LeerCoilStatus();           //Leer bits de cambio de estado
void LeerInputStatus();         //Leer bits de estado de operación
void LeerHoldingRegisters();    //Leer registros Holding
void LeerInputRegisters();      //Leer registros Inputs

```

```

void Escribir_Coils();          //Escribir bits de cambio de estado
void Escribir_Holding();      //Escribir registro Holding
void ModbusExcep(Uint16);
interrupt void scia_recibir(void);    //Interrupción de recepción de datos
interrupt void scia_enviar(void);     //Interrupción de envío de datos
Uint16 AddrCheck(Uint16,Uint16,Uint16); //chequeo de dirección
Uint16 CRC16(Uint16 *, Uint16);      //chequeo cíclico redundante

#define COIL_1                0
#define COIL_2                1
#define COIL_3                2

int* DiscreteInput[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int* HoldingRegister[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int* InputRegister[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

Uint16 address=5;            //dirección del dispositivo
Uint16 Led=0;

```

Se ve la declaración de las funciones Modbus vistas en la tabla 3.2. Cada una de estas funciones se debe implementar de acuerdo al estándar del protocolo. Se declaran además dos funciones de tipo interrupción, una para la recepción de datos y otra para el envío. De esta forma el DSP puede realizar la comunicación en paralelo a sus tareas principales. También es importante implementar la función CRC encargada del chequeo de errores. Luego de las funciones se declaran los registros y los coils. En este caso se declararon 3 coils, los cuales pueden corresponder a 3 leds, relés, entre otros. El largo máximo de los registros y el valor inicial de éstos depende de las necesidades del usuario, en este caso los registros se inician en cero y tienen un largo de 20 espacios. En este ejemplo la dirección del equipo es 5. A continuación se muestra la inicialización de los periféricos y las interrupciones.

```

void main(void)
{
    InitSysCtrl();
    InitSciaGpio();
    DINT;
    InitPieCtrl(); //Inicializa el control de interrupciones
                  //DSP2833x_PieCtrl.c
    IER = 0x0000;
    IFR = 0x0000;
    InitPieVectTable(); //Inicializa el abanico de interrupciones
                       //DSP2833x_PieVect.c
    EALLOW;
    PieVectTable.SCIRXINTA = &scia_recibir; //apunta a la función declarada
    PieVectTable.SCITXINTA = &scia_enviar; //apunta a la función declarada
    EDIS;

```

```

PieCtrlRegs.PIECTRL.bit.ENPIE = 1; // Habilita el bloque PIE
PieCtrlRegs.PIEIER9.bit.INTx1 = 1; // Habilita SCIRXINTA
PieCtrlRegs.PIEIER9.bit.INTx2 = 1; // Habilita SCITXINTA
IER |= M_INT9; // Enable CPU Interrupt 9
EINT; // Enable Global interrupt INTM

EALLOW; //Iniciar configuración de los GPIO
GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0; // 0=GPIO, 1=ECAP1, 2=Resv, 3=Resv
GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; // 1=OUTPUT, 0=INPUT
GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1; //Inicia en 0 [V]
EDIS; //Finalizar configuración de los GPIO

