Universidad Autónoma de Querétaro
Facultad de Ingeniería.

Reconversión en Hardware de una máquina SMD RX-4A.

TESIS

Que como parte de los requisitos
Para obtener el grado de

Maestro en Ciencias en Mecatrónica

Presenta

César Barrón Romero

San Juan del Río, Querétaro Septiembre de 2012

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias (Mecatrónica)

Reconversión en Hardware de una máquina SMD RX-4A

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias

Presenta:
César Barrón Romero

Dirigido por:
Dr. Roque Alfredo Osornio Ríos

SINODALES

Dr. Roque Alfredo Osornio Ríos
Presidente — Firma

Dr. René de Jesús Romero Troncoso
Secretario — Firma

Dr. Luis Morales Velázquez
Vocal — Firma

Dr. Miguel Trejo Hernández
Suplente — Firma

Dr. Juan Primo Benítez Rangel
Suplente — Firma

Dr. Aurelio Domínguez González
Director de la Facultad de Ingeniería

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Septiembre de 2012
México

RESUMEN

La industria electrónica nacional es un pilar central de nuestra economía. México es un país líder en la manufactura de productos electrónicos de gran demanda a nivel mundial, como teléfonos celulares, aparatos de video juegos, computadoras, electrodomésticos y automotrices, los cuales utilizan la tecnología SMD (Surface Mounting Device) para el ensamble de componentes electrónicos chip sobre tableta PCB. Actualmente existen más de 730 plantas manufactureras relacionadas con la industria electrónica en México y es uno de los sectores de más rápido crecimiento en el país en términos de potencial exportador. Una estrategia que fortalece la competitividad y el desarrollo de este sector, es la investigación aplicada y los proyectos de colaboración entre las universidades y las empresas. Por lo anterior, en este trabajo se presenta la reconversión en Hardware de una máquina de componentes SMD RX-4A mediante IP cores basados en FPGA (Field Programmable Gate Array), esta reconversión, además permitir el uso de tecnología de nueva generación aplicada a la máquina en su sistema de selección de componentes y posicionamiento x-y, es compatible con el estándar RS-274X de EIA (Electronic Industry Association). El sistema desarrollado e implementado consta de un controlador FPGA para la etapa de posicionamiento x-y que en operatividad reduce en un factor de 100 el error contra un controlador comercial, un controlador difuso para la selección de componentes en carrusel y un controlador para la rotación en 8 direcciones. El sistema completo fue sometido a pruebas de operatividad para compararlo con su condición de operación de fábrica, obteniendo como resultado un menor error de posicionamiento, una mayor velocidad de operación, un sistema de control de arquitectura abierta, flexible, reconfigurable y de bajo costo.

(Palabras clave: SMD**,** controlador, FPGA, posicionamiento x-y, IP cores**)**

ABSTRACT

The national electronics industry is a central pillar of our economy. Mexico is a leading manufacturer of electronic products in great demand worldwide as cell phones, video games, computers, home appliances and automotives, which uses SMD technology (Surface Mounting Device) for the assembly of electronic components tablet chip on PCB. Currently there are more than 730 manufacturing facilities related to the electronics industry in Mexico and is one of the fastest growing sectors in the country in terms of export potential. A strategy that strengthens the competitiveness and development of this sector is the applied research and collaborative projects between universities and enterprises. Therefore, in this paper, the conversion in hardware on a components machine SMD RX-4A using IP cores based on FPGA (Field Programmable Gate Array) which also allow the use of new generation technology applied to the machine in his system component selection and positioning x-y, is compatible with the standard RS-274X EIA (Electronic Industry Association). The system developed and implemented consists of a FPGA controller for the stage of positioning x-y stage, that in operation reduces by a factor of 100 the error against a commercial controller, a fuzzy controller for selection of components on the rotatory table, and a controller for rotation 8 directions. The complete system was tested for operability to compare it with its factory operating condition, resulting in a lower positioning error, greater operating speed control, a system of open architecture, flexible, reconfigurable and inexpensive.

(**Keywords:** SMD, controller, FPGA, x-y positioning, IP cores)

AGRADECIMIENTOS

Agradezco a mis padres por darme la existencia, Paulina Romero y José Luis Barrón que desafortunadamente no le alcanzo el tiempo para ver cumplir la culminación de este proyecto, pero que en su momento me aconsejo superarme día con día. También muy en especial a mi hermano Adolfo que me enseño que la constancia y perseverancia llevan al éxito en las metas que uno se fija y al resto de mi familia por su apoyo.

A mi asesor, Dr. Roque Alfredo Osornio Ríos por sus observaciones siempre acertadas en el desarrollo del proyecto, y las facilidades brindadas. A mi co-asesor, Dr. Luis Morales Velázquez que siempre tuvo la disposición de ayudarme y disipar mis dudas. A mis sinodales, Dr. René de J. Romero Troncoso, Dr. Miguel Trejo Hernández y Dr. Juan Primo Benítez Rangel que de una u otra manera aprendí que un proyecto siempre se puede mejorar tomando en cuenta su gran experiencia y comentarios.

A mis compañeros y amigos Juan, Francisco, Ángel y Mario que me brindaron su amistad durante el tiempo que duro la carrera y me apoyaron en todo lo que necesite.

A la Universidad Autónoma de Querétaro por permitir superarme profesionalmente, al poner a mi disposición un gran cumulo de conocimiento.

A CONACYT por la beca otorgada durante el desarrollo de este proyecto.

Índice general

Índice de figuras

Figura

Índice de tablas

Tabla

Capítulo I

Introducción

1.1 Antecedentes

Desde la adopción de la computadora, y consigo las tarjetas electrónicas, las tecnologías de montaje y colocación de componentes superficial han tenido un crecimiento exponencial, esto debido a la gran demanda de productos electrónicos que invaden casi todas las actividades de la vida cotidiana. Si bien la construcción de tarjetas o placas electrónicas prototipo puede lograrse con cierta habilidad de manera manual, actualmente un proceso que es ampliamente aceptado en la industria electrónica es la tecnología de montaje superficial (SMD). La producción en serie de tarjetas electrónicas y la miniaturización de componentes utilizados en el ensamble de éstas, tales como transistores, resistencias, capacitores e inductores, requieren el uso de máquinas-herramienta que realicen las operaciones de selección, rotación y colocación precisa de estos componentes; para tal efecto se utilizan específicamente las máquinas de montaje chip (SMD).

La importancia de las máquinas-herramienta ha cobrado gran interés durante las últimas dos o tres décadas, ya que la incorporación de la electrónica y la informática, han permitido automatizar diversas operaciones que en gran medida se enfocan en los procesos de maquinado y manufactura de componentes electrónicos como por ejemplo: centros de torneado, centros de fresado, máquinas de electroerosión, centros de fabricación y ensamble de componentes electrónicos, etc. Este interés ha llegado hasta el punto de que hoy día en gran medida se diseñan y construyen unidades de control electrónico muy complejas para la industria, capaces de controlar la operación de una sola máquina, de un grupo de máquinas o incluso de toda una planta de manufactura, y solo en algunos casos se opta por la reconversión o modificación de las máquinas-herramienta tanto en Software como Hardware por la misma industria manufacturera.

Hasta la fecha, en la Universidad Autónoma de Querétaro se han realizado diferentes trabajos relacionados con la automatización, control y reconversión de máquinas-herramientas entre los cuales se encuentra el de Ronquillo (2002), quien desarrolló un control digital de

servomotor sin escobillas de corriente directa, el cual pretende dar origen al dominio de una tecnología que permita la producción nacional de sistemas de control de movimientos los cuales pudieran ser integrados a sistemas de control numérico computarizado, mediante la integración parcial de resultados en sistemas de posicionamiento de precisión, empleados en la fabricación de circuitos impresos y otros componentes en empresas locales de Querétaro. Osornio (2004) por su parte diseñó y construyó una tarjeta controladora de 3 ejes, con la cual se disminuyen los costos con respecto a las tarjetas comerciales, pero lo más importante es la generación de una independencia tecnológica en el área relacionada con los controles de posicionamiento y la aportación de ser un trabajo base para desarrollos subsecuentes. Trejo (2006) realizó un módulo de maquinado y monitoreo, aplicando control difuso en un proceso de torneado el cual mejora cuantitativamente el proceso reduciendo el tiempo de maquinado y aumenta el nivel de producción, mediante algoritmos de control adaptable. Mejía (2008) describió el diseño e implementación en software con C++, de un compilador para el desarrollo de un sistema de control para Máquina Herramienta de Control Numérico (MHCN) basado en la interpretación de código estándar G y M, así como también el desarrollo de una interfaz de usuario que permita visualizar y manufacturar piezas a partir de programas CNC (Control Numérico por Computadora) generados en software comercial CAD/CAM (Diseñó y Manufactura Asistido por Computadora respectivamente). Por su parte Muñoz (2009) desarrolló la arquitectura de un procesador PLC en un FPGA mediante lenguaje de descripción de hardware y su interfaz de entradas y salidas, para controlar los eventos discretos de una máquina de control numérico o de máquinas-herramientas que en alguna medida hacen uso de las señales discretas. Recientemente Vera (2010) trabajó sobre un sistema de medición y análisis de vibraciones inalámbrico basado en FPGA, aplicado a una máquina ensambladora de componentes electrónicos de chip, el cual sirvió como medio de monitoreo de las vibraciones y dinámica de la máquina, esto con la finalidad de diagnosticar y determinar que mejoras se pueden implementar en la máquina de montaje de componentes electrónicos. Cabe mencionar que los trabajos antes mencionados se enfocan en el mejoramiento de la operatividad de máquinas-herramienta y equipo existente, además de generar opciones tecnológicas independientes en las aéreas relacionadas con su campo de aplicación.

A nivel internacional se encuentran algunos trabajos sobre todo relacionados con la simulación mediante software o de algún algoritmo de optimización en tiempo de colocación de componentes electrónicos, como el de Wang et al. (1999) el cual presentó resultados experimentales usando algoritmos genéticos para optimizar el aplicador de alta velocidad paralelo de componentes en una máquina multiestación de ensamble SMT. Tirpak et al. (2002) trabajó en el desarrollo de un simulador cuya plataforma fue C++, basado en la clasificación genérica y la programación orientada a objetos el cual permite la jerarquización de la estructura física de la máquina y su operación. Cabe mencionar que el juego de herramientas del programa puede ser configurado para realizar la simulación de una máquina en particular. Kumar y Luo (2003) utilizaron un modelo TSP sobre una máquina de montaje de componentes FUJI FCP-IV para la optimización en tiempo de ensamble. La formulación da una descripción matemática del tiempo de ensamble a ser optimizado como un problema completamente de programación. Su simulación ha sido ejecutada en tarjetas de Lexmark, Inc., Lexington, KY. Wang y Zhang (2006) por su parte trabajaron sobre un algoritmo de alta velocidad para centrar y rotar el ángulo de posicionamiento de componentes electrónicos, utilizando el área mínima de sensado sobre la geometría de los componentes y aumentando la precisión y velocidad de colocación. El algoritmo computacional y procesamiento de imagen fue desarrollado en lenguaje de programación Visual C++ 6.0. Por su parte Ho y Ji (2009) propusieron 2 modelos matemáticos cuya solución con algoritmos genéticos minimiza el tiempo de ensamble de componentes electrónicos realizados por una máquina de inserción automática del tipo mesa movible x-y, alimentadores y torreta giratoria múltiple de componentes electrónicos.

En base a los antecedentes anteriores, y que en la UAQ se cuenta con una máquina de montaje de componentes superficial SMD RX-4A, misma que ha caído en obsolescencia, se hace necesario el proyecto que se encargue de su actualización, que mediante el estudio y aplicación de nuevas tecnologías, pretende su reconversión en Hardware enfocado a la operatividad de la máquina, la cual como algunas otras máquinas-herramienta consta de un sistema de posicionamiento de precisión, pero que además cuenta con un sistema síncrono complejo de alimentación, detección, despacho, fijación y colocación de componentes electrónicos.

Se espera que los eventos involucrados en estas operaciones, sean funcionales en su operatividad secuencial, de la colocación y ensamble de componentes, mediante su reconversión utilizando las herramientas teóricas y prácticas obtenidas en la formación de la maestría.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar la reconversión de una máquina de componentes chip SMD ARX-4A que permita el control de los eventos que involucra la máquina durante la selección, rotación y montaje de componentes electrónicos, para obtener una mejora en su operatividad y en la arquitectura del hardware con respecto a su condición original, mediante desarrollo e integración de IP cores propios basados en FPGA.

1.2.2 Objetivos Particulares

- Instalar y acondicionar un controlador de posicionamiento basado en FPGA para el control en la bancada de la máquina SMD ARX-4A.

- Interpretar los datos provenientes de un código estándar entregado por un software de diseño de PCB para generar las referencias al controlador mediante un módulo basado en FPGA, que se aplicarán a los ejes de la máquina.

- Desarrollar un módulo de adquisición A/D basado en FPGA para controlar y monitorear las señales que requiera el proceso de funcionamiento de la máquina ARX-4A.

- Desarrollar e instrumentar un controlador de posicionamiento para el sistema de selección de componentes (carrusel), utilizando las técnicas de control difuso basado en FPGA.

- Integrar una interfaz de comunicación USB basada en FPGA para la comunicación de la máquina con el control y la interfaz de usuario.

- Instrumentar y acondicionar los eventos de la máquina ARX-4 mediante la integración de elementos mecatrónicos para su integración con el resto de los módulos generados.

- Desarrollar pruebas funcionales mediante la integración de los módulos basados en FPGA y la interfaz desarrollada en otro proyecto para comprobar el funcionamiento esencial del hardware.

1.3 Justificación

En México un sector importante de la industria manufacturera está enfocado a satisfacer las demandas del ramo automotriz, que para mantener los estándares de calidad y los niveles de producción exigidos por los mercados nacionales y extranjeros, requiere de soluciones en automatización y control que se adecuen a su entorno y sus posibilidades de inversión para su mejora y crecimiento.

Un aspecto importante por el cual atraviesa en buena medida la industria nacional es el hecho de que en su mayoría son pequeñas empresas, las cuales cuentan con máquinas-herramientas o CNC de fabricación extranjera obsoleta, esto aunado a que los industriales optan por importar maquinaria e inclusive transferencia de líneas completas que son ya consideradas como obsoletas en los países desarrollados, como Estados Unidos, Japón, Inglaterra, Alemania y Francia, entre otros. Dejando así a la industria Mexicana con equipos de producción muy poco rentables debido a su obsolescencia.

Si bien la adquisición de equipos nuevos es una solución a este problema, esto representa altos costos de inversión que en gran medida este tipo de empresas no pueden pagar. Por ende la reconversión de estas máquinas-herramientas, por medio de sistemas electrónicos de medición, regulación y control, o dicho de otra manera, la automatización de una máquina, puede ser una mejor alternativa a la solución del problema.

Como bien se menciona en los antecedentes, el mejoramiento de operatividad e independencia tecnológica son aspectos importantes, sin embargo, otros puntos importantes para trabajar en la reconversión de máquinas-herramientas, en este caso el hardware de una de montaje de componentes (SMD), es que en la Facultad de Ingeniería de la Universidad Autónoma de Querétaro, no hay algún precedente en la elaboración de trabajos o proyectos de automatización relacionados con este tipo de máquinas, esto aunado a la necesidad de la empresa Clarion de regresar la máquina RX-4A a su vida útil productiva con mejoras en su operatividad, abre un área de oportunidad para realizar la aplicación de las nuevas tecnologías digitales basadas en FPGA de forma tangible.

1.4 Planteamiento General

Un sistema de control de movimiento o posicionamiento es un tipo de automatización comúnmente requerida en la industria nacional, algunos ejemplos de ellos son: Las máquinas de medición de coordenadas (CMM), las máquinas de control numérico computarizado (CNC), los proyectores de perfiles, las máquinas de montaje de componentes, entre otras. Estos equipos se utilizan principalmente en aquellas empresas que se dedican al sector manufacturero.

La automatización o reconversión de maquinaria representa un gran problema para muchas de estas empresas, ya que su principal obstáculo es la solvencia económica, pues no pueden invertir en la compra de equipos nuevos o la modernización de ellos, mediante tecnología de punta que usualmente es de arquitectura cerrada que no son en realidad económicos.

El desarrollo de este proyecto para lograr la reconversión en hardware de una máquina SMD RX-4A, se ha divido en varias etapas, entre las cuales las de mayor importancia son: la instalación y el acondicionamiento de un controlador basado en FPGA de la bancada o sistema de posicionamiento (x,y), sincronización y posicionamiento del carrusel de componentes electrónicos con la bancada, desarrollo de módulos A/D para monitorear y controlar las señales que se requieran, realizar pruebas funcionales mediante la integración de módulos basados en FPGA.

Y una interfaz de un proyecto alterno que enviará los datos de un código generado por algún software de diseño de PCB. El diagrama general se muestra en la Figura 1.

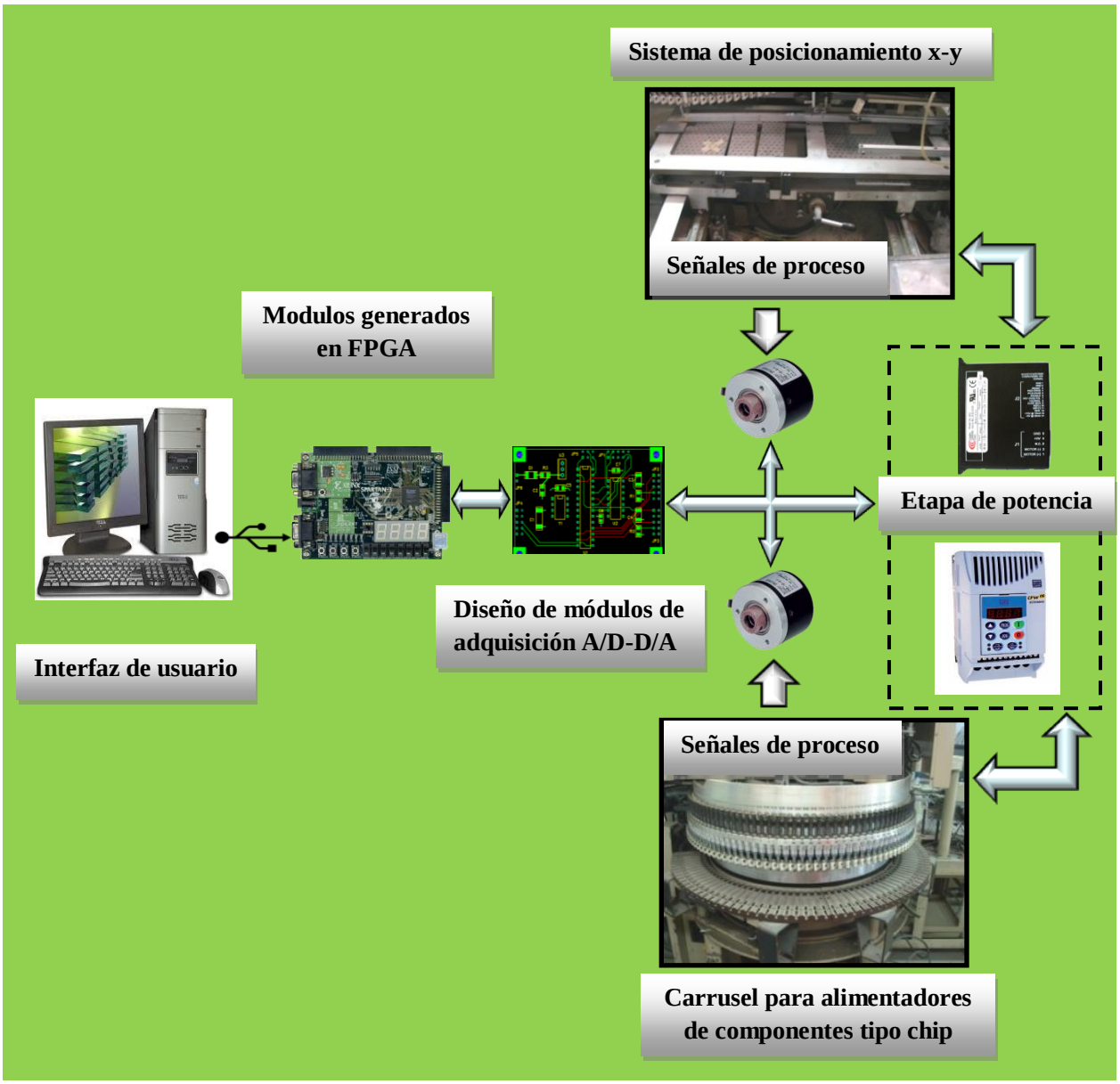


Figura 1.1. Diagrama a bloques de la máquina.

Capítulo II

Revisión de literatura

2.1 Estado del arte

La evolución del hombre y en particular de su tecnología se ha basado en la utilización de máquinas-herramienta, estas eran como la prolongación de las manos humanas, y le permitían realizar con mayor facilidad las actividades de su entorno.

En los primeros años de la electrónica, los componentes electrónicos tales como capacitores, resistores y conectores, fueron ensamblados manualmente. Pero con el tiempo, las máquinas-herramienta especializadas en el montaje de componentes electrónicos han ido tomando el trabajo de ensamblar estos componentes. El deseo de conectar componentes electrónicos en un modo fácil ha existido desde el surgimiento de la electrónica. La técnica con PCB (Tableta de Circuito Impreso) fue inventada por Paul Eisler alrededor de la segunda guerra mundial. Después de la segunda guerra mundial la técnica fue liberada para usos comerciales. En esta técnica, las pistas de cobre son impresas en una tableta no conductiva, donde las pistas conectan los diferentes componentes montados sobre la tableta. Hay diferentes tipos de tarjetas de circuito impreso en la cuales, los componentes electrónicos pueden ser ensamblados a través de tecnología TH (Thru-Hole, Agujeros Pasantes) o por SMT (Surface Mount Technology, Tecnología de Montaje Superficial) como se muestra en la figura 2.1. El ensamble manual de componentes sobre el PCB fue el primer acercamiento con la tecnología de montaje y aun se utiliza en casos especiales. Por tanto así como los componentes se han ido desarrollando, a la par de estos las máquinas-herramienta de montaje de componentes electrónicos han sido creadas y perfeccionadas, por lo cual se intuye que cuando se habla del desarrollo de alguna tecnología en componentes también se está hablando de alguna máquina-herramienta que lo pueda montar sobre la PCB.

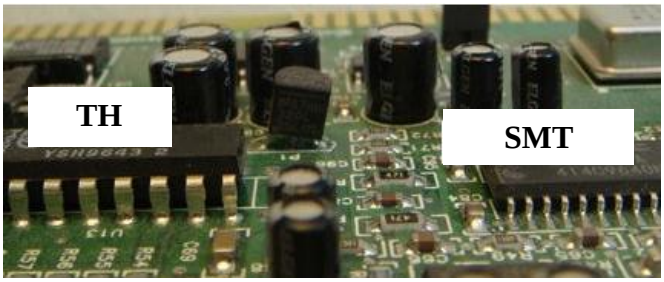


Figura 2.1 Componentes electrónicos montados con tecnologías TH y SMT.
Fuente: Woodgate, 2005.

SMT no es un concepto nuevo, este tiene sus raíces en viejas tecnologías llamadas FT (Flat Pack, Empaques Planos) e Hibrida. Los extensos trabajos hechos los Estados Unidos para la industria militar. Por ejemplo, la reducción en el tamaño del empaque para grandes cantidades de terminales. Mientras que la industria electrónica en los Estados Unidos estuvo preocupada por los sustratos de componentes para el mercado limitado militar, los Japoneses y Europeos estuvieron respondiendo a las demandas del mercado consumidor. En 1960 N.V Philips, una compañía Europea inventó dispositivos de montaje para la industria de relojes Suizos. Estos dispositivos son conocidos como circuitos integrados de contorno pequeño (SOIC). En 1970 los Japoneses comenzaron a construir calculadoras con paquetes cuadrados (quad packs), los cuales tienen terminales en los 4 lados. Los japoneses también perfeccionaron la versión actual de la tecnología de montaje superficial, sobre los altos volúmenes de los productos de consumo, tales como radios, televisiones y reproductores grabadores de esa época. Para la mayor parte, de estos productos se requieren dispositivos de montaje pasivo para sus circuitos analógicos. En Japón hacia 1980 fue caracterizado por el uso de dispositivos activos a nivel mundial en aplicaciones en masa para telecomunicaciones y computadoras. A la par en los Estados Unidos, la industria electrónica es manejada hacia los circuitos digitales para telecomunicaciones, computadoras y el mercado militar. En la actualidad estos países incluyendo la comunidad Europea se rigen por los estándares SMT de la Asociación de Industrias Electrónicas (EIA) y otras (Prasad, 1997).

De esta forma las máquinas-herramienta han llegado a ser más avanzadas, mucho más componentes pueden ser montados por la máquina con mayor precisión, velocidad y menos fallas. Las máquinas-herramienta de montaje pueden ser clasificadas en 5 categorías.

Las cinco clases son:1) máquinas de montaje dual, 2) máquinas multiestacion, 3) máquinas de montaje de estilo torreta, 4) máquinas de montaje de multicabeza, 5) máquinas secuenciales escoger y montar. Cada una de estas resuelve el problema de la tarea de ensamble en diferente forma.

Esta tesis se concentra en una máquina del tipo 5, en la figura 2.2 se muestra la bancada de un sistema secuencial escoger y montar (Wiklund, 2010).

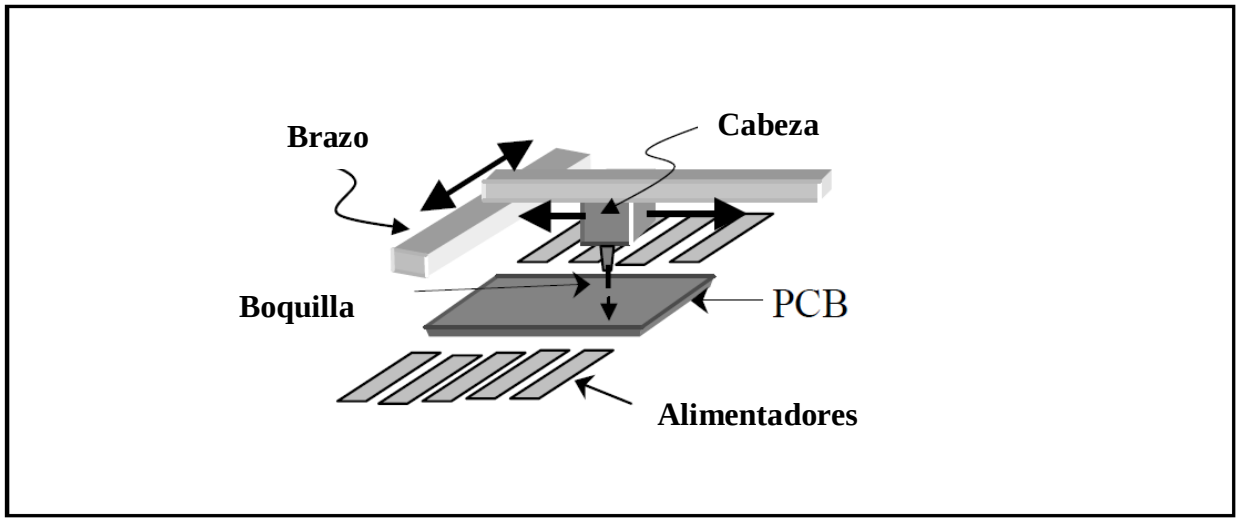


Figura 2.2 Bancada de un sistema secuencial escoger y montar.

2.2 Sistemas de control de movimiento

Los sistemas de control de movimiento programable (PMCS) se emplean en todos los sistemas mecánicos que implican movimiento controlado por computadora. Un manipulador robótico, una máquina de ensamble de componentes, una máquina CNC, una mesa XYZ y los sistemas de control de equipo de construcción son ejemplos de la aplicación de los sistemas de control de movimiento programable. Como su nombre lo indica los PMCS son mecanismos de movimiento donde el movimiento se controla con una computadora digital y de aquí que sean programables. Estos son buenos ejemplos de sistemas mecatrónicos, puesto que implican un sistema de movimiento mecánico, varios actuadores, sensores, y control por computadora.

Cada sistema de control de movimiento integrado principalmente por los componentes siguientes (Fig. 2.3).

1. Controlador.
2. Actuadores (motor, amplificador, fuente de potencia).
3. Sensores (codificadores, tacómetros, sensores de voltaje, etcétera).
4. Mecanismos de transmisión de movimiento (engranes, tornillos de avance).
5. Dispositivos de interfaz del operador (también denominados interfaz hombre-máquina).

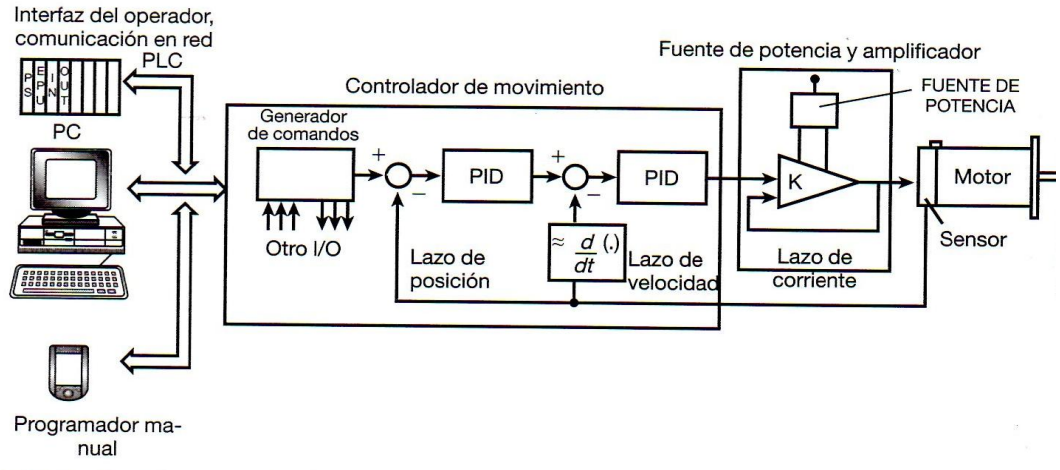


Figura 2.3. Sistema de control de movimiento de un eje servocontrolado. (Cetinkunt, 2007).

2.3 Controladores

Un controlador automático compara el valor real de la señal de una planta con la entrada de referencia (el valor deseado), determina la desviación y produce una señal de control que reduce la desviación a cero o a un valor pequeño. La manera en la cual el controlador automático produce la señal de control se denomina acción de control. Un sistema que mantiene la relación entre la salida y la entrada de referencia, comparándolas y usando la diferencia como medio de control se denomina sistema en lazo cerrado. Los sistemas en los cuales la salida no tiene efecto sobre la acción correctora se denominan sistemas en lazo abierto. El controlador es el elemento

encargado de procesar la señal de error y generar una señal encargada de disminuir el valor de dicho error con el objetivo de lograr la máxima precisión del sistema de control, en la figura 2.4 se muestra un diagrama de un sistema de control industrial de lazo cerrado.

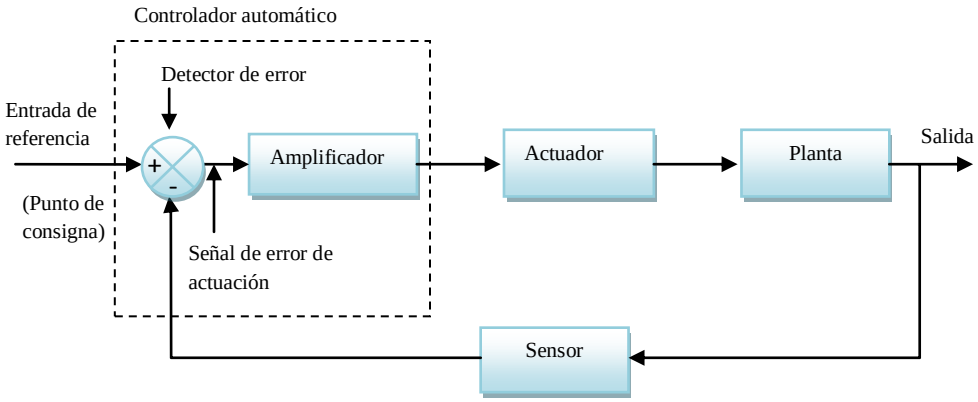


Figura 2.4. Diagrama de bloques de un sistema de control industrial.

Los controladores clásicos se clasifican, de acuerdo con sus acciones de control, como:

1. **De dos posiciones o controladores on-off**

   En un sistema de control de 2 posiciones, el elemento de actuación solo tiene 2 posiciones fijas, que en muchos casos, son simplemente encendido y apagado. En el control de 2 posiciones, la señal u(t) permanece en un valor ya sea máximo o mínimo, dependiendo de si la señal de error es positiva o negativa. De este modo

   $$u(t) = U1 \quad para\ e(t) > 0 \tag{2.1}$$
   $$u(t) = U2 \quad para\ e(t) < 0 \tag{2.2}$$

2. **Controladores proporcionales**

   La relación entre la salida del controlador *u(t)* y la señal de error *e(t)* es

   $$u(t) = Kpe(t) \tag{2.3}$$

   donde *Kp* se considera la ganancia proporcional, el control proporcional es, en esencia, un amplificador con ganancia ajustable.

3. **Controladores integrales**

   En un controlador con acción de control integral, el valor de la salida del controlador *u(t)* se cambia a una razón proporcional a la señal de error *e(t)*. Es decir,

   donde *Ki* es una ganancia ajustable.

$$u(t) = Ki \int_0^t e(t)dt \qquad (2.4)$$

4. **Controladores proporcionales-integrales**

   La acción de control de un controlador proporcional-integral (PI) se define mediante

$$u(t) = Kpe(t) + \frac{Kp}{Ti} \int_0^t e(t)dt \qquad (2.5)$$

   donde *Ti* se denomina tiempo integral.

5. **Controladores proporcionales-derivativos**

   La acción de control de un controlador proporcional-derivativa (PD) se define mediante

$$u(t) = Kp\, e(t) + KpTd \frac{de(t)}{dt} \qquad (2.6)$$

   donde *Td* es el tiempo derivativo.

6. **Controladores proporcionales-integrales-derivativos**

   La combinación de la acción de control proporcional, la acción de control integral y la acción de control derivativa se, se denomina acción de control proporcional-integral-derivativa. Esta acción combinada tiene las ventajas de las 3 acciones de control individuales. La ecuación de un controlador con esta acción combinada está dada por:

$$u(t) = Kp\, e(t) + \frac{Kp}{Ti} \int_0^t e(t)dt + KpTd \frac{de(t)}{dt} \qquad (2.7)$$

   Donde *Kp*, es la ganancia proporcional, *Ti* es el tiempo integral y *Td* es el tiempo derivativo (Ogata, 2008).

2.3.1 Control difuso

En los últimos años los sistemas difusos (SDs) se han venido consolidando como una herramienta útil para tratar y modelar sistemas complejos y no lineales. Especialmente en áreas como el control, el procesamiento de imágenes, la robótica y la electrónica de consumo, los SDs han sido incorporados en un gran número de productos y procesos.

El control difuso incorpora el conocimiento experto dentro de su estructura. Lo que permite tener una herramienta muy valiosa, en el control de procesos, y sistemas, que presentan dificultades, para ser controlados, debido a su complejidad incluso en aquellos sistemas en donde el modelo matemático incluso no existe.

El control difuso o control borroso se inspira en el tipo de control que ejerce un operador humano experto para controlar procesos en los que otros tipos de control fallarían.

La lógica difusa fue diseñada para representar y razonar sobre una forma particular de conocimiento, expresado de forma verbal. Lo difuso de una propiedad subyace en la ausencia de límites bien definidos del conjunto de objetos a los cuales se aplica dicha propiedad. El pensamiento humano generalmente toma decisiones evaluando el grado de verdad de una variable, esto es debido a que no vemos variables discretas, es decir, aquellas variables cuyo valor es si o no. Este tipo de razonamiento es difícil de reproducir mediante métodos normales, en las que las variables tienen que pertenecer completamente a algún grupo y por lo tanto no pertenecer a otro. La lógica difusa es una manera de resolver este problema.

En la lógica difusa una aseveración puede ser más o menos cierta, mientras que en la lógica clásica, una aseveración o es cierta o es falsa (nada intermedio). La lógica difusa extiende la lógica clásica de tal manera que permite valores intermedios entre cero y uno (Passino, 1998).

2.3.1.1 Conjunto difuso

Los conjuntos difusos se distinguen de los conjuntos clásicos, en que un objeto puede pertenecer parcialmente a un conjunto o hasta puede pertenecer parcialmente a varios conjuntos.

En cuánto pertenece o no un objeto a un conjunto difuso, se llama su grado de pertenencia o membrecía, y este valor está definido por la función de pertenencia o membrecía del conjunto, que por lo general se representa mediante la letra griega μ.

Este grado de pertenencia se define mediante la función característica asociada al conjunto difuso, para cada valor que pueda tomar un elemento o variable de entrada x la función característica (x) A μ proporciona el grado de pertenencia de este valor de x al conjunto difuso A (Zadeh, 1973).

Además se puede decir que el conjunto A es matemáticamente equivalente a su función de pertenencia o característica μ $A(x)$, ya que conocer μ $A(x)$ es lo mismo que conocer A. Un conjunto difuso en el universo U se caracteriza por una función de pertenencia μ $A(x)$ que toma valores en el intervalo [0,1] y puede representarse como un conjunto de pares ordenados de un elemento x y su valor de pertenencia al conjunto:

$$A = \{(x, \mu A(x))\ x \in U\} \tag{2.8}$$

2.3.1.2 Universo

Los miembros de un conjunto difuso son tomados de un universo. El universo contiene todos los objetos que puedan caer dentro de la consideración de comparar al conjunto U con la definición de conjunto difuso. Por lo tanto el universo es el rango de todos los posibles valores aplicables a una variable de un sistema (Dobois, 1980).

2.3.1.3 Dominio

El rango de valores sobre el cual se ubica el ancho de una función de membrecía se conoce como dominio

2.3.1.4 Funciones de pertenencia

El grado de pertenencia en los conjuntos difusos puede ser representado por medio de una función matemática la cual indica el grado de pertenencia que presenta un elemento dentro de un conjunto difuso.

En otras palabras, si A es un conjunto difuso, entonces la función de pertenencia μ A(x) mide el grado con el cual el elemento *x* pertenece al conjunto A (Witold, 1996). Las funciones de pertenencia se pueden clasificar según su forma, las formas de funciones más comunes y usadas debido a su simplicidad y fácil manejo son las formas: trapezoidal, triangular, campana de gauss, gamma, etc. Como se muestra en la figura 2.5.

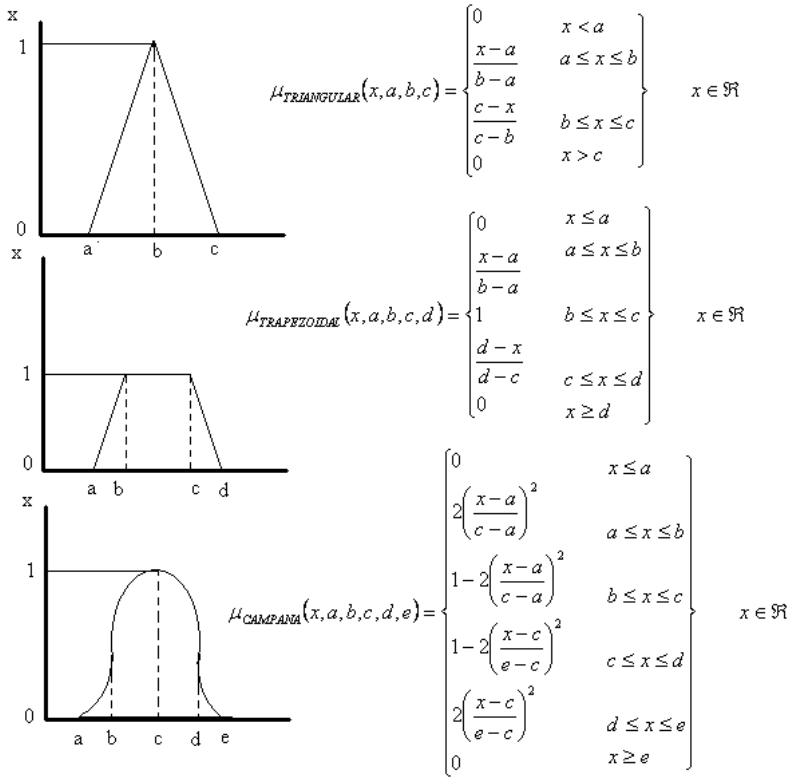


Figura 2.5. Funciones de pertenencia comúnmente utilizas

2.3.1.5 Controlador lógico difuso

La finalidad de un controlador difuso es la de modificar el comportamiento de la planta mediante el cambio de una o varias entradas del sistema, de acuerdo a un conjunto de reglas y un proceso de inferencia que permita obtener las salidas deseadas.

Un controlador lógico difuso, emplea principios de lógica difusa y permite convertir estrategias de control lingüístico, basado en conocimiento experto, en una estrategia de control automático (Hopgood, 2001). Los sistemas de control difuso se basan en reglas difusas que representan un mecanismo de decisión de control, para ajustar los cambios no deseados provenientes de la planta. Normalmente, los sistemas de control difuso (figura 2.6) sustituyen las habilidades de un operador humano por un sistema basado en reglas difusas. Un sistema difuso reemplaza las ecuaciones diferenciales del modelo matemático, por un modelo construido en base de un número de reglas.

Las partes principales de un sistema de control difuso son:

- Base de reglas
- Máquina de inferencia
- Fusificador
- Defusificador

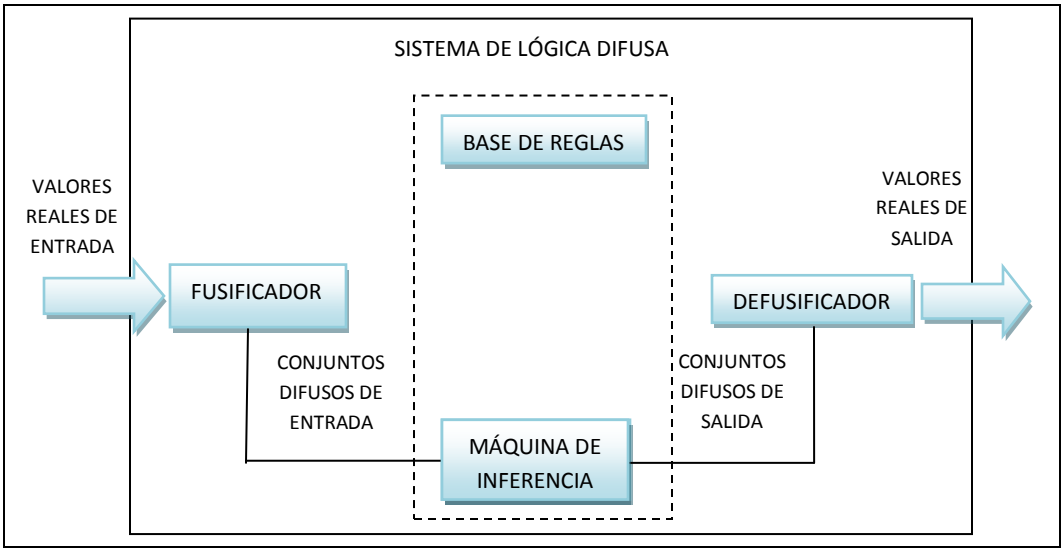


Figura 2.6. Partes de un sistema de lógica difusa.

(Lee, 1990)

Base de reglas: contiene el conocimiento experto resumido en un conjunto de reglas del tipo if-then (si-entonces).

Máquina de inferencias: simula el proceso de la toma de decisiones del experto, teniendo en cuenta la base de reglas y el conocimiento difuso del proceso.

Fusificador: es una interfaz que transforma la información de entrada al controlador en información lingüística que puede ser interpretada por la base de reglas y la máquina de inferencias.

Defusificador: esta interfaz convierte las conclusiones de la máquina de inferencias en acciones de control (Altrock, 1995).

2.4 Identificación de procesos

El término identificación de procesos adoptado en la década de los sesenta en el ámbito de los especialistas en Ingeniería de Sistemas y Automática, podría ser definido como el conjunto de estudios, teorías y algoritmos que permiten obtener la estructura y los parámetros de un modelo matemático (generalmente dinámico) que produce, con suficiente exactitud para los fines de control automático, las variables de salida del proceso o sistema real objeto de estudio ante el mismo conjunto de variables de entrada.

El concepto de identificación de procesos, se pueden abordar desde distintos enfoques o métodos (Aguado, 2000).

1. **Identificación analítica o de primeros principios.** Consiste en desarrollar un modelo basado en las relaciones físico-químicas del proceso a identificar, planteando ecuaciones cinemáticas, dinámicas, de balance de masa, de energía, de cinética química, etc. Este enfoque conduce generalmente a modelos complejos y no lineales que deben ser sometidos a un proceso de simplificación y línealización. El inconveniente principal de este enfoque consiste en que se requiere un conocimiento muy especializado sobre la tecnología del proceso, no siempre disponible.

2. **Identificación experimental mediante señales especiales.** Este enfoque, que algunos autores denominan como identificación clásica, resulta generalmente el más directo y el que permite obtener el modelo de un proceso a más corto plazo. Las señales utilizadas con más frecuencia son los escalones y las llamadas secuencias binarias pseudoaliatorias. Una limitación muy importante de estos métodos es que, como resultado del proceso de identificación, se obtienen exclusivamente modelos determinísticos, sin considerar los posibles modelos de perturbación.

3. **Identificación paramétrica.** El método más común de identificación paramétrica está basado en los denominados métodos de minimización del error de predicción, derivados de la tradicional teoría de mínimos cuadrados. La adaptación de esta teoría a la identificación de procesos se basa en aceptar como cierto de que el proceso puede ser representado por un modelo de estructura fija, generalmente una ecuación lineal en diferencias, lo que implica que dicho modelo tiene naturaleza discreta. La aplicación de este algoritmo puede hacerse fuera de línea o no recursiva, mediante el uso de toda la información disponible de entrada y salida del proceso para determinar un modelo invariante en el tiempo. También puede hacerse de manera recursiva, de modo que partiendo de una estimación inicial, generalmente arbitraria, de los parámetros del modelo se va actualizando y mejorando con cada nueva información de entrada y salida obtenida, lo que permite obtener una estimación de parámetros variables en el tiempo.

2.5 Sistemas de conversión D/A y A/D

Un convertidor D/A convierte un número digital en una señal analógica, es decir, por lo general un nivel de voltaje. Un convertidor A/D hace lo opuesto. Convierte una señal analógica en un número digital. Los convertidores D/A y A/D son componentes esenciales al interrelacionar el mundo digital con el mundo analógico.

Los procesos de conversión de señales de forma analógica en forma digital implican dos operaciones: Muestrear la señal y cuantificar la señal.

Muestreo es el proceso en el cual un número finito de muestras de una señal continua se toma y se convierte en una secuencia de números discretos. Las muestras son la única información disponible respecto a señal en la computadora.

La señal analógica pasa por un filtro de anti alisado, después la señal se muestrea y se mantiene en el nivel muestreado momento en el cual la señal es cuantificada de manera digital.

El convertidor D/A convierte un número digital en señales analógicas. A esto también se le refiere como la etapa de reconstrucción de la señal (Cetinkunt, 2007).

2.6 Lógica programable

Una de las herramientas más poderosas con que se cuenta actualmente para el diseño de sistemas digitales es el uso de lenguajes descriptivos de circuitos que permiten jerarquizar las diferentes etapas de diseño para lograr una repartición óptima del trabajo. Poniendo como ejemplo, el proceso de diseño de un microprocesador, tradicionalmente primero se daban las especificaciones del microprocesador, mas tarde se definía la arquitectura, en un tercer paso se definían los bloques funcionales a nivel de compuerta y finalmente se realizaba el diseño a nivel transistor. Cada una de las etapas del diseño estaba condicionada con la terminación de la etapa anterior y el tiempo del proceso crecía con la complejidad del sistema.

Con los lenguajes descriptivos, se puede diseñar y simular el funcionamiento de cada uno de los niveles de abstracción, tratando a los niveles como cajas negras y por lo tanto se pueden definir las características del microprocesador, su arquitectura, su estructura, función y realización a nivel transistor de manera simultánea, disminuyendo los tiempos de diseño.

Esto es posible gracias al uso de elementos repetitivos en cada una de las partes y sobre todo, a la existencia de lenguajes que permiten el modelado y simulación de sistemas digitales en cada uno de los niveles de abstracción. Solamente se necesitan dar las especificaciones en cada nivel para que a la hora de conjuntar el diseño, se cumplan las expectativas originales. Aun sigue siendo un problema complejo, pero ya no tanto con la ayuda de estas herramientas. A pesar de que existen diversos lenguajes descriptivos, el VHDL (Very High-speed Integrated Circuit

Hardware Description Language) es el más popular y su uso es cada vez mayor, dejando relegados los lenguajes a un segundo plano. Su popularidad se debe principalmente a que es un estándar del IEEE, particularmente el estándar 1164 (Romero, 2004).

2.7 Instrumentación y sensores

La medición de variables físicas es necesaria para fines de monitoreo y control. Las variables comunes que se pueden medir en un sistema de adquisición de datos y control son:

1. Posición, velocidad, aceleración.
2. Fuerza, par de torsión, deformación, presión.
3. Temperatura.
4. Gasto.
5. Humedad.

El dispositivo de medición se denomina sensor. Este se coloca en el entorno donde se debe medir una variable. El sensor está expuesto a la variable medida. Hay tres fenómenos básicos en efecto en cualquier operación del sensor.

I. El cambio (o el valor absoluto) en la variable física medida (es decir, presión, temperatura, desplazamiento) se traslada a una carga en la propiedad (resistencia, capacitancia, acoplamiento magnético) del sensor. A ésto se le denomina transducción, el cambio de la variable medida se convierte en un cambio equivalente de la propiedad del sensor.

II. El cambio en la propiedad del sensor se traduce en una señal eléctrica de nivel de baja potencia en la forma de voltaje o corriente.

III. Esta señal del sensor de baja potencia se amplifica, se acondiciona y se transmite a un dispositivo inteligente para su procesamiento, es decir, se representa de manera visual o se emplea en un algoritmo de control de lazo cerrado.

Los tipos de sensores varían en la etapa de transducción a medir una variable física. En respuesta a la variable física un sensor se puede diseñar para cambiar sus resistencias, capacitancia, inductancia, corriente inducida o voltaje inducido (Cetinkunt, 2007).

Capítulo III

Metodología

El presente capítulo muestra el desarrollo realizado para la reconversión de la máquina de montaje de componentes electrónicos chip RX-4A. El proyecto se desarrollo en diferentes etapas las cuales se muestran en la figura 3.1. El sistema principal de la máquina es el de posicionamiento x-y, el cual utilizaba servomotores trifásicos de la marca y serie OKUMA BL-S10E-30S respectivamente, el control de éstos se realizaba mediante varias tarjetas electrónicas con tecnología obsoleta y que además presentaban daño en su operatividad, por lo cual se remplazaron por servomotores de corriente directa con sus respectivos encoders. El acoplamiento mecánico se realizó directamente sobre la mesa de posicionamiento o bancada x-y, para esto solo se construyeron bases de sujeción, a la medida de los servomotores para alinear el eje del tornillo de avance del eje x, y el sistema banda polea del eje y. Una vez realizada la instalación y acondicionamiento del sistema de posicionamiento, se llevó a cabo el proceso de identificación y sintonización para cada uno de los controladores propios, utilizando como elemento de medición el encoder, y además haciendo uso de la interfaz USB para el proceso anteriormente mencionado. Finalmente se tomaron como referencia los datos generados por algún programa de diseño a partir del formato de datos estándar extendido RS-274X regulado por la EIA.

Paralelamente al proceso anterior se realizó la instrumentación y control de la mesa rotatoria o carrusel en el cual se deben cargar los alimentadores con bobinas o carrilleras, estos deben contener los componentes electrónicos tipo SMD que serán colocados sobre la tarjeta PCB, como el caso anterior las tarjetas de control son obsoletas y están dañadas, sin embargo, para realizar el control de este proceso, se preserva el servomotor trifásico de corriente alterna y se utilizan las técnicas de control difuso basadas en FPGA, para su posicionamiento y selección.

Para complementar el proceso anterior se implementa un controlador de rotación de componentes en 8 direcciones, el cual utiliza como actuador final un motor a pasos. Finalmente

se realizó la integración y pruebas de los eventos que intervienen en el proceso de selección y montaje de componentes electrónicos tipo chip.

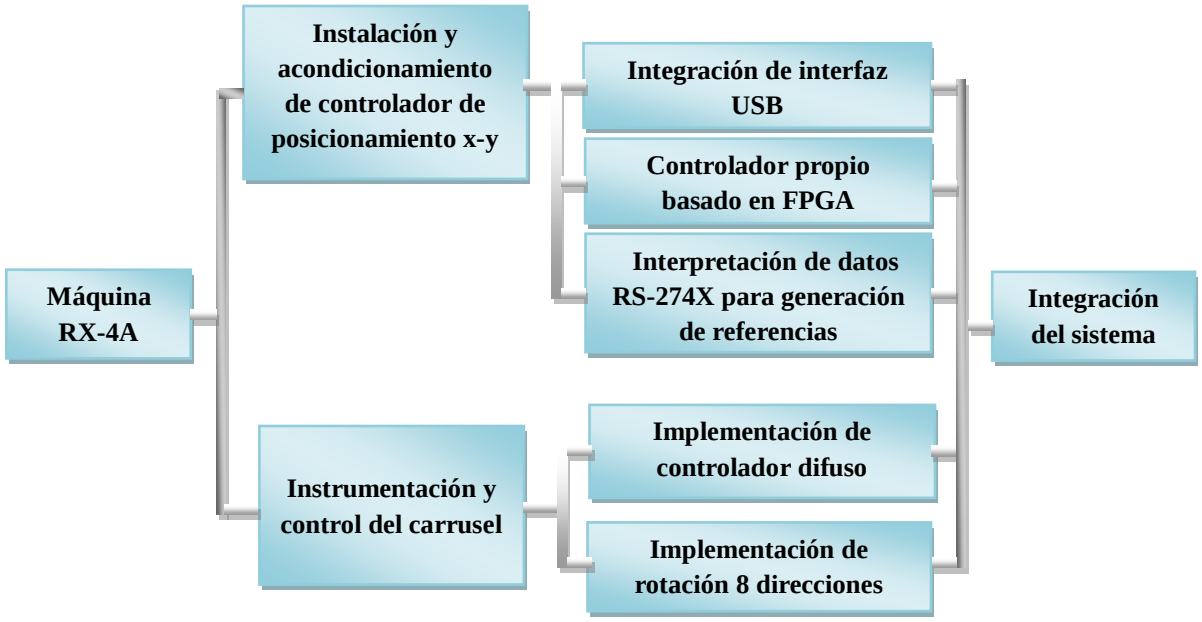


Figura 3.1. Metodología a seguir para el desarrollo del proyecto

3.1 Instalación y acondicionamiento del controlador de posicionamiento x-y

Para el control de posicionamiento x-y de la máquina es necesaria la selección de los servomotores de corriente directa, para lo cual se utilizaron como referencia las características técnicas de los servomotores trifásicos, sobre todo en el parámetro de torque de salida 10 Kg-cm, los servomotores seleccionados fueron de la serie SSC 23SMDC-LCSS cuya compañía es Copley Control, que cuentan con un reductor planetario 5:1 y un encoder incremental de 1000 PPR (Pulsos por Revolución). Adicionalmente es necesario construir tarjetas de optoacoplamiento eléctricas de activación para los interruptores de límite de la bancada y las señales de habilitación tanto de los servoamplificadores como del controlador. Para el control de los servomotores es necesario el uso de servoamplificadores los cuales acondicionan la señal de control a magnitudes de potencia adecuadas para la manipulación de los actuadores (servomotores). En la selección de éstos se tomaron en cuenta los siguientes requerimientos: la corriente continua máxima, la cual debe ser mayor a la requerida por los servomotores, y el voltaje de alimentación de los motores debe estar dentro del rango que maneja el amplificador, en la figura 3.2 se muestra el servoamplificador Copley Control Modelo 403 seleccionado y sus características para su funcionamiento. Cada servoamplificador tiene 3 señales de habilitación:

ENABLE, POS ENABLE y NEG ENABLE, estas señales controlan la habilitación del servomotor en cada sentido de giro mediante los interruptores de límite.