scia_configurar();
InicializarRegistros();
//-----Ciclo principal-----
for(;;){
    if (Led==1) GpioDataRegs.GPBSET.bit.GPIO34 = 1;
    else GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;
}

```

Las funciones **InitPieCtrl()** e **InitPieVectTable()** vienen por defecto en el compilador y son esenciales para el uso de interrupciones tanto internas como externas. En este caso se utilizaron los comandos **PieVectTable** y **PieCtrlRegs** para la configuración de las interrupciones de recepción (SCIRXINTA) y transmisión (SCITXINTA) de datos. Cabe destacar que cada interrupción va ligada a una función declarada al comienzo del código. Luego estas funciones se ejecutarán cada vez que la interrupción sea gatillada. En este ejemplo se configuró el GPIO34 como salida para controlar el estado de un led. Éste se puede encender o apagar mediante una petición del maestro en la red Modbus. Si la variable **Led** es 1 el led se enciende, de lo contrario se apaga. Esta variable es del tipo **Uint16** por lo tanto se debe inicializar como un *holding register* (registro de 16 bits). Mediante la función **Escribir\_Holding()** se puede modificar este registro.

Finalmente se deben implementar las funciones declaradas al comienzo, las cuales se observan en el Anexo. Para verificar la rutina se puede usar algún software de simulación para Modbus. La presente rutina corresponde a un dispositivo esclavo por lo tanto el software debe emular un maestro. Esta rutina fue verificada con éxito mediante el software gratuito *Modbus Tester*.

## Capítulo 4

# Diseño de un CER para laboratorio

### 4.1. Topología

La topología del CER a diseñar se basa en la estructura que posee el equipo de Polpaico (figura 2.24). El equipo posee un filtro para la 5<sup>a</sup> armónica que estará permanentemente conectado. Además posee dos bancos de condensadores para la compensación reactiva cuando sea necesario. Los bancos serán operados mediante relés mecánico. También se tiene un módulo TCR encargado de absorber los reactivos que trabaja coordinadamente con los bancos de condensadores (MSC) teniendo en cuenta la compensación permanente que entrega el filtro. El transformador tiene conexión Y-D y se conecta a la red eléctrica de distribución.

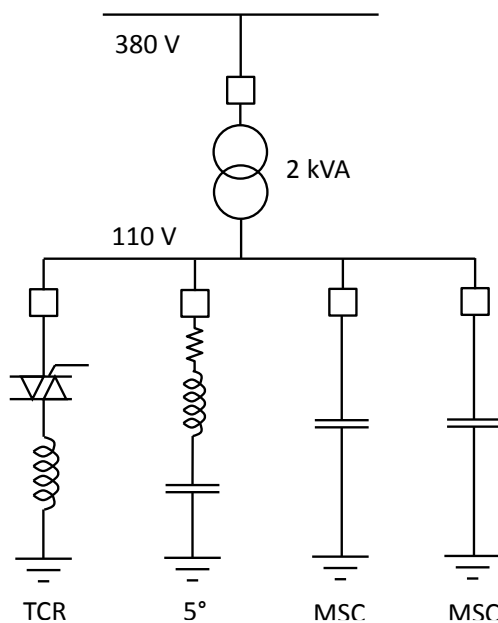


Figura 4.1: Diagrama unilineal del CER a construir

## 4.2. Dimensionamiento del CER

Los componentes a dimensionar en el CER son los siguientes:

- Banco de condensadores (MSC)
- Banco de inductores (TCR)
- Filtro de 5<sup>a</sup> armónica

Para definir el tamaño del filtro y los inductores se definirá primero el tamaño del banco de condensadores, ya que éstos se venden en el comercio con capacidades fijas. El SVC que opera actualmente en el laboratorio de energía del DIE (figura 2.27) posee una capacidad de +660/-740 VAR. Se desea ampliar la capacidad de entrega de reactivos, para ello se debe utilizar condensadores con mayor capacitancia. Si se usan dos bancos con condensadores de 55 uF se obtendría una capacidad de reactivos de 1260 VAR, el cual tendría un 70 % más de capacidad respecto al actual.

Cabe recordar según la figura 2.13 que la 5<sup>a</sup> armónica representa un 5 % de la corriente fundamental. Considerando el banco de 1260 VAR, se tendría una corriente de línea de 3.8 A, por lo tanto el filtro debe tener al menos una capacidad de 65 VAR. Para el diseño del filtro se utilizó la ecuación 2.12. De esta manera el cálculo teórico arroja los siguientes parámetros para el filtro:

Parámetro	Valor	Unidad
<b>Capacitancia</b>	8	uF
<b>Inductancia</b>	50	mH
<b>Resistencia</b>	0,79	$\Omega$
$Q_{3\phi}$	95	VAR capacitivo

Tabla 4.1: Parámetros del filtro de 5<sup>a</sup> armónica

Para el tamaño del banco de inductores se debe tener en cuenta que el filtro entrega reactivos de manera permanente, por lo tanto se debe tener en cuenta la siguiente restricción:

$$Q_{TCR} \leq Q_{filtro} + Q_{banco} \quad (4.1)$$

Luego el módulo TCR debe tener al menos 1245 VAR. Si se utilizan inductancias de 90 mH se tendría un banco de inductancias de 1284 VAR, lo cual cumple con la restricción 4.1. Finalmente se tendría un SVC de capacidad +1284/-1355 VAR.

## 4.3. Diseño de los inductores

En la mayoría de los equipos SVC de potencia se utilizan inductores con núcleo de aire [5], pero para el presente prototipo se utilizó núcleo de hierro debido a que los primeros alcanzan dimensiones no

compatibles para laboratorio. Para el diseño de las inductancias se trabajó con las relaciones conocidas de flujo magnético y reluctancias (sin saturación).

$$\begin{aligned}\phi &= B \cdot A \\ \phi \cdot \mathcal{R} &= N \cdot I \\ \mathcal{R} &= \frac{l}{A \cdot \mu} \\ L &= \frac{N^2}{\mathcal{R}_{eq}}\end{aligned}\tag{4.2}$$

El núcleo a diseñar se muestra en la figura 4.2. La profundidad del núcleo es  $2a$ . Luego el diseño depende del valor de la inductancia  $L$  que se haya obtenido en el dimensionamiento del componente. Este proceso es iterativo, se debe comenzar con datos previamente definidos ( $d, d', a, g$ ) para luego observar las variables resultantes  $N$  y  $\phi$ . El entrehierro  $g$  se debe ir variando hasta encontrar la distancia en la cuál el núcleo no se sature. Es recomendable diseñar un núcleo sobredimensionado para que no se sature al sobrepasar ligeramente la tensión nominal. Es importante tener en cuenta que la bobina debe caber dentro de la ventana del núcleo, de lo contrario se debe disminuir el número de vueltas o agrandar la ventana.

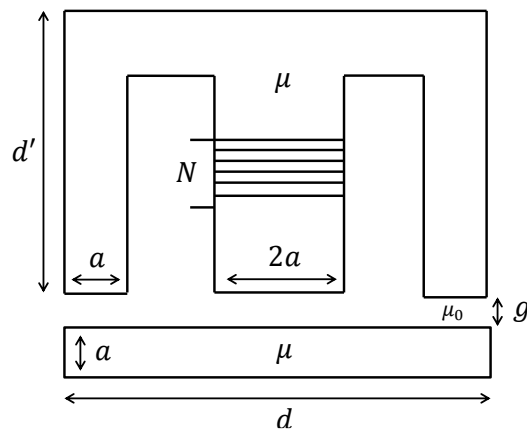


Figura 4.2: Parámetros de diseño del inductor

Realizado el diseño se obtuvieron los resultados que se muestran en la tabla 4.2.

Parámetro	Valor	Unidad
$V$	110	V
$L$	115	mH
$a$	2.25	cm
$d$	13.5	cm
$d'$	9	cm
$g$	1.6	mm
$N$	300	vueltas
$\mu$	$500\mu_0$	$\text{Hm}^{-1}$

Tabla 4.2: Resultado del diseño del inductor

#### 4.4. Transformador de bajada

El transformador de bajada debe ser capaz de no saturarse en el estado "full capacitivo", es decir, cuando el SVC está entregando su máxima cantidad de reactivos. Para esto se debe diseñar un transformador que opere al menos con tensiones de 1.1 pu. La razón de transformación del transformador para el presente proyecto es de 380/110 V entre fases, con conexión Y-D y conformado por 3 unidades monofásicas. En la figura 4.3 se aprecia el esquema de conexión. La capacidad del transformador debe ser superior a la máxima capacidad de entrega del SVC, por lo tanto una capacidad de 1.5 kVA es suficiente para operar de manera holgada. Cada unidad monofásica debe tener una razón de transformación de 220/63.5 V y una capacidad de 500 VA.

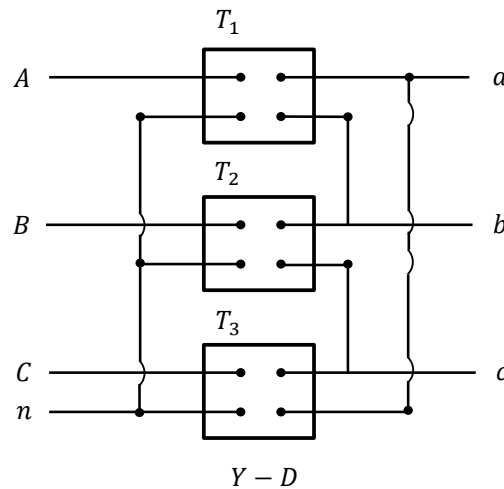


Figura 4.3: Diagrama de conexión del transformador



## **4.5. Controlador principal**

Como se vió en la sección 2.5.1, el SVC requiere un procesador capaz de trabajar las señales medidas para luego efectuar el control en los módulos TCR y MSC. En el presente proyecto se utilizó el procesador TMS320F28335 (DSP) descrito en la sección 3.3. Este microcontrolador fue programado de manera que efectue un control PI con referencia de voltaje, como el descrito en la sección 2.5.3.

## **4.6. Dispositivos de medición**

### **4.6.1. Transductores**

Se utilizaron transductores de voltaje y de corriente, cuyos modelos son LV 20-P y LA 55-P respectivamente, ambos fabricados por LEM. El objetivo de los transductores es obtener una señal proporcional al voltaje y la corriente en la red de manera que pueda ser procesada por la electrónica. El transductor de voltaje tolera hasta 500 V y el de corriente tolera 50 A. Ambos transductores emiten una señal del orden de los miliamperes, la cual debe ser ajustada por un circuito de opamps para ser leída por el DSP.

### **4.6.2. Amplificador operacional**

Se utilizaron amplificadores operacionales para la adaptación de la señal a la salida de los transductores, de manera que la señal quedara entre 0 y 3 V, según el requerimiento del DSP. En específico se utilizó el modelo TLC274 por su velocidad y precisión además de contener 4 opamps en un solo integrado.

### **4.6.3. Optoacoplador**

Se utilizó optoacopladores para el circuito detector de cruce por cero. De esta manera se aísla galvánicamente el circuito de control de posibles fallas en la red. Se utilizó el modelo 4N26 fabricado por FAIRCHILD.

## **4.7. Dispositivos de accionamiento**

### **4.7.1. Tiristores**

El modelo utilizado para los tiristores SCR es el 2N6508 fabricado por ON SEMICONDUCTOR. El máximo voltaje tolerable es de 600 V y la máxima corriente es de 25 A. Este tiristor no necesita alimentación externa, para conducir sólo necesita un pulso de corriente en la compuerta.

### **4.7.2. Optoacoplador de salida triac**

Se utilizaron optoacopladores como intermediarios entre el DSP y los tiristores, ya que su función es aislar eléctricamente el DSP de las corrientes elevadas presentes en el tiristor. De esta manera el optoacoplador recibe la señal de disparo del DSP y luego enciende el tiristor. El modelo usado es MOC3011 fabricado por FAIRCHILD.

### **4.7.3. Buffer XOR**

Se usaron puertas lógicas XOR, específicamente el modelo 74HC86, para proporcionar la corriente necesaria y poder gatillar los transistores y las optocuplas con los pines del DSP. Estos pines proporcionan máximo una corriente de 4 ma, y los dispositivos recién mencionados requieren corrientes de alrededor de 10ma para encender, por lo que se usaron buffers de lógica XOR tal que cuando un pin del DSP está en alto, la salida del buffer se pone en alto también y puede proporcionar hasta 25ma. El buffer tiene además como propósito proteger los pines del DSP en caso de falla, ya que no drena corriente del pin, solo lee el estado (HIGH/LOW).

### **4.7.4. Relés**

El modelo de los relés utilizados es Zelio fabricado por *Schneider*. Estos relés son utilizados en la operación de los bancos de condensadores y del filtro. Su capacidad nominal es de 12 A y 250 V alterno. El voltaje de accionamiento del relé es 12 V continuo.

### **4.7.5. Transistores**

Se utilizaron transistores NPN como intermediarios entre el DSP y los relés, ya que los puertos GPIO no entregan la suficiente corriente para activar los relés. El modelo de transistor utilizado es BD53 fabricado por DARLINGTON.

## 4.8. Dispositivos de comunicación

### 4.8.1. RS-232

Para la comunicación serial con un computador se utilizó el transceptor que viene incluido en el DSP, el cual deja una salida que se comunica directamente con el computador.

### 4.8.2. RS-485

Para conectar el DSP a la red Modbus con la interfaz RS-485 se utilizó el transceptor MAX3485 que transforma las señales de 3.3 V al voltaje requerido para la red Modbus.

## 4.9. Diseño de placas

El diseño de las placas circuitales se realizó en el software *Altium* debido a su amigable interfaz y su facilidad para crear bibliotecas de componentes. A continuación se muestran los circuitos impresos (PCB) de las placas realizadas. Las pistas azules corresponden a la capa inferior del circuito y las pistas rojas a la capa superior (PCB de doble capa).

### 4.9.1. Placa de control

La placa de control es la placa principal del proyecto ya que contiene al DSP. Esta placa contiene los opamps de ajuste para las señales de medida y también contiene los dispositivos de comunicación. La configuración utilizada en los opamps es la que se muestra en la figura 4.4

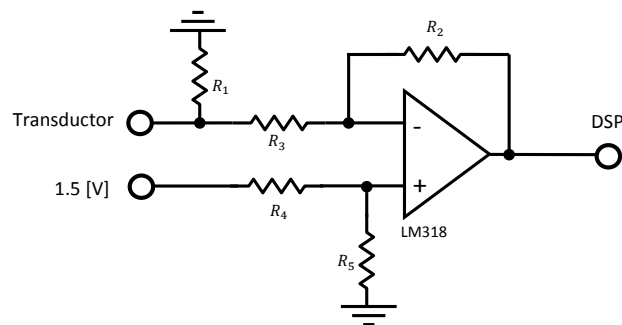


Figura 4.4: Circuito de ajuste para la salida de los transductores LEM

Corresponde a la configuración de amplificador diferencial y se encarga de ajustar la señal para que varíe entre los 0 y 3 V. Desde ésta salen también todas las conexiones a las otras placas: voltajes, corrientes, disparo y detección de cruces por cero.

En la figura 4.5 se observa los distintos componentes que conforman la placa de control, compuesta principalmente por el DSP y los opamps.

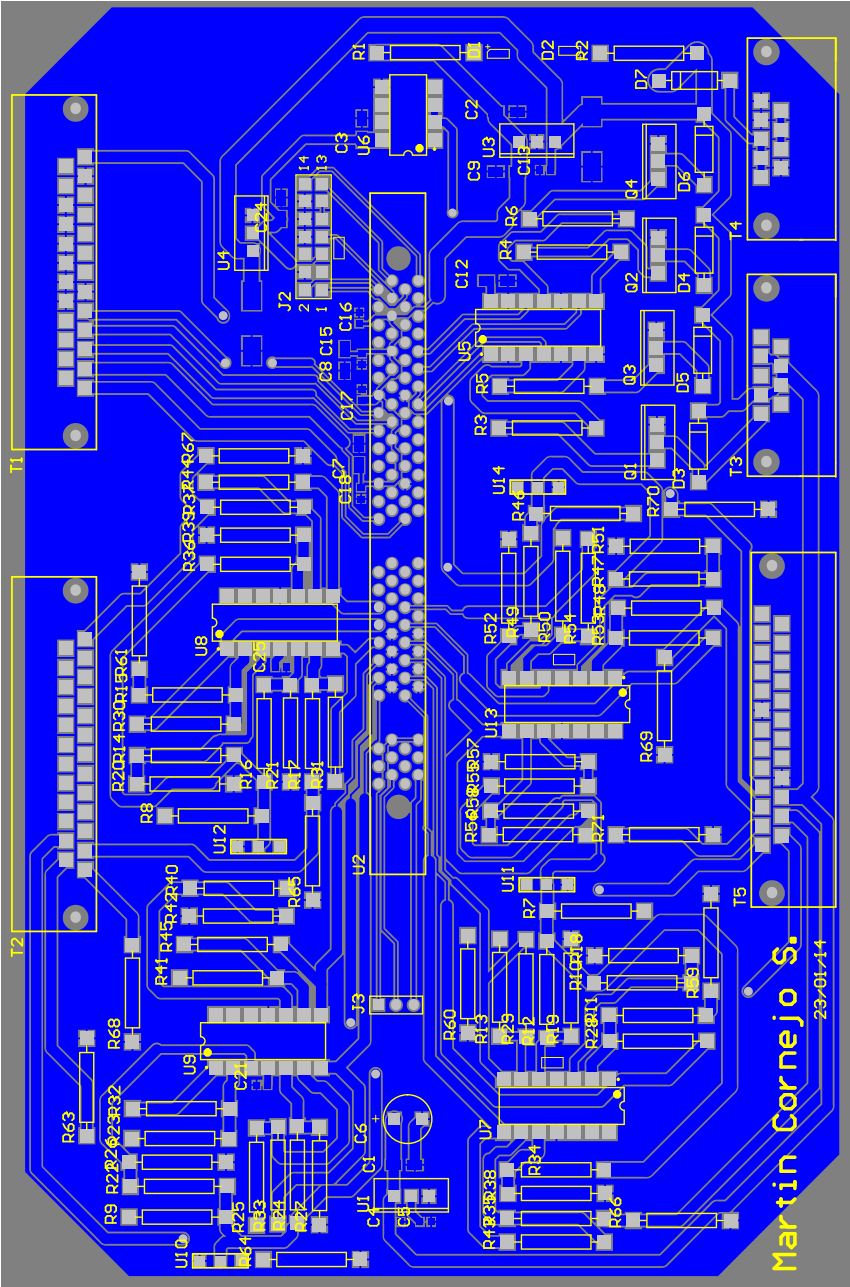


Figura 4.5: Capa inferior de la placa de control

Además en esta placa se encuentran los terminales para conectar los relés de operación, cuyo esquemático se muestra en la figura 4.6. Se utiliza un diodo conectado en antiparalelo a la bobina del relé para proteger al transistor de las corrientes de rebote de la bobina.

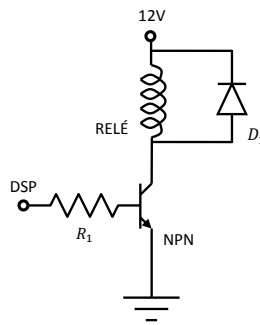


Figura 4.6: Esquemático de los relés

#### 4.9.2. Placa de adquisición y disparo

En esta placa se realiza la adquisición de voltaje, la detección de cruce por cero y el disparo de los tiristores. En la figura 4.7 se muestran 3 transductores de voltaje, uno por fase, conectados a la red por medio de una resistencia en serie. Esta resistencia tiene como objetivo ajustar la corriente primaria del transductor a un valor cercano a los 10 mA rms nominal, según las especificaciones del fabricante. La razón de transformación es de 25:10, por lo tanto la corriente de salida es de 25 mA rms nominal. La señal de salida debe ser ajustada una señal que oscile entre los 0 y los 3 V de manera que pueda ser leída por el DSP.

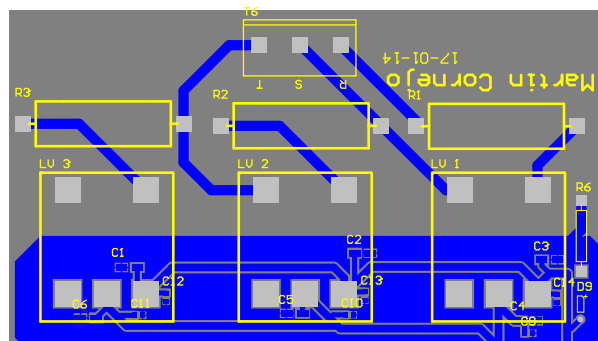


Figura 4.7: PCB de lems de tensiones

En la figura 4.8 se observa el circuito que detecta los cruces por cero en cada fase. Las resistencias sirven para atenuar la corriente al orden de los miliamperes. El integrado DB106S es un puente rectificador de onda completa, el cual está conectado a la compuerta óptica del optoacoplador. Cada vez que el voltaje pasa por cero se produce un pulso positivo en el pin 5 de la optocoupla, el cual es leído por el DSP. La figura 4.9 muestra la placa circuital para la detección de ceros en las 3 fases de la red.

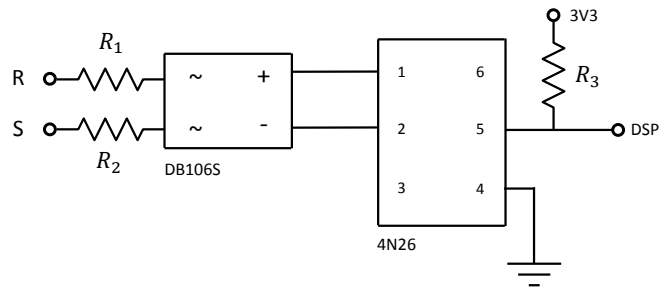


Figura 4.8: Circuito detector de cruces por cero

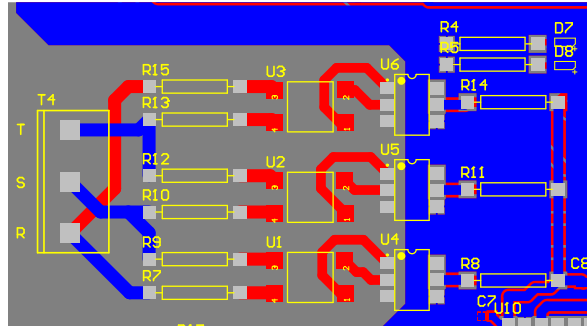


Figura 4.9: PCB del circuito de detección de ceros

Para el disparo se utiliza un optotriac, que es una optocoupla especializada en el manejo de triacs y tiristores. El esquemático de disparo se muestra en la figura 4.10, el cual es recomendado por el fabricante del optotriac. En este ejemplo se muestra conectado entre las fases R y S, sin embargo existe uno por cada fase. La resistencia  $R_2$  y el condensador  $C_1$  componen la rama *snubber* del circuito, que tiene como objetivo evitar falsos disparos en los tiristores. El SCR 2 conduce en el semiciclo positivo y el SCR 1 en el negativo. El DSP y los tiristores se encuentran aislados eléctricamente ya que el optotriac se excita mediante una señal luminosa proveniente de un led conectado entre los pines 1 y 2. En la figura 4.11 se muestra el PCB de la placa de disparo, en donde se observan los terminales de conexión para las inductancias y el terminal de conexión para la red de 110 V.

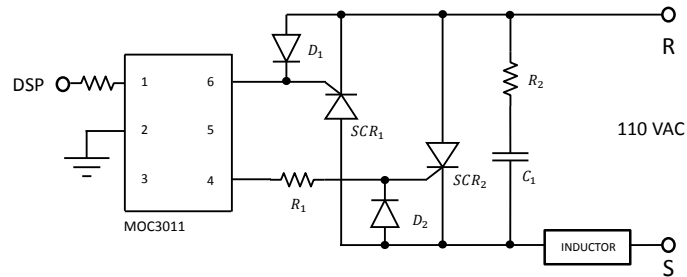


Figura 4.10: Circuito de disparo de los tiristores

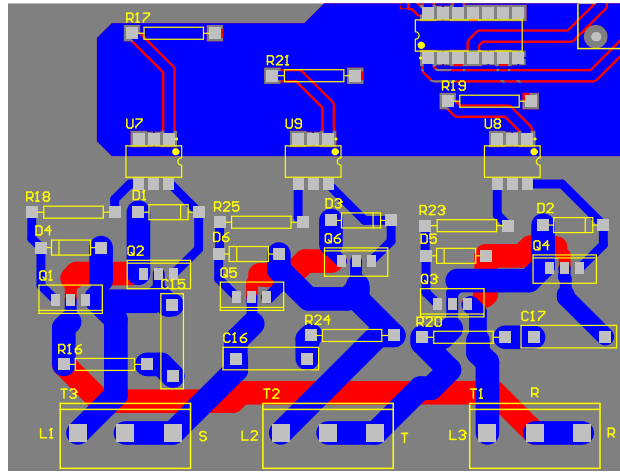


Figura 4.11: PCB de la placa de disparo

### 4.9.3. Placa de alimentación

La función de esta placa es alimentar las placas de control y de adquisición y disparo. Ésta entrega 4 niveles de voltaje continuo: +15,-15,+5 y +24 V. Los voltajes +15 y -15 V son para alimentar los Lems de voltaje y corrientes. El voltaje 5 V es necesario para alimentar el microcontrolador y los 24 V son necesarios para activar/desactivar los contactores de los bancos de condensadores. En la figura 4.12 se puede observar el diseño pcb.

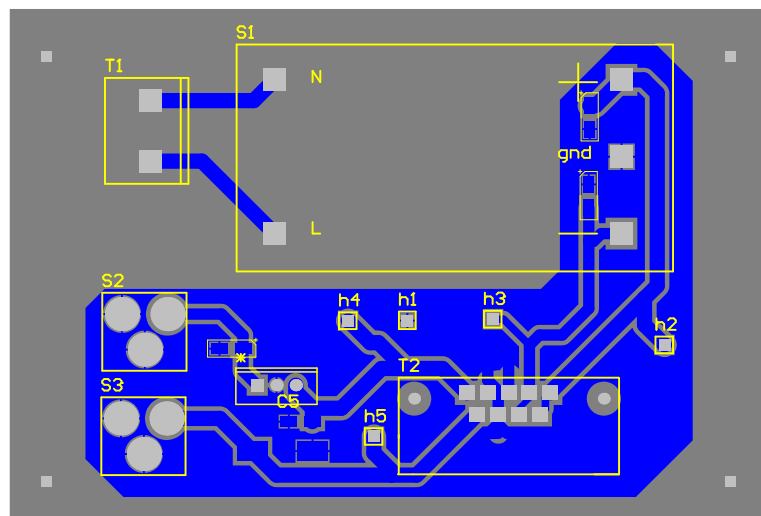


Figura 4.12: PCB de la placa de alimentación

Para generar +15/-15 V se usó una fuente AC/DC modelo RAC06-15DC montable para pcb. Para generar 5 V se utilizó un regulador LM7805 y para generar 24 V se usa un adaptador AC/DC con salida *jack dc*.

## 4.10. Algoritmo del CER

### 4.10.1. Adquisición de variables

Para la adquisición de voltajes se usó el circuito mostrado en la figura 4.7 y para ajustar la señal según las especificaciones del DSP (leer voltajes entre 0 y 3 V) se utilizó el circuito de la figura 4.4. La salida de este último circuito se conecta a los pines de conversión ADC del DSP.

El conversor ADC se configuró con una frecuencia de muestreo de 4500 Hz, más de 50 veces la frecuencia de la red (50 Hz), de esta manera se evita el *Aliasing*. El algoritmo contiene 3 etapas mostradas en la figura 4.13.

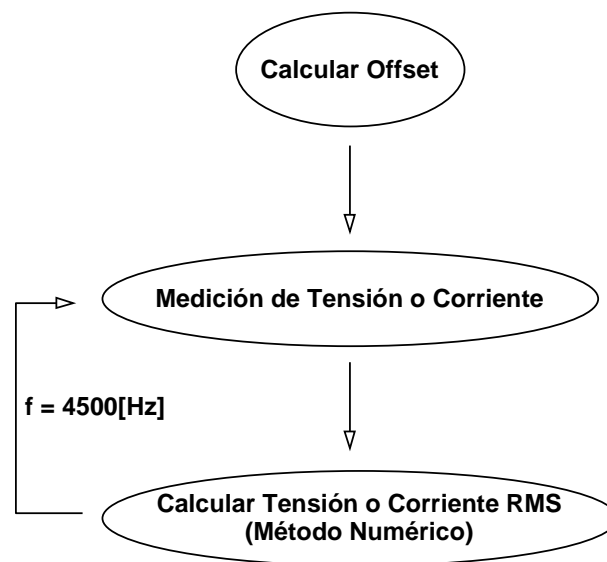


Figura 4.13: Diagrama de flujo del conversor ADC

La primera etapa consiste en determinar el "cero" o referencia de la senoide, ya que la señal oscila entre 0 y 3 V. Esta etapa ocurre solo una vez, al iniciar el DSP. Una vez determinado el nivel de referencia, el conversor ADC toma muestras de la senoide respecto al cero recién calculado y las procesa con un método numérico para obtener la señal RMS. Es en esta etapa donde destaca el nivel de procesamiento y cálculo del DSP, ya que el cálculo RMS requiere adquirir datos e integrar simultáneamente, tarea que no cualquier microcontrolador puede realizar.

### 4.10.2. Sincronizador de disparos

Para la detección de cruces por cero se utilizó el circuito mostrado en la figura 4.8 para las 3 fases. La salida de este circuito va a los pines de interrupción externa del DSP (XINTF) configurados para detectar flancos ascendentes. La señal de salida para la fase R se muestra en la figura 4.14. Su comportamiento es igual para las otras 2 fases de voltaje, sólo que desfasadas en  $120^\circ$ .



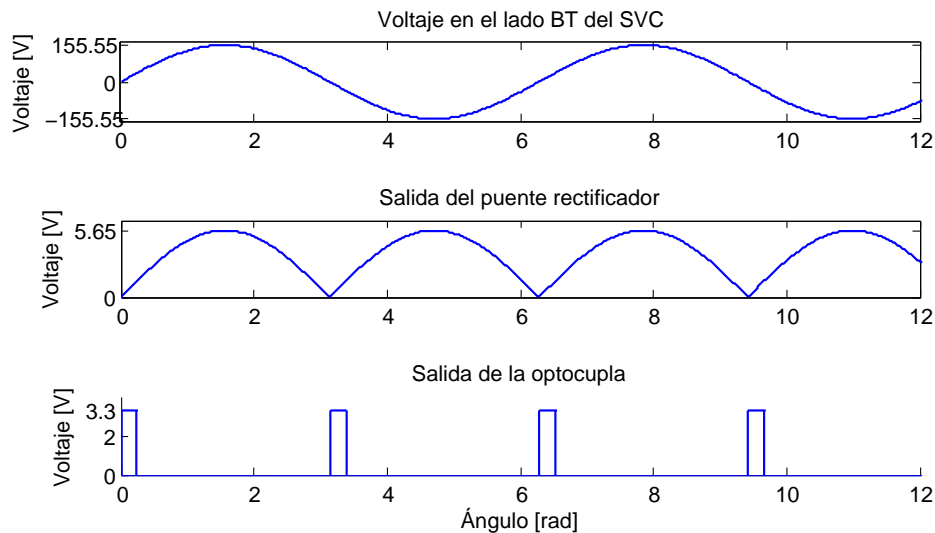


Figura 4.14: Señal de interrupción para sincronizador de disparos en la fase R

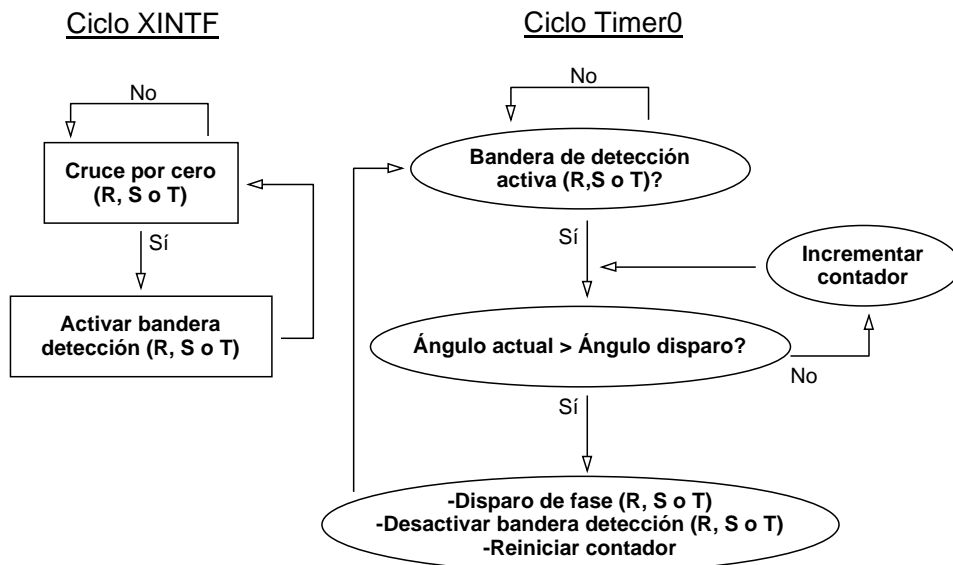


Figura 4.15: Diagrama de flujo del sincronizador de disparos

La interrupción por cruce se produce cada 10 ms en una red ideal de 50 Hz, rango de tiempo disponible para el disparo del tiristor según el ángulo requerido. Por ejemplo, si el ángulo de disparo es de  $90^\circ$ , el tiristor debe ser accionado en la mitad del semiciclo, es decir a los 5 ms.

Para llevar un conteo del tiempo que transcurre entre cada detección de cruce se utiliza el Timer0 del microcontrolador. Este reloj se configuró a una resolución de aproximadamente  $1^\circ$  de ángulo de disparo. En otras palabras, el reloj cuenta con pasos de  $55[\mu s]$  dando un total de 182 pasos entre cruce y cruce  $\left(\frac{10ms}{55\mu s} = 181.82\right)$ . Para configurar un ángulo de disparo de  $90^\circ$ , el Timer0 debe contar  $\frac{5ms}{55\mu s} = 90.9$ , aproximadamente 91 pasos para luego accionar el disparo. Al producirse un nuevo disparo se reinicia el contador de pasos del Timer0.

## Capítulo 5

# Construcción de un CER para laboratorio

### 5.1. Construcción de los inductores

Para la construcción de las inductancias se usaron láminas de acero silicoso en forma de E y de I, además de un carrete de plástico para el enrollado del alambre esmaltado. Se usó papel prespán para el entrehierro. Todos los materiales se pueden encontrar en transformadores Pailamilla. Según los parámetros de diseño de la tabla 4.2 se obtuvieron los siguientes resultados:

<b>Voltaje V</b>	110
<b>Potencia VAr</b>	340.6
<b>Potencia W</b>	16
<b>Corriente A</b>	3.1
<b>R <math>\Omega</math></b>	1.7
<b>L mH</b>	113

Tabla 5.1: Valores nominales del inductor

Con estos valores se obtiene un banco de inductancias capaz de absorber 1020 VAr. Además se realizaron pruebas de voltaje al inductor para observar su comportamiento en voltajes cercanos al nominal, obteniendo los siguientes resultados:

<b>Voltaje V</b>	80	90	100	110	120	130
<b>Corriente A</b>	2,30	2,56	2,75	3,10	3,30	3,61
<b>Impedancia <math>\Omega</math></b>	34,78	35,16	36,36	35,48	36,36	36,01

Tabla 5.2: Pruebas de energización del inductor

De la tabla 5.2 se observa que la impedancia varía levemente (menos del 5%), esto significa que el inductor opera en la zona lineal de la curva de magnetización.

En la figura 5.1 se observa uno de los 3 inductores realizados.

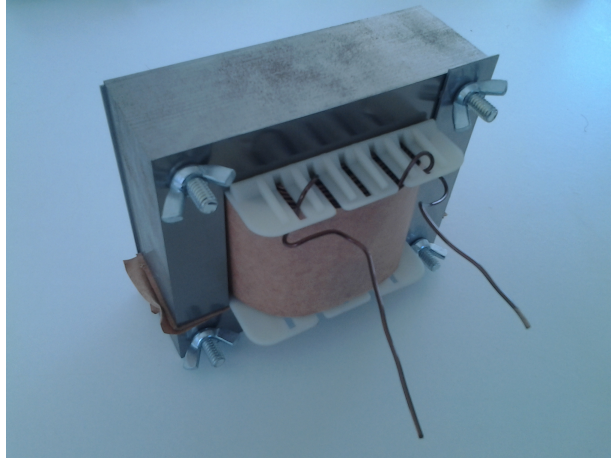


Figura 5.1: Inductor final

## 5.2. Construcción de placas circuitales

### 5.2.1. Placa de adquisición y disparo

El resultado final de la placa de adquisición y disparo se muestra en la figura 5.2.

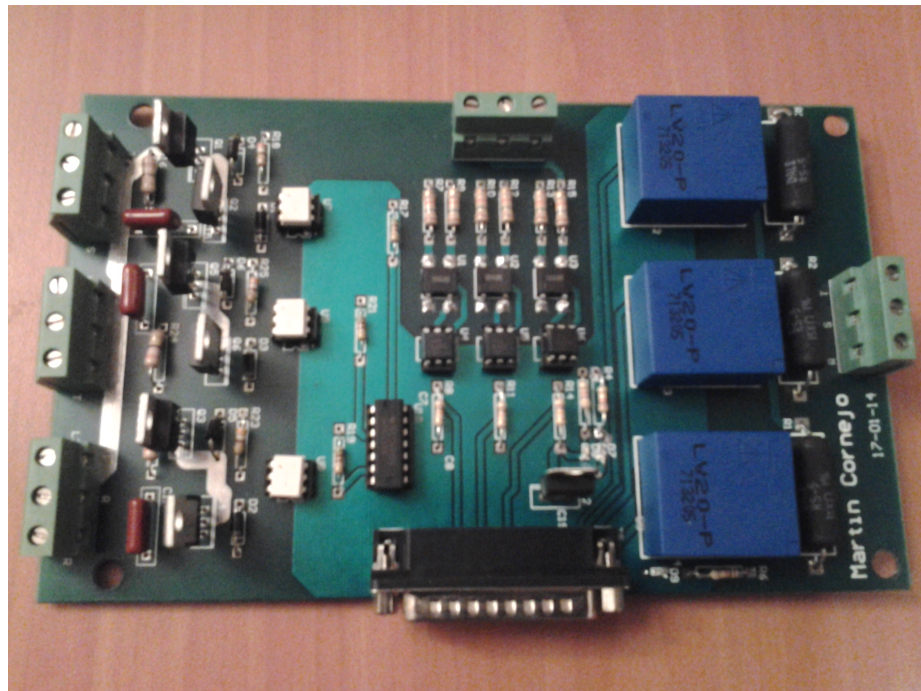


Figura 5.2: Placa de adquisición y disparo

Al costado derecho de la placa se encuentran los Lems de tensión encargados de medir el voltaje en la barra de conexión del CER. Al centro de la placa se encuentra el circuito detector de ceros y al lado izquierdo se encuentra el circuito de disparo con los tiristores respectivos.

Para las conexiones entre la placa y el exterior (inductores y barra de voltaje) se usaron los terminales verdes observados al borde de la placa, los cuales soportan un nivel de tensión de 300 V y corrientes de 9 A. Para las conexiones entre ésta y la placa de control se usó un conector DB25 que se observa en la parte centro-inferior de la figura 5.2.