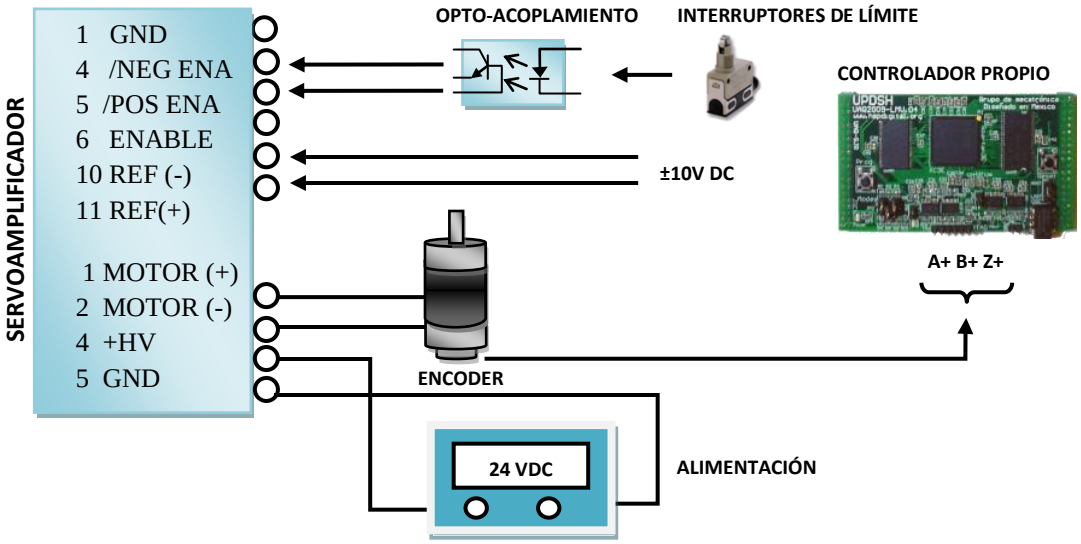


Figura 3.2. Interconexión eléctrica para el funcionamiento de los ejes x-y

Los elementos que conforman el sistema de posicionamiento x-y, o bancada de la máquina se muestran se muestran en la figura 3.3, se pueden observar los 2 nuevos servomotores de corriente directa acoplados e instrumentados a los 2 ejes de movimiento.

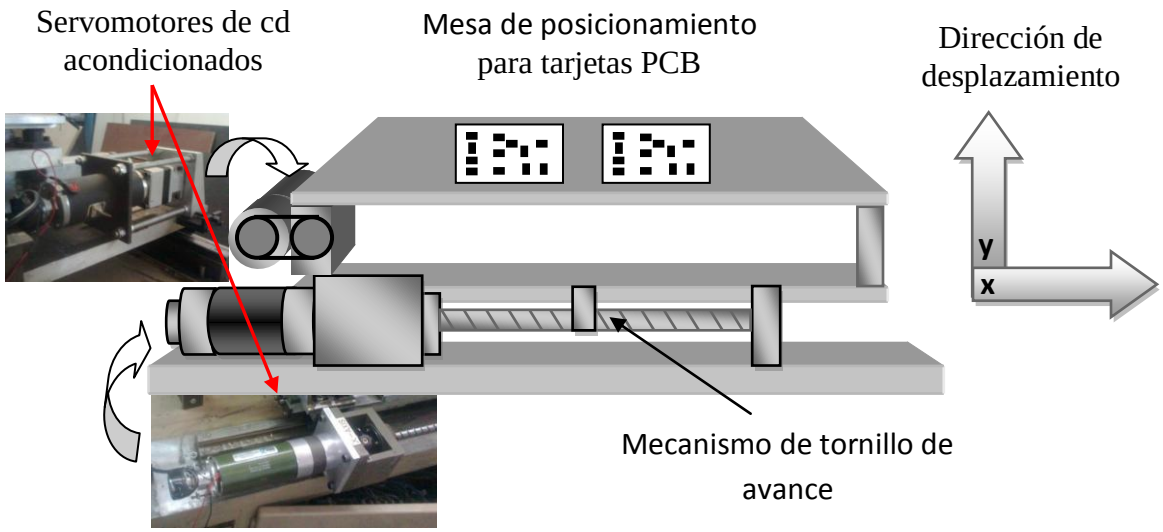


Figura 3.3 Sistema de posicionamiento x-y acondicionado.

Para realizar el cálculo de la resolución de cada uno de los ejes, se toma en cuenta la relación de los reductores y los sensores (encoders) utilizados, para determinar este parámetro se hace uso de la fórmula 3.1, que rige al mecanismo de conversión de movimiento de uso común y su relación de entrada-salida.

$$Resolución = \frac{Movimiento\ lineal\ del\ tornillo\ guia\ por\ revolución}{Cuentas\ de\ encoder\ por\ revolución\ x\ relación\ del\ del\ reductor} \quad (3.1)$$

Debido al juego mecánico la fricción estática, el desgaste mecánico, entre otros, la resolución puede variar.

De acuerdo a la especificación contenida en el manual de operación de la máquina AVIMOUNT, el desplazamiento de la bancada podía ser realizado, en avances de 0.2 mm y 0.5 mm, usualmente esta máquina para efectos de producción masiva en línea, estaba configurada en avances de 0.5 mm.

3.1.1 Integración de interfaz USB

Este bloque implementa el protocolo de comunicación USB de acuerdo a Morales et al. (2009) y es el medio de comunicación entre el controlador y la interfaz gráfica de usuario, sirve para enviar por medio de la interfaz de usuario las constantes requeridas por los bloques del controlador y enviar del controlador la señal de encoder y de referencia a la interfaz para lograr el movimiento en la bancada x-y. Para esta aplicación la interfaz USB integrada cuenta con un *endpoint* de salida de 32 bytes y un *endpoint* de entrada de 16 bytes. Los 32 bytes de salida configuran el controlador PID. Los 16 bytes de entrada reciben los datos de posición deseada, posición real, error, señal de control, y lectura del monitor de corriente.

3.1.2 Controlador propio basado en FPGA

El diagrama a bloques de la Figura 3.4 muestra de forma general la arquitectura interna del controlador propio. Consiste en un módulo USB a través del cual se envían datos en forma bidireccional hacia el sistema como son: las constantes de la ecuación en diferencias mostrada en la ecuación (3.2), los cuales se obtienen mediante previa sintonización. A su vez en respuesta el

controlador envía una señal analógica de ± 10 Volts, mediante un DAC hacia el servoamplificador para controlar el servomotor tanto en posición como en velocidad.

$$y(k) = a_0x(k) + a_1x(k-1) + a_2x(k-2) + b_1(k-1) + b_2y(k-2) \quad (3.2)$$

$$a_{0=}k_p + k_i + k_d \quad (3.2a)$$

$$a_{1=} - (k_p + 2k_d) \quad (3.2b)$$

$$a_{2=}k_d \quad (3.2c)$$

$$b_{1=}1 \quad (3.2d)$$

$$b_{0=}0 \quad (3.2e)$$

Una interfaz desarrollada en lenguaje C++ y GTK+ (librería gráfica), permite el manejo del sistema de control, en donde el usuario puede cargar las coordenadas de posicionamiento punto a punto o un archivo conteniendo las coordenadas de las posiciones centrales donde el componente electrónico debe ser colocado. El módulo permite monitorear la posición instantánea de la bancada, mediante la lectura del encoder y su conversión a milímetros en ambos ejes.

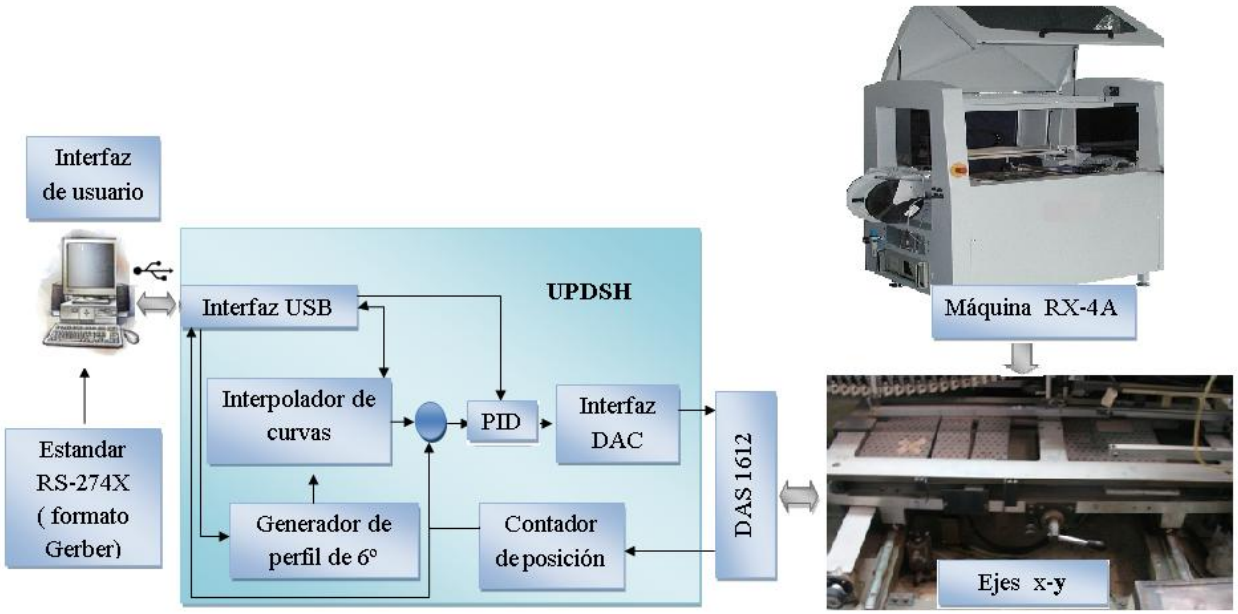


Figura 3.4 Arquitectura interna del controlador propio basado en FPGA.

3.1.3 Interpretación de datos RS-274X para generación de referencias

Usualmente las máquinas de montaje de componentes parten de un código generado por algún software de diseño de PCB, es decir, los datos más relevantes son presentados como listas de componentes con sus respectivos atributos, estos datos son enviados hacia el controlador bajo el estándar RS-274X, la figura 3.5, muestra una pantalla conteniendo una lista usual de componentes a ser montados sobre una PCB (1-7), en la cual se observan las características más significativas y necesarias para poder generar la secuencia de colocación y rotación de componentes sobre el sistema de posicionamiento x-y.

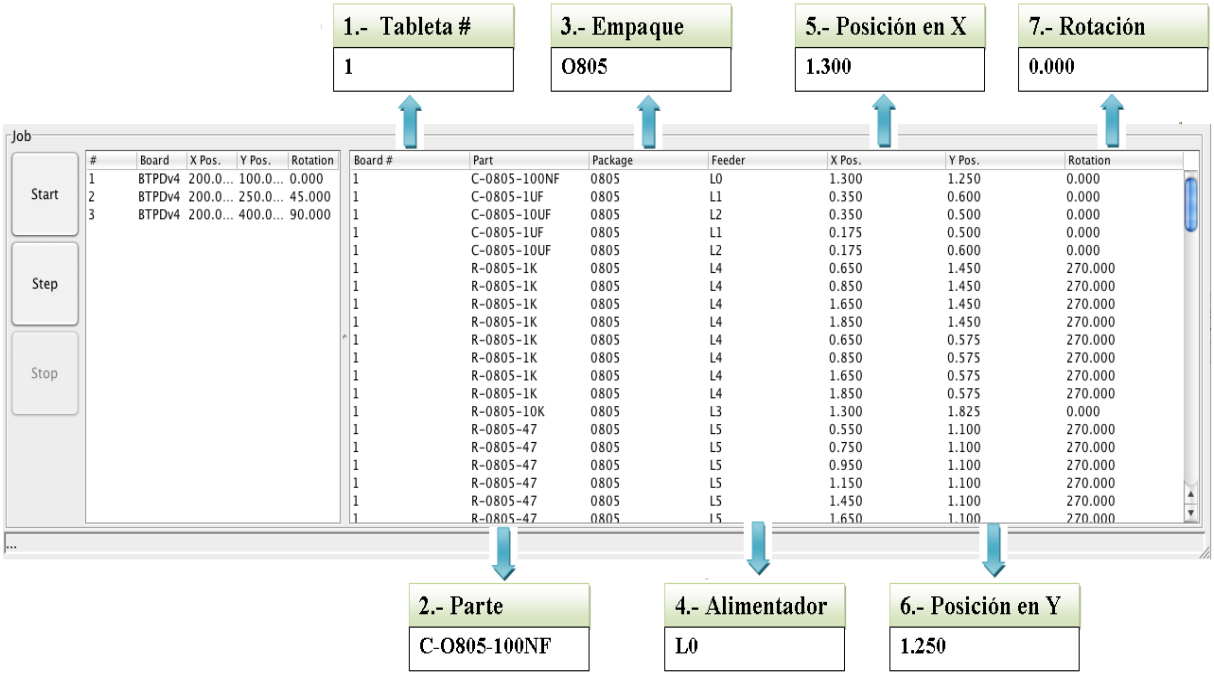


Figura 3.5 Lista de componentes a ensamblar solo tipo chip

De la figura anterior los parámetros que servirán para generar un archivo de referencias de posicionamiento son los etiquetados como (5,6 y 7) y que se cargaran mediante la interfaz de usuario que permite el control del este evento.

3.2 Instrumentación y acondicionamiento de control de carrusel

Para llevar a cabo la selección del componente deseado, el carrusel cuenta con una base de 60 porta alimentadores enumerados de forma consecutiva, los cuales pueden ser cargados con componentes electrónicos tipo chip de la serie CR 2012 (2 mm de largo, 1.2 mm de ancho), CR-3819 (3.8 mm de largo, 1.9 mm de ancho), SOT23 (similar a 2012), tamaños estándar de acuerdo a las normas internacionales EIA.

La selección de cualquiera de ellos se logra mediante 120 boquillas, de las cuales 60 toman componentes de manera serial o secuencial, mediante succión a través de las boquillas, y las otras 60 se encuentran cargadas con componentes, listas para ser utilizadas por el aplicador. Para poder colocar alguno de estos componentes sobre la tableta PCB, que se encuentre localizada sobre la bancada del sistema de posicionamiento, es necesario posicionar el carrusel en el alimentador correcto, mediante giros o pasos de 3°, el sistema encargado de realizar esta función es el sistema mecánico de posicionamiento rotativo polea-banda y servomotor trifásico. La figura 3.6 muestra el sistema rotativo carrusel.

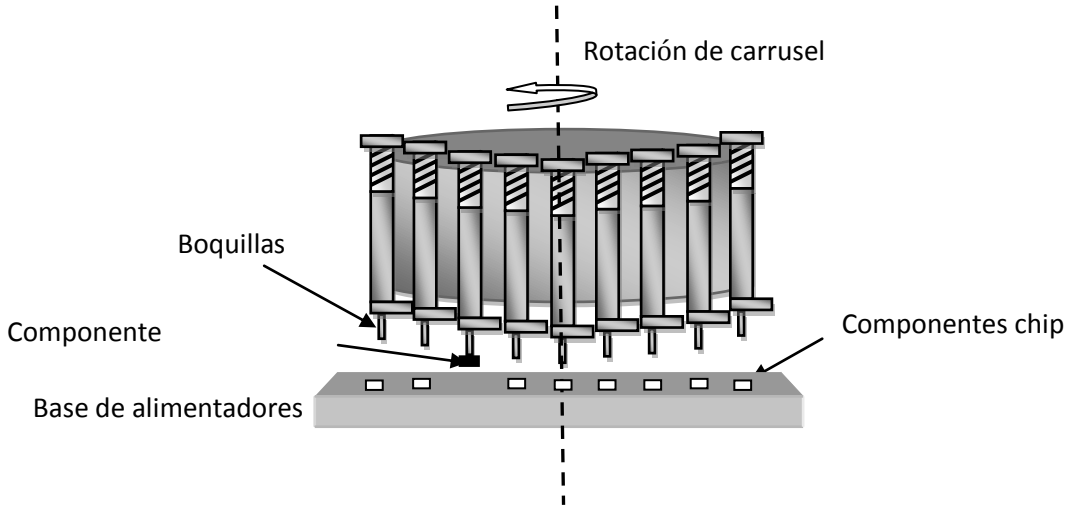


Figura 3.6. Carrusel de selección de componentes tipo chip

Este sistema de selección de componentes también fué instrumentado y acondicionado en su parte eléctrica ya que como su análogo, el sistema de posicionamiento contaba con tarjetas

electrónicas y encoder obsoleto e inservible. A diferencia del sistema de posicionamiento x-y, en el sistema de rotación se conservó el servomotor trifásico y su acoplamiento banda polea con el carrusel. Para lograr su instrumentación se tomaron en cuenta las características técnicas del servomotor BL-50E-20T marca OKUMA, y los parámetros principales a tomar en cuenta es el torque de salida 49 Kg-cm y la corriente máxima 5 Amperios. Además un encoder de 1500 PPR fue acoplado al servomotor para poder monitorear su posición. Para lograr el movimiento de este servomotor se utilizó un inversor marca WEG serie 08, con la característica en potencia suficiente para mover el servomotor y así el carrusel. En la figura 3.5 se muestra la interconexión para el funcionamiento del sistema carrusel.

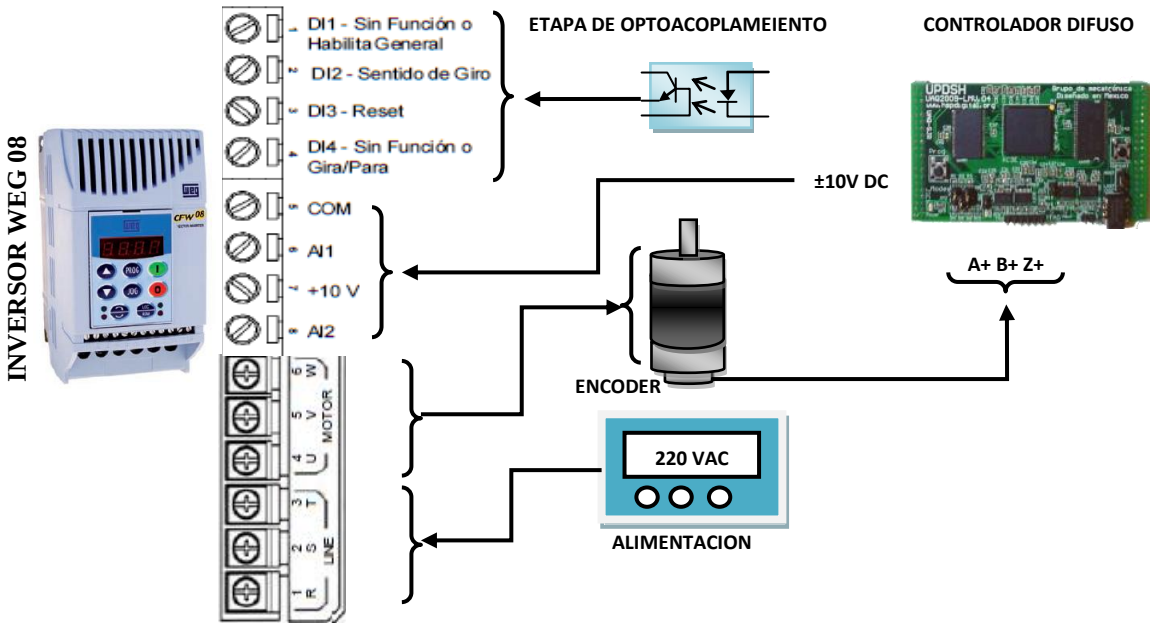


Figura 3.7. Interconexión eléctrica para el funcionamiento del carrusel

Para el control de giro del carrusel se desarrolló un algoritmo basado en lógica difusa mediante el desarrollo de IP cores en FPGA, esta metodología a diferencia del control de posicionamiento x-y que implican una identificación y sintonización, la utilización del

servomotor trifásico y su control solo tomó en cuenta el conocimiento heurístico del proceso para obtener el posicionamiento y velocidad del carrusel deseadas.

Sin embargo, una restricción inherente en cuanto a la velocidad de rotación del carrusel se debió a las características mecánicas de éste, debido al momento de inercia que presentan el tren de engranes y la carga, además de la velocidad de respuesta del inversor WEG-08, y restringen por ello la velocidad de giro del carrusel. No así la gran velocidad del control difuso basado en FPGA.

3.2.1 Implementación de controlador difuso

La lógica difusa fue diseñada para representar y razonar sobre una forma particular, es decir, como lo hace el ser humano, por ello para implementar el control de posición y velocidad sobre el carrusel no fue necesario hacer uso de alguna fórmula, expresión matemática que describiera el modelo del servomotor trifásico o algún método de identificación de la planta . A continuación se muestra en la figura 3.6 el diagrama de un controlador difuso en lazo cerrado aplicado al sistema carrusel, como se puede observar la posición deseada, es el número de alimentador cargado en la base de alimentadores, que contiene el componente que se pretende colocar, las entrada al control difuso son el error de posición y el error de velocidad.

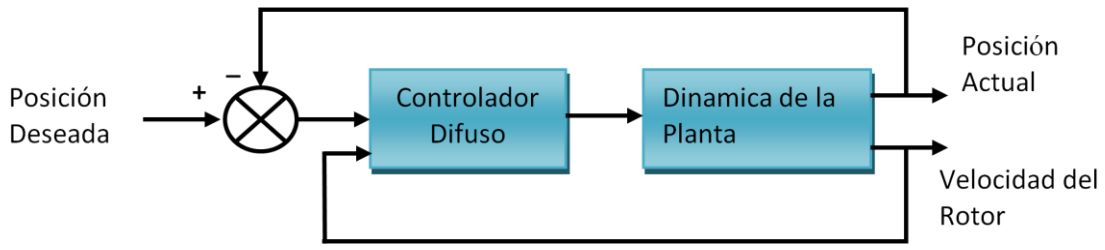


Figura 3.8. Sistema de control difuso en lazo cerrado.

Para la implementación del controlador difuso se utilizó la metodología Mamdani que se muestra en la figura 3.7.

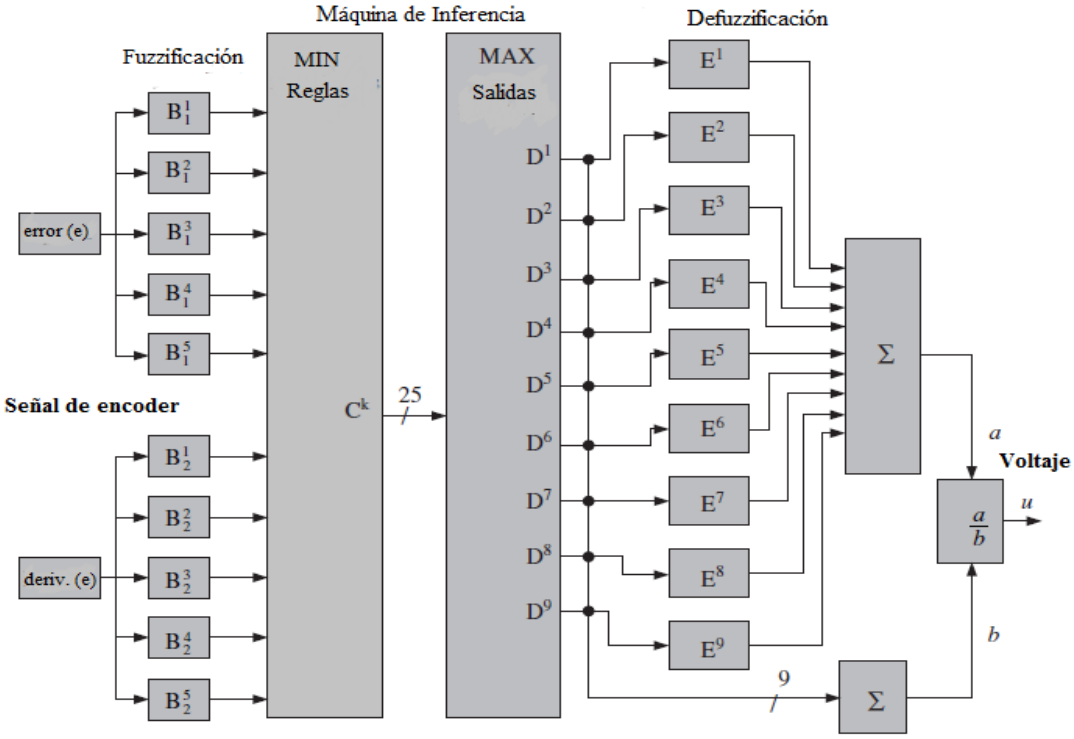


Figura 3.9 Diagrama a bloques de controlador difuso

Como se puede observar la posición deseada o punto de consigna y la posición actual son utilizadas para determinar por simple resta el error (e) figura 3.9 y es la primera señal de entrada al controlador. Esta señal es derivada para complementar la segunda entrada del sistema, para el cálculo de la primera etapa o fuzzificacion se toma en cuenta el mismo universo de discurso para ambas variables de entrada y reducir los cálculos, este proceso toma 2 ciclos de reloj ya que en el primer ciclo se procesa la primer variable y se guarda el resultado, el procesamiento de la segunda variable se completa en el segundo ciclo y se guarda también el resultado. Dentro del proceso de la máquina de inferencias, cada variable procesada por el Fusificador solo puede tomar 2 valores de pertenencia a la vez así que, en total 4 valores son procesados primero para obtener los valores mínimos, los cuales para reducir el tiempo de procesamiento se enfocan a las reglas relevantes e ignora las irrelevantes. En la etapa de defuzzificacion se utiliza el método de promedio de centros para obtener el valor final que será aplicado a la entrada analógica del inversor. Cabe mencionar que el método de división con restauración fue implementado y permite obtener el signo de la división para así, poder controlar el sentido de giro del carrusel.