### 5.2.2. Placa de control

En la figura 5.3 se muestra la placa de control. Al centro se encuentra el microcontrolador TMS320F28335 conectado mediante un conector tipo DIMM-100 de orejas blancas. Al lado superior izquierdo se encuentra un circuito de opamps con sus respectivas resistencias de ganancia, estos son los encargados de adaptar las señales provenientes de los Lems de la placa de adquisición. Al lado inferior derecho se encuentran los transistores encargados de accionar los relés de operación. Se observa que la placa tiene espacio para soldar más componentes, esto se debe a que el modelo de opamp pensado para el ajuste (TLC274) resultó ser de baja calidad debido al gran ruido que introduce a las mediciones, por lo que se optó por realizar una placa de ajuste aparte con otro modelo de opamp. Para la salida de conexiones de la placa de control hacia la placa de adquisición y disparo y hacia la fuente se usaron los conectores DB25 y DB9. Además se dejaron habilitados 3 puertos de comunicación, el puerto serial RS232 cuya salida de 3 pines (conector polarizado macho) se observa a la izquierda del microcontrolador. El puerto JTAG, el cual permite la programación y depuración de los programas del DSP, se observa al lado superior derecho del microcontrolador y consiste en 14 pines ordenados en una hilera de 2x7. Por último el puerto de comunicación RS485 a la salida del conector DB25 de la derecha (para más detalle ver el esquemático en anexos).

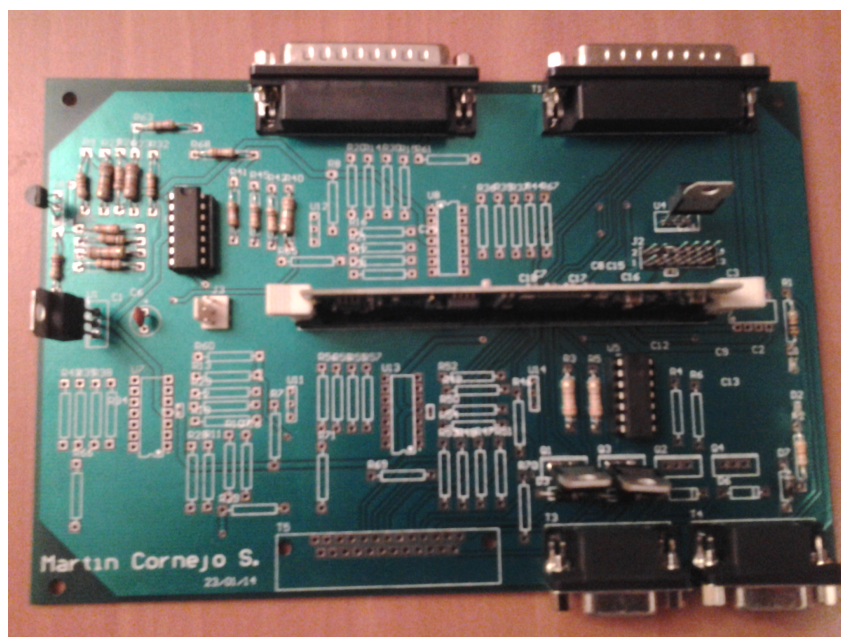


Figura 5.3: Placa de Control

### 5.2.3. Placa de alimentación

La placa de alimentación, como se mencionó en el capítulo 4.9.3, tiene una fuente AC/DC montable para pcb que genera +15/-15 V. Para conectar la fuente a la red (220 o 110 V) se tiene un terminal de tornillos. Para los voltajes 5 y 24 V se tienen dos conectores *jack dc* a los cuales se conectan adaptadores AC/DC de pared. Para la salida de estas alimentaciones se utilizó un conector DB9 como se ve en la figura 5.4.

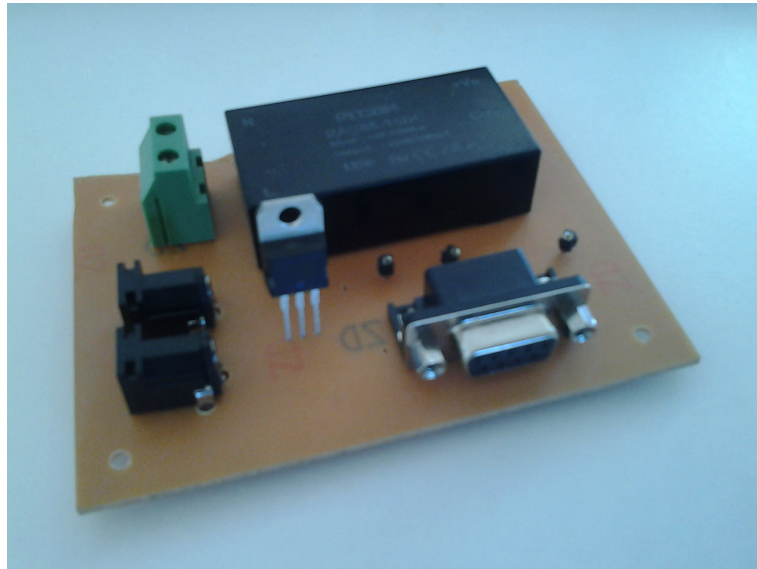


Figura 5.4: Placa de alimentación

### 5.2.4. Placa de ajuste

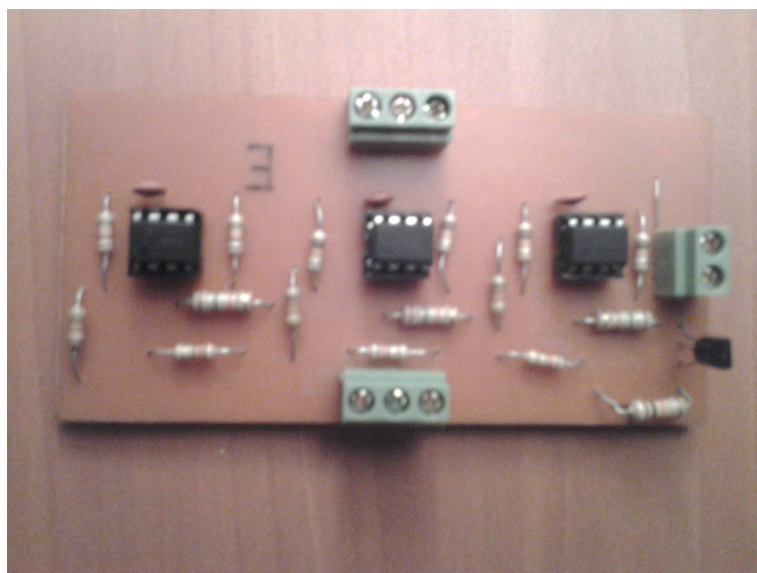


Figura 5.5: Nueva placa de ajuste

Pese a haber hecho pruebas previas con los opamps modelo TLC274 con un osciloscopio y no haber tenido problemas, al trabajar junto al DSP se obtiene ruido considerable a la entrada del conversor ADC que genera un margen de error del 40 %. Como esto se descubrió después de realizar la placa de control, se optó por crear una placa circuital adicional con opamps modelo OP07 debido a su reconocida precisión para asegurar una adquisición de variables precisa. Luego de realizar mediciones se redujo el ruido considerablemente (menos del 5 % de error).

## Capítulo 6

### Conclusion

Para la realización del trabajo de título se requirió de una gran investigación previa sobre los Compensadores Estáticos de Reactivos (CER) tanto en potencia como a escala, sin embargo más tiempo se dedicó al ámbito electrónico del trabajo, dividido en tres partes importantes: Aprendizaje y programación del microprocesador TMS320F28335 , Aprendizaje y utilización del software *Altium* para diseño de los circuitos electrónicos impresos, Finalización y verificación de las placas circuitales. En cada una de estas partes surgieron dificultades, al mencionar las más importantes se tienen respectivamente: Aprender a programar DSP desde cero, ya que la única fuente de tutoriales y aprendizaje eran los manuales oficiales. El diseño de placas circuitales, principalmente la placa de control, fue un importante obstáculo debido a la gran cantidad de salidas provenientes de la tarjeta de control (100 posiciones) teniendo en cuenta las normas básicas de distancia entre pistas y pines que se deben considerar en un circuito impreso. Por último la verificación de las placas circuitales después del soldado de los componentes, ya que se debe revisar que no existan cortocircuitos en cada pista y pin del circuito.

Como resultado se obtuvo un CER a escala con un banco de inductancias que opera en modo manual, es decir, tanto el ángulo de disparo en los tiristores como la conexión y desconexión de bancos de condensadores debe ser ingresado manualmente al DSP. Estas variables pueden ser ingresadas directamente en el código del programa o a través de la comunicación Modbus.

Al final de este trabajo se adquirieron importantes conocimientos tanto en *Software* relacionado con microcontroladores y diversos protocolos de comunicación entre ellos Modbus y UART, como también en *Hardware* relacionado con el diseño y construcción de circuitos electrónicos para control de potencia.

## 6.1. Trabajo futuro y recomendaciones

Se propone terminar la etapa de potencia del CER, instalando el banco de condensadores y el transformador para la conexión a la red. Además es posible implementar un control PI para el control de voltaje por referencia, ya que actualmente el CER opera en modo manual. Si bien en el presente trabajo se mencionó sobre los filtros de armónica e incluso se diseñaron los parámetros de un filtro de 5<sup>a</sup> armónica, éste no se llegó a implementar, por lo que queda propuesto su construcción.

Para la realización de este tipo de prototipos electrónicos se recomienda la especialización en temas relacionado con la programación de software orientado a hardware y también en placas circuitales previo a trabajar. Debido a la inexperiencia inicial del autor en estos temas se utilizó gran parte del tiempo en ensayo y error tanto en diseño como en construcción.



# Bibliografía

- [1] ABB: *SVC to Increase Reliability and Reduce Congestion over Multiple 500 kV Lines*. Application Note A02-0207 E.
- [2] ABB: *Static Var Compensator, An insurance for improved grid system stability and reliability*. 2010. A02-0100 E, 2010-11, Elanders Sverige AB.
- [3] ABB: *FACTS for grid voltage stabilization and increased power transmission capability in Chile*. 2012. Application Note 1JNS012661.
- [4] ABB: *STATCOM and beyond*. 2012. A02-0165 E, 2010-03, Elanders Sverige AB.
- [5] Castro, Jaime Muñoz: *Prototipo de equipos FACTS de baja potencia*. Memoria de Título, Departamento de Ingeniería Eléctrica, Universidad de Chile, 2006.
- [6] Chopade, P., M. Bikdash, I. Kateeb y A.D. Kelkar: *Reactive power management and voltage control of large Transmission System using SVC (Static VAR Compensator)*. En *Southeastcon, 2011 Proceedings of IEEE*, páginas 85–90, 2011.
- [7] Dixon, J., L. Moran, J. Rodriguez y R. Domke: *Reactive Power Compensation Technologies: State-of-the-Art Review*. Proceedings of the IEEE, 93(12):2144–2164, 2005, ISSN 0018-9219.
- [8] Dong, L., M.L. Crow, Z. Yang, C. Shen, L. Zhang y S. Atcitty: *A reconfigurable FACTS system for university laboratories*. Power Systems, IEEE Transactions on, 19(1):120–128, 2004, ISSN 0885-8950.
- [9] Haro, P.Z. y J.M. Ramirez Arredondo: *Experimental results on a lab scale single-phase TCSC*. En *Power Engineering Society Summer Meeting, 2002 IEEE*, volumen 3, páginas 1433–1438 vol.3, 2002.
- [10] Mendoza-Araya, P., J.M. Castro, J.C. Nolasco y R.E. Palma-Behnke: *Lab-Scale TCR-Based SVC System for Educational and DG Applications*. Power Systems, IEEE Transactions on, 26(1):3–11, 2011, ISSN 0885-8950.
- [11] Muñoz, Alfredo: *Filtros sintonizados y compensación de reactivos en sistemas perturbados por armónicas*. Apuntes del autor, 1997.

- [12] Muñoz, J., P. Mendoza, J. Cotos y R. Palma: *Lab-scale three-phase TCR-based SVC system for educational purpose in dynamic and steady-state analysis*. En *Power Symposium, 2007. NAPS '07. 39th North American*, páginas 636–643, 2007.
- [13] Noroozian, M., A.N. Petersson, B. Thorvaldson, B.A. Nilsson y C.W. Taylor: *Benefits of SVC and STATCOM for electric utility application*. En *Transmission and Distribution Conference and Exposition, 2003 IEEE PES*, volumen 3, páginas 1192–1199, 2003.
- [14] Rashid, Muhammad H.: *Electrónica de potencia: Circuitos, dispositivos y aplicaciones*. PRETINCE HALL HISPANOAMERICANA, segunda edición, 1993.
- [15] Rashid, Muhammad H.: *Power Electronics Handbook*. ACADEMIC PRESS, primera edición, 2001.
- [16] Shicheng, Zheng, Fang Sian y Zhang Gaoyu: *Research on TCR type SVC system and MATLAB simulation*. En *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, páginas 2110–2114, 2010.
- [17] Siemens: *Modbus Information*. Site Collection Documents.
- [18] SIEMENS: *FACTS - FLEXIBLE AC Transmission Systems: Static Var Compensators*. 2011. [www.siemens.com/energy/facts](http://www.siemens.com/energy/facts).
- [19] Venu Yarlagadda, K. R. M. Rao, B. V. Sankar Ram: *Hardware Circuit Implementation of Automatic Control of Static Var Compensator (SVC) using Micro Controller*. International Journal of Instrumentation, Control and Automation (IJICA), Volume 1, Issue 2, 2011, ISSN 2231-1890.
- [20] W. Brokering, R. Palma, L. Vargas: *Ñom Lufke (El Rayo Domado) o Los Sistemas Eléctricos de Potencia*. PEARSON Prentice Hall, primera edición, 2008.
- [21] Warner, J., J. Fenn, S. Hutchinson y R. Platt: *Impact of system changes on the harmonic performance and rating of an SVC*. En *Transmission and Distribution Conference and Exposition (T D), 2012 IEEE PES*, páginas 1–7, 2012.
- [22] Yehui, Han, J.Y. Chen y Hu Cheng: *SVC multi-control system design and implementation*. En *Power System Technology, 2002. Proceedings. PowerCon 2002. International Conference on*, volumen 4, páginas 2294–2297 vol.4, 2002.
- [23] Zemerick, S.A., P. Klinkhachorn y A. Feliachi: *Prototype design of a personal static VAR compensator*. En *System Theory, 2002. Proceedings of the Thirty-Fourth Southeastern Symposium on*, páginas 311–315, 2002.

## **Apéndice A**

# **Esquemáticos de las placas**

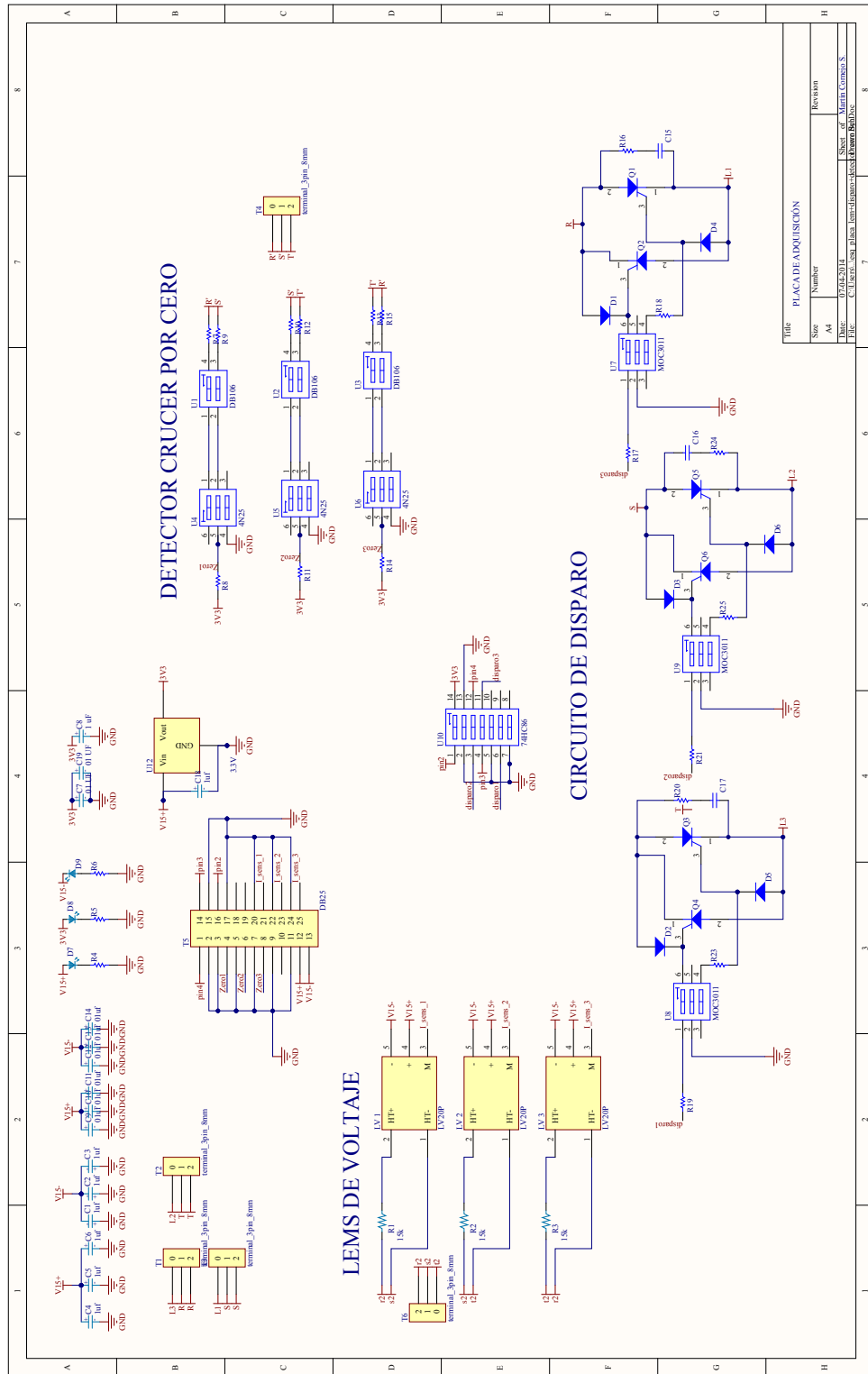


Figura A.1: Esquemático de la placa de adquisición

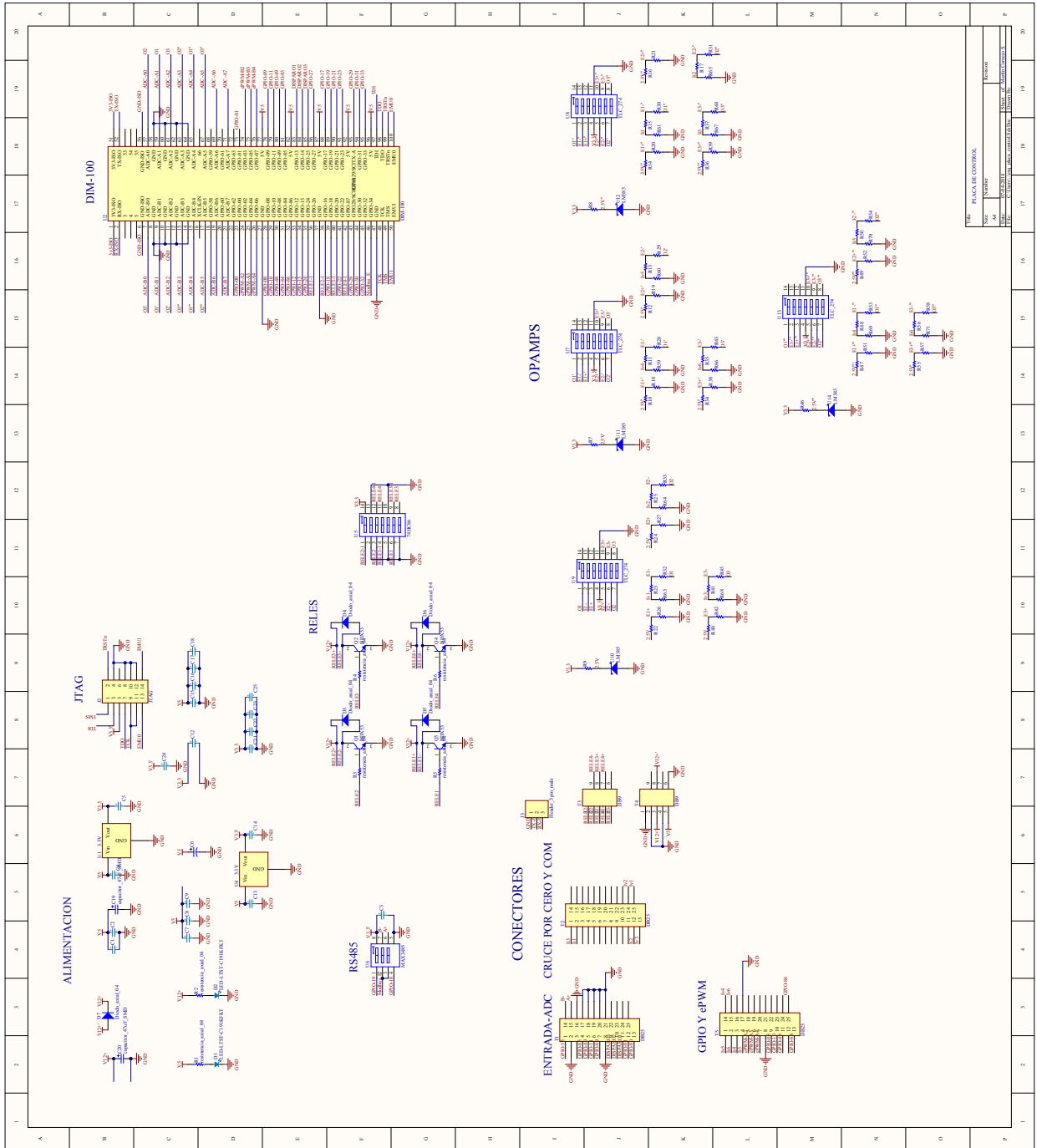
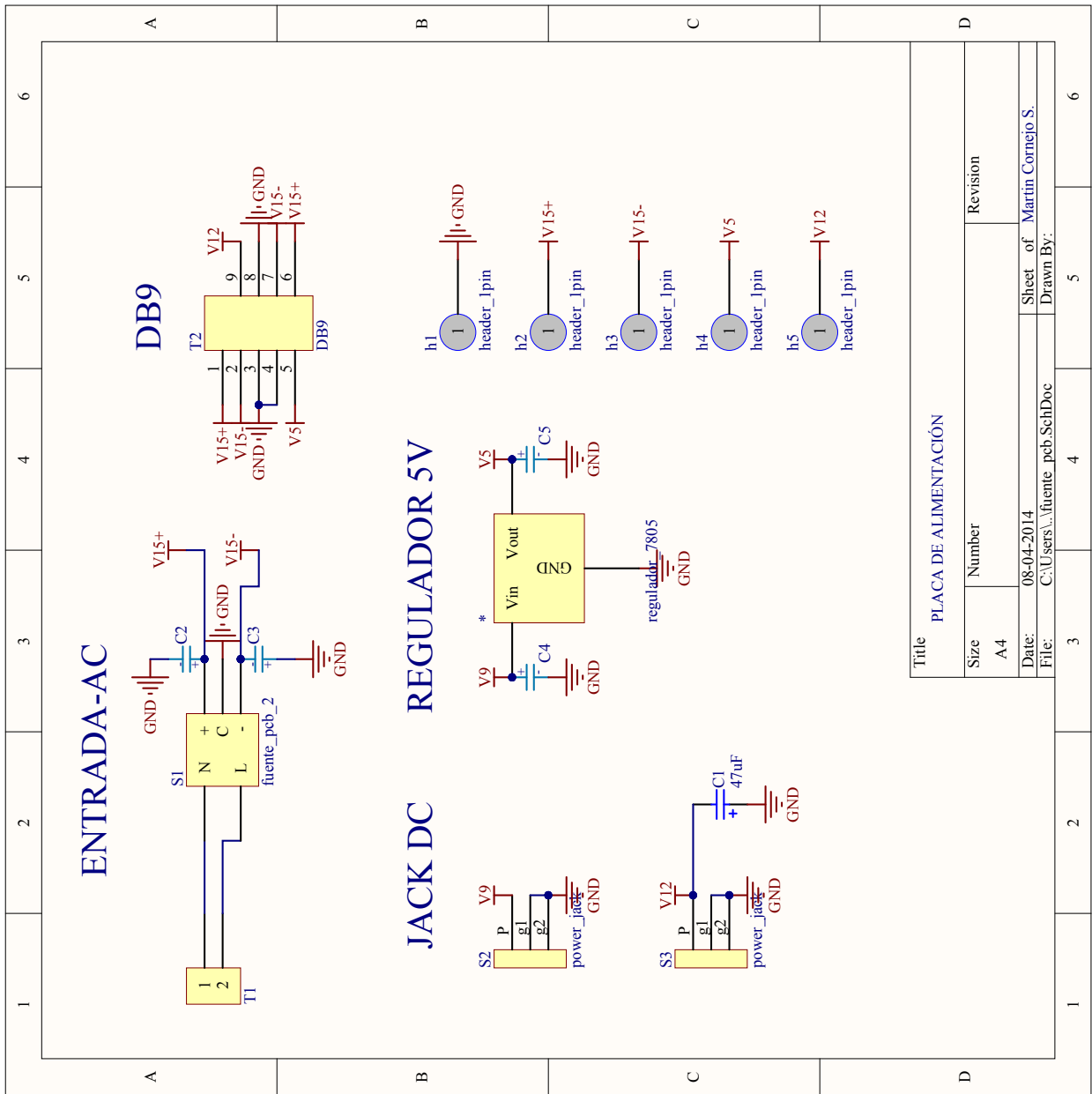


Figura A.2: Esquemático de la placa de control



Title		
Size	Number	Revision
A4		
Date:	08-04-2014	
File:	C:\Users\...fuente_pcb.SchDoc	
		Sheet of
		Martin Cornejo S.
		Drawn By:

Figura A.3: Esquemático de la placa de alimentación

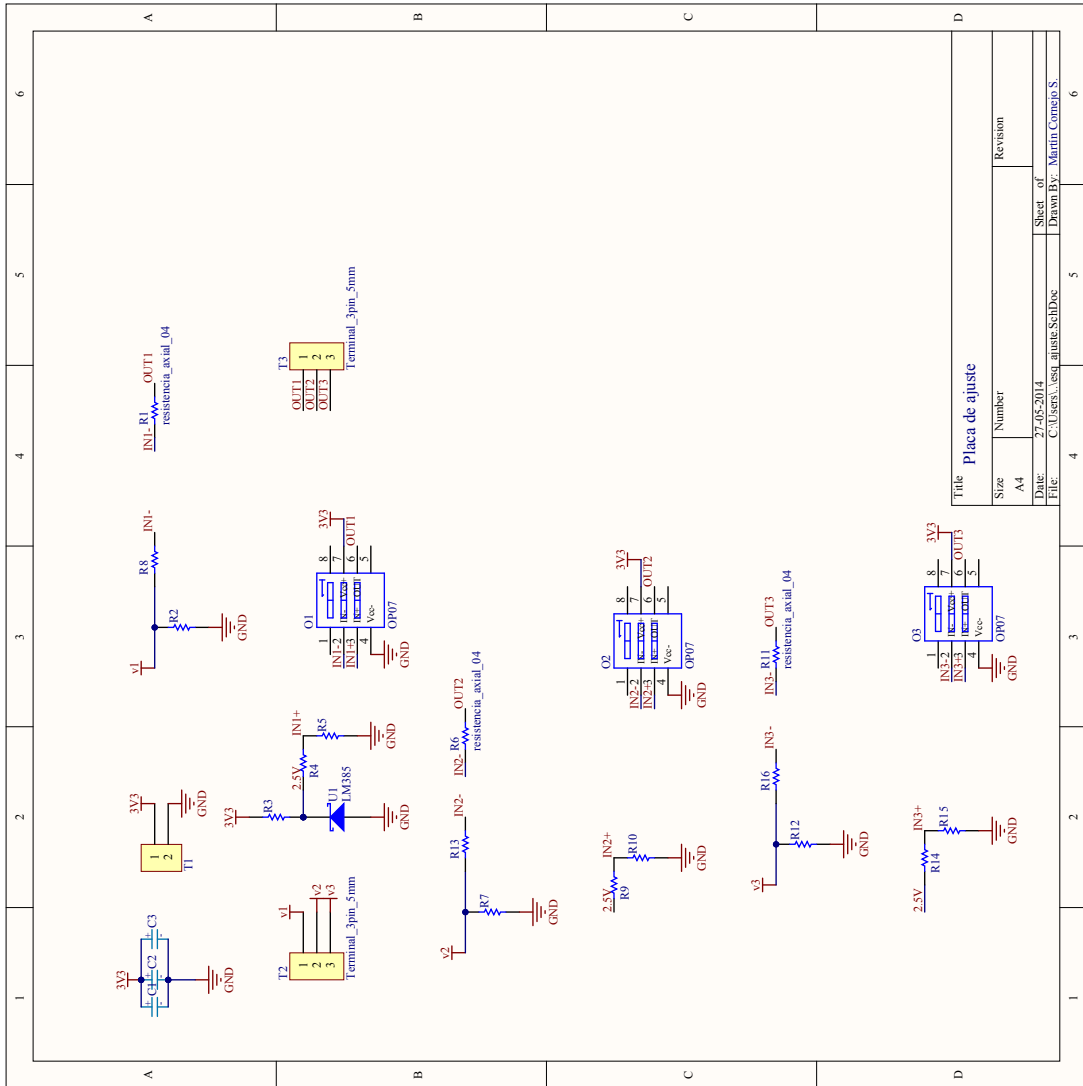


Figura A.4: Esquemático de la placa de ajuste

## **Apéndice B**

### **Código del programa**



svc4-Main.c