3.2.2 Control de rotación de ángulo de componentes

Para controlar el posicionamiento del componente electrónico sobre la tableta, además de seleccionar el componente deseado sobre el carrusel, es necesario conocer el ángulo de rotación, en el cual será situado el componente, para ello la máquina consta con sistema de rotación de componentes en 8 direcciones que puede girar simétricamente cada 45 °. La tabla de direcciones o ángulos de giro, en las cuales puede ser montado el componente se muestran en la tabla 3.1. Los datos contenidos ahí corresponden a la dirección de empaquetado que cada proveedor tiene disponible, ejemplo de ello es la empresa Asiática ROHM, cuyo empaquetado puede cumplir con la tabla siguiente.

	C*0	**C*1**	**C*2**	**C*3**	**C*4**	**C*5**	**C*6**	**C*7**
M*0	0	90	0	90	45	135	45	135
M*1	0	90	0	90	45	135	45	135
M*2	0	90	180	270	45	135	225	345
M*3	180	270	0	90	225	315	45	135
M*4	90	0	90	0	135	45	135	45
M*5	180	270	0	90	225	315	45	135
M*6	0	90	180	270	45	135	225	315
M*7	270	0	90	180	315	45	135	225
M*8	90	180	270	0	135	225	315	45

Tabla 3.1. Nivel de rotación de componentes.

Donde M* significa la dirección de empaquetado de fábrica y C* la dirección en que puede ser colocado, dependiendo de sus características físicas.

Este proceso es un sistema mecatrónico el cual mediante un motor a pasos que mueve un brazo mecánico, y a su vez mueve un centrador para realizar la rotación al ángulo deseado. En la figura 3.10. Se observa el centrador de 8 direcciones, el cual recibe del código generado por algún software de diseño que genere el formato RS-274X, y parte de la trama correspondiente al ángulo de rotación es decodificada mediante un programa realizado en VHDL.

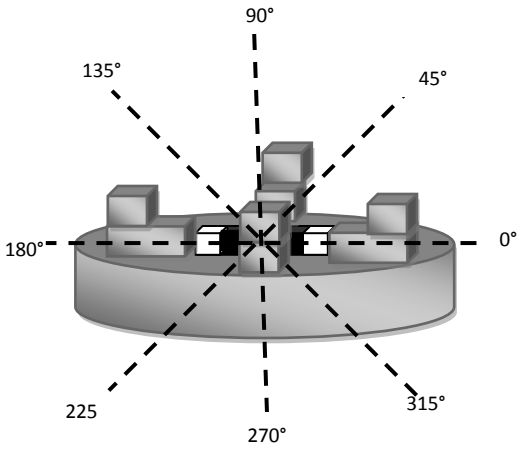


Figura 3,10. Sistema de posicionamiento 8 direcciones.

En la figura 3.11 se puede observar el sistema de rotación de 8 direcciones, el cual fue diseñado para girar en pasos completos en dirección de las manecillas del reloj (CW), pasos completos en dirección contraria a las manecillas del reloj (CCW), para lograr los avances rápidos.

Y para el logro de los posicionamientos intermedios, es decir, las rotaciones a 45°, se implementaron pasos medios en ambas direcciones de giro (CW y CCW), en esta etapa se utilizó toda la etapa mecánica y eléctrica del sistema siendo solo necesaria una etapa de opto-acoplamientos.

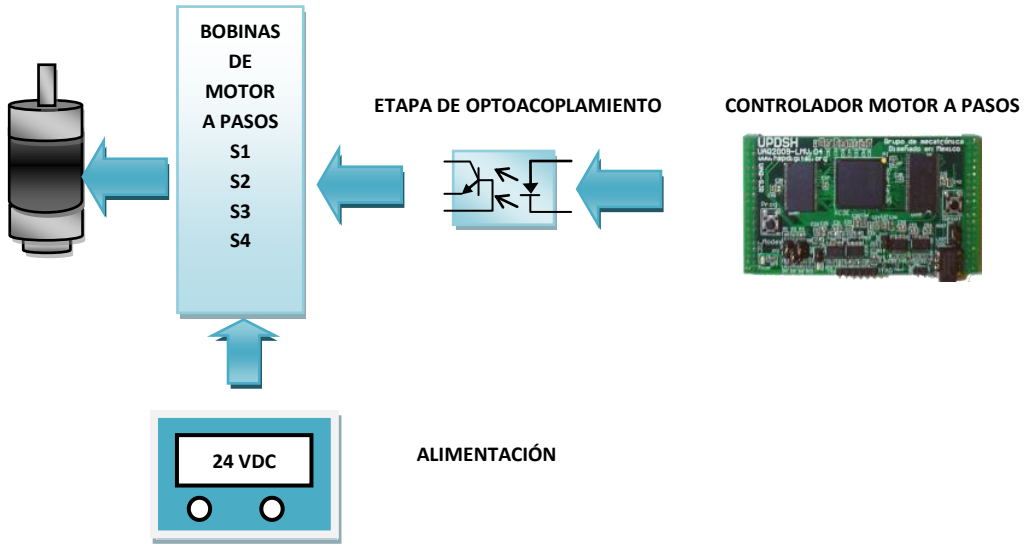


Figura 3.11. Sistema de rotación en 8 direcciones.

Capítulo IV

Pruebas y Resultados

En el presente capítulo se describen las pruebas realizadas para la reconversión en el hardware de la máquina de montaje de componentes SMD RX-4A. Para probar la funcionalidad del sistema de control y posicionamiento x-y, y carrusel diseñados e implementados, se ejecutaron diferentes pruebas realizando movimientos punto a punto de una secuencia de montaje de componentes electrónicos chip sobre una tableta, a partir de un circuito electrónico genérico diseñado en el software de integración y diseño FABMASTER de la compañía IFR international, la referencia de comparación se realizó contra un controlador comercial Galil DMC-1832, estableciendo las mismas condiciones para seguimiento de sus trayectorias. En el caso del sistema comercial se utilizó el algoritmo de auto sintonización (Auto-Crossover-Frequency) desarrollado por el fabricante.

4.1 Controlador propio basado en FPGA

Esta prueba tiene la finalidad de observar las ventajas que se obtienen al utilizar el generador de perfil polinomial de movimiento del controlador propio, la prueba consistió en realizar una comparación de la dinámica de movimiento obtenida con el controlador propio y un controlador Galil en un desplazamiento de 500 cuentas, la señal de posición se obtuvo por medio de las cuentas adquiridas durante el movimiento, para obtener las gráficas de velocidad, la aceleración y jerk se deriva consecutivamente la posición.

En el caso del controlador Galil se utilizó un perfil trapezoidal simétrico de velocidad, siendo esta la opción que ofrece el controlador, el perfil es diseñado con una velocidad máxima de 1000 cuentas/s y una aceleración de 25000 cuentas/s², la dinámica de movimiento obtenida se muestra en la figura 4.1.

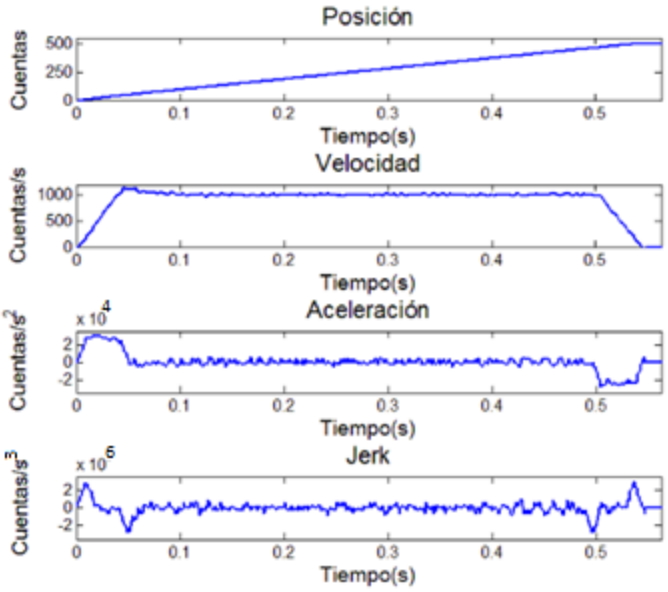


Figura 4.1 Dinámica de movimiento obtenida para el controlador Galil DMC-1832.

Para la prueba del controlador propio se utilizó un perfil polinomial simétrico de sexto grado en posición, con los siguientes parámetros de diseño: velocidad máxima 1000 cuentas/s, aceleración máxima 20000 cuentas/s², la dinámica de movimiento teórica esperada se muestra en la figura 4.2.

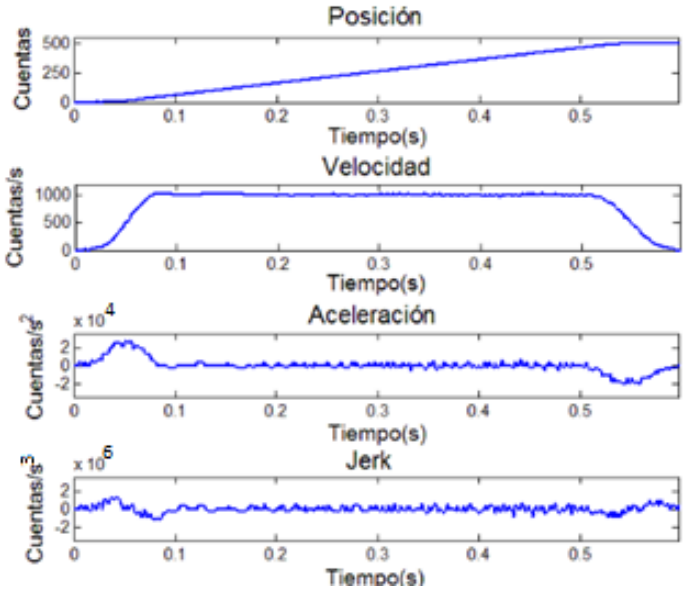


Figura 4.2 Dinámica de movimiento obtenida para el controlador propio basado en FPGA.

Comparando la dinámica de ambos controladores se aprecia en la gráfica que muestra el historial del parámetro Jerk, como con el controlador propio se obtienen movimientos más suaves sin sobrepaso, movimiento deseable en todo sistema de control de posicionamiento debido a la inclusión de un generador de perfil polinomial de movimiento.

Respecto a la resolución que presentó el controlador propio basado en FPGA (utilizando la ecuación 3.2), contra la especificación del manual de la máquina AVIMONT se obtuvieron los resultados de la tabla 4.1.

	MÁQUINA RX-4A	**CONTROLADOR PROPIO BASADO EN FPGA**
EJE X	0.2 mm	0.721 µm
EJE Y	0.2 mm	0.681 µm

Tabla 4.1. Resolución del sistema de posicionamiento basado en controlador propio.

Debido al juego mecánico por el uso de la bancada y la fricción estática, tanto la resolución que maneja el manual y la resolución calculada, pueden variar de la resolución exacta.

4.2 Interpretación de datos RS-274X para generación de referencias

Como se mencionó al principio de este capítulo se parte de un circuito diseñado en algún software de diseño, para este caso en la figura 4.3 se muestras las trayectorias y secuencias que debe seguir el sistema de control, para la colocación de componentes SMD, el cual genera un archivo con las posiciones x-y, y el ángulo de rotación en el que deben ir montados, siendo el estándar industrial RS-274X extendido regulado por la EIA.

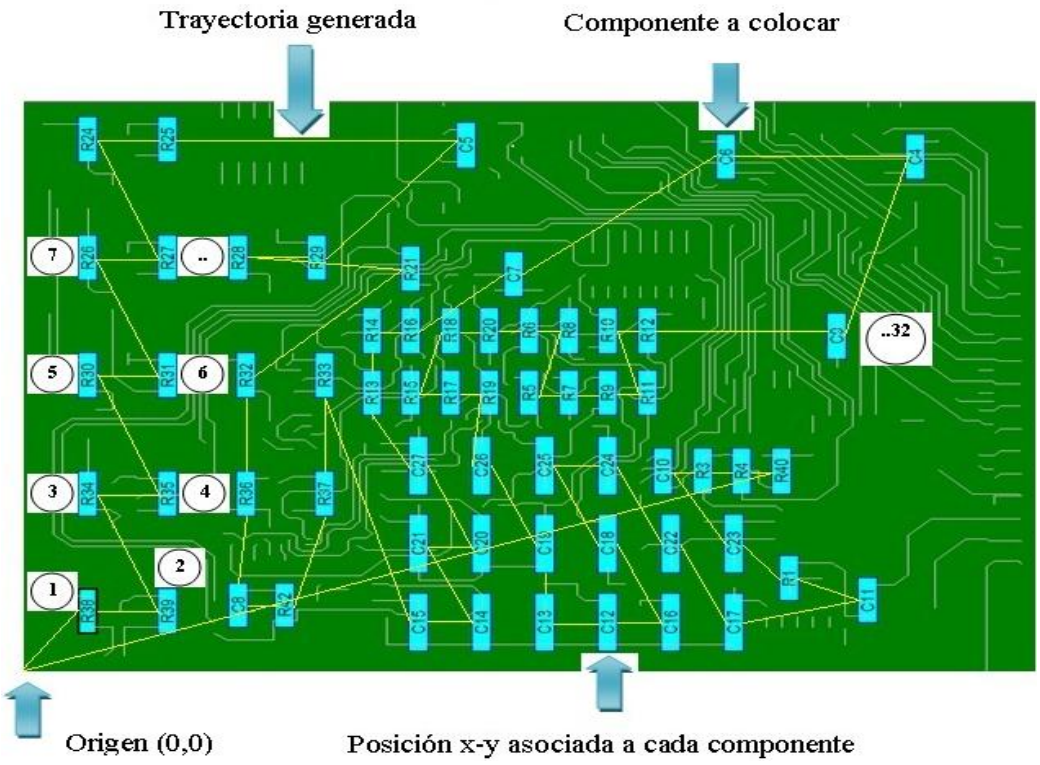


Figura 4.3. Trayectorias de colocación de componentes

Para cargar las coordenadas de posición y rotación se puede realizar de dos maneras, la primera es ingresar la coordenada x-y una a una para correr una secuencia, y otra que es la más usual es cargar un archivo con extensión .txt que puede generar algún software de diseño, esto se hizo por medio de la interfaz de usuario para probar el sistema. Dicha interfaz se muestra en la figura 4.4.

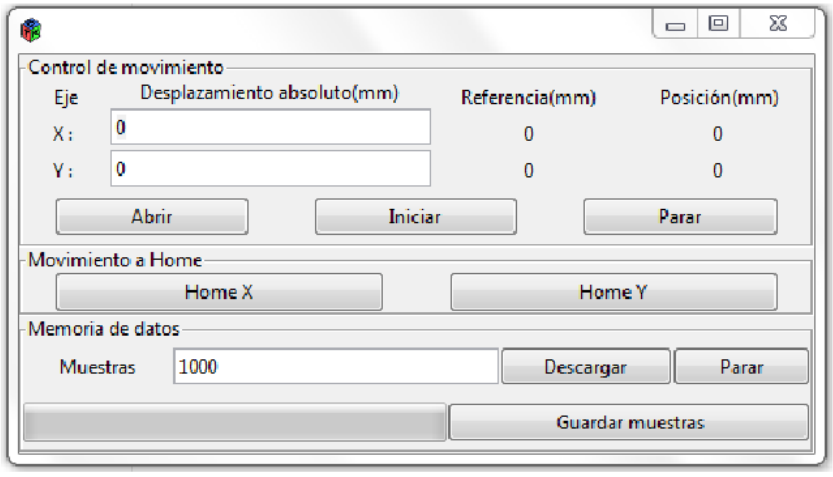


Figura 4.4. Interfaz de usuario para controlador propio.

Para realizar la validación de los datos obtenidos en la experimentación, debido a que no se cuenta con un patrón de referencia se hace un comparativo entre el controlador comercial Galil DMC-1832 y el controlador propio utilizando IP cores basados en FPGA, siguiendo la trayectoria de la figura 4.3, de ambos ejes x-y. En color azul se muestra la trayectoria de referencia y color rojo la trayectoria de seguimiento de ambos controladores figuras 4.5 y 4.6.

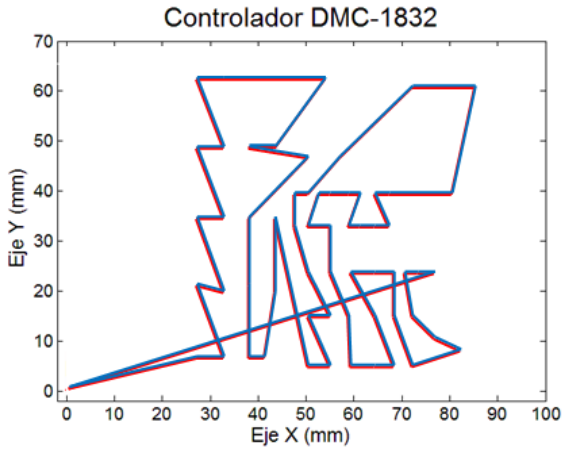


Figura 4.5. Trayectorias de ejes coordenados de controlador DMC-1832

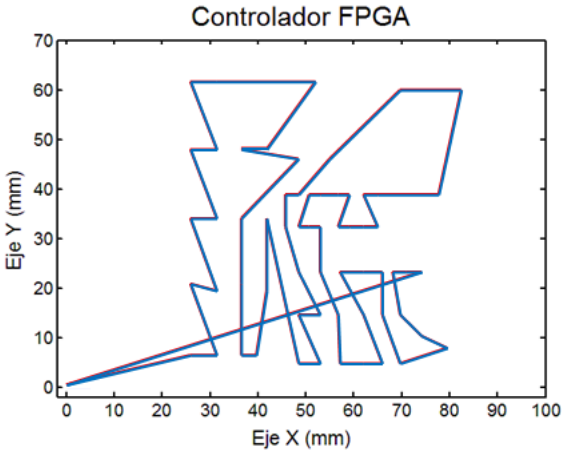


Figura 4.6. Trayectorias de ejes coordenados de controlador FPGA

Para tener una evaluación más acertada, sobre el error generado por los controladores se calcularon sobre el error los siguientes parámetros: $\bar{x}$ la media del error, y la desviación estándar σ. Sobre un tamaño de muestras de 100 000 datos.

Los cuales dieron como resultado los datos de la tabla 4.2 y sus gráficas correspondientes en las figuras 4.7 y 4.8.

Controlador	$\bar{x}$	σ	Error máximo
Controlador DMC-1832	0.22738	0.00819	0.26160
Controlador basado en FPGA	0.00254	0.00023	0.03600

Tabla 4.2. Mediciones estadísticas de datos obtenidos de posicionamiento.

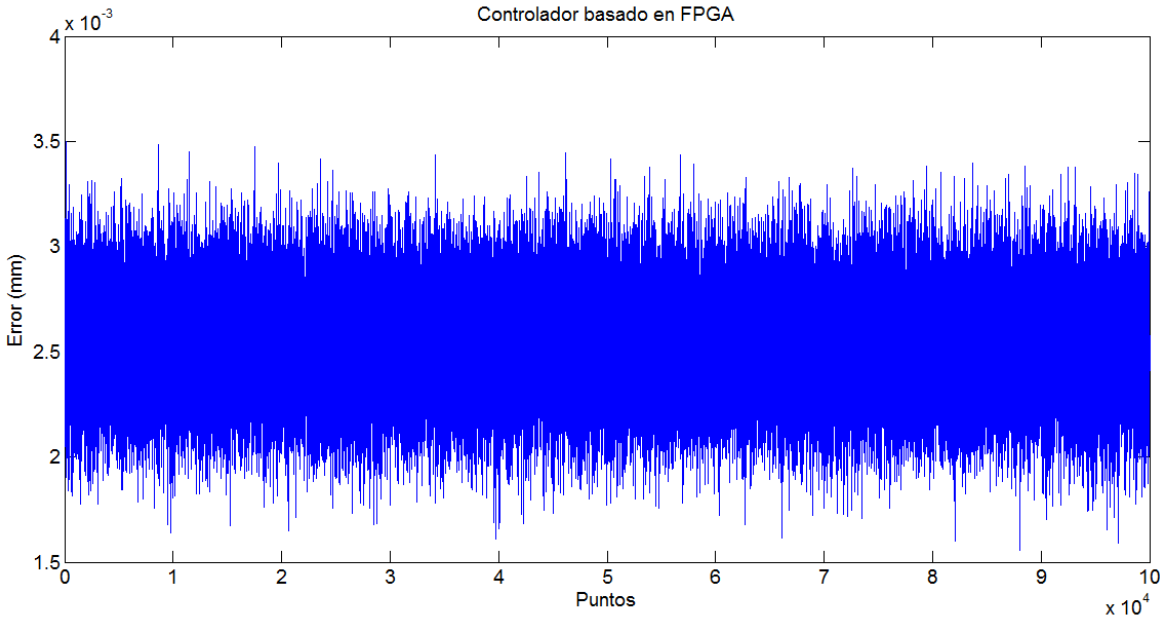


Figura 4.7. Gráfica del error de controlador FPGA

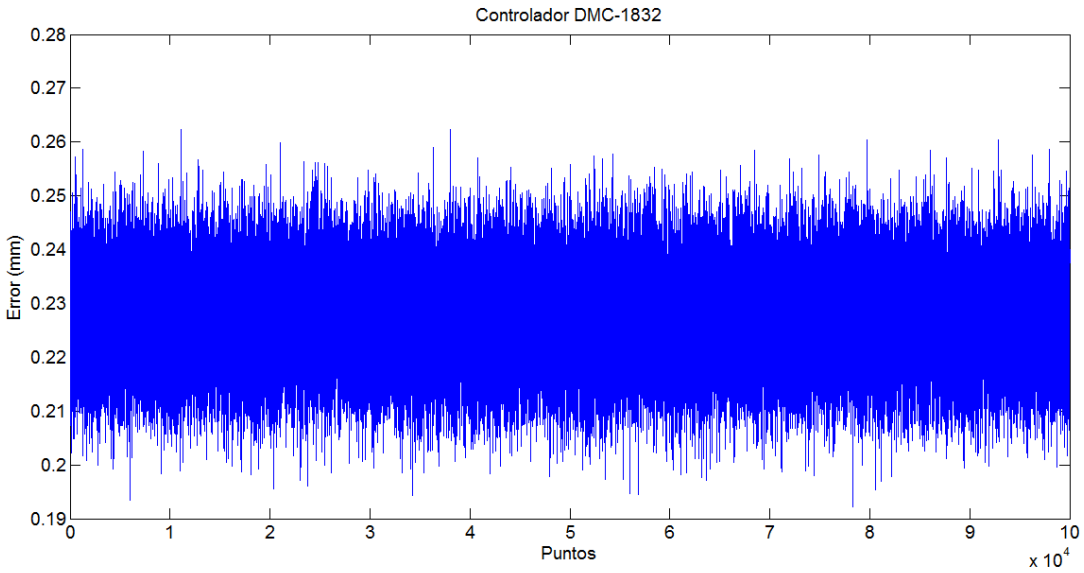
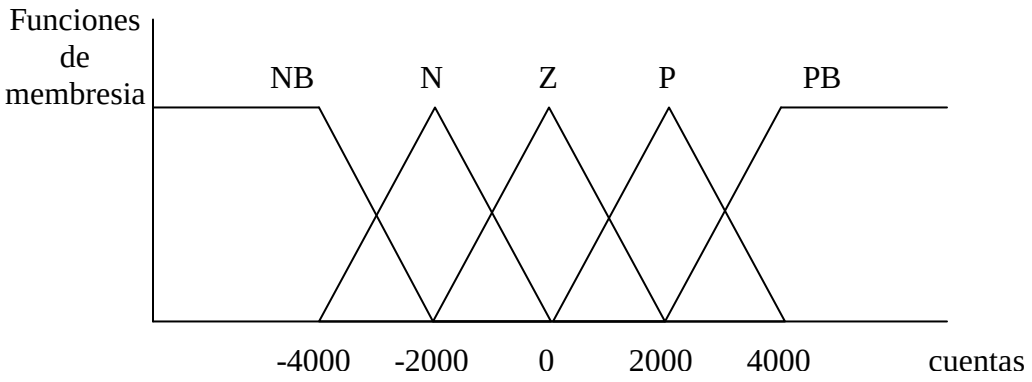


Figura 4.8. Gráfica del error de controlador DMC-1832.

De acuerdo a los resultados observados en las figuras 4.7 y 4.7 , la tabla 1 muestra claramente que la media de los datos analizados del controlador basado en FPGA, es 100 veces más pequeña comparada con el controlador comercial Galil, respecto a la desviación estándar, se observa que los datos se concentran o presentan una mayor cercanía hacia la media y el valor de error máximo en el controlador propio, lo cual es deseable para el control de posicionamiento en el sistema x-y de la máquina de inserción de componentes.

4.3 Resultados de la instrumentación y control del carrusel

Para poder evaluar la respuesta del controlador difuso ante una entrada deseada o punto de consigna, de acuerdo a la metodología de lógica difusa se definieron las funciones de membrecía triangulares de acuerdo al conocimiento heurístico del carrusel y se les asigno valores lingüísticos NB=negativo grande, N=negativo, Z=cero, P=Positivo, PB=Positivo grande y con un dominio de [-4000,4000] cuentas de encoder que corresponden a la posición actual del carrusel, donde como ya se mencionó el universo de discurso es igual para el error (e) y la derivada del error (ė) como se puede observar en la figura 4.9.



4.9 Funciones de membrecía para la señal de e y ė en la etapa de fuzzificacion.

El siguiente paso es implementar la máquina de inferencias y la base de reglas de acuerdo a la ecuación:

$$Nmaxr = Nvl^{Nvar} \tag{3.3}$$

Donde *Nmaxr*=Número máximo de reglas

Nvl=Numero de valores lingüísticos

Nvar=Numero de variables de entrada

Por lo que tomando la ecuación anterior y tomado los valores correspondientes Nmaxr=5^2=25 se calcula la cantidad de reglas como se muestra en la tabla 4.3.

Voltaje	**Derivada del error (ė)**					
Error(e)		NB	N	Z	P	PB
	NB	PVB	PB	P	PS	Z
	N	PB	P	PS	Z	NS
	Z	P	PS	Z	NS	Z
	P	PS	Z	NS	N	NB
	PB	Z	NS	N	NB	NVB

Tabla 4.3. Reglas para control de posición del motor en el carrusel.

Para la variable de salida las funciones de membrecía también son también triangulares y se establecen conforme a la tabla 4. En un rango de [-2000,000], esto con el propósito de cubrir el rango de voltaje de salida de la tarjeta controladora la cual proporcionará el voltaje para el movimiento de rotación del carrusel mediante el inversor.

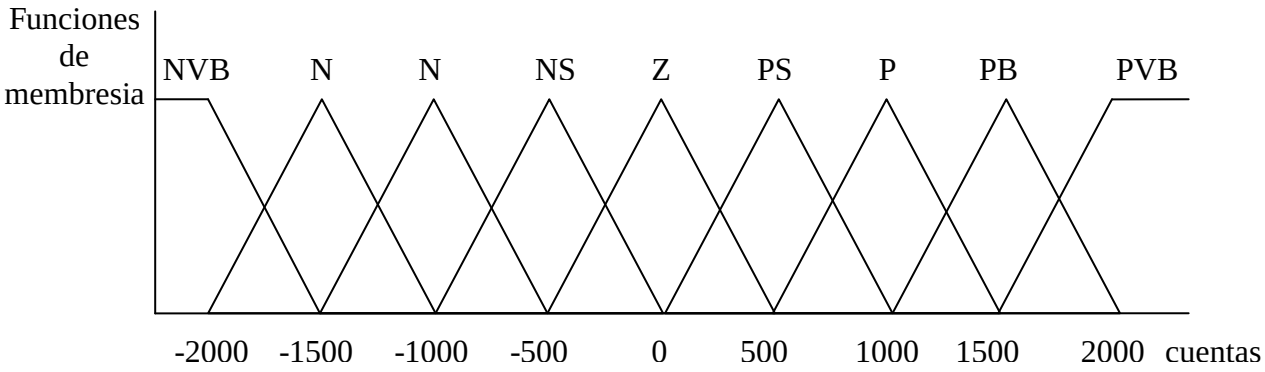


Figura 4.10. Funciones de membrecía para la variable de salida.

Como se mencionó anteriormente de acuerdo a la figura 3.7 la división con restauración fue implementada en VHDL para concluir con el proceso del desarrollo e implementación del controlador difuso.

Para el diseño y prueba del controlador difuso basado en FPGA, se utilizó la herramienta FIS de matlab, la figura 4.11 muestra los resultados obtenidos.

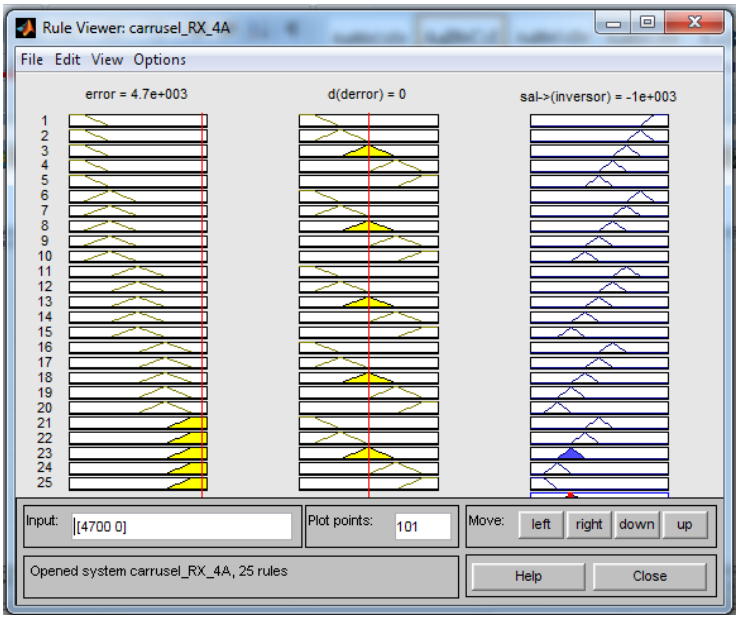


Figura 4.11. Simulación del controlador difuso

Evidentemente este simulador es solo un referente para saber rápidamente como cambia la variable de salida si se cambian algunas reglas, funciones o variable de entrada y así lograr depurar el controlador difuso implementado en VHDL.

La figura 4.12 muestra la implementación del controlador difuso bajo VHDL de los mismos valores de (e) y (ė) que fueron implementados en FIS figura 4.11.

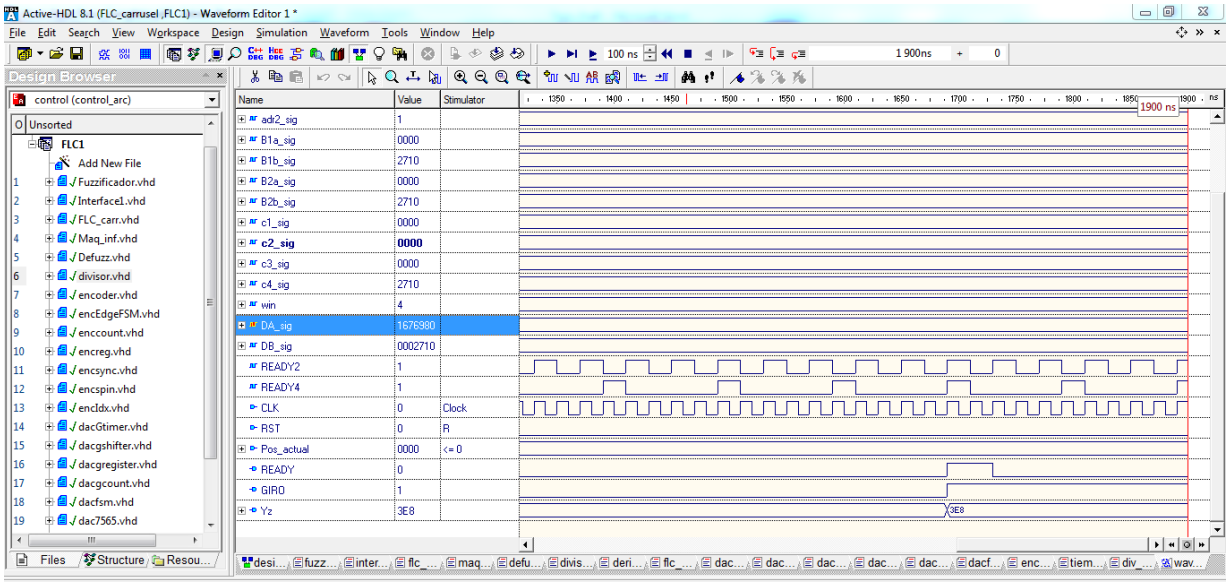
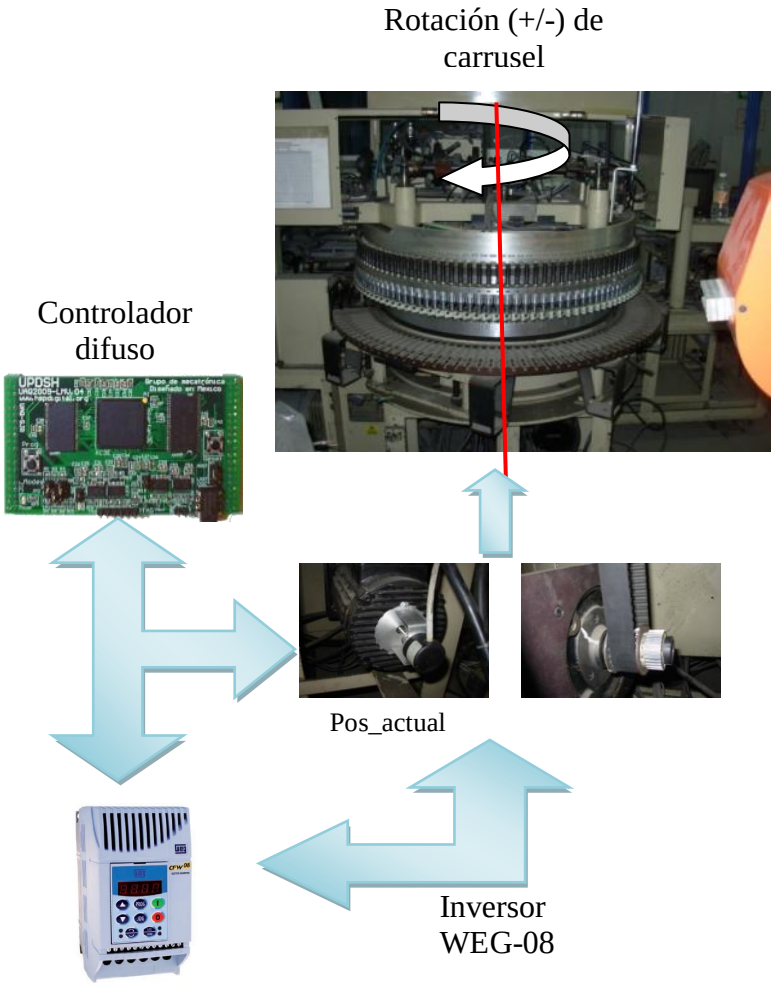


Figura 4.12 Controlador difuso implementado en VHDL.

La posición deseada o requerida (3° de rotación) está declarada como un constante, dentro del código de programa, las variables que se pueden observar en la simulación son la señal de encoder leída (Pos_actual=0), la señal en la cual debe girar el servomotor (GIRO=1/0) y el valor de voltaje que debe entregar la tarjeta controladora al inversor (Yz= 3E8 hex =1000 dec).

Para realizar las pruebas de operatividad sobre carrusel una corrida de una vuelta completa (360°) de carrusel fue realizada, para así poder calcular el error de posición de las boquillas de succión de componentes electrónicos.

El diagrama de la figura 4.13 muestra el flujo de cómo fue implementado el sistema de control difuso para el posicionamiento del carrusel.



4.13. Implementación para calcular el error de posición del carrusel

El resultado de la operatividad respecto a posicionamiento del carrusel para selección de componentes se puede observar en la figura 4.13, donde se puede observar el error una vuelta o giro completo de carrusel.

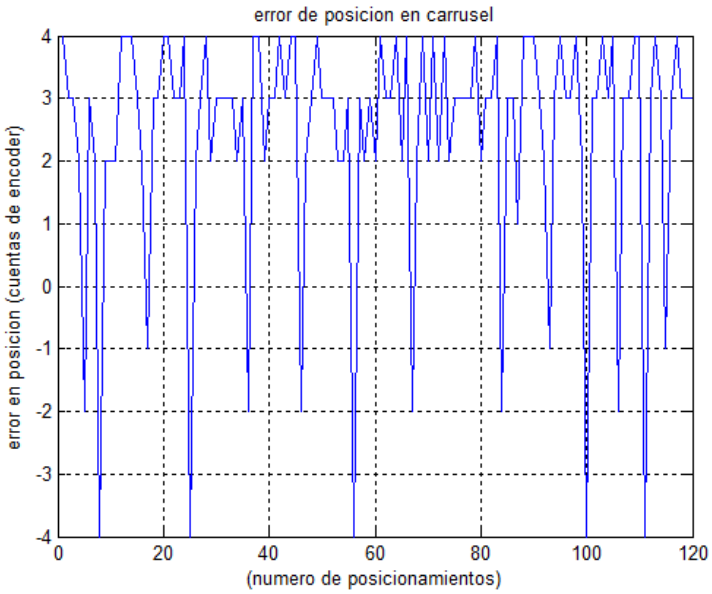


Figura 4.14. Error de posición de carrusel.

Se obtiene una mejor idea de los datos presentados en la gráfica si se obtienen los datos estadísticos de ellos como se muestran en la tabla 4.4.

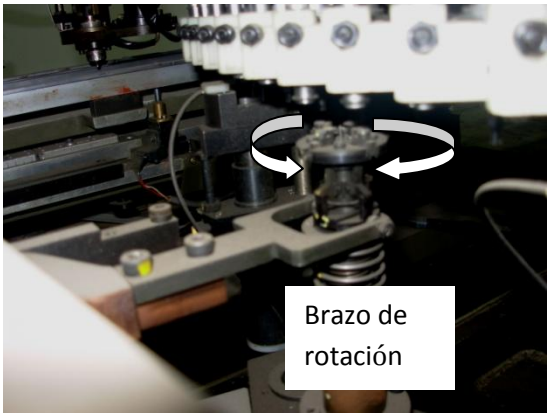
Controlador	$\bar{x}$ (°)	σ	Error máximo
Controlador basado en FPGA	2.4833 cuentas $(1.6x10^{-3}\,^{\circ})$	$(1.1x10^{-3}\,^{\circ})$	8 cuentas $(15.3x10^{-3}\,^{\circ})$
Maquina RX-4A (Manual AVIMONT)	No existe referencia	No existe referencia	No existe referencia

Tabla 4.4. Análisis estadístico del error de posición de carrusel.

Los datos presentados en la tabla 5, son valores de error medido sobre el eje el servomotor, sin embargo, el error se reduce al orden de milésimas debido a la relación de movimiento entre éste y el posicionamiento final de la boquilla montada sobre el carrusel. En este cálculo su comparación fue contra la operatividad de la máquina en línea de producción, debido a que un mal posicionamiento en este sistema causaba que el componente a montar no se tomara, o llevara una posición incorrecta que causaba la caída del componente de la boquilla correspondiente, lo cual causaba también que el ensamble total de la PCB tomara más tiempo.

4.3.1 Rotación 8 direcciones

Para determinar el funcionamiento del sistema de rotación en 8 direcciones del sistema, se desarrolló en VHDL un controlador para cumplir las condiciones que se pueden encontrar en la tabla 1, para posicionar en el ángulo correcto cada uno de los componentes que llevan las boquillas.



4.15 Sistema físico de rotación 8 direcciones

Este brazo funciona mediante un motor unipolar a pasos y toma un punto de referencia mediante un sensor óptico que indica la posición de origen, es decir, el ángulo de montaje igual a 0°, el programa implementado en VHDL se puede observar en la figura 4.16.

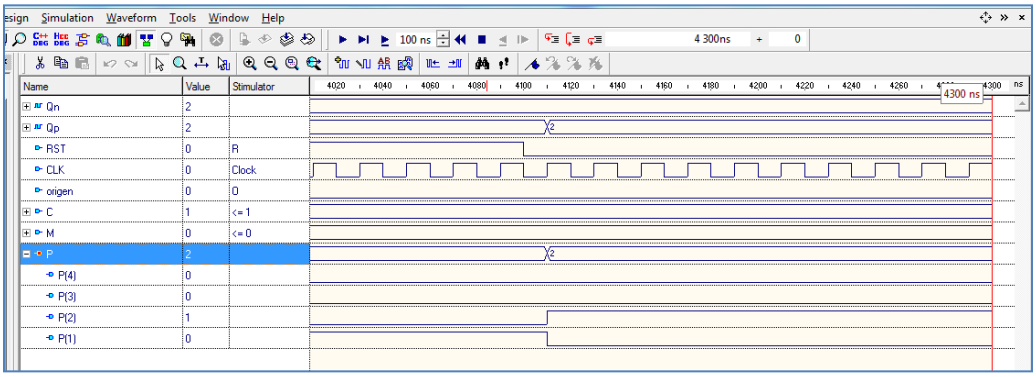


Figura 4.16. Controlador de motor a pasos para rotación 8 direcciones

Como se observa en el programa las variables de entrada son el sensor de origen, los parámetros del tipo de componente a montar M, y el ángulo C con el que debe ser montado el componente sobre la PCB. Por lo anterior a cada ángulo de rotación le corresponde la activación

de una o un par de bobinas del motor a pasos que se presentan en la figura anterior como las salidas activas de P (P1, P2, P3, P4).

4.3 Resultados de integración del sistema

Para probar la operatividad esencial del sistema se realizó la integración de los módulos basados en FPGA, que permiten controlar los eventos que intervienen en la colocación de componentes electrónicos tipo chip. Para ello se genera una máquina de estados finita (FSM) que permita controlar la secuencia de activación de cada evento como se muestra en la figura 4.17.

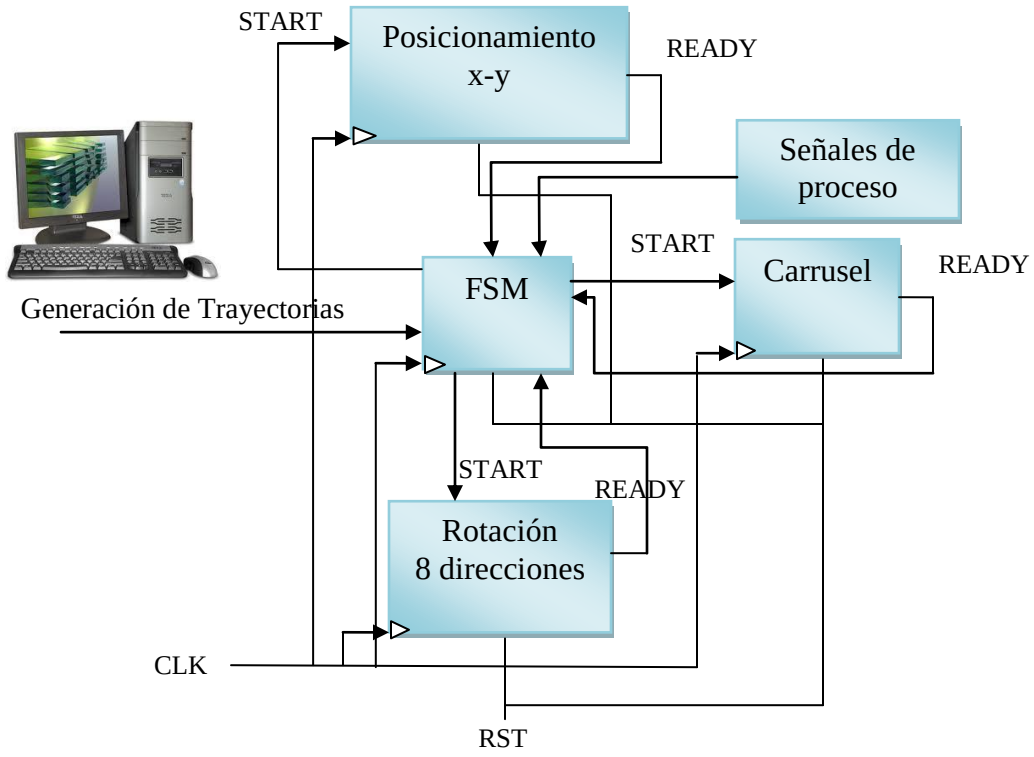


Figura 4.17. Máquina de estados para el control de los eventos.

Como se puede observar en la figura 4.17 adicionalmente a los módulos desarrollados una serie de señales que indican los estados de cada proceso son tomadas en cuenta para el funcionamiento o paro de una secuencia de montaje si una señal de estas no está activa en el estado lógico correspondiente ocasiona automáticamente que la máquina pare y este error se indicará por el encendido de un led en la tarjeta controladora, para cada diferente tipo de error.

La tabla 4.5 describe cada una de señales adicionales al proceso de montaje.

Señal	Descripción de señal	Acción lógica
S1	Presencia ausencia de componente	Utilizar otra boquilla
S2	Rotación incorrecta	Expulsar componente
S3	Detección de componente después de rotación	Utilizar otra boquilla
S4	Origen de carrusel	Rotar hasta posición inicial
S5	Limite x+ de bancada	Paro de la máquina
S6	Limite x- de bancada	Paro de la máquina
S7	Limite y+ de bancada	Paro de la máquina
S8	Limite y+ de bancada	Paro de la máquina

Tabla 4.5. Descripción de las señales que intervienen en el proceso de montaje.

Como resultado de la prueba para posicionar los componentes de la figura 4.3, se obtuvo el siguiente resultado haciendo énfasis en el parámetro de tiempo de ejecución de la prueba.

La ecuación que determina el promedio de montaje en la máquina utilizando la reconversión en hardware es

$$AVR = \frac{T}{n} \tag{3.4}$$

Donde T=número total de pasos

n= Tiempo total de pasos

Y tomando en cuenta la consideración de que este sistema utiliza para el control del servomotor del carrusel un inversor, el cual debido a su construcción tiene un retardo de tiempo en aceleración de 0.1 segundo, una velocidad constante de 0.1 segundo y un tiempo de desaceleración de 0.1s, lo cual da como resultado una velocidad de montaje de 0.3 componentes por segundo. Aunque el sistema basado en FPGA tiene una velocidad de procesamiento muy superior a esa velocidad como se muestra en la tabla 4.6.

Condición de operación	Velocidad de Inserción
Máquina antes de la reconversión	0.5 shots/ segundo
Máquina después de la reconversión en Hardware	0.33 shots/ segundo

Tabla 4.6. Velocidad de montaje de componentes sobre PCB.

Se puede observar claramente como en promedio la velocidad de la máquina es aumentada lo cual se deduce que aumentará su porcentaje de productividad.

Capítulo V

Conclusiones y Prospectivas

El desarrollo del presente trabajo ha permitido la reconversión en hardware de una máquina RX-4A en los procesos de posicionamiento y rotación en el ensamble de componentes electrónicos tipo chip, lo cual da como resultado que mediante el uso de IP cores propios basados en FPGA y hardware de control, permiten el uso de tecnología de nueva generación, y que es compatible con el estándar industrial extendido RS-274X de EIA, utilizado en el ensamble de componentes electrónicos sobre PCB en la industria electrónica.

Un punto muy importante que se logró con la implementación del controlador difuso sobre el carrusel, fue la de reducir error en posición que presentaba la máquina en este proceso en su condición original o de fábrica. Y seguramente disminuir el desperdicio o merma de componentes electrónicos atribuidos a este error de posicionamiento.

Con la implementación de este tipo sistemas de control, se obtienen como resultado sistemas de control de arquitectura abierta, flexibles, reconfigurables y de bajo costo que muestran una mejora en la operatividad de este tipo de máquina con respecto a su sistema del posicionamiento y rotación, lo cual también da como beneficio el contar con un sistema flexible que permita la reconversión de maquinaria de este tipo y permitir que su vida útil pueda ser extendida.

La pruebas de validación han demostrado también que la reconversión de hardware mediante IP cores basados en FPGA, con tecnología propia pueden ser un área de oportunidad para las empresas que hacen uso de transferencia de maquinaria hacia nuestro país, y de la simple integración de equipo o accesorios que no son realmente una solución para aumentar la vida útil productiva de una máquina –herramienta.

Como resultado de este proyecto de tesis se logró la publicación de un artículo referente a trayectorias generadas a partir de códigos Gerber, en el 8° Congreso Internacional de Ingeniería.

El siguiente paso en el desarrollo de este proyecto puede ser el desarrollo de un sistema de visión basado en FPGA, para controlar el OFF-SET de la secuencia de ensamblado sobre el sistema de posicionamiento x-y, el diseño mecatrónico de un aplicador de componentes tipo chip y diseño de boquillas intercambiables que permitan seleccionar una mayor gama de diferentes tamaños de componentes. Finalmente se pueden desarrollar más IP cores que eficienten todos los procesos que involucra la máquina y que fácilmente pueden ser integrados.

Bibliografía

Aguado B.A. 2000. Temas de identificación y control adaptable. Instituto de Cibernética, Matemática y Física. Impresa en los talleres de la empresa de comunicación de Ciencia y tecnología, PALCIEN, La Habana, Cuba. ISBN 959-7056-11-9.

Altrock. C.V., 1995. Fuzzy Logic and Neuro Fuzzy applications explained, Prentice Hall.

Dobois D. and Prade H. 1980. Fuzzy Sets and Systems. Academic Press. Orlando F.L.

Hopwood Adrian A. 2001. Intelligent Systems for Engineers and Scientists. CRC Press.

Instruction Manual AVIMONT 1984. for models RX-4A, Rev. Level 0.

Jaen Cuellar, Arturo Y. 2011. Desarrollo de perfiles polinomiales 3D basado en FPGA para control de posición en máquina fresadora CNC. Tesis Maestría. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Jun Wang, Mingzhu Zhang. 2006. A New High-Speed Algorithm for Center and Rotate Angle of Electronic Components. IEEE International Conference on Mechatronics and Automation.

Kumar, R. Zhonghui Luo. 2003. Optimizing the operation sequence of a chip placement machine using TSP model. IEEE International Conference on Mechatronics and Automation.

Lee C.C. 1990. Fuzzy Logic in Control Systems, Transactions on Systems and Cybernetics. Vol. 20. Merlapr.

Mejía Ugalde Mario. 2009, Tesis de Grado Maestría. Desarrollo de un Control Numérico Computarizado para torno. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Morales Velázquez L., 2010, Diseñó de plataforma hardware-software para el desarrollo de aplicaciones industriales basadas en FPGA, Tesis de Doctorado, Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Morales Velázquez L., Romero Troncoso R. de J., Osornio Ríos R. A., Herrera Ruiz G. y De Santiago Pérez J. J., 2009, Special purpose processor for parameter identification on CNC second order servo Systems on a low-cost FPGA platform, Journal Elsevier

Muñoz Barrón Benigno. 2009. Tesis de Grado Ingeniería. Diseño de un procesador PLC basado en FPGA para aplicación en maquinaria CNC. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Osornio Ríos R. A., Romero Troncoso R. J., 2008, Herrera Ruiz G. y Castañeda Miranda R., FPGA implementation of higher degree polinomial acceleration profiles for peak jerk reduction on servomotors, Journal Science Direct.

Osornio , 2004. Tesis de Grado Maestría. Diseño y construcción de una tarjeta controladora de 3 ejes. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Rivera Guillén, Jesús R. 2007., Perfiles Polinomiales de Movimiento para Máquinas CNC. Tesis Maestría. Universidad de Guanajuato. FIMEE.

Ronquillo, 2002. Tesis de Grado Maestría. Control digital de servomotor sin escobillas de corriente directa. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Tirpak, T.M. Mohapatra, P.K. Nelson, P.C. Rajbhandari, R.R. 2002. A generic classification and object-oriented simulation toolkit for SMT assembly equipment. IEEE International Conference on Systems, Man and Cybernetics.

Trejo Hernandez Miguel, 2004. Tesis de Grado Maestría. Módulo de maquinado y monitoreo, aplicando control difuso en un proceso de torneado. Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Weihsin Wang, Nelson, P.C., Tirpak, T.M. 2002. Optimization of high-speed multistation SMT placement machines using evolutionary algorithms. IEEE International Conference on Systems, Man and Cybernetics.

Wiklund Kristoffer. 2010. Optimization of the feeder assignment for PCB assembly machines University of Gothenburg, Deparment of Science and Engineering.

William Ho and Pin Ji. 2009. Integrated component scheduling models for chip shooter machines. Journal Science Direct.

Witold Pedrycs. 1996. Fuzzy Control and Fuzzy Systems. John Wiley & Sons Inc. 2th. Edition

Woodgate Ralph W. 2005. The Handbook of Machine Soldering SMT & TH. A Whiley Interscience Publication, 3th. Edition.

Apéndice A

Módulos VHDL

En este apéndice se presentan las unidades de más alta jerarquía de la arquitectura completa del sistema digital que fue integrado e implementado, de acuerdo a lo descrito en el capítulo 3.