```

1 // DESCRIPCIÓN
2 //
3 // Rutina de control del SVC, modo manual.
4 // Envia datos por Puerto Serie A.
5 // El angulo de disparo es elegido por el usuario
6 //#####
7 // $Microcontrolador: TMS320F283335 $
8 // $Fecha: 24 de Marzo, 2014 $
9 // $Autor: Martín Cornejo Saavedra $
10 //#####
11 //
12
13 #include "DSP28x_Project.h" // Device Headerfile and Examples Include File
14 #include "math.h"
15 // #include "time.h"
16 // #include <stdio.h>
17 // #include <stdlib.h>
18
19 //=====funciones a
    utilizar=====
20
21 void Rutina_ADC(void);
22 void ConfigurarEPWM(void);
23 void ConfigurarADC(void);
24 void ConfigurarGPIO(void);
25 void ConfigurarXINT(void);
26 void adc_init_offset(void);
27
28 void scia_configurar(void);
29 void scia_fifo_iniciar(void);
30 void scia_emitir(int a);
31 void scia_msg(char *msg);
32
33 interrupt void cpu_timer0_isr(void);
34 interrupt void adc_isr(void);
35 interrupt void epwm1_isr(void);
36 interrupt void xint1_isr(void);
37 interrupt void xint2_isr(void);
38 interrupt void xint3_isr(void);
39
40 //=====constantes=====
    =====
41 //universales
42 #define FALSE 0
43 #define TRUE 1
44
45 //definiciones de constantes
46 #define UNODIVRAIZTRES 0.5773502691896258
47 // #define DOS_PI 6.2831853071795865
48
49 // ADC start parameters
50 #define ADC_MODCLK 0x2 // HSPCLK = SYSCLKOUT/2*ADC_MODCLK2 = 150/(2*2) = 37.5 MHz
51 #define BUF_SIZE 2048 // Sample buffer size
52 #define ADC_CLKPS 0x1 // ADCCLK = HSPCLK/(2*ADC_CLKPS*(CPS+1))
53 #define ADC_CPS 0x0
54 #define ADC_SHCLK 0xf // S/H width in ADC module periods = 16 ADC
    clocks

```

svc4-Main.c

```
55
56 //ePWM parameters
57 #define EPWM_TIMER_TBPRD    17858 // Indica hasta dónde cuenta el PWM en ciclos de TBCLK,
    cuenta para arriba y para abajo, luego el período es el doble en ciclos TBCLK
58
59 //GPIOs disparo
60 #define PIN_DISPARO_AB    GPIO13
61 #define PIN_DISPARO_BC    GPIO14
62 #define PIN_DISPARO_CA    GPIO25
63
64 #define PIN_CONTACTOR_1    GPIO16
65 #define PIN_CONTACTOR_2    GPIO26
66
67 //=====variables=====
    =====
68 //Bancos de condensadores
69 Uint16 numero_bancos=0;
70
71 //Setpoint
72 Uint16 V_setpoint=0;
73
74 //deteccion cruce por cero
75 Uint16 deteccion_AB=FALSE;
76 Uint16 deteccion_BC=FALSE;
77 Uint16 deteccion_CA=FALSE;
78 Uint16 angulo_por_cero=90; //es el angulo de control que se tiene cuando se detecta el cruce
    por cero
79 Uint16 contador_AB=0; //contador de cruces por cero, sirve para debuggear
80 Uint16 contador_BC=0; //contador de cruces por cero
81 Uint16 contador_CA=0; //contador de cruces por cero
82
83 //Timer y disparo
84 Uint16 step_counter_AB=0;
85 Uint16 step_counter_BC=0;
86 Uint16 step_counter_CA=0;
87 //Uint16 freqStep = 500000; //55 uSeg, resolucion de 180 para 100Hz (frecuencia de semi-ciclo)
88 Uint16 step_deseado=68;
89 Uint16 step_deseado_AB=0;
90 Uint16 step_deseado_BC=0;
91 Uint16 step_deseado_CA=0;
92
93 Uint16 estado_CA_1=FALSE;
94
95 Uint16 estado_BC_1=FALSE;
96
97 Uint16 estado_AB_1=FALSE;
98
99 Uint16 angulo_disparo=0;
100
101
102 // variables auxiliares
103 Uint16 aux1=0;
104
105 //Calculo de Offset ADC
106 int Offset_cnt = 0;
107 float offsetVbt_AB = 0, offsetVbt_BC = 0, offsetVbt_CA = 0;
108
```

svc4-Main.c

```

109 float KI_offset = 0.998, KP_offset = 0.001999;
110
111 //Calculo de variables electricas
112 float KgainVbt_AB = 0.243645, KgainVbt_BC = 0.247982, KgainVbt_CA= 0.247982;
113 float Vbt_AB = 0, Vbt_BC = 0,Vbt_CA = 0;
114 float Vbt_A = 0, Vbt_B = 0, Vbt_C = 0;
115 float Vbt_AB_2 = 0, Vbt_BC_2 = 0, Vbt_CA_2 = 0;
116 float Vbt_AB_f = 0, Vbt_AB_22 = 0, Vbt_AB_21 = 0, Vbt_AB_f2 = 0, Vbt_AB_f1 = 0;
117 float Vbt_BC_f = 0, Vbt_BC_22 = 0, Vbt_BC_21 = 0, Vbt_BC_f2 = 0, Vbt_BC_f1 = 0;
118 float Vbt_CA_f = 0, Vbt_CA_22 = 0, Vbt_CA_21 = 0, Vbt_CA_f2 = 0, Vbt_CA_f1 = 0;
119 float Vbt_AB_rms = 0, Vbt_BC_rms = 0, Vbt_CA_rms = 0;
120 float Vbt_rms = 0;
121
122 //=====Acá empieza el main=====
123
124 main()
125 {
126
127     InitSysCtrl(); // PLL, WatchDog, enable Peripheral Clocks
128     InitSciaGpio(); //Inicia el puerto serial SCI-A
129
130     EALLOW;
131     SysCtrlRegs.HISPCP.all = ADC_MODCLK; // HSPCLK = SYSCLKOUT/ADC_MODCLK
132     EDIS;
133
134     DINT;
135
136     InitPieCtrl();
137
138 // Disable CPU interrupts and clear all CPU interrupt flags:
139     IER = 0x0000;
140     IFR = 0x0000;
141
142     InitPieVectTable();
143
144     InitAdc(); // For this example, init the ADC
145
146     InitCpuTimers();
147     ConfigCpuTimer(&CpuTimer0, 150,75);
148
149     ConfigurarEPWM();
150     ConfigurarADC();
151     ConfigurarGPIO();
152     ConfigurarXINT();
153     scia_fifo_iniciar(); // Inicia el FIFO
154     scia_configurar(); // Configura el SCI-A
155
156     EALLOW;
157     PieVectTable.TINT0 = &cpu_timer0_isr;
158     PieVectTable.EPWM1_INT = &epwm1_isr;
159     PieVectTable.ADCINT = &adc_init_offset;
160     PieVectTable.XINT1 = &xint1_isr;
161     PieVectTable.XINT2 = &xint2_isr;
162     PieVectTable.XINT3 = &xint3_isr;
163     EDIS;
164
165     PieCtrlRegs.PIECTRL.bit.ENPIE = 1; // Enable the PIE block

```

svc4-Main.c