A.1 Modulo de posicionamiento de bancada x-y

```
1   ----------------------------------------------------------------------------
2   --
3   -- Universidad Autonoma de Queretaro
4   -- UAQ-SJR mechatronics group
5   --
6   -- Design      :
7   -- Author      : Luis Morales Velazquez
8   --
9   ----------------------------------------------------------------------------
10  --
11  -- Description :
12  --
13  ----------------------------------------------------------------------------
14  library IEEE;
15  use IEEE.std_logic_1164.all;
16  use IEEE.numeric_std.all;
17
18  entity mcUAQ4x is
19    port(
20          RST : in STD_LOGIC;
21          CLK : in STD_LOGIC;
22          START1: in STD_LOGIC;
23          READY1:out STD_LOGIC;
24          DRC_SLOT : in STD_LOGIC_VECTOR(2 downto 0);
25          USB_DP : inout STD_LOGIC;
26          USB_DM : inout STD_LOGIC;
27          ENC_APX : in STD_LOGIC;
28          ENC_AMX : in STD_LOGIC;
29          ENC_BPX : in STD_LOGIC;
30          ENC_BMX : in STD_LOGIC;
31          ENC_IDXPX : in STD_LOGIC;
32          ENC_IDXMX : in STD_LOGIC;
33          ENC_APY : in STD_LOGIC;
```

```
34            ENC_AMY : in STD_LOGIC;
35            ENC_BPY : in STD_LOGIC;
36            ENC_BMY : in STD_LOGIC;
37            ENC_IDXPY : in STD_LOGIC;
38            ENC_IDXMY : in STD_LOGIC;
39            ENC_APZ : in STD_LOGIC;
40            ENC_AMZ : in STD_LOGIC;
41            ENC_BPZ : in STD_LOGIC;
42            ENC_BMZ : in STD_LOGIC;
43            ENC_IDXPZ : in STD_LOGIC;
44            ENC_IDXMZ : in STD_LOGIC;
45            ENC_APW : in STD_LOGIC;
46            ENC_AMW : in STD_LOGIC;
47            ENC_BPW : in STD_LOGIC;
48            ENC_BMW : in STD_LOGIC;
49            ENC_IDXPW : in STD_LOGIC;
50            ENC_IDXMW : in STD_LOGIC;
51            DAC_SCLK : out STD_LOGIC;
52            DAC_SYNC : out STD_LOGIC;
53            DAC_DIN : out STD_LOGIC;
54            DAC_LDAC : out STD_LOGIC;
55            DAC_RST : out STD_LOGIC;
56            ADC_DCLK : out STD_LOGIC;
57            ADC_DIN : out STD_LOGIC;
58            ADC_BUSY : in STD_LOGIC;
59            ADC_DOUT : in STD_LOGIC;
60          SW_LIM : in STD_LOGIC_VECTOR(11 downto 0);
61            DRAM_CLK : out STD_LOGIC;
62            DRAM_CKE : out STD_LOGIC;
63            DRAM_CS : out STD_LOGIC;
64            DRAM_WE : out STD_LOGIC;
65            DRAM_CAS : out STD_LOGIC;
66            DRAM_RAS : out STD_LOGIC;
67            DRAM_DQML : out STD_LOGIC;
68            DRAM_DQMH : out STD_LOGIC;
69            DRAM_BA : out STD_LOGIC_VECTOR(1 downto 0);
70            DRAM_A : out STD_LOGIC_VECTOR(12 downto 0);
71            DRAM_DQ : inout STD_LOGIC_VECTOR(15 downto 0);
72          DIG_OUTS : out STD_LOGIC_VECTOR(7 downto 0);
73            I2C_SCL : out STD_LOGIC;
74            I2C_SDA : inout STD_LOGIC;
75            RAM_CS: out STD_LOGIC;
76            RAM_OE: out STD_LOGIC;
77            RAM_WE: out STD_LOGIC;
78            RAM_DATA: inout STD_LOGIC_VECTOR(7 downto 0);
79            RAM_ADDR: out STD_LOGIC_VECTOR(18 downto 0)
80      );
81    end mcUAQ4x;
82
83    architecture mcUAQ4x of mcUAQ4x is
84
85       component pidPID_4D
86         generic(ex : integer := 5;
87                     fx : integer := 13;
88                     ey : integer := 5;
89                     fy : integer := 13;
90                     ec : integer := 5;
91                     fc : integer := 13);
92         port(
93             RST : in STD_LOGIC;
94             CLK : in STD_LOGIC;
95             STR : in STD_LOGIC;
96             RDY : out STD_LOGIC;
97             CLR : in STD_LOGIC;
98             XK_1  : in  STD_LOGIC_VECTOR(ex+fx-1 downto 0);
99             YK_1  : out STD_LOGIC_VECTOR(ey+fy-1 downto 0);
```

```
100         a0_1,a1_1,a2_1   : in  STD_LOGIC_VECTOR(ec+fc-1 downto 0);
101         XK_2  : in  STD_LOGIC_VECTOR(ex+fx-1 downto 0);
102         YK_2  : out STD_LOGIC_VECTOR(ey+fy-1 downto 0);
103         a0_2,a1_2,a2_2   : in  STD_LOGIC_VECTOR(ec+fc-1 downto 0);
104         XK_3  : in  STD_LOGIC_VECTOR(ex+fx-1 downto 0);
105         YK_3  : out STD_LOGIC_VECTOR(ey+fy-1 downto 0);
106         a0_3,a1_3,a2_3   : in  STD_LOGIC_VECTOR(ec+fc-1 downto 0);
107         XK_4  : in  STD_LOGIC_VECTOR(ex+fx-1 downto 0);
108         YK_4  : out STD_LOGIC_VECTOR(ey+fy-1 downto 0);
109         a0_4,a1_4,a2_4   : in  STD_LOGIC_VECTOR(ec+fc-1 downto 0));
110     end component;
111
112     component nbsNurbs
113      generic(ex : integer := 16;
114              fx : integer := 0;
115              ew : integer := 5;
116              fw : integer := 11;
117              et : integer := 2;
118              ft : integer := 14);
119       port(
120           RST : in STD_LOGIC;
121           CLK : in STD_LOGIC;
122           CLR : in STD_LOGIC;
123           STR : in STD_LOGIC;
124           RDY : out STD_LOGIC;
125           LDP : in STD_LOGIC;
126           PX : in STD_LOGIC_VECTOR(ex+fx-1 downto 0);
127           PY : in STD_LOGIC_VECTOR(ex+fx-1 downto 0);
128           PZ : in STD_LOGIC_VECTOR(ex+fx-1 downto 0);
129           PW : in STD_LOGIC_VECTOR(ex+fx-1 downto 0);
130           WW : in STD_LOGIC_VECTOR(ew+fw-1 downto 0);
131           LDT : in STD_LOGIC;
132           TK : in STD_LOGIC_VECTOR(et+ft-1 downto 0);
133           T : in STD_LOGIC_VECTOR(et+ft-1 downto 0);
134           X : out STD_LOGIC_VECTOR(ex+fx-1 downto 0);
135           Y : out STD_LOGIC_VECTOR(ex+fx-1 downto 0);
136           Z : out STD_LOGIC_VECTOR(ex+fx-1 downto 0);
137           W : out STD_LOGIC_VECTOR(ex+fx-1 downto 0));
138     end component;
139
140     component prfProfile
141      generic(et : integer := 32;
142              ft : integer := 32;
143              ec : integer := 32;
144              fc : integer := 32;
145              n  : integer := 32);
146       port(
147           RST : in STD_LOGIC;
148           CLK : in STD_LOGIC;
149           FS : in STD_LOGIC;
150           CLR : in STD_LOGIC;
151           JOG : in STD_LOGIC;
152           RDY : out STD_LOGIC;
153           EOP : out STD_LOGIC;
154           TS : in STD_LOGIC_VECTOR(ft-1 downto 0);
155           Ka1,Ka2,Ka3,Kd1,Kd2,Kd3 : in STD_LOGIC_VECTOR(ec+fc-1 downto 0);
156           Kv : in STD_LOGIC_VECTOR(2*fc-1 downto 0);
157           K1,K2,K3 : in STD_LOGIC_VECTOR(n-1 downto 0);
158           X : out STD_LOGIC_VECTOR(ft-1 downto 0));
159     end component;
160
161      component encEncoder
162       generic(n : integer := 16;
163               m : integer := 4);
164       port(
165           RST : in STD_LOGIC;
```

```
166         CLK : in STD_LOGIC;
167         CLR : in STD_LOGIC;
168         DIF : in STD_LOGIC;
169         RD : in STD_LOGIC;
170         K : in STD_LOGIC_VECTOR(m-1 downto 0);
171         D : out STD_LOGIC_VECTOR(n-1 downto 0);
172         IDX : out STD_LOGIC_VECTOR(n-1 downto 0);
173         ENC_AP : in STD_LOGIC;
174         ENC_AM : in STD_LOGIC;
175         ENC_BP : in STD_LOGIC;
176         ENC_BM : in STD_LOGIC;
177         ENC_IDXP : in STD_LOGIC;
178         ENC_IDXM : in STD_LOGIC);
179     end component;
180
181     component DAC7565
182       port(
183         RST : in STD_LOGIC;
184         CLK : in STD_LOGIC;
185         WR : in STD_LOGIC;
186         EOW : out STD_LOGIC;
187         DRST : in STD_LOGIC;
188         Ch0 : in STD_LOGIC_VECTOR(15 downto 0);
189        Ch1 : in STD_LOGIC_VECTOR(15 downto 0);
190        Ch2 : in STD_LOGIC_VECTOR(15 downto 0);
191        Ch3 : in STD_LOGIC_VECTOR(15 downto 0);
192         DAC_SCLK : out STD_LOGIC;
193         DAC_SYNC : out STD_LOGIC;
194         DAC_DIN : out STD_LOGIC;
195         DAC_LDAC : out STD_LOGIC;
196         DAC_RST : out STD_LOGIC);
197     end component;
198
199     component ADS7841
200       port(
201         RST : in STD_LOGIC;
202         CLK : in STD_LOGIC;
203         RD : in STD_LOGIC;
204         EOR : out STD_LOGIC;
205         Ch0 : out STD_LOGIC_VECTOR(15 downto 0);
206         Ch1 : out STD_LOGIC_VECTOR(15 downto 0);
207         Ch2 : out STD_LOGIC_VECTOR(15 downto 0);
208         Ch3 : out STD_LOGIC_VECTOR(15 downto 0);
209         ADC_DCLK : out STD_LOGIC;
210         ADC_DIN : out STD_LOGIC;
211         ADC_BUSY : in STD_LOGIC;
212         ADC_DOUT : in STD_LOGIC);
213     end component;
214
215     component dramDRAM
216       generic(n : integer := 256;
217                 m : integer := 4);
218       port(
219         RST : in STD_LOGIC;
220         CLK : in STD_LOGIC;
221         CS : in STD_LOGIC;
222         WR : in STD_LOGIC;
223         RD : in STD_LOGIC;
224         RDY : out STD_LOGIC;
225         K : in STD_LOGIC_VECTOR(m-1 downto 0);
226         Din : in STD_LOGIC_VECTOR(n-1 downto 0);
227         Dout : out STD_LOGIC_VECTOR(n-1 downto 0);
228         DRAM_CLK : out STD_LOGIC;
229         DRAM_CKE : out STD_LOGIC;
230         DRAM_CS : out STD_LOGIC;
231         DRAM_WE : out STD_LOGIC;
```

```
232         DRAM_CAS : out STD_LOGIC;
233         DRAM_RAS : out STD_LOGIC;
234         DRAM_DQML : out STD_LOGIC;
235         DRAM_DQMH : out STD_LOGIC;
236         DRAM_BA : out STD_LOGIC_VECTOR(1 downto 0);
237         DRAM_A : out STD_LOGIC_VECTOR(12 downto 0);
238         DRAM_DQ : inout STD_LOGIC_VECTOR(15 downto 0));
239     end component;
240
241     component usbUSB_v6_0
242     generic(VID : integer := 0;
243             PID : integer := 0);
244      port (
245         RST: in STD_LOGIC;
246         CLK48: in STD_LOGIC;
247         SLOT: in STD_LOGIC_VECTOR (7 downto 0);
248         USB_DP : inout STD_LOGIC;
249         USB_DM : inout STD_LOGIC;
250         CS: in STD_LOGIC_VECTOR (2 downto 1);
251         RDY: out STD_LOGIC_VECTOR (2 downto 1);
252         AD01 : in STD_LOGIC_VECTOR(5 downto 0);
253         EPD1 : out STD_LOGIC_VECTOR(7 downto 0);
254         WEP2 : in STD_LOGIC;
255         AD02 : in STD_LOGIC_VECTOR(5 downto 0);
256         EPD2 : in STD_LOGIC_VECTOR(7 downto 0));
257     end component;
258
259     component pktTerminal
260       port(
261         RST : in STD_LOGIC;
262         CLK : in STD_LOGIC;
263         DRC : in STD_LOGIC_VECTOR(15 downto 0);
264         SLOT : in STD_LOGIC_VECTOR(7 downto 0);
265         RxValid : in STD_LOGIC;
266         RxReady : out STD_LOGIC;
267         TxValid : out STD_LOGIC;
268         TxReady : in STD_LOGIC;
269         DI : in STD_LOGIC_VECTOR(7 downto 0);
270         IDXR : out STD_LOGIC_VECTOR(5 downto 0);
271         WR : out STD_LOGIC;
272         DO : out STD_LOGIC_VECTOR(7 downto 0);
273         IDXW : out STD_LOGIC_VECTOR(5 downto 0);
274         LDR : out STD_LOGIC;
275         CLR : out STD_LOGIC;
276         Dout : out STD_LOGIC_VECTOR(7 downto 0);
277         ADDR : out STD_LOGIC_VECTOR(13 downto 0);
278         Din : in STD_LOGIC_VECTOR(7 downto 0);
279         ADDW : out STD_LOGIC_VECTOR(13 downto 0));
280     end component;
281
282     component srvError
283      generic(n : integer := 8);
284       port(
285         RST : in STD_LOGIC;
286         CLK : in STD_LOGIC;
287         CLR : in STD_LOGIC;
288         K : in STD_LOGIC_VECTOR(n-1 downto 0);
289         ERL : in STD_LOGIC_VECTOR(n-1 downto 0);
290         ERR : out STD_LOGIC);
291     end component;
292
293     component swtLimitSwitch
294      generic(n : integer := 8);
295       port(
296         RST : in STD_LOGIC;
297         CLK : in STD_LOGIC;
```

```
298         LDR : in STD_LOGIC;
299         CLR : in STD_LOGIC;
300         Din : in STD_LOGIC_VECTOR(n-1 downto 0);
301         Dout : out STD_LOGIC_VECTOR(n-1 downto 0));
302     end component;
303
304     component tbTimer
305       generic(n : integer := 8);
306       port(
307         RST : in STD_LOGIC;
308         CLK : in STD_LOGIC;
309         K : in STD_LOGIC_VECTOR(n-1 downto 0);
310         ENI : in STD_LOGIC;
311         ENO : out STD_LOGIC);
312     end component;
313
314     component pktReg
315      generic(n : integer := 8);
316       port(
317         RST : in STD_LOGIC;
318         CLK : in STD_LOGIC;
319         CLR : in STD_LOGIC;
320         LDR : in STD_LOGIC;
321         Di : in STD_LOGIC_VECTOR(n-1 downto 0);
322         Do : out STD_LOGIC_VECTOR(n-1 downto 0));
323     end component;
324
325     component srvFSM
326       port(
327         RST : in STD_LOGIC;
328         CLK : in STD_LOGIC;
329         EOP : in STD_LOGIC;
330         NPR : in STD_LOGIC;
330         NPR : in STD_LOGIC;
331         ENP : out STD_LOGIC);
332     end component;
333
334     component MC24LC256
335      generic(Fosc : integer := 50000000;
336              Fi2c : integer := 400000);
337       port(
338         RST : in STD_LOGIC;
339         CLK : in STD_LOGIC;
340         CS : in STD_LOGIC;
341         RD : in STD_LOGIC;
342         WR : in STD_LOGIC;
343         RDY : out STD_LOGIC;
344         DEV : in STD_LOGIC_VECTOR(2 downto 0);
345         Din : in STD_LOGIC_VECTOR(63 downto 0);
346         Dout : out STD_LOGIC_VECTOR(63 downto 0);
347         I2C_SCL : out STD_LOGIC;
348         I2C_SDA : inout STD_LOGIC);
349     end component;
350
351      component sramSRAM
352       generic(n : integer := 16;
353                 m : integer := 2);
354       port(
355         RST : in STD_LOGIC;
356         CLK : in STD_LOGIC;
357         CS : in STD_LOGIC;
358         RD : in STD_LOGIC;
359         WR : in STD_LOGIC;
360         RDY : out STD_LOGIC;
361         K : in STD_LOGIC_VECTOR(m-1 downto 0);
362         Din : in STD_LOGIC_VECTOR(n-1 downto 0);
```

```
363           Dout : out STD_LOGIC_VECTOR(n-1 downto 0);
364           RAM_CS: out STD_LOGIC;
365           RAM_OE: out STD_LOGIC;
366           RAM_WE: out STD_LOGIC;
367           RAM_DATA: inout STD_LOGIC_VECTOR(7 downto 0);
368           RAM_ADDR: out STD_LOGIC_VECTOR(18 downto 0));
369       end component;
370
371       signal STR,FS,CLR     : std_logic;
372       signal SYNC           : std_logic_vector(6 downto 0);
373       signal TB             : std_logic_vector(31 downto 0);
374
375       signal JOG,EOP,STP,ENP: std_logic;
376       signal TS,K1,K2,K3    : std_logic_vector(31 downto 0);
377       signal TK,T           : std_logic_vector(32 downto 0);
378       signal Ka1,Ka2,Ka3    : std_logic_vector(63 downto 0);
379       signal Kd1,Kd2,Kd3,Kv : std_logic_vector(63 downto 0);
380
381       signal LDP,LDT        : std_logic;
382       signal PX,PY,PZ,PW,WW : std_logic_vector(31 downto 0);
383       signal X,Y,Z,W        : std_logic_vector(31 downto 0);
384
385       signal DIF,MODE       : std_logic;
386       signal Tsup           : std_logic_vector(7 downto 0);
387       signal CTM            : std_logic_vector(7 downto 0);
388       signal ENCX,ENCY          : std_logic_vector(31 downto 0);
389       signal ENCZ,ENCW      : std_logic_vector(31 downto 0);
390       signal IDXX,IDXY      : std_logic_vector(31 downto 0);
391       signal IDXZ,IDXW      : std_logic_vector(31 downto 0);
392       signal ENCDX,ENCDY    : std_logic_vector(31 downto 0);
393       signal ENCDZ,ENCDW    : std_logic_vector(31 downto 0);
394       signal SENX,SENY      : std_logic_vector(31 downto 0);
395       signal SENZ,SENW      : std_logic_vector(31 downto 0);
396       signal VELX,VELY      : std_logic_vector(31 downto 0);
397       signal VELZ,VELW      : std_logic_vector(31 downto 0);
398
399       signal EX,EY,EZ,EW    : std_logic_vector(34 downto 0);
400
401       signal a0_1,a1_1,a2_1 : std_logic_vector(35 downto 0);
402       signal a0_2,a1_2,a2_2 : std_logic_vector(35 downto 0);
403       signal a0_3,a1_3,a2_3 : std_logic_vector(35 downto 0);
404       signal a0_4,a1_4,a2_4 : std_logic_vector(35 downto 0);
405       signal YX,YY,YZ,YW    : std_logic_vector(15 downto 0);
406       signal UX,UY,UZ,UW    : std_logic_vector(15 downto 0);
407
408       signal DRST           : std_logic;
409       signal ERX,ERY,ERZ,ERW: std_logic;
410       signal SWT            : std_logic_vector(11 downto 0);
411       signal EXL,EYL,EZL,EWL: std_logic_vector(31 downto 0);
412
413       signal CSM,RDM,WRM    : std_logic;
414       signal WRMEN          : std_logic;
415       signal DRI,DRO        : std_logic_vector(255 downto 0);
416
417       signal WR             : std_logic;
418       signal CS,RDY         : std_logic_vector(2 downto 1);
419       signal DRC            : std_logic_vector(15 downto 0);
420       signal SLOT           : std_logic_vector(7 downto 0);
421       signal IDXRD,IDXWR    : std_logic_vector(5 downto 0);
422       signal DI,DO          : std_logic_vector(7 downto 0);
423
424       signal TLDR,TCLR      : std_logic;
425       signal RxD,TxD        : std_logic_vector(7 downto 0);
426       signal ADDR,ADDW      : std_logic_vector(13 downto 0);
427       signal LD             : std_logic_vector(195 downto 0);
428       signal CL,CR          : std_logic_vector(43 downto 0);
```

```
429
430     signal CTRL0          : std_logic_vector(7 downto 0);
431     signal CTRL1          : std_logic_vector(4 downto 0);
432     signal CTRL2          : std_logic_vector(1 downto 0);
433     signal STATUS         : std_logic_vector(7 downto 0);
434
435     signal CSE,RDE,WRE,RDYE   : std_logic;
436     signal EECMD          : std_logic_vector(1 downto 0);
437     signal EEDI,EEDO      : std_logic_vector(63 downto 0);
438
439      signal EOADC             : std_logic;
440      signal Ch0,Ch1,Ch2,Ch3: std_logic_vector(15 downto 0);
441      signal AD0,AD1,AD2,AD3: std_logic_vector(31 downto 0);
442
443      signal CSSM,RDSM,WRSM,RDYSM : std_logic;
444      signal SMMOD          : std_logic_vector(3 downto 0);
445      signal DiSM,DoSM      : std_logic_vector(15 downto 0);
446
447   begin
448   ------------------------------------------------------------------------------------------------------------------
449   -- Controller --
450
451     T(32) <= '0';
452     TK(0) <= '0';
453
454     TimeBase:   tbTimer    generic map (32) port map (RST,CLK,TB,STR,FS);
455
456     Profile:    prfProfile generic map (32,32,32,32,32)
457                            port map (RST,CLK,STP,CLR,JOG,SYNC(0),EOP,TS,Ka1,Ka2,Ka3,Kd1,Kd2,Kd3,Kv,K1,K2,K3,T(31 downto 0));
458
459     Nurbs:      nbsNurbs   generic map (32,0,6,26,2,31)
460                            port map (RST,CLK,CLR,SYNC(0),SYNC(1),LDP,PX,PY,PZ,PW,WW,LDT,TK,T,X,Y,Z,W);
461
462     Encoder_X:  encEncoder generic map (32,8) port map (RST,CLK,CLR,DIF,FS,Tsup,ENCX,IDXX,
463                                ENC_APX,ENC_AMX,ENC_BPX,ENC_BMX,ENC_IDXPX,ENC_IDXMX);
464
465     Encoder_Y:  encEncoder generic map (32,8) port map (RST,CLK,CLR,DIF,FS,Tsup,ENCY,IDXY,
466                                ENC_APY,ENC_AMY,ENC_BPY,ENC_BMY,ENC_IDXPY,ENC_IDXMY);
467
468     Encoder_Z:  encEncoder generic map (32,8) port map (RST,CLK,CLR,DIF,FS,Tsup,ENCZ,IDXZ,
469                                ENC_APZ,ENC_AMZ,ENC_BPZ,ENC_BMZ,ENC_IDXPZ,ENC_IDXMZ);
470
471     Encoder_W:  encEncoder generic map (32,8) port map (RST,CLK,CLR,DIF,FS,Tsup,ENCW,IDXW,
472                                ENC_APW,ENC_AMW,ENC_BPW,ENC_BMW,ENC_IDXPW,ENC_IDXMW);
473
474      RegEncDX:   pktReg generic map (32) port map (RST,CLK,CLR,FS,ENCX,ENCDX);
475      RegEncDY:   pktReg generic map (32) port map (RST,CLK,CLR,FS,ENCY,ENCDY);
476      RegEncDZ:   pktReg generic map (32) port map (RST,CLK,CLR,FS,ENCZ,ENCDZ);
477      RegEncDW:   pktReg generic map (32) port map (RST,CLK,CLR,FS,ENCW,ENCDW);
478
479      VELX <= std_logic_vector(unsigned(ENCX)-unsigned(ENCDX));
480      VELY <= std_logic_vector(unsigned(ENCY)-unsigned(ENCDY));
481      VELZ <= std_logic_vector(unsigned(ENCZ)-unsigned(ENCDZ));
482      VELW <= std_logic_vector(unsigned(ENCW)-unsigned(ENCDW));
483
484      AD0(31 downto 16) <= (others => Ch0(15)); AD0(15 downto 0) <= Ch0;
485      AD1(31 downto 16) <= (others => Ch1(15)); AD1(15 downto 0) <= Ch1;
486      AD2(31 downto 16) <= (others => Ch2(15)); AD2(15 downto 0) <= Ch2;
487      AD3(31 downto 16) <= (others => Ch3(15)); AD3(15 downto 0) <= Ch3;
488
489      SENX <= VELX when CTM(0)='1' else AD0 when CTM(4)='1' else ENCX;
490      SENY <= VELY when CTM(1)='1' else AD1 when CTM(5)='1' else ENCY;
491      SENZ <= VELZ when CTM(2)='1' else AD2 when CTM(6)='1' else ENCZ;
492      SENW <= VELW when CTM(3)='1' else AD3 when CTM(7)='1' else ENCW;
493
494     EX <= std_logic_vector(unsigned(X)-unsigned(SENX)) & "000";
495     EY <= std_logic_vector(unsigned(Y)-unsigned(SENY)) & "000";
496     EZ <= std_logic_vector(unsigned(Z)-unsigned(SENZ)) & "000";
497     EW <= std_logic_vector(unsigned(W)-unsigned(SENW)) & "000";
498
499     PID:        pidPID_4D  generic map (32,3,16,0,18,18)
500                            port map (RST,CLK,FS,SYNC(2),CLR,
501                                      EX,YX,a0_1,a1_1,a2_1,EY,YY,a0_2,a1_2,a2_2,
502                                      EZ,YZ,a0_3,a1_3,a2_3,EW,YW,a0_4,a1_4,a2_4);
503
504      UX <= DoSM when SMMOD(0)='1' else YX;
505      UY <= DoSM when SMMOD(1)='1' else YY;
506      UZ <= DoSM when SMMOD(2)='1' else YZ;
507      UW <= DoSM when SMMOD(3)='1' else YW;
508
509     DAC:        DAC7565    port map (RST,CLK,SYNC(2),SYNC(3),DRST,UX,UY,UZ,UW,
510                            DAC_SCLK,DAC_SYNC,DAC_DIN,DAC_LDAC,DAC_RST);
511
512     Limit:      swtLimitSwitch generic map (12) port map (RST,CLK,FS,CLR,SW_LIM,SWT);
513     LimEX:      srvError generic map (32) port map (RST,CLK,CLR,EXL,EX(34 downto 3),ERX);
514     LimEY:      srvError generic map (32) port map (RST,CLK,CLR,EYL,EY(34 downto 3),ERY);
515     LimEZ:      srvError generic map (32) port map (RST,CLK,CLR,EZL,EZ(34 downto 3),ERZ);
516     LimEW:      srvError generic map (32) port map (RST,CLK,CLR,EWL,EW(34 downto 3),ERW);
517
518      ADC:            ADS7841 port map (RST,CLK,FS,EOADC,Ch0,Ch1,Ch2,Ch3,
519                                        ADC_DCLK,ADC_DIN,ADC_BUSY,ADC_DOUT);
520
521     Control:    srvFSM port map (RST,CLK,EOP,LDP,ENP);
```

```
522
523   ----------------------------------------------------------------------------------------------------------------
524   -- Data storage --
525
526      DRI( 15 downto   0) <= X(15 downto  0) when MODE='0' else UX;
527      DRI( 31 downto  16) <= X(31 downto 16) when MODE='0' else Ch0;
528      DRI( 63 downto  32) <= SENX;
529      DRI( 79 downto  64) <= Y(15 downto  0) when MODE='0' else UY;
530      DRI( 95 downto  80) <= Y(31 downto 16) when MODE='0' else Ch1;
531      DRI(127 downto  96) <= SENY;
532      DRI(143 downto 128) <= Z(15 downto  0) when MODE='0' else UZ;
533      DRI(159 downto 144) <= Z(31 downto 16) when MODE='0' else Ch2;
534      DRI(191 downto 160) <= SENZ;
535      DRI(207 downto 192) <= W(15 downto  0) when MODE='0' else UW;
536      DRI(223 downto 208) <= W(31 downto 16) when MODE='0' else Ch3;
537      DRI(255 downto 224) <= SENW;
538
539       WRM <= '0' when RST='1' else SYNC(2) AND WRMEN when rising_edge(CLK);
540
541      Memory:     dramDRAM generic map (256,4)
542                        port map (RST,CLK,CSM,WRM,RDM,SYNC(4),"1111",DRI,DRC,
543                        DRAM_CLK,DRAM_CKE,DRAM_CS,DRAM_WE,DRAM_CAS,DRAM_RAS,DRAM_DQML,DRAM_DQMH,
544                        DRAM_BA,DRAM_A,DRAM_DQ);
545
546      EEPROM:     MC24LC256 generic map (48000000,400000)
547                        port map (RST,CLK,CSE,RDE,WRE,RDYE,"000",EEDI,EEDO,I2C_SCL,I2C_SDA);
548
549       RDSM <= WRM;
550       SRAM:           sramSRAM generic map (16,2)
551                              port map (RST,CLK,CSSM,RDSM,WRSM,RDYSM,"01",DiSM,DoSM,
552                              RAM_CS,RAM_OE,RAM_WE,RAM_DATA,RAM_ADDR);
553
554   ----------------------------------------------------------------------------------------------------------------
555   -- Communication interface ---
556
557      DRC  <= "0000000000000001";
558      SLOT <= "00000" & DRC_SLOT;
559
560      STR  <= CTRL0(0) AND NOT(ERX OR ERY OR ERZ OR ERW);
561      CLR  <= CTRL0(1);
562      STP  <= CTRL0(2) AND SYNC(2) AND ENP;
563      DRST <= CTRL0(3) OR (ERX OR ERY OR ERZ OR ERW);
564      CSM  <= CTRL0(4);
565      DIF  <= CTRL0(6);
566      MODE <= CTRL0(7);
567      LDF  <= CTRL1(0);
568      LDT  <= CTRL1(1);
569      JOG  <= CTRL1(2);
570      RDM  <= CTRL1(3);
571
572      CSE  <= CTRL0(5);
573      RDE  <= EECMD(0);
574      WRE  <= EECMD(1);
575
576       WRMEN<= CTRL2(0);
577
578       CSSM <= CTRL2(1);
579       WRSM <= CTRL1(4);
580
581      STATUS <= ERZ & ERY & ERX & NOT ENP & DRST & STP & CLR & STR;
582
583      USB:        usbUSB_v6_0 generic map (9731,9217) port map (RST,CLK,SLOT,USB_DP,USB_DM,CS,RDY,IDXRD,DI,WR,IDXWR,DO);
584
585      Terminal:   pktTerminal port map (RST,CLK,DRC,SLOT,RDY(1),CS(1),CS(2),RDY(2),DI,IDXRD,WR,DO,IDXWR,
586                                        TLDR,TCLR,RxD,ADDR,TxD,ADDW);
587
588      CR <= (others => '0') when RST='1' else CL when rising_edge(CLK);
589
590      RegCTRL0:  pktReg generic map (8) port map (RST,CLK,CR(0),LD(0),RxD,CTRL0);
591      RegCTRL1:  pktReg generic map (5) port map (RST,CLK,CR(1),LD(1),RxD(4 downto 0),CTRL1);
592      RegTB_0:   pktReg generic map (8) port map (RST,CLK,CR(2),LD(2),RxD,TB( 7 downto  0));
593      RegTB_1:   pktReg generic map (8) port map (RST,CLK,CR(2),LD(3),RxD,TB(15 downto  8));
594      RegTB_2:   pktReg generic map (8) port map (RST,CLK,CR(2),LD(4),RxD,TB(23 downto 16));
595      RegTB_3:   pktReg generic map (8) port map (RST,CLK,CR(2),LD(5),RxD,TB(31 downto 24));
596      RegTS_0:   pktReg generic map (8) port map (RST,CLK,CR(3),LD(6),RxD,TS( 7 downto  0));
597      RegTS_1:   pktReg generic map (8) port map (RST,CLK,CR(3),LD(7),RxD,TS(15 downto  8));
598      RegTS_2:   pktReg generic map (8) port map (RST,CLK,CR(3),LD(8),RxD,TS(23 downto 16));
599      RegTS_3:   pktReg generic map (8) port map (RST,CLK,CR(3),LD(9),RxD,TS(31 downto 24));
600      RegK1_0:   pktReg generic map (8) port map (RST,CLK,CR(4),LD(10),RxD,K1( 7 downto  0));
601      RegK1_1:   pktReg generic map (8) port map (RST,CLK,CR(4),LD(11),RxD,K1(15 downto  8));
602      RegK1_2:   pktReg generic map (8) port map (RST,CLK,CR(4),LD(12),RxD,K1(23 downto 16));
603      RegK1_3:   pktReg generic map (8) port map (RST,CLK,CR(4),LD(13),RxD,K1(31 downto 24));
604      RegK2_0:   pktReg generic map (8) port map (RST,CLK,CR(5),LD(14),RxD,K2( 7 downto  0));
605      RegK2_1:   pktReg generic map (8) port map (RST,CLK,CR(5),LD(15),RxD,K2(15 downto  8));
606      RegK2_2:   pktReg generic map (8) port map (RST,CLK,CR(5),LD(16),RxD,K2(23 downto 16));
607      RegK2_3:   pktReg generic map (8) port map (RST,CLK,CR(5),LD(17),RxD,K2(31 downto 24));
608      RegK3_0:   pktReg generic map (8) port map (RST,CLK,CR(6),LD(18),RxD,K3( 7 downto  0));
609      RegK3_1:   pktReg generic map (8) port map (RST,CLK,CR(6),LD(19),RxD,K3(15 downto  8));
610      RegK3_2:   pktReg generic map (8) port map (RST,CLK,CR(6),LD(20),RxD,K3(23 downto 16));
611      RegK3_3:   pktReg generic map (8) port map (RST,CLK,CR(6),LD(21),RxD,K3(31 downto 24));
612      RegKa1_0:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(22),RxD,Ka1( 7 downto  0));
613      RegKa1_1:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(23),RxD,Ka1(15 downto  8));
614      RegKa1_2:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(24),RxD,Ka1(23 downto 16));
615      RegKa1_3:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(25),RxD,Ka1(31 downto 24));
616      RegKa1_4:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(26),RxD,Ka1(39 downto 32));
617      RegKa1_5:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(27),RxD,Ka1(47 downto 40));
618      RegKa1_6:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(28),RxD,Ka1(55 downto 48));
619      RegKa1_7:  pktReg generic map (8) port map (RST,CLK,CR(7),LD(29),RxD,Ka1(63 downto 56));
```

```
620     RegKa2_0:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(30),RxD,Ka2( 7 downto  0));
621     RegKa2_1:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(31),RxD,Ka2(15 downto  8));
622     RegKa2_2:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(32),RxD,Ka2(23 downto 16));
623     RegKa2_3:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(33),RxD,Ka2(31 downto 24));
624     RegKa2_4:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(34),RxD,Ka2(39 downto 32));
625     RegKa2_5:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(35),RxD,Ka2(47 downto 40));
626     RegKa2_6:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(36),RxD,Ka2(55 downto 48));
627     RegKa2_7:  pktReg generic map (8) port map (RST,CLK,CR(8),LD(37),RxD,Ka2(63 downto 56));
628     RegKa3_0:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(38),RxD,Ka3( 7 downto  0));
629     RegKa3_1:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(39),RxD,Ka3(15 downto  8));
630     RegKa3_2:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(40),RxD,Ka3(23 downto 16));
631     RegKa3_3:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(41),RxD,Ka3(31 downto 24));
632     RegKa3_4:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(42),RxD,Ka3(39 downto 32));
633     RegKa3_5:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(43),RxD,Ka3(47 downto 40));
634     RegKa3_6:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(44),RxD,Ka3(55 downto 48));
635     RegKa3_7:  pktReg generic map (8) port map (RST,CLK,CR(9),LD(45),RxD,Ka3(63 downto 56));
636     RegKd1_0:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(46),RxD,Kd1( 7 downto  0));
637     RegKd1_1:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(47),RxD,Kd1(15 downto  8));
638     RegKd1_2:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(48),RxD,Kd1(23 downto 16));
639     RegKd1_3:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(49),RxD,Kd1(31 downto 24));
640     RegKd1_4:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(50),RxD,Kd1(39 downto 32));
641     RegKd1_5:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(51),RxD,Kd1(47 downto 40));
642     RegKd1_6:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(52),RxD,Kd1(55 downto 48));
643     RegKd1_7:  pktReg generic map (8) port map (RST,CLK,CR(10),LD(53),RxD,Kd1(63 downto 56));
644     RegKd2_0:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(54),RxD,Kd2( 7 downto  0));
645     RegKd2_1:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(55),RxD,Kd2(15 downto  8));
646     RegKd2_2:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(56),RxD,Kd2(23 downto 16));
647     RegKd2_3:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(57),RxD,Kd2(31 downto 24));
648     RegKd2_4:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(58),RxD,Kd2(39 downto 32));
649     RegKd2_5:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(59),RxD,Kd2(47 downto 40));
650     RegKd2_6:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(60),RxD,Kd2(55 downto 48));
651     RegKd2_7:  pktReg generic map (8) port map (RST,CLK,CR(11),LD(61),RxD,Kd2(63 downto 56));
652     RegKd3_0:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(62),RxD,Kd3( 7 downto  0));
653     RegKd3_1:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(63),RxD,Kd3(15 downto  8));
654     RegKd3_2:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(64),RxD,Kd3(23 downto 16));
655     RegKd3_3:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(65),RxD,Kd3(31 downto 24));
656     RegKd3_4:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(66),RxD,Kd3(39 downto 32));
657     RegKd3_5:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(67),RxD,Kd3(47 downto 40));
658     RegKd3_6:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(68),RxD,Kd3(55 downto 48));
659     RegKd3_7:  pktReg generic map (8) port map (RST,CLK,CR(12),LD(69),RxD,Kd3(63 downto 56));
660     RegKv_0:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(70),RxD,Kv( 7 downto  0));
661     RegKv_1:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(71),RxD,Kv(15 downto  8));
662     RegKv_2:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(72),RxD,Kv(23 downto 16));
663     RegKv_3:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(73),RxD,Kv(31 downto 24));
664     RegKv_4:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(74),RxD,Kv(39 downto 32));
665     RegKv_5:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(75),RxD,Kv(47 downto 40));
666     RegKv_6:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(76),RxD,Kv(55 downto 48));
667     RegKv_7:   pktReg generic map (8) port map (RST,CLK,CR(13),LD(77),RxD,Kv(63 downto 56));
668     RegPX_0:   pktReg generic map (8) port map (RST,CLK,CR(14),LD(78),RxD,PX( 7 downto  0));
669     RegPX_1:   pktReg generic map (8) port map (RST,CLK,CR(14),LD(79),RxD,PX(15 downto  8));
670     RegPX_2:   pktReg generic map (8) port map (RST,CLK,CR(14),LD(80),RxD,PX(23 downto 16));
671     RegPX_3:   pktReg generic map (8) port map (RST,CLK,CR(14),LD(81),RxD,PX(31 downto 24));
672     RegPY_0:   pktReg generic map (8) port map (RST,CLK,CR(15),LD(82),RxD,PY( 7 downto  0));
673     RegPY_1:   pktReg generic map (8) port map (RST,CLK,CR(15),LD(83),RxD,PY(15 downto  8));
674     RegPY_2:   pktReg generic map (8) port map (RST,CLK,CR(15),LD(84),RxD,PY(23 downto 16));
675     RegPY_3:   pktReg generic map (8) port map (RST,CLK,CR(15),LD(85),RxD,PY(31 downto 24));
676     RegPZ_0:   pktReg generic map (8) port map (RST,CLK,CR(16),LD(86),RxD,PZ( 7 downto  0));
677     RegPZ_1:   pktReg generic map (8) port map (RST,CLK,CR(16),LD(87),RxD,PZ(15 downto  8));
678     RegPZ_2:   pktReg generic map (8) port map (RST,CLK,CR(16),LD(88),RxD,PZ(23 downto 16));
679     RegPZ_3:   pktReg generic map (8) port map (RST,CLK,CR(16),LD(89),RxD,PZ(31 downto 24));
680     RegPW_0:   pktReg generic map (8) port map (RST,CLK,CR(17),LD(90),RxD,PW( 7 downto  0));
681     RegPW_1:   pktReg generic map (8) port map (RST,CLK,CR(17),LD(91),RxD,PW(15 downto  8));
682     RegPW_2:   pktReg generic map (8) port map (RST,CLK,CR(17),LD(92),RxD,PW(23 downto 16));
683     RegPW_3:   pktReg generic map (8) port map (RST,CLK,CR(17),LD(93),RxD,PW(31 downto 24));
684     RegTK_0:   pktReg generic map (8) port map (RST,CLK,CR(18),LD(94),RxD,TK( 8 downto  1));
685     RegTK_1:   pktReg generic map (8) port map (RST,CLK,CR(18),LD(95),RxD,TK(16 downto  9));
```

```
686     RegTK_2:   pktReg generic map (8) port map (RST,CLK,CR(18),LD(96),RxD,TK(24 downto 17));
687     RegTK_3:   pktReg generic map (8) port map (RST,CLK,CR(18),LD(97),RxD,TK(32 downto 25));
688     RegA01_0:  pktReg generic map (8) port map (RST,CLK,CR(19),LD(98),RxD,a0_1( 7 downto  0));
689     RegA01_1:  pktReg generic map (8) port map (RST,CLK,CR(19),LD(99),RxD,a0_1(15 downto  8));
690     RegA01_2:  pktReg generic map (8) port map (RST,CLK,CR(19),LD(100),RxD,a0_1(23 downto 16));
691     RegA01_3:  pktReg generic map (8) port map (RST,CLK,CR(19),LD(101),RxD,a0_1(31 downto 24));
692     RegA01_4:  pktReg generic map (4) port map (RST,CLK,CR(19),LD(102),RxD(3 downto 0),a0_1(35 downto 32));
693     RegA11_0:  pktReg generic map (8) port map (RST,CLK,CR(20),LD(103),RxD,a1_1( 7 downto  0));
694     RegA11_1:  pktReg generic map (8) port map (RST,CLK,CR(20),LD(104),RxD,a1_1(15 downto  8));
695     RegA11_2:  pktReg generic map (8) port map (RST,CLK,CR(20),LD(105),RxD,a1_1(23 downto 16));
696     RegA11_3:  pktReg generic map (8) port map (RST,CLK,CR(20),LD(106),RxD,a1_1(31 downto 24));
697     RegA11_4:  pktReg generic map (4) port map (RST,CLK,CR(20),LD(107),RxD(3 downto 0),a1_1(35 downto 32));
698     RegA21_0:  pktReg generic map (8) port map (RST,CLK,CR(21),LD(108),RxD,a2_1( 7 downto  0));
699     RegA21_1:  pktReg generic map (8) port map (RST,CLK,CR(21),LD(109),RxD,a2_1(15 downto  8));
700     RegA21_2:  pktReg generic map (8) port map (RST,CLK,CR(21),LD(110),RxD,a2_1(23 downto 16));
701     RegA21_3:  pktReg generic map (8) port map (RST,CLK,CR(21),LD(111),RxD,a2_1(31 downto 24));
702     RegA21_4:  pktReg generic map (4) port map (RST,CLK,CR(21),LD(112),RxD(3 downto 0),a2_1(35 downto 32));
703     RegA02_0:  pktReg generic map (8) port map (RST,CLK,CR(22),LD(113),RxD,a0_2( 7 downto  0));
704     RegA02_1:  pktReg generic map (8) port map (RST,CLK,CR(22),LD(114),RxD,a0_2(15 downto  8));
705     RegA02_2:  pktReg generic map (8) port map (RST,CLK,CR(22),LD(115),RxD,a0_2(23 downto 16));
706     RegA02_3:  pktReg generic map (8) port map (RST,CLK,CR(22),LD(116),RxD,a0_2(31 downto 24));
707     RegA02_4:  pktReg generic map (4) port map (RST,CLK,CR(22),LD(117),RxD(3 downto 0),a0_2(35 downto 32));
708     RegA12_0:  pktReg generic map (8) port map (RST,CLK,CR(23),LD(118),RxD,a1_2( 7 downto  0));
709     RegA12_1:  pktReg generic map (8) port map (RST,CLK,CR(23),LD(119),RxD,a1_2(15 downto  8));
710     RegA12_2:  pktReg generic map (8) port map (RST,CLK,CR(23),LD(120),RxD,a1_2(23 downto 16));
711     RegA12_3:  pktReg generic map (8) port map (RST,CLK,CR(23),LD(121),RxD,a1_2(31 downto 24));
712     RegA12_4:  pktReg generic map (4) port map (RST,CLK,CR(23),LD(122),RxD(3 downto 0),a1_2(35 downto 32));
713     RegA22_0:  pktReg generic map (8) port map (RST,CLK,CR(24),LD(123),RxD,a2_2( 7 downto  0));
714     RegA22_1:  pktReg generic map (8) port map (RST,CLK,CR(24),LD(124),RxD,a2_2(15 downto  8));
715     RegA22_2:  pktReg generic map (8) port map (RST,CLK,CR(24),LD(125),RxD,a2_2(23 downto 16));
716     RegA22_3:  pktReg generic map (8) port map (RST,CLK,CR(24),LD(126),RxD,a2_2(31 downto 24));
717     RegA22_4:  pktReg generic map (4) port map (RST,CLK,CR(24),LD(127),RxD(3 downto 0),a2_2(35 downto 32));
718     RegA03_0:  pktReg generic map (8) port map (RST,CLK,CR(25),LD(128),RxD,a0_3( 7 downto  0));
719     RegA03_1:  pktReg generic map (8) port map (RST,CLK,CR(25),LD(129),RxD,a0_3(15 downto  8));
720     RegA03_2:  pktReg generic map (8) port map (RST,CLK,CR(25),LD(130),RxD,a0_3(23 downto 16));
721     RegA03_3:  pktReg generic map (8) port map (RST,CLK,CR(25),LD(131),RxD,a0_3(31 downto 24));
722     RegA03_4:  pktReg generic map (4) port map (RST,CLK,CR(25),LD(132),RxD(3 downto 0),a0_3(35 downto 32));
723     RegA13_0:  pktReg generic map (8) port map (RST,CLK,CR(26),LD(133),RxD,a1_3( 7 downto  0));
724     RegA13_1:  pktReg generic map (8) port map (RST,CLK,CR(26),LD(134),RxD,a1_3(15 downto  8));
725     RegA13_2:  pktReg generic map (8) port map (RST,CLK,CR(26),LD(135),RxD,a1_3(23 downto 16));
726     RegA13_3:  pktReg generic map (8) port map (RST,CLK,CR(26),LD(136),RxD,a1_3(31 downto 24));
727     RegA13_4:  pktReg generic map (4) port map (RST,CLK,CR(26),LD(137),RxD(3 downto 0),a1_3(35 downto 32));
728     RegA23_0:  pktReg generic map (8) port map (RST,CLK,CR(27),LD(138),RxD,a2_3( 7 downto  0));
729     RegA23_1:  pktReg generic map (8) port map (RST,CLK,CR(27),LD(139),RxD,a2_3(15 downto  8));
730     RegA23_2:  pktReg generic map (8) port map (RST,CLK,CR(27),LD(140),RxD,a2_3(23 downto 16));
731     RegA23_3:  pktReg generic map (8) port map (RST,CLK,CR(27),LD(141),RxD,a2_3(31 downto 24));
732     RegA23_4:  pktReg generic map (4) port map (RST,CLK,CR(27),LD(142),RxD(3 downto 0),a2_3(35 downto 32));
733     RegA04_0:  pktReg generic map (8) port map (RST,CLK,CR(28),LD(143),RxD,a0_4( 7 downto  0));
734     RegA04_1:  pktReg generic map (8) port map (RST,CLK,CR(28),LD(144),RxD,a0_4(15 downto  8));
735     RegA04_2:  pktReg generic map (8) port map (RST,CLK,CR(28),LD(145),RxD,a0_4(23 downto 16));
736     RegA04_3:  pktReg generic map (8) port map (RST,CLK,CR(28),LD(146),RxD,a0_4(31 downto 24));
737     RegA04_4:  pktReg generic map (4) port map (RST,CLK,CR(28),LD(147),RxD(3 downto 0),a0_4(35 downto 32));
738     RegA14_0:  pktReg generic map (8) port map (RST,CLK,CR(29),LD(148),RxD,a1_4( 7 downto  0));
739     RegA14_1:  pktReg generic map (8) port map (RST,CLK,CR(29),LD(149),RxD,a1_4(15 downto  8));
740     RegA14_2:  pktReg generic map (8) port map (RST,CLK,CR(29),LD(150),RxD,a1_4(23 downto 16));
741     RegA14_3:  pktReg generic map (8) port map (RST,CLK,CR(29),LD(151),RxD,a1_4(31 downto 24));
742     RegA14_4:  pktReg generic map (4) port map (RST,CLK,CR(29),LD(152),RxD(3 downto 0),a1_4(35 downto 32));
743     RegA24_0:  pktReg generic map (8) port map (RST,CLK,CR(30),LD(153),RxD,a2_4( 7 downto  0));
744     RegA24_1:  pktReg generic map (8) port map (RST,CLK,CR(30),LD(154),RxD,a2_4(15 downto  8));
745     RegA24_2:  pktReg generic map (8) port map (RST,CLK,CR(30),LD(155),RxD,a2_4(23 downto 16));
746     RegA24_3:  pktReg generic map (8) port map (RST,CLK,CR(30),LD(156),RxD,a2_4(31 downto 24));
747     RegA24_4:  pktReg generic map (4) port map (RST,CLK,CR(30),LD(157),RxD(3 downto 0),a2_4(35 downto 32));
748     RegEXL_0:  pktReg generic map (8) port map (RST,CLK,CR(31),LD(160),RxD,EXL( 7 downto  0));
749     RegEXL_1:  pktReg generic map (8) port map (RST,CLK,CR(31),LD(161),RxD,EXL(15 downto  8));
750     RegEXL_2:  pktReg generic map (8) port map (RST,CLK,CR(31),LD(162),RxD,EXL(23 downto 16));
751     RegEXL_3:  pktReg generic map (8) port map (RST,CLK,CR(31),LD(163),RxD,EXL(31 downto 24));
```

```
785
786       RegSMMOD:  pktReg generic map (4) port map (RST,CLK,CR(42),LD(193),RxD(3 downto 0),SMMOD);
787       RegSMDI_0: pktReg generic map (8) port map (RST,CLK,CR(43),LD(194),RxD,DiSM( 7 downto 0));
788       RegSMDI_1: pktReg generic map (8) port map (RST,CLK,CR(43),LD(195),RxD,DiSM(15 downto 8));
789
790       Demux: process(ADDR,TLDR,TCLR)
791       begin
792          LD <= (others => '0');
793          CL <= (others => '0');
794          case ADDR is
795             when "00000000000000" => LD(0) <= TLDR; CL(0) <= TCLR;   -- [0] CTRL0
796             when "00000001000000" => LD(1) <= TLDR; CL(1) <= TCLR;   -- [1] CTRL1
797             when "00000010000000" => LD(2) <= TLDR; CL(2) <= TCLR;   -- [2] TB
798             when "00000010000001" => LD(3) <= TLDR; CL(2) <= TCLR;
799             when "00000010000010" => LD(4) <= TLDR; CL(2) <= TCLR;
800             when "00000010000011" => LD(5) <= TLDR; CL(2) <= TCLR;
801             when "00000011000000" => LD(6) <= TLDR; CL(3) <= TCLR;   -- [3] TS
802             when "00000011000001" => LD(7) <= TLDR; CL(3) <= TCLR;
803             when "00000011000010" => LD(8) <= TLDR; CL(3) <= TCLR;
804             when "00000011000011" => LD(9) <= TLDR; CL(3) <= TCLR;
805             when "00000100000000" => LD(10) <= TLDR; CL(4) <= TCLR;  -- [4] K1
806             when "00000100000001" => LD(11) <= TLDR; CL(4) <= TCLR;
807             when "00000100000010" => LD(12) <= TLDR; CL(4) <= TCLR;
808             when "00000100000011" => LD(13) <= TLDR; CL(4) <= TCLR;
809             when "00000101000000" => LD(14) <= TLDR; CL(5) <= TCLR;  -- [5] K2
810             when "00000101000001" => LD(15) <= TLDR; CL(5) <= TCLR;
811             when "00000101000010" => LD(16) <= TLDR; CL(5) <= TCLR;
812             when "00000101000011" => LD(17) <= TLDR; CL(5) <= TCLR;
813             when "00000110000000" => LD(18) <= TLDR; CL(6) <= TCLR;  -- [6] K3
814             when "00000110000001" => LD(19) <= TLDR; CL(6) <= TCLR;
815             when "00000110000010" => LD(20) <= TLDR; CL(6) <= TCLR;
816             when "00000110000011" => LD(21) <= TLDR; CL(6) <= TCLR;
817             when "00000111000000" => LD(22) <= TLDR; CL(7) <= TCLR;  -- [7] Ka1
818             when "00000111000001" => LD(23) <= TLDR; CL(7) <= TCLR;
819             when "00000111000010" => LD(24) <= TLDR; CL(7) <= TCLR;
820             when "00000111000011" => LD(25) <= TLDR; CL(7) <= TCLR;
821             when "00000111000100" => LD(26) <= TLDR; CL(7) <= TCLR;
822             when "00000111000101" => LD(27) <= TLDR; CL(7) <= TCLR;
823             when "00000111000110" => LD(28) <= TLDR; CL(7) <= TCLR;
824             when "00000111000111" => LD(29) <= TLDR; CL(7) <= TCLR;
825             when "00001000000000" => LD(30) <= TLDR; CL(8) <= TCLR;  -- [8] Ka2
826             when "00001000000001" => LD(31) <= TLDR; CL(8) <= TCLR;
827             when "00001000000010" => LD(32) <= TLDR; CL(8) <= TCLR;
828             when "00001000000011" => LD(33) <= TLDR; CL(8) <= TCLR;
829             when "00001000000100" => LD(34) <= TLDR; CL(8) <= TCLR;
830             when "00001000000101" => LD(35) <= TLDR; CL(8) <= TCLR;
831             when "00001000000110" => LD(36) <= TLDR; CL(8) <= TCLR;
832             when "00001000000111" => LD(37) <= TLDR; CL(8) <= TCLR;
833             when "00001001000000" => LD(38) <= TLDR; CL(9) <= TCLR;  -- [9] Ka3
834             when "00001001000001" => LD(39) <= TLDR; CL(9) <= TCLR;
835             when "00001001000010" => LD(40) <= TLDR; CL(9) <= TCLR;
836             when "00001001000011" => LD(41) <= TLDR; CL(9) <= TCLR;
837             when "00001001000100" => LD(42) <= TLDR; CL(9) <= TCLR;
838             when "00001001000101" => LD(43) <= TLDR; CL(9) <= TCLR;
839             when "00001001000110" => LD(44) <= TLDR; CL(9) <= TCLR;
840             when "00001001000111" => LD(45) <= TLDR; CL(9) <= TCLR;
841             when "00001010000000" => LD(46) <= TLDR; CL(10) <= TCLR;  -- [10] Kd1
842             when "00001010000001" => LD(47) <= TLDR; CL(10) <= TCLR;
843             when "00001010000010" => LD(48) <= TLDR; CL(10) <= TCLR;
844             when "00001010000011" => LD(49) <= TLDR; CL(10) <= TCLR;
845             when "00001010000100" => LD(50) <= TLDR; CL(10) <= TCLR;
846             when "00001010000101" => LD(51) <= TLDR; CL(10) <= TCLR;
847             when "00001010000110" => LD(52) <= TLDR; CL(10) <= TCLR;
848             when "00001010000111" => LD(53) <= TLDR; CL(10) <= TCLR;
849             when "00001011000000" => LD(54) <= TLDR; CL(11) <= TCLR;  -- [11] Kd2
850             when "00001011000001" => LD(55) <= TLDR; CL(11) <= TCLR;
```

```
851         when "00001011000010" => LD(56) <= TLDR; CL(11) <= TCLR;
852         when "00001011000011" => LD(57) <= TLDR; CL(11) <= TCLR;
853         when "00001011000100" => LD(58) <= TLDR; CL(11) <= TCLR;
854         when "00001011000101" => LD(59) <= TLDR; CL(11) <= TCLR;
855         when "00001011000110" => LD(60) <= TLDR; CL(11) <= TCLR;
856         when "00001011000111" => LD(61) <= TLDR; CL(11) <= TCLR;
857         when "00001100000000" => LD(62) <= TLDR; CL(12) <= TCLR;  -- [12] Kd3
858         when "00001100000001" => LD(63) <= TLDR; CL(12) <= TCLR;
859         when "00001100000010" => LD(64) <= TLDR; CL(12) <= TCLR;
860         when "00001100000011" => LD(65) <= TLDR; CL(12) <= TCLR;
861         when "00001100000100" => LD(66) <= TLDR; CL(12) <= TCLR;
862         when "00001100000101" => LD(67) <= TLDR; CL(12) <= TCLR;
863         when "00001100000110" => LD(68) <= TLDR; CL(12) <= TCLR;
864         when "00001100000111" => LD(69) <= TLDR; CL(12) <= TCLR;
865         when "00001101000000" => LD(70) <= TLDR; CL(13) <= TCLR;  -- [13] Kv
866         when "00001101000001" => LD(71) <= TLDR; CL(13) <= TCLR;
867         when "00001101000010" => LD(72) <= TLDR; CL(13) <= TCLR;
868         when "00001101000011" => LD(73) <= TLDR; CL(13) <= TCLR;
869         when "00001101000100" => LD(74) <= TLDR; CL(13) <= TCLR;
870         when "00001101000101" => LD(75) <= TLDR; CL(13) <= TCLR;
871         when "00001101000110" => LD(76) <= TLDR; CL(13) <= TCLR;
872         when "00001101000111" => LD(77) <= TLDR; CL(13) <= TCLR;
873         when "00001110000000" => LD(78) <= TLDR; CL(14) <= TCLR;  -- [14] PX
874         when "00001110000001" => LD(79) <= TLDR; CL(14) <= TCLR;
875         when "00001110000010" => LD(80) <= TLDR; CL(14) <= TCLR;
876         when "00001110000011" => LD(81) <= TLDR; CL(14) <= TCLR;
877         when "00001111000000" => LD(82) <= TLDR; CL(15) <= TCLR;  -- [15] PY
878         when "00001111000001" => LD(83) <= TLDR; CL(15) <= TCLR;
879         when "00001111000010" => LD(84) <= TLDR; CL(15) <= TCLR;
880         when "00001111000011" => LD(85) <= TLDR; CL(15) <= TCLR;
881         when "00010000000000" => LD(86) <= TLDR; CL(16) <= TCLR;  -- [16] PZ
882         when "00010000000001" => LD(87) <= TLDR; CL(16) <= TCLR;
883         when "00010000000010" => LD(88) <= TLDR; CL(16) <= TCLR;
884         when "00010000000011" => LD(89) <= TLDR; CL(16) <= TCLR;
885         when "00010001000000" => LD(90) <= TLDR; CL(17) <= TCLR;  -- [17] PW
886         when "00010001000001" => LD(91) <= TLDR; CL(17) <= TCLR;
887         when "00010001000010" => LD(92) <= TLDR; CL(17) <= TCLR;
888         when "00010001000011" => LD(93) <= TLDR; CL(17) <= TCLR;
889         when "00010010000000" => LD(94) <= TLDR; CL(18) <= TCLR;  -- [18] TK
890         when "00010010000001" => LD(95) <= TLDR; CL(18) <= TCLR;
891         when "00010010000010" => LD(96) <= TLDR; CL(18) <= TCLR;
892         when "00010010000011" => LD(97) <= TLDR; CL(18) <= TCLR;
893         when "00010011000000" => LD(98) <= TLDR; CL(19) <= TCLR;  -- [19] a0_1
894         when "00010011000001" => LD(99) <= TLDR; CL(19) <= TCLR;
895         when "00010011000010" => LD(100) <= TLDR; CL(19) <= TCLR;
896         when "00010011000011" => LD(101) <= TLDR; CL(19) <= TCLR;
897         when "00010011000100" => LD(102) <= TLDR; CL(19) <= TCLR;
898         when "00010100000000" => LD(103) <= TLDR; CL(20) <= TCLR;  -- [20] a1_1
899         when "00010100000001" => LD(104) <= TLDR; CL(20) <= TCLR;
900         when "00010100000010" => LD(105) <= TLDR; CL(20) <= TCLR;
901         when "00010100000011" => LD(106) <= TLDR; CL(20) <= TCLR;
902         when "00010100000100" => LD(107) <= TLDR; CL(20) <= TCLR;
903         when "00010101000000" => LD(108) <= TLDR; CL(21) <= TCLR;  -- [21] a2_1
904         when "00010101000001" => LD(109) <= TLDR; CL(21) <= TCLR;
905         when "00010101000010" => LD(110) <= TLDR; CL(21) <= TCLR;
906         when "00010101000011" => LD(111) <= TLDR; CL(21) <= TCLR;
907         when "00010101000100" => LD(112) <= TLDR; CL(21) <= TCLR;
908         when "00010110000000" => LD(113) <= TLDR; CL(22) <= TCLR;  -- [22] a0_2
909         when "00010110000001" => LD(114) <= TLDR; CL(22) <= TCLR;
910         when "00010110000010" => LD(115) <= TLDR; CL(22) <= TCLR;
911         when "00010110000011" => LD(116) <= TLDR; CL(22) <= TCLR;
912         when "00010110000100" => LD(117) <= TLDR; CL(22) <= TCLR;
913         when "00010111000000" => LD(118) <= TLDR; CL(23) <= TCLR;  -- [23] a1_2
914         when "00010111000001" => LD(119) <= TLDR; CL(23) <= TCLR;
915         when "00010111000010" => LD(120) <= TLDR; CL(23) <= TCLR;
916         when "00010111000011" => LD(121) <= TLDR; CL(23) <= TCLR;
```

```
917         when "00010111000100" => LD(122) <= TLDR; CL(23) <= TCLR;
918         when "00011000000000" => LD(123) <= TLDR; CL(24) <= TCLR;  -- [24] a2_2
919         when "00011000000001" => LD(124) <= TLDR; CL(24) <= TCLR;
920         when "00011000000010" => LD(125) <= TLDR; CL(24) <= TCLR;
921         when "00011000000011" => LD(126) <= TLDR; CL(24) <= TCLR;
922         when "00011000000100" => LD(127) <= TLDR; CL(24) <= TCLR;
923         when "00011001000000" => LD(128) <= TLDR; CL(25) <= TCLR;  -- [25] a0_3
924         when "00011001000001" => LD(129) <= TLDR; CL(25) <= TCLR;
925         when "00011001000010" => LD(130) <= TLDR; CL(25) <= TCLR;
926         when "00011001000011" => LD(131) <= TLDR; CL(25) <= TCLR;
927         when "00011001000100" => LD(132) <= TLDR; CL(25) <= TCLR;
928         when "00011010000000" => LD(133) <= TLDR; CL(26) <= TCLR;  -- [26] a1_3
929         when "00011010000001" => LD(134) <= TLDR; CL(26) <= TCLR;
930         when "00011010000010" => LD(135) <= TLDR; CL(26) <= TCLR;
931         when "00011010000011" => LD(136) <= TLDR; CL(26) <= TCLR;
932         when "00011010000100" => LD(137) <= TLDR; CL(26) <= TCLR;
933         when "00011011000000" => LD(138) <= TLDR; CL(27) <= TCLR;  -- [27] a2_3
934         when "00011011000001" => LD(139) <= TLDR; CL(27) <= TCLR;
935         when "00011011000010" => LD(140) <= TLDR; CL(27) <= TCLR;
936         when "00011011000011" => LD(141) <= TLDR; CL(27) <= TCLR;
937         when "00011011000100" => LD(142) <= TLDR; CL(27) <= TCLR;
938         when "00011100000000" => LD(143) <= TLDR; CL(28) <= TCLR;  -- [28] a0_4
939         when "00011100000001" => LD(144) <= TLDR; CL(28) <= TCLR;
940         when "00011100000010" => LD(145) <= TLDR; CL(28) <= TCLR;
941         when "00011100000011" => LD(146) <= TLDR; CL(28) <= TCLR;
942         when "00011100000100" => LD(147) <= TLDR; CL(28) <= TCLR;
943         when "00011101000000" => LD(148) <= TLDR; CL(29) <= TCLR;  -- [29] a1_4
944         when "00011101000001" => LD(149) <= TLDR; CL(29) <= TCLR;
945         when "00011101000010" => LD(150) <= TLDR; CL(29) <= TCLR;
946         when "00011101000011" => LD(151) <= TLDR; CL(29) <= TCLR;
947         when "00011101000100" => LD(152) <= TLDR; CL(29) <= TCLR;
948         when "00011110000000" => LD(153) <= TLDR; CL(30) <= TCLR;  -- [30] a2_4
949         when "00011110000001" => LD(154) <= TLDR; CL(30) <= TCLR;
950         when "00011110000010" => LD(155) <= TLDR; CL(30) <= TCLR;
951         when "00011110000011" => LD(156) <= TLDR; CL(30) <= TCLR;
952         when "00011110000100" => LD(157) <= TLDR; CL(30) <= TCLR;
953         when "00011111000000" => LD(160) <= TLDR; CL(31) <= TCLR;  -- [31] EXL
954         when "00011111000001" => LD(161) <= TLDR; CL(31) <= TCLR;
955         when "00011111000010" => LD(162) <= TLDR; CL(31) <= TCLR;
956         when "00011111000011" => LD(163) <= TLDR; CL(31) <= TCLR;
957         when "00100000000000" => LD(164) <= TLDR; CL(32) <= TCLR;  -- [32] EYL
958         when "00100000000001" => LD(165) <= TLDR; CL(32) <= TCLR;
959         when "00100000000010" => LD(166) <= TLDR; CL(32) <= TCLR;
960         when "00100000000011" => LD(167) <= TLDR; CL(32) <= TCLR;
961         when "00100001000000" => LD(168) <= TLDR; CL(33) <= TCLR;  -- [33] EZL
962         when "00100001000001" => LD(169) <= TLDR; CL(33) <= TCLR;
963         when "00100001000010" => LD(170) <= TLDR; CL(33) <= TCLR;
964         when "00100001000011" => LD(171) <= TLDR; CL(33) <= TCLR;
965         when "00100010000000" => LD(172) <= TLDR; CL(34) <= TCLR;  -- [34] EWL
966         when "00100010000001" => LD(173) <= TLDR; CL(34) <= TCLR;
967         when "00100010000010" => LD(174) <= TLDR; CL(34) <= TCLR;
968         when "00100010000011" => LD(175) <= TLDR; CL(34) <= TCLR;
969         when "00100011000000" => LD(176) <= TLDR; CL(35) <= TCLR;  -- [35] OUTS
970         when "00100100000000" => LD(177) <= TLDR; CL(36) <= TCLR;  -- [36] EECMD
971         when "00100101000000" => LD(178) <= TLDR; CL(37) <= TCLR;  -- [37] EEDI
972         when "00100101000001" => LD(179) <= TLDR; CL(37) <= TCLR;
973         when "00100101000010" => LD(180) <= TLDR; CL(37) <= TCLR;
974         when "00100101000011" => LD(181) <= TLDR; CL(37) <= TCLR;
975         when "00100101000100" => LD(182) <= TLDR; CL(37) <= TCLR;
976         when "00100101000101" => LD(183) <= TLDR; CL(37) <= TCLR;
977         when "00100101000110" => LD(184) <= TLDR; CL(37) <= TCLR;
978         when "00100101000111" => LD(185) <= TLDR; CL(37) <= TCLR;
979         when "00100110000000" => LD(186) <= TLDR; CL(38) <= TCLR;  -- [38] Tsup
980         when "00100111000000" => LD(187) <= TLDR; CL(39) <= TCLR;  -- [39] WW
981         when "00100111000001" => LD(188) <= TLDR; CL(39) <= TCLR;
982         when "00100111000010" => LD(189) <= TLDR; CL(39) <= TCLR;
```

```
983           when "00100111000011" => LD(190) <= TLDR; CL(39) <= TCLR;
984           when "00101000000000" => LD(191) <= TLDR; CL(40) <= TCLR;  -- [40] CTM
985           when "00101001000000" => LD(192) <= TLDR; CL(41) <= TCLR;  -- [41] CTRL2
986           when "00101010000000" => LD(193) <= TLDR; CL(42) <= TCLR;  -- [42] SMMOD
987           when "00101011000000" => LD(194) <= TLDR; CL(43) <= TCLR;  -- [43] DiSM
988           when "00101011000001" => LD(195) <= TLDR; CL(43) <= TCLR;  --
989           when      others      =>
990        end case;
991     end process Demux;
992
993     Mux: process(ADDW,STATUS,X,Y,Z,W,SENX,SENY,SENZ,SENW,IDXX,IDXY,IDXZ,IDXW,YX,YY,YZ,YW,
994                  ERX,ERY,ERZ,ERW,SWT,T,DRO,EEDO,Ch0,Ch1,Ch2,Ch3)
995     begin
996        case ADDW is
997           when "00000000000000" => TxD <= STATUS;            -- [0] Status
998           when "00000001000000" => TxD <= X( 7 downto  0);   -- [1] X
999           when "00000001000001" => TxD <= X(15 downto  8);
1000          when "00000001000010" => TxD <= X(23 downto 16);
1001          when "00000001000011" => TxD <= X(31 downto 24);
1002          when "00000010000000" => TxD <= Y( 7 downto  0);   -- [2] Y
1003          when "00000010000001" => TxD <= Y(15 downto  8);
1004          when "00000010000010" => TxD <= Y(23 downto 16);
1005          when "00000010000011" => TxD <= Y(31 downto 24);
1006          when "00000011000000" => TxD <= Z( 7 downto  0);   -- [3] Z
1007          when "00000011000001" => TxD <= Z(15 downto  8);
1008          when "00000011000010" => TxD <= Z(23 downto 16);
1009          when "00000011000011" => TxD <= Z(31 downto 24);
1010          when "00000100000000" => TxD <= W( 7 downto  0);   -- [4] W
1011          when "00000100000001" => TxD <= W(15 downto  8);
1012          when "00000100000010" => TxD <= W(23 downto 16);
1013          when "00000100000011" => TxD <= W(31 downto 24);
1014          when "00000101000000" => TxD <= SENX( 7 downto  0);   -- [5] ENCX
1015          when "00000101000001" => TxD <= SENX(15 downto  8);
1016          when "00000101000010" => TxD <= SENX(23 downto 16);
1017          when "00000101000011" => TxD <= SENX(31 downto 24);
1018          when "00000110000000" => TxD <= SENY( 7 downto  0);   -- [6] ENCY
1019          when "00000110000001" => TxD <= SENY(15 downto  8);
1020          when "00000110000010" => TxD <= SENY(23 downto 16);
1021          when "00000110000011" => TxD <= SENY(31 downto 24);
1022          when "00000111000000" => TxD <= SENZ( 7 downto  0);   -- [7] ENCZ
1023          when "00000111000001" => TxD <= SENZ(15 downto  8);
1024          when "00000111000010" => TxD <= SENZ(23 downto 16);
1025          when "00000111000011" => TxD <= SENZ(31 downto 24);
1026          when "00001000000000" => TxD <= SENW( 7 downto  0);   -- [8] ENCW
1027          when "00001000000001" => TxD <= SENW(15 downto  8);
1028          when "00001000000010" => TxD <= SENW(23 downto 16);
1029          when "00001000000011" => TxD <= SENW(31 downto 24);
1030          when "00001001000000" => TxD <= IDXX( 7 downto  0);   -- [9] IDXX
1031          when "00001001000001" => TxD <= IDXX(15 downto  8);
1032          when "00001001000010" => TxD <= IDXX(23 downto 16);
1033          when "00001001000011" => TxD <= IDXX(31 downto 24);
1034          when "00001010000000" => TxD <= IDXY( 7 downto  0);   -- [10] IDXY
1035          when "00001010000001" => TxD <= IDXY(15 downto  8);
1036          when "00001010000010" => TxD <= IDXY(23 downto 16);
1037          when "00001010000011" => TxD <= IDXY(31 downto 24);
1038          when "00001011000000" => TxD <= IDXZ( 7 downto  0);   -- [11] IDXZ
1039          when "00001011000001" => TxD <= IDXZ(15 downto  8);
1040          when "00001011000010" => TxD <= IDXZ(23 downto 16);
1041          when "00001011000011" => TxD <= IDXZ(31 downto 24);
1042          when "00001100000000" => TxD <= YX( 7 downto  0);   -- [12] YX
1043          when "00001100000001" => TxD <= YX(15 downto  8);
1044          when "00001101000000" => TxD <= YY( 7 downto  0);   -- [13] YY
1045          when "00001101000001" => TxD <= YY(15 downto  8);
1046          when "00001110000000" => TxD <= YZ( 7 downto  0);   -- [14] YZ
1047          when "00001110000001" => TxD <= YZ(15 downto  8);
1048          when "00001111000000" => TxD <= YW( 7 downto  0);   -- [15] YW
```

```
1049            when "00001111000001" => TxD <= YW(15 downto  8);
1050            when "00010000000000" => TxD <= SWT(7 downto 0);    -- [16] SWT
1051            when "00010000000001" => TxD <= "0000" & SWT(11 downto 8);
1052            when "00010001000001" => TxD <= "0000" & ERX & ERY & ERZ & ERW; -- [17] ERRLIM
1053            when "00010010000000" => TxD <= T( 8 downto  1);   -- [18] T
1054            when "00010010000001" => TxD <= T(16 downto  9);
1055            when "00010010000010" => TxD <= T(24 downto 17);
1056            when "00010010000011" => TxD <= T(32 downto 25);
1057            when "00010011000000" => TxD <= EEDO( 7 downto   0); -- [19] EEDO
1058            when "00010011000001" => TxD <= EEDO(15 downto   8);
1059            when "00010011000010" => TxD <= EEDO(23 downto  16);
1060            when "00010011000011" => TxD <= EEDO(31 downto  24);
1061            when "00010011000100" => TxD <= EEDO(39 downto  32);
1062            when "00010011000101" => TxD <= EEDO(47 downto  40);
1063            when "00010011000110" => TxD <= EEDO(55 downto  48);
1064            when "00010011000111" => TxD <= EEDO(63 downto  56);
1065            when "00010100000000" => TxD <= DRO(  7 downto   0); -- [20] DRO
1066            when "00010100000001" => TxD <= DRO( 15 downto   8);
1067            when "00010100000010" => TxD <= DRO( 23 downto  16);
1068            when "00010100000011" => TxD <= DRO( 31 downto  24);
1069            when "00010100000100" => TxD <= DRO( 39 downto  32);
1070            when "00010100000101" => TxD <= DRO( 47 downto  40);
1071            when "00010100000110" => TxD <= DRO( 55 downto  48);
1072            when "00010100000111" => TxD <= DRO( 63 downto  56);
1073            when "00010100001000" => TxD <= DRO( 71 downto  64);
1074            when "00010100001001" => TxD <= DRO( 79 downto  72);
1075            when "00010100001010" => TxD <= DRO( 87 downto  80);
1076            when "00010100001011" => TxD <= DRO( 95 downto  88);
1077            when "00010100001100" => TxD <= DRO(103 downto  96);
1078            when "00010100001101" => TxD <= DRO(111 downto 104);
1079            when "00010100001110" => TxD <= DRO(119 downto 112);
1080            when "00010100001111" => TxD <= DRO(127 downto 120);
1081            when "00010100010000" => TxD <= DRO(135 downto 128);
1082            when "00010100010001" => TxD <= DRO(143 downto 136);
1083            when "00010100010010" => TxD <= DRO(151 downto 144);
1084            when "00010100010011" => TxD <= DRO(159 downto 152);
1085            when "00010100010100" => TxD <= DRO(167 downto 160);
1086            when "00010100010101" => TxD <= DRO(175 downto 168);
1087            when "00010100010110" => TxD <= DRO(183 downto 176);
1088            when "00010100010111" => TxD <= DRO(191 downto 184);
1089            when "00010100011000" => TxD <= DRO(199 downto 192);
1090            when "00010100011001" => TxD <= DRO(207 downto 200);
1091            when "00010100011010" => TxD <= DRO(215 downto 208);
1092            when "00010100011011" => TxD <= DRO(223 downto 216);
1093            when "00010100011100" => TxD <= DRO(231 downto 224);
1094            when "00010100011101" => TxD <= DRO(239 downto 232);
1095            when "00010100011110" => TxD <= DRO(247 downto 240);
1096            when "00010100011111" => TxD <= DRO(255 downto 248);
1097            when "00010101000000" => TxD <= IDXW( 7 downto  0);   -- [21] IDXW
1098            when "00010101000001" => TxD <= IDXW(15 downto  8);
1099            when "00010101000010" => TxD <= IDXW(23 downto 16);
1100            when "00010101000011" => TxD <= IDXW(31 downto 24);
1101            when "00010110000000" => TxD <= Ch0( 7 downto  0);    -- [22] Ch0
1102            when "00010110000001" => TxD <= Ch0(15 downto  8);
1103            when "00010111000000" => TxD <= Ch1( 7 downto  0);    -- [23] Ch1
1104            when "00010111000001" => TxD <= Ch1(15 downto  8);
1105            when "00011000000000" => TxD <= Ch2( 7 downto  0);    -- [24] Ch2
1106            when "00011000000001" => TxD <= Ch2(15 downto  8);
1107            when "00011001000000" => TxD <= Ch3( 7 downto  0);    -- [55] Ch3
1108            when "00011001000001" => TxD <= Ch3(15 downto  8);
1109            when      others      => TxD <= (others => '0');
1110        end case;
1111    end process Mux;
1112
1113 end mcUAQ4x;
1114
```

A.2 Modulo de posicionamiento de carrusel

```
 library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity flc_completo is
    generic(n : integer := 16;
            m : integer := 4);
     port(
         RST : in STD_LOGIC;
         CLK : in STD_LOGIC;
         START2:in STD_LOGIC;
         IDX : out STD_LOGIC_VECTOR(n-1 downto 0);
         ENC_AP : in STD_LOGIC;
         ENC_AM : in STD_LOGIC;
         ENC_BP : in STD_LOGIC;
         ENC_BM : in STD_LOGIC;
         ENC_IDXP : in STD_LOGIC;
         ENC_IDXM : in STD_LOGIC;
         DAC_SCLK : out STD_LOGIC;
         DAC_SYNC : out STD_LOGIC;
         DAC_DIN : out STD_LOGIC;
         DAC_LDAC : out STD_LOGIC;
         DAC_RST : out STD_LOGIC;
         READY2:out std_logic;
         GIRO:out std_logic
         );
end flc_completo;


architecture flc_completo_arch of flc_completo is
component encEncoder
     generic(n : integer := 16;
             m : integer := 4);
     port(
        RST : in STD_LOGIC;
        CLK : in STD_LOGIC;
        CLR : in STD_LOGIC;
        DIF : in STD_LOGIC;
        RD : in STD_LOGIC;
        K : in STD_LOGIC_VECTOR(m-1 downto 0);
        D : out STD_LOGIC_VECTOR(n-1 downto 0);
        IDX : out STD_LOGIC_VECTOR(n-1 downto 0);
        ENC_AP : in STD_LOGIC;
        ENC_AM : in STD_LOGIC;
        ENC_BP : in STD_LOGIC;
        ENC_BM : in STD_LOGIC;
        ENC_IDXP : in STD_LOGIC;
        ENC_IDXM : in STD_LOGIC
        );
end component;

component  Control
       port(
       CLK:in std_logic;
       RST: in std_logic;
       START2:in std_logic;
       Pos_actual:in std_logic_vector(14 downto 0);
       READY2:out std_logic;
       GIRO: out std_logic;
       Yz:out std_logic_vector(11 downto 0)
       );
end component;

component DAC7565
    port(
        RST : in STD_LOGIC;
        CLK : in STD LOGIC;
```

```
        WR : in STD_LOGIC;
        EOW : out STD_LOGIC;
        DRST : in STD_LOGIC;
        Ch0 : in STD_LOGIC_VECTOR(15 downto 0);
        Ch1 : in STD_LOGIC_VECTOR(15 downto 0);
        Ch2 : in STD_LOGIC_VECTOR(15 downto 0);
        Ch3 : in STD_LOGIC_VECTOR(15 downto 0);
        DAC_SCLK : out STD_LOGIC;
        DAC_SYNC : out STD_LOGIC;
        DAC_DIN : out STD_LOGIC;
        DAC_LDAC : out STD_LOGIC;
        DAC_RST : out STD_LOGIC
        );
end component;

signal CLR,DIF,RD,WR,EOW,DRST,START:std_logic;
signal Pos_actual: std_logic_vector(14 downto 0);
signal D : STD_LOGIC_VECTOR(n-1 downto 0);
signal K :STD_LOGIC_VECTOR(m-1 downto 0);
signal Yz:std_logic_vector(11 downto 0);
signal Ch0,Ch1,Ch2,Ch3 :STD_LOGIC_VECTOR(15 downto 0);
begin
    CLR<='0';
    DIF<='0';
    RD<='1';
    WR<='1';
    DRST<='0';
    K<="0000";
    Pos_actual<=D(14 downto 0);
    Ch0<= Yz &"0000";

    Modulo_1:encEncoder port map(RST,CLK,CLR,DIF,RD,K,D,IDX,ENC_AP,ENC_AM,ENC_BP,ENC_BM,ENC_IDXP,ENC_IDXM);

   Modulo_2:Control port map(CLK,RST,START2,Pos_actual,READY2,GIRO,Yz);

    Modulo_3:DAC7565 port map(RST,CLK,WR,EOW,DRST,Ch0,Ch1,Ch2,Ch3,DAC_SCLK,DAC_SYNC,DAC_DIN,DAC_LDAC,DAC_RST);

    end flc_completo_arch;
```