```

166 PieCtrlRegs.PIEIER1.bit.INTx4 = 1;           //Enable XINT1
167 PieCtrlRegs.PIEIER1.bit.INTx5 = 1;           //Enable XINT2
168 PieCtrlRegs.PIEIER12.bit.INTx1 = 1;          //Enable XINT3
169 PieCtrlRegs.PIEIER1.bit.INTx6 = 1; //adc
170 PieCtrlRegs.PIEIER1.bit.INTx7 = 1; //timer0
171 PieCtrlRegs.PIEIER3.bit.INTx1 = 1; // ePWM1
172     IER |= M_INT1;
173     IER |= M_INT3;
174     IER |= M_INT12;
175 EINT;           // Enable Global interrupt INTM
176 ERTM;          // Enable Global realtime interrupt DBGEM
177
178 CpuTimer0Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
179
180 for(;;)
181 {
182
183 }
184 }
185
186 //=====Termina el
main=====
187
188 //=====Implementación de funciones e
interrupciones=====
189
190 void ConfigurarEPWM()
191 {
192     EALLOW;
193     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0; // Deshabilita el reloj del EPWM
194     EDIS;
195
196     // >>>> CONFIGURACIÓN EPWM1 <<<<<
197     // Setup TBCLK
198     EPwm1Regs.TBPRD = EPWM_TIMER_TBPRD;           // Fijar período del PWM
199     EPwm1Regs.TBCTR = 0x0000;                   // Clear counter
200     // Setup counter mode
201     EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Modo simétrico (ie contador triangular)
202     EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;     // TBLK = SYSCLKOUT/(CLKDIV x HSPCLKDIV)
    este registro es pa tener un control más fino de la frecuencia
203     EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;       // Clock ratio to SYSCLKOUT en este caso:
    TB_DIV1 -> TBLK = SYSCLKOUT
204     // Set actions
205     EPwm1Regs.AQCTLB.all = 0x0600;
206     // Interrupt where we will change the Compare Values
207     EPwm1Regs.ETSEL.bit.INTSEL = ET_CTR_ZERO;   // Select INT on Zero event
208     EPwm1Regs.ETSEL.bit.INTEN = 1;             // Enable INT
209     EPwm1Regs.ETPS.bit.INTPRD = ET_1ST;        // Genera interrupción cada vez
210
211     EALLOW;
212     SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1; // Habilitar el reloj del EPWM
213     EDIS;
214 }
215
216 void ConfigurarADC(void){
217     // Specific ADC setup for this example:
218

```

svc4-Main.c

```

219     AdcRegs.ADCTRL1.bit.ACQ_PS = ADC_SHCLK; // Duración del sample and hold
220     AdcRegs.ADCTRL1.bit.CPS = ADC_CPS; //ADCCLK=FCLK/1
221     AdcRegs.ADCTRL3.bit.ADCCLKPS = ADC_CLKPS; // Divisor para el reloj ADC, ADCCLK =
HSPCLK/(2*ADC_CLKPS*(CPS+1))
222     AdcRegs.ADCTRL3.bit.SMODE_SEL = 0x1; //Setup simultaneous sampling mode
223     AdcRegs.ADCTRL1.bit.SEQ_CASC = 0x1; // 0x0 Non-Cascaded Mode 0x1 Cascaded Mode
224     AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 0x1; // Habilita la interrupción de la secuencia 1
que se gatilla cada EOS (end of sequence)
225     AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1; // Resetea el contador de SEQ1
226     AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0; // Setup ADCINA0 y ADCINB0 (0x0) como la
primera conversión de la secuencia (CONV0x0)
227     AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x1;
228     AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x2;
229     AdcRegs.ADCCHSELSEQ1.bit.CONV03 = 0x3;
230     AdcRegs.ADCCHSELSEQ2.bit.CONV04 = 0x4;
231     AdcRegs.ADCCHSELSEQ2.bit.CONV05 = 0x5;
232     AdcRegs.ADCMAXCONV.bit.MAX_CONV1 = 0x0005; // En cada SOC se ejecutan 6 conversiones
dobles (canales A y B, desde 0 a 5, 12 conversiones).
233 }
234
235 void ConfigurarGPIO(void){
236     EALLOW;
237     GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0;
238     GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;
239     GpioDataRegs.GPASET.bit.GPIO31 = 1;
240     GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;
241     GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;
242     GpioDataRegs.GPBSET.bit.GPIO34 = 1;
243
244     //configuramos los pines de interrupcion externa
245     GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0; // GPIO
246     GpioCtrlRegs.GPADIR.bit.GPIO0 = 0; // input
247     GpioCtrlRegs.GPAQSEL1.bit.GPIO0 = 0; // Xint1 Synch to SYSCLKOUT only
248
249     GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0; // GPIO
250     GpioCtrlRegs.GPADIR.bit.GPIO1 = 0; // input
251     GpioCtrlRegs.GPAQSEL1.bit.GPIO1 = 0; //Xint2 Synch to SYSCLKOUT
252
253     GpioCtrlRegs.GPBMUX1.bit.GPIO33 = 0; // GPIO
254     GpioCtrlRegs.GPBDIR.bit.GPIO33 = 0; // input
255     GpioCtrlRegs.GPBQSEL1.bit.GPIO33 = 0; //Xint3 Synch to SYSCLKOUT
256
257     //declaramos GPIO0 como XINT1
258     GpioIntRegs.GPIOXINT1SEL.bit.GPIOSEL = 0; // Xint1 is GPIO0
259     GpioIntRegs.GPIOXINT2SEL.bit.GPIOSEL = 1; // XINT2 is GPIO1
260     GpioIntRegs.GPIOXINT3SEL.bit.GPIOSEL = 1; // XINT3 is GPI33
261
262     GpioCtrlRegs.GPAPUD.bit.PIN_DISPARO_AB = 0; // Enable pull-up
263     GpioCtrlRegs.GPAMUX1.bit.PIN_DISPARO_AB = 0;
264     GpioCtrlRegs.GPADIR.bit.PIN_DISPARO_AB = 1;
265     GpioDataRegs.GPACLEAR.bit.PIN_DISPARO_AB = 1;
266
267     GpioCtrlRegs.GPAPUD.bit.PIN_DISPARO_BC = 0; // Enable pull-up
268     GpioCtrlRegs.GPAMUX1.bit.PIN_DISPARO_BC = 0;
269     GpioCtrlRegs.GPADIR.bit.PIN_DISPARO_BC = 1;
270     GpioDataRegs.GPACLEAR.bit.PIN_DISPARO_BC = 1;
271

```

svc4-Main.c

```

272     GpioCtrlRegs.GPAPUD.bit.PIN_DISPARO_CA = 0;    // Enable pull-up
273     GpioCtrlRegs.GPAMUX2.bit.PIN_DISPARO_CA = 0;
274     GpioCtrlRegs.GPADIR.bit.PIN_DISPARO_CA = 1;
275     GpioDataRegs.GPACLEAR.bit.PIN_DISPARO_CA = 1;
276
277     GpioCtrlRegs.GPAPUD.bit.PIN_CONTACTOR_1 = 0;    // Enable pull-up
278     GpioCtrlRegs.GPAMUX2.bit.PIN_CONTACTOR_1 = 0;
279     GpioCtrlRegs.GPADIR.bit.PIN_CONTACTOR_1 = 1;
280     GpioDataRegs.GPACLEAR.bit.PIN_CONTACTOR_1 = 1;
281
282     GpioCtrlRegs.GPAPUD.bit.PIN_CONTACTOR_2 = 0;    // Enable pull-up
283     GpioCtrlRegs.GPAMUX2.bit.PIN_CONTACTOR_2 = 0;
284     GpioCtrlRegs.GPADIR.bit.PIN_CONTACTOR_2 = 1;
285     GpioDataRegs.GPACLEAR.bit.PIN_CONTACTOR_2 = 1;
286
287     EDIS;
288
289 }
290
291 void ConfigurarXINT(void){
292
293     XIntruptRegs.XINT1CR.bit.POLARITY = 1;    //Xint1 Flanco de subida
294     XIntruptRegs.XINT2CR.bit.POLARITY = 1;    //Xint2 Flanco de subida
295     XIntruptRegs.XINT3CR.bit.POLARITY = 1;    //Xint3 Flanco de subida
296
297     XIntruptRegs.XINT1CR.bit.ENABLE = 1;    // Enable Xint1
298     XIntruptRegs.XINT2CR.bit.ENABLE = 1;    // Enable Xint2
299     XIntruptRegs.XINT3CR.bit.ENABLE = 1;    // Enable Xint3
300 }
301
302 interrupt void epwm1_isr(void) // Inicio de ciclo. Aquí se hace SOC y se fija la bandera que
    usa la interrupción del adc pa saber en q parte del pwm estamos
303 {
304     //contador1++;
305     AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1; // gatilla la interrupcion para el ADC
306     // Clear INT flag for this timer
307     EPwm1Regs.ETCLR.bit.INT = 1;
308     // Acknowledge this interrupt to receive more interrupts from group 3
309     PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
310 }
311 interrupt void adc_init_offset(void){
312
313     Offset_cnt++;
314
315     offsetVbt_AB = KI_offset*offsetVbt_AB+KP_offset*((float)(AdcRegs.ADCRESULT0 >>4));
316     offsetVbt_BC = KI_offset*offsetVbt_BC+KP_offset*((float)(AdcRegs.ADCRESULT2 >>4));
317     offsetVbt_CA = KI_offset*offsetVbt_CA+KP_offset*((float)(AdcRegs.ADCRESULT4 >>4));
318
319
320     if(Offset_cnt >= 30000){
321         EALLOW;
322         PieVectTable.ADCINT = &adc_isr;
323         EDIS;
324     }
325
326     //Limpiar ADC
327     AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;    // Resetea el contador de SEQ1

```

svc4-Main.c

```

328     AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;    // Clear INT SEQ1 bit
329     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE
330 }
331
332 interrupt void adc_isr(void)
333 {
334     Rutina_ADC(); //rutina de medicion voltajes y corrientes rms
335
336     AdcRegs.ADCTRL2.bit.RST_SEQ1 = 0x1;    // Resetea el contador de SEQ1
337     AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;    // Clear INT SEQ1 bit
338     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; // Acknowledge interrupt to PIE
339
340 }
341
342 void Rutina_ADC(void){
343     //Lectura ADC
344     //Vbt_AB=KgainVbt_AB*((float)(AdcRegs.ADCRESULT0 >>4)-offsetVbt_AB);
345     Vbt_AB=0.11*((float)(AdcRegs.ADCRESULT0 >>4)-offsetVbt_AB);
346     Vbt_BC=KgainVbt_BC*((float)(AdcRegs.ADCRESULT2 >>4)-offsetVbt_BC);
347     Vbt_CA=KgainVbt_CA*((float)(AdcRegs.ADCRESULT4 >>4)-offsetVbt_CA);
348
349     //Fase-Fase a Fase-Neutro
350     Vbt_A=0.3333333333333333*(Vbt_AB-Vbt_CA);
351     Vbt_B=0.3333333333333333*(Vbt_BC-Vbt_AB);
352     Vbt_C=0.3333333333333333*(Vbt_CA-Vbt_BC);
353
354     //RMS
355     Vbt_AB_2 = Vbt_AB*Vbt_AB;
356     Vbt_BC_2 = Vbt_BC*Vbt_BC;
357     Vbt_CA_2 = Vbt_CA*Vbt_CA;
358
359     Vbt_AB_f=8.85174031474353e-005*Vbt_AB_21+8.85174031474353e-005*Vbt_AB_22-
(-1.98109467563163*Vbt_AB_f1+0.98127171043792*Vbt_AB_f2);
360     Vbt_AB_22=Vbt_AB_21;
361     Vbt_AB_21=Vbt_AB_2;
362     Vbt_AB_f2=Vbt_AB_f1;
363     Vbt_AB_f1=Vbt_AB_f;
364
365     Vbt_BC_f=8.85174031474353e-005*Vbt_BC_21+8.85174031474353e-005*Vbt_BC_22-
(-1.98109467563163*Vbt_BC_f1+0.98127171043792*Vbt_BC_f2);
366     Vbt_BC_22=Vbt_BC_21;
367     Vbt_BC_21=Vbt_BC_2;
368     Vbt_BC_f2=Vbt_BC_f1;
369     Vbt_BC_f1=Vbt_BC_f;
370
371     Vbt_CA_f=8.85174031474353e-005*Vbt_CA_21+8.85174031474353e-005*Vbt_CA_22-
(-1.98109467563163*Vbt_CA_f1+0.98127171043792*Vbt_CA_f2);
372     Vbt_CA_22=Vbt_CA_21;
373     Vbt_CA_21=Vbt_CA_2;
374     Vbt_CA_f2=Vbt_CA_f1;
375     Vbt_CA_f1=Vbt_CA_f;
376
377     //Vbt_AB_rms = sqrt(Vbt_AB_f)*0.222222-138.8889;
378     Vbt_AB_rms = sqrt(Vbt_AB_f);
379     Vbt_BC_rms = sqrt(Vbt_BC_f);
380     Vbt_CA_rms = sqrt(Vbt_CA_f);
381

```

svc4-Main.c

```

382     Vbt_rms = (Vbt_AB_rms + Vbt_BC_rms + Vbt_CA_rms)*UNODIVRAIZTRES/3;
383
384 }
385
386 interrupt void xint1_isr(void)
387 {
388     //GpioDataRegs.GPATOGGLE.bit.GPIO31=1;      //Enciende o apaga el LED cuando se entra en
    la interrupcion
389     contador_AB++;
390
391     step_deseado_AB=step_deseado+contador_AB/10;
392     if(step_deseado_AB > 130) {contador_AB=0;}
393     //Variable que sirve para visualizar en watch expresions, para debugear.
394     deteccion_AB=TRUE;
395     step_counter_AB=0;                          //reseteamos el contador de ciclo
396     estado_AB_1=FALSE;                          //el pin de disparo AB está apagado
397
398     //enviar datos por puerto serie cada 20ms (interrupciones pares), ya que la interrupcion
    ocurre cada 10ms (cada semiciclo)
399     if (contador_AB%2==0){
400         scia_emitir(Vbt_rms);
401         scia_emitir(numero_bancos);
402         scia_emitir(angulo_disparo);
403         scia_emitir(V_setpoint);
404         scia_emitir('\n');
405     }
406
407     // Acknowledge this interrupt to get more from group 1
408
409     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
410 }
411
412 interrupt void xint2_isr(void)
413 {
414     //GpioDataRegs.GPATOGGLE.bit.GPIO31=1;      //Enciende o apaga el LED cuando se entra en
    la interrupcion
415     contador_BC++;
416     step_deseado_BC=step_deseado+contador_BC/10;
417     if(step_deseado_BC > 130) {contador_BC=0;}
418     deteccion_BC=TRUE;
419     step_counter_BC=0;                          //reseteamos el contador de ciclo
420     estado_BC_1=FALSE;                          //el pin de disparo BC está apagado
421
422     // Acknowledge this interrupt to get more from group 1
423     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
424 }
425
426 interrupt void xint3_isr(void)
427 {
428     //GpioDataRegs.GPATOGGLE.bit.GPIO31=1;      //Enciende o apaga el LED cuando se entra en
    la interrupcion
429     contador_CA++;
430
431     step_deseado_CA=step_deseado+contador_CA/10;
432     if(step_deseado_CA > 130) {contador_CA=0;}
433     deteccion_CA=TRUE;
434     step_counter_CA=0;                          //reseteamos el contador de ciclo

```



svc4-Main.c

```

435     estado_CA_1=FALSE;           //el pin de disparo CA está apagado
436
437     // Acknowledge this interrupt to get more from group 12
438     PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
439 }
440
441 interrupt void cpu_timer0_isr(void)
442 {
443
444 //Algoritmo de disparo en fase AB
445
446 if (deteccion_AB==TRUE){
447     if(step_counter_AB>step_deseado_AB && estado_AB_1==FALSE){
448         GpioDataRegs.GPASET.bit.PIN_DISPARO_AB = 1;
449         estado_AB_1=TRUE;
450     }
451     else if(step_counter_AB>step_deseado_AB && estado_AB_1==TRUE){
452         GpioDataRegs.GPACLEAR.bit.PIN_DISPARO_AB = 1;
453         deteccion_AB=FALSE;
454     }
455     else step_counter_AB++;
456 }
457
458 //Algoritmo de disparo en fase BC
459
460 if (deteccion_BC==TRUE){
461     if(step_counter_BC>step_deseado_BC && estado_BC_1==FALSE){
462         GpioDataRegs.GPASET.bit.PIN_DISPARO_BC = 1;
463         estado_BC_1=TRUE;
464     }
465     else if(step_counter_BC>step_deseado_BC && estado_BC_1==TRUE){
466         GpioDataRegs.GPACLEAR.bit.PIN_DISPARO_BC = 1;
467         deteccion_BC=FALSE;
468     }
469     else step_counter_BC++;
470 }
471
472 //Algoritmo de disparo en fase CA
473
474 if (deteccion_CA==TRUE){
475     if(step_counter_CA>step_deseado_CA && estado_CA_1==FALSE){
476         GpioDataRegs.GPASET.bit.PIN_DISPARO_CA = 1;
477         estado_CA_1=TRUE;
478     }
479     else if(step_counter_CA>step_deseado_CA && estado_CA_1==TRUE){
480         GpioDataRegs.GPACLEAR.bit.PIN_DISPARO_CA = 1;
481         deteccion_CA=FALSE;
482     }
483     else step_counter_CA++;
484 }
485
486 // Acknowledge this interrupt to receive more interrupts from group 1
487 PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //Si no se pone esto la interrupcion deja de servir
488                                           //Esta sentencia es una forma de resetear la
interrupcion
489                                           //O permitir que sigan ocurriendo interrupciones
del grupo 1 del vector pie

```

```

490 }
491
492 void scia_configurar()
493 {
494     SciaRegs.SCICCR.all =0x0007;    // 1 stop bit, No loopback
495                                     // Sin paridad,8 char bits,
496                                     // async mode, idle-line protocol
497     SciaRegs.SCICTL1.all =0x0003;   // Habilitar TX, RX, internal SCICLK,
498                                     // Disable RX ERR, SLEEP, TXWAKE
499     SciaRegs.SCICTL2.all =0x0003;
500     SciaRegs.SCICTL2.bit.TXINTENA =1;
501     SciaRegs.SCICTL2.bit.RXBKINTENA =1;
502     #if (CPU_FRQ_150MHZ)
503         SciaRegs.SCIHBAUD    =0x0001; // 9600 baud @LSPCLK = 37.5MHz.
504         SciaRegs.SCILBAUD    =0x00E7;
505     #endif
506     #if (CPU_FRQ_100MHZ)
507         SciaRegs.SCIHBAUD    =0x0001; // 9600 baud @LSPCLK = 20MHz.
508         SciaRegs.SCILBAUD    =0x0044;
509     #endif
510     SciaRegs.SCICTL1.all =0x0023;   // Relinquish SCI from Reset
511 }
512
513 void scia_emitir(int a) //Transmite un caracter
514 {
515     while (SciaRegs.SCIFFTX.bit.TXFFST != 0) {}
516     SciaRegs.SCITXBUF=(int)a;
517 }
518
519 void scia_msg(char * msg) //Transmite un arreglo de caracteres
520 {
521     int i;
522     i = 0;
523     while(msg[i] != '\0')
524     {
525         scia_emitir(msg[i]);
526         i++;
527     }
528 }
529
530 void scia_fifo_iniciar()
531 {
532     SciaRegs.SCIFFTX.all=0xE040;
533     SciaRegs.SCIFFRX.all=0x204f;
534     SciaRegs.SCIFFCT.all=0x0;
535
536 }
537
538 //=====
539 // Fin del código
540 //=====
541
542

```

## Modbus\_SCIB-Main.c

```
1 // DESCRIPCIÓN
2 //
3 // Rutina de comunicación Modbus del SVC.
4 // Ejemplo para monitorear, encender o apagar un LED utilizando holding register o coils
5 // Comunicación realizada por Puerto Serie B.
6 //#####
7 // $Microcontrolador: TMS320F283335 $
8 // $Fecha: 24 de Marzo, 2014 $
9 // $Autor: Andrés Vargas $
10 // $Modificado por: Martín Cornejo S.
11 //#####
12
13 #include "DSP28x_Project.h" // Device Headerfile and Examples Include File
14
15 Uint16 contador_rx=0;
16 Uint16 contador_tx=0;
17
18 // Prototype statements for functions found within this file.
19 void Scib_configurar(void);
20 void Scib_fifo_init(void);
21 void delay_loop();
22 void EnviarDatosModbus();
23 void InicializarRegistros();
24 void ReadCoilStatus(); //Leer bits de cambio de estado
25 void ReadInputStatus(); //Leer bits de estado de operación
26 void ReadHoldingRegisters(); //Leer floats modificables
27 void ReadInputRegisters(); //Leer floats de estado de operación
28 void ForceMultipleCoils(); //Modificar bits de cambio de estado
29 void PresetMultipleRegisters(); //Modificar registros float
30 void PresetSingleRegister(); //Modificar registros float
31 void ModbusExcep(Uint16);
32 interrupt void ScibRxFifoIsr(void);
33 interrupt void ScibTxFifoIsr(void);
34 interrupt void cpu_timer0_isr(void);
35 Uint16 AddrCheck(Uint16,Uint16,Uint16);
36 Uint16 CRC16(Uint16 *, Uint16);
37
38
39 #define SCI_PRD      243
40
41 //Variables Globales
42 #define FALSE      0
43 #define TRUE      1
44
45 //Variables de entrada
46 int Encendido=FALSE;
47 int holding = 8;
48 int analoginput = 7;
49
50 //Variables auxiliares
51 Uint16 ReceivedChar;
52 Uint16 contador=10;
53 char *msg;
54
55 //Estados Coil
56 #define COIL_LED      0
57 #define COIL_2      1
```

Modbus\_SCIB-Main.c

```
58
59 // Asignación de pines GPIO
60 #define PIN_MODBUS_DE          GPIO34
61 #define PIN_LED                 GPIO31
62 #define PIN_LED_COMUNICACIONES GPIO20
63
64 //Constantes para Modbus
65 #define MaxI                    75 //Max input variables que definen el tamaño máximo de los
    mensajes y las direcciones máximas de los registros
66 #define MaxQ                    32 //Max Coil
67 #define MaxAI                   75 // Max Analog Input
68 #define MaxHold                  32 // Max Holding Registers
69 #define ILLEGAL_FUNCTION        0x01 // Códigos de los errores de modbus
70 #define ILLEGAL_DATA_ADDRESS    0x02
71 #define ILLEGAL_DATA_VALUE      0x03
72 #define SLAVE_DEVICE_FAILURE    0x04
73 #define ACKNOWLEDGE             0x05
74 #define SLAVE_DEVICE_BUSY       0x06
75
76 //Matrices auxiliares para cálculo del CRC en protocolo Modbus
77 const Uint16 auchCRChi[] = {
78 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
79 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
80 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
81 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
82 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
83 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
84 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
85 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
86 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
87 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
88 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
89 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
90 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
91 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
92 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
93 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
94 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
95 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
96 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
97 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
98 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
99 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
100 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
101 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
102 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
103 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
104 } ;
105
106 const Uint16 auchCRCLo[] = {
107 0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
108 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
109 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
110 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
111 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
112 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0xD3, 0xD3,
113 0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
```



Modbus\_SCIB-Main.c

```

163 IER = 0x0000;
164 IFR = 0x0000;
165 InitPieVectTable();
166
167 InicializarRegistros();
168
169 EALLOW; // This is needed to write to EALLOW protected registers
170 PieVectTable.TINT0 = &cpu_timer0_isr;
171 PieVectTable.SCIRXINTB = &ScibRxFifoIsr;
172 PieVectTable.SCITXINTB = &ScibTxFifoIsr;
173 EDIS;
174
175 InitCpuTimers(); // For this example, only initialize the Cpu Timers
176
177 #if (CPU_FRQ_150MHZ) //Mi DSP es de 150 MHz
178 // Configure CPU-Timer 0 to interrupt every 500 milliseconds:
179 // 150MHz CPU Freq, 500 millisecond Period (in uSeconds)
180 ConfigCpuTimer(&CpuTimer0, 150, 10000); //Lo cambie a 1 segundo
181 #endif
182 #if (CPU_FRQ_100MHZ)
183 // Configure CPU-Timer 0 to interrupt every 500 milliseconds:
184 // 100MHz CPU Freq, 500 millisecond Period (in uSeconds)
185 ConfigCpuTimer(&CpuTimer0, 100, 500000);
186 #endif
187
188 CpuTimer0Regs.TCR.all = 0x4001;
189
190 EALLOW;
191 GpioCtrlRegs.GPAMUX2.bit.GPIO20 = 0;
192 GpioCtrlRegs.GPADIR.bit.GPIO20 = 1;
193 GpioDataRegs.GPACLEAR.bit.GPIO20 = 1;
194 GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0;
195 GpioCtrlRegs.GPADIR.bit.GPIO31 = 1;
196 GpioDataRegs.GPASET.bit.GPIO31 = 1;
197 GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;
198 GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;
199 GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;
200 EDIS;
201
202
203 IER |= M_INT1; //Siempre despues de configurar una interrupcion se debe habilitar mediante
IER y luego el comando
204 //Correspondiente a la interrupción
205 PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
206
207 PieCtrlRegs.PIECTRL.bit.ENPIE = 1; // Habilita el bloque PIE
208 PieCtrlRegs.PIEIER9.bit.INTx3 = 1; // Habilita SCIRXINTA
209 PieCtrlRegs.PIEIER9.bit.INTx4 = 1; // Habilita SCITXINTA
210 IER |= M_INT9; // Enable CPU Interrupt 9
211
212 EINT; // Enable Global interrupt INTM
213 ERTM; // Enable Global realtime interrupt DBGM
214 //Scib_fifo_init(); // Initialize the SCI FIFO
215 Scib_configurar(); // Initalize SCI for echoback
216
217 for(;;){
218

```

Modbus\_SCIB-Main.c

```

219     if(holding == 1 || Encendido == TRUE) GpioDataRegs.GPACLEAR.bit.PIN_LED = 1;
220     else GpioDataRegs.GPASET.bit.PIN_LED = 1;
221 }
222
223 }
224
225 // Test 1,Scib DLB, 8-bit word, baud rate 0x000F, default, 1 STOP bit, no parity
226 void Scib_configurar()
227 {
228     ScibRegs.SCICCR.all=0x0007; // 1 stop bit, Sin Loopback, Paridad par, 8 bits de datos,
    Async, Protocolo idle-line // antes era: ScibRegs.SCICCR.all=0x00E7; // 2 stop bits, Sin
    Loopback, Paridad par, 8 bits de datos, Async, Protocolo idle-line
229     ScibRegs.SCICTL1.all=0x0043; // Habilita TX, RX, SCICLK interno, RX ERR, Sleep,
    deshabilita TXWAKE// antes era: ScibRegs.SCICTL1.all=0x0003; // Habilita TX, RX, SCICLK
    interno, deshabilita RX ERR, Sleep, TXWAKE
230     ScibRegs.SCICTL2.bit.TXINTENA =1;
231     ScibRegs.SCICTL2.bit.RXBKINTENA =1;
232     ScibRegs.SCIHBAUD = (((int)(SCI_PRD)) >> 8) & 0x00FF;
233     ScibRegs.SCILBAUD = (int)(SCI_PRD) & 0x00FF;
234     ScibRegs.SCICTL1.all =0x0023; // Habilita SCI
235 }
236
237 // Initalize the SCI FIFO
238
239 interrupt void ScibRxFifoIsr(void)
240 {
241     GpioDataRegs.GPASET.bit.PIN_LED_COMUNICACIONES = 1;
242     contador_rx++;
243     if(ContadorFinDeCuadroModbus==0) {
244         RdataA_Num=0;
245     }
246     //if (RdataA_Num >= 1) GpioDataRegs.GPACLEAR.bit.PIN_LED = 1; //para probar si lee mas
    desde el 0
247     RdataA[RdataA_Num++]=ScibRegs.SCIRXBUF.all;
248     AnteriorModbus=RdataA[RdataA_Num-1];
249     if(GpioDataRegs.GPBDAT.bit.PIN_MODBUS_DE == 0 && ContadorEnvioModbus>0){ // Si no está
    enviando, activa el contador para recibir
250         ContadorFinDeCuadroModbus=10;
251         ContadorLedComunicaciones=200;
252     }
253     if(RdataA_Num>=69){
254         RdataA_Num=2;
255     }
256     PieCtrlRegs.PIEACK.all |= 0x100; // Emitir Ack para el PIE
257     ScibRegs.SCIFFRX.bit.RXFFOVRCLR=1; // Borrar bandera overflow
258     ScibRegs.SCIFFRX.bit.RXFFINTCLR=1; // Borrar bandera interrupción
259 }
260
261 interrupt void ScibTxFifoIsr(void)
262 {
263     if(ComenzarEnvio==TRUE){
264         //GpioDataRegs.GPACLEAR.bit.PIN_LED = 1;
265         ContadorLedComunicaciones=200;
266         CRCCheck=CRC16(RdataA,RdataA_Num-2);
267         if((CRCCheck==(RdataA[RdataA_Num-1]+(RdataA[RdataA_Num-2]<<8)))){ // Calcula CRC
268             switch(RdataA[1]){
269                 case 1:

```

Modbus\_SCIB-Main.c

```

270         ReadCoilStatus(); //Leer bits de cambio de estado
271         break;
272     case 2:
273         ReadInputStatus(); //Leer bits de estado de operación
274         break;
275     case 3:
276         ReadHoldingRegisters(); //Leer floats modificables
277         break;
278     case 4:
279         ReadInputRegisters(); //Leer floats de estado de operación
280         break;
281     case 6:
282         PresetSingleRegister(); //Leer floats de estado de operación
283         break;
284     case 15:
285         ForceMultipleCoils(); //Modificar bits de cambio de estado
286         break;
287     case 16: //case 6:
288         PresetMultipleRegisters(); //Modificar registros float
289         break;
290     default:
291         ModbusExcep(ILLEGAL_FUNCTION);
292         break;
293     }
294     ComenzarEnvio=FALSE;
295 }
296 }else{
297     if(ContadorEnvioModbus<SdataA_Num){
298         ScibRegs.SCITXBUF=SdataA[ContadorEnvioModbus++] & 0x00FF; // Send data
299     }
300 }
301
302 ScibRegs.SCIFFTX.bit.TXFFINTCLR=1; // Borrar bandera interrupción
303 PieCtrlRegs.PIEACK.all|=0x100; // Emitir Ack para el PIE
304 }
305
306 void InicializarRegistros(){
307     Uint16 i=0;
308     DiscreteInput[0]=&Encendido;
309
310     HoldingRegister[i]=(int *)&holding;
311     HoldingRegister[i++]=HoldingRegister[i++]+1;
312     i=0;
313     InputRegister[i]=(int *)&analoginput;
314     InputRegister[i++]=InputRegister[i++]+1;
315 }
316 }
317
318 Uint16 CRC16(Uint16 *puchMsg, Uint16 usDataLen)
319 {
320     Uint16 uchCRCHi = 0xFF ; // βCRC×Ö%Ú³õÊ%~ */
321     Uint16 uchCRCLo = 0xFF ; // μÍCRC ×Ö%Ú³õÊ%~ */
322     Uint32 uIndex ; // CRCÑ-»·ÖĐμÄË÷Öý */
323     while (usDataLen-->0) // * «ËäïüÏϕ»³äçø */
324     {
325         uIndex = uchCRCHi ^ *puchMsg++ ; /* %ÄËäCRC */
326         uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;

```



Modbus\_SCIB-Main.c

```

327     uchCRCLo = auchCRCLo[uIndex] ;
328 }
329 return (uchCRCHi << 8 | uchCRCLo) ;
330 }
331
332 void EnviarDatosModbus(){
333     if(ContadorFinDeCuadroModbus==0) { // Sólo si el bus no está en uso
334         ContadorEnvioModbus=0;
335         contador_tx++;
336         GpioDataRegs.GPBSET.bit.PIN_MODBUS_DE = 1;
337         ScibRegs.SCITXBUF=SdataA[ContadorEnvioModbus++] & 0x00FF; // Send data
338     }
339 }
340
341 void ReadCoilStatus(){ //Leer bits de cambio de estado
342     Uint16 AddrStart=0;
343     Uint16 TempAddr=0;
344     Uint16 CoilCount=0;
345     Uint16 ByteCount=0;
346     Uint16 TempData=0;
347     Uint16 i=0,k=0;
348
349     AddrStart = (RdataA[2]<<8) + RdataA[3];
350     CoilCount = (RdataA[4]<<8) + RdataA[5];
351     ByteCount = CoilCount/8;
352     if(CoilCount%8 != 0){
353         ByteCount+=1;
354     }
355     if(AddrCheck(AddrStart,CoilCount,MaxQ)){
356         return;
357     }
358
359     TempAddr=AddrStart;
360     for(k=0;k<ByteCount;k++){
361         SdataA[k + 3] = 0;
362         for(i=0;i<8;i++){
363             switch(TempAddr){
364                 case COIL_LED:
365                     TempData=Encendido;
366                     break;
367                 case COIL_2:
368                     break;
369
370                 default:
371                     TempData=0;
372                     break;
373             }
374             SdataA[k + 3] |= TempData << i;
375             TempAddr++;
376             if(TempAddr >= AddrStart+CoilCount){
377                 k=ByteCount;
378                 i=8;
379             }
380         }
381     }
382
383     SdataA[0] = Local;

```

Modbus\_SCIB-Main.c

```

384 SdataA[1] = RdataA[1];
385 SdataA[2] = ByteCount;
386
387 CRCCheck = CRC16(SdataA,ByteCount+3);
388 SdataA[ByteCount+3] = CRCCheck >> 8;
389 SdataA[ByteCount+4] = CRCCheck & 0xff;
390
391 SdataA_Num = ByteCount+5;
392
393 EnviarDatosModbus();
394 }
395
396 void ReadInputStatus(){ //Leer bits de estado de operación
397     Uint16 AddrStart=0;
398     Uint16 TempAddr=0;
399     Uint16 InputCount=0;
400     Uint16 ByteCount=0;
401     Uint16 TempData=0;
402     Uint16 i=0,k=0;
403
404     AddrStart = (RdataA[2]<<8) + RdataA[3];
405     TempAddr=AddrStart;
406     InputCount = (RdataA[4]<<8) + RdataA[5];
407     ByteCount = InputCount / 8;
408     if(InputCount%8 != 0)
409         ByteCount+=1;
410     if(AddrCheck(AddrStart,InputCount,MaxI)){
411         return;
412     }
413     for(k=0;k<ByteCount;k++){
414         SdataA[k + 3] = 0;
415         for(i=0;i<8;i++){
416             TempData=*DiscreteInput[TempAddr];
417             SdataA[k + 3] |= TempData << i;
418             TempAddr++;
419             if(TempAddr >= AddrStart+InputCount){
420                 k=ByteCount;
421                 i=8;
422             }
423         }
424     }
425
426     SdataA[0] = Local;
427     SdataA[1] = RdataA[1];
428     SdataA[2] = ByteCount;
429
430     CRCCheck = CRC16(SdataA,ByteCount+3);
431     SdataA[ByteCount+3] = CRCCheck >> 8;
432     SdataA[ByteCount+4] = CRCCheck & 0xff;
433
434     SdataA_Num = ByteCount+5;
435
436     EnviarDatosModbus();
437 }
438
439 void ReadHoldingRegisters(){ //Leer floats modificables
440     Uint16 AddrStart=0;

```

Modbus\_SCIB-Main.c

```

441  Uint16 TempAddr=0;
442  Uint16 RegistersCount=0;
443  Uint16 ByteCount=0;
444  Uint16 TempData=0;
445  Uint16 i=0;
446
447  AddrStart = (RdataA[2]<<8) + RdataA[3];
448  TempAddr=AddrStart;
449  RegistersCount = (RdataA[4]<<8) + RdataA[5];
450  ByteCount = RegistersCount * 2;
451
452  if(AddrCheck(AddrStart,RegistersCount,MaxHold)){
453      return;
454  }
455
456  for(i=0;i<ByteCount;i+=2,TempAddr++){
457      TempData=*HoldingRegister[TempAddr];
458      SdataA[i+3] = TempData >> 8;
459      SdataA[i+4] = TempData & 0xff;
460  }
461
462
463  SdataA[0] = Local;
464  SdataA[1] = RdataA[1];
465  SdataA[2] = ByteCount;
466  CRCCheck = CRC16(SdataA,ByteCount+3);
467  SdataA[ByteCount+3] = CRCCheck >> 8;
468  SdataA[ByteCount+4] = CRCCheck & 0xff;
469  SdataA_Num = ByteCount + 5;
470
471  EnviarDatosModbus();
472 }
473
474 void ReadInputRegisters(){ //Leer floats de estado de operación
475     Uint16 AddrStart=0;
476     Uint16 TempAddr=0;
477     Uint16 RegistersCount=0;
478     Uint16 ByteCount=0;
479     Uint16 TempData = 0;
480     Uint16 i=0;
481
482     AddrStart = (RdataA[2]<<8) + RdataA[3];
483     TempAddr=AddrStart;
484     RegistersCount = (RdataA[4]<<8) + RdataA[5];
485     ByteCount = RegistersCount * 2;
486
487     if(AddrCheck(AddrStart,RegistersCount,MaxAI)){
488         return;
489     }
490
491     for(i=0;i<ByteCount;i+=2,TempAddr++)
492     {
493         TempData=*InputRegister[TempAddr];
494         SdataA[i+3] = TempData >> 8;
495         SdataA[i+4] = TempData & 0xff;
496     }
497     SdataA[0] = Local;

```

```

498   SdataA[1] = RdataA[1];
499   SdataA[2] = ByteCount;
500   CRCCheck = CRC16(SdataA,ByteCount+3);
501   SdataA[ByteCount+3] = CRCCheck >> 8;
502   SdataA[ByteCount+4] = CRCCheck & 0xff;
503   SdataA_Num = ByteCount + 5;
504
505   EnviarDatosModbus();
506 }
507
508 void ForceMultipleCoils(){
509     Uint16     AddrStart=0;
510     Uint16     TempAddr=0;
511     Uint16     SetCount=0;
512     unsigned   TempState=0;
513
514     AddrStart = (RdataA[2]<<8) + RdataA[3];
515     TempAddr = AddrStart;
516     SetCount = (RdataA[4]<<8) + RdataA[5];
517
518     if(SetCount>1){
519         ModbusExcep(ILLEGAL_DATA_ADDRESS);
520         return;
521     }
522     if(AddrCheck(AddrStart,SetCount,MaxQ)){
523         return;
524     }
525
526     TempState = (RdataA[7]) & 0x01;
527
528     switch(TempAddr){
529     case COIL_LED:
530         if(TempState){
531             Encendido= TRUE;
532             SdataA[0] = Local;
533             SdataA[1] = RdataA[1];
534             SdataA[2] = AddrStart >> 8;
535             SdataA[3] = AddrStart & 0xff;
536             SdataA[4] = SetCount >> 8;
537             SdataA[5] = SetCount & 0xff;
538             CRCCheck = CRC16(SdataA,6);
539             SdataA[6] = CRCCheck >> 8;
540             SdataA[7] = CRCCheck & 0xff;
541             SdataA_Num = 8;
542             EnviarDatosModbus();
543         }
544         break;
545     case COIL_2:
546         if(TempState){
547             Encendido= FALSE;
548             SdataA[0] = Local;
549             SdataA[1] = RdataA[1];
550             SdataA[2] = AddrStart >> 8;
551             SdataA[3] = AddrStart & 0xff;
552             SdataA[4] = SetCount >> 8;
553             SdataA[5] = SetCount & 0xff;
554             CRCCheck = CRC16(SdataA,6);

```

Modbus\_SCIB-Main.c

```

555             SdataA[6] = CRCCheck >> 8;
556             SdataA[7] = CRCCheck & 0xff;
557             SdataA_Num = 8;
558             EnviarDatosModbus();
559         }
560         break;
561     default:
562         ModbusExcep(ILLEGAL_DATA_ADDRESS);
563         break;
564     }
565 }
566
567 void ModbusExcep(Uint16 ExceptionCode){
568     SdataA[0] = Local;
569     SdataA[1] = RdataA[1]+0x80;
570     SdataA[2] = ExceptionCode;
571
572     CRCCheck = CRC16(SdataA,3);
573     SdataA[3] = CRCCheck >> 8;
574     SdataA[4] = CRCCheck & 0xFF;
575
576     SdataA_Num = 5;
577
578     EnviarDatosModbus();
579 }
580
581 void PresetMultipleRegisters(){ //Modificar registros float
582     Uint16 AddrStart=0;
583     Uint16 TempAddr=0;
584     Uint16 SetCount=0;
585     Uint16 TempData=0;
586     Uint16 i=0;
587
588     AddrStart = (RdataA[2]<<8) + RdataA[3];
589     TempAddr = AddrStart;
590     SetCount = (RdataA[4]<<8) + RdataA[5];
591
592     if(AddrCheck(AddrStart,SetCount,MaxHold)){
593         return;
594     }
595
596     for(i=0;i<SetCount;i++,TempAddr++){
597         TempData = (RdataA[i*2+7]<<8) + RdataA[i*2+8];
598         *HoldingRegister[TempAddr]=TempData;
599     }
600
601     SdataA[0] = Local;
602     SdataA[1] = RdataA[1];
603     SdataA[2] = AddrStart >> 8;
604     SdataA[3] = AddrStart & 0xff;
605     SdataA[4] = SetCount >> 8;
606     SdataA[5] = SetCount & 0xff;
607     CRCCheck = CRC16(SdataA,6);
608     SdataA[6] = CRCCheck >> 8;
609     SdataA[7] = CRCCheck & 0xff;
610     SdataA_Num = 8;
611

```

```

612     EnviarDatosModbus();
613 }
614
615 void PresetSingleRegister(){
616     Uint16 AddrStart=0;
617     Uint16 TempAddr=0;
618     //Uint16 SetCount=0;
619     Uint16 TempData=0;
620     //Uint16 i=0;
621
622     AddrStart = (RdataA[2]<<8) + RdataA[3];
623     TempAddr = AddrStart;
624     //SetCount = (RdataA[4]<<8) + RdataA[5];
625
626     //if(AddrCheck(AddrStart,SetCount,MaxHold)){
627     // return;
628     //}
629
630     TempData = (RdataA[4]<<8) + RdataA[5];
631     *HoldingRegister[TempAddr]=TempData;
632
633     SdataA[0] = Local;
634     SdataA[1] = RdataA[1];
635     SdataA[2] = AddrStart >> 8;
636     SdataA[3] = AddrStart & 0xff;
637     SdataA[4] = RdataA[4] >> 8;
638     SdataA[5] = RdataA[5] & 0xff;
639     CRCCheck = CRC16(SdataA,6);
640     SdataA[6] = CRCCheck >> 8;
641     SdataA[7] = CRCCheck & 0xff;
642     SdataA_Num = 8;
643
644     EnviarDatosModbus();
645
646 }
647
648 Uint16 AddrCheck(Uint16 AddrStart,Uint16 Num,Uint16 AddrMax)
649 {
650     Uint16 Result=0;
651     if(AddrStart+Num>AddrMax){
652         ModbusExcep(ILLEGAL_DATA_ADDRESS);
653         Result=1;
654     }
655     return Result;
656 }
657
658 interrupt void cpu_timer0_isr(void)
659 {
660     if(ContadorFinDeCuadroModbus>0){
661         ContadorFinDeCuadroModbus--;
662         if((RdataA[0]==Local)&&(ContadorFinDeCuadroModbus==0)){
663             ComenzarEnvio=TRUE;
664             ScibRegs.SCICTL1.bit.TXWAKE=1;
665             ScibRegs.SCITXBUF=0x00DC; // Da lo mismo que se pone acá, no se va a mandar.
666             Sólo se busca conseguir una interrupción del SCI
667         }
668     }
669 }

```

Modbus\_SCIB-Main.c

```
668 // FIN Código para encontrar fin de cuadro Modbus
669
670 // Manejo Pin MODBUS_DE
671 if(ContadorEnvioModbus>=SdataA_Num){
672     ContadorEnvioModbus++;
673     if(ContadorEnvioModbus>=SdataA_Num+15){
674         GpioDataRegs.GPBCLEAR.bit.PIN_MODBUS_DE = 1;
675     }
676 }
677
678 // Código para manejo de LEDs
679 //GpioDataRegs.GPACLEAR.bit.PIN_PROCESAMIENTO = 1; // Para ver cuanto tiempo de
procesamiento está usado
680 if(ContadorLedComunicaciones>0){
681     ContadorLedComunicaciones--;
682     if(ContadorLedComunicaciones==0){
683         GpioDataRegs.GPACLEAR.bit.PIN_LED_COMUNICACIONES = 1;
684     }
685 }else{
686     GpioDataRegs.GPBCLEAR.bit.PIN_MODBUS_DE= 1;
687     ContadorEnvioModbus=1;
688     ContadorFinDeCuadroModbus=0;
689 }
690
691 PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
692 }
693
694
```

## **Apéndice C**

# **Interfaz en Matlab-Simulink**



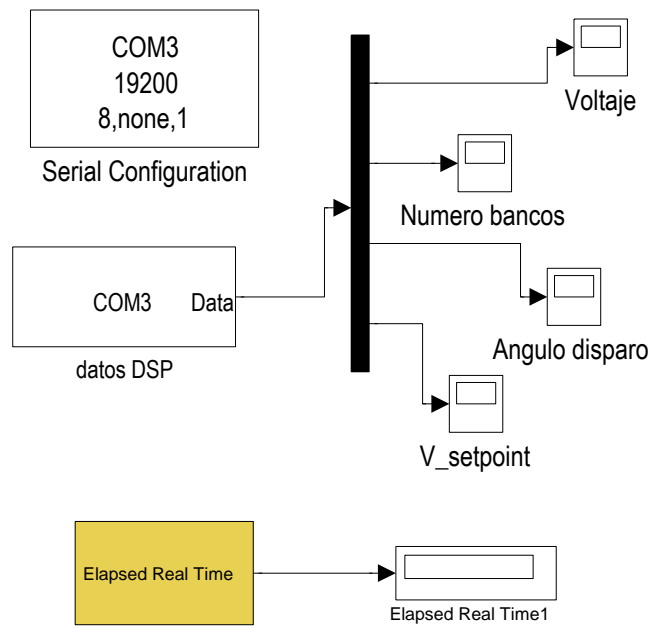


Figura C.1: Interfaz para visualizar las variables del CER