A.3 Módulo de control de rotación de componentes

```
library IEEE;
use IEEE.std_logic_1164.all;

entity motor_control is
    port(
    RST:in std_logic;
    CLK:in std_logic;
    START3:in std_logic;
    READY3:out std_logic;
    origen:in std_logic;
    C:in std_logic_vector(2 downto 0);
    M:in std_logic_vector(3 downto 0);
    P:  out std_logic_vector(4 downto 1)
    );
end motor_control;
```

Apéndice B

Artículo publicado en el 8° congreso Internacional de Ingeniería

Generación de trayectorias en una máquina de montaje de componentes chip SMD RX-4A

8° CONGRESO INTERNACIONAL DE INGENIERÍA

ÁREAS: DISEÑO E INNOVACIÓN, BIOSISTEMAS, INGENIERÍA CIVIL Y AMBIENTAL, AUTOMATIZACIÓN, MECATRÓNICA, ARQUITECTURA Y MATEMÁTICAS PURAS Y APLICADAS.

La UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
a través de la FACULTAD DE INGENIERÍA,
otorga la presente

CONSTANCIA

a:

Ing. Cesar Barrón Romero

Por haber participado con la Ponencia:

"Generación de trayectorias en una máquina de montaje de componente chip SMD RX-4ª"

en el Congreso Internacional de Ingeniería en su octava edición
realizado del 23 al 27 de Abril del 2012
en la ciudad de San Juan del Río, Querétaro, México.

CONCYTEQ

CONACYT
Consejo Nacional de Ciencia y Tecnología

Dr. Aurelio Domínguez González
Director

M.L. Adriana Medellín Gómez
Comité Organizador

creandoconciencia

Controlador FPGA aplicado a los ejes en una máquina de montaje de componentes chip RX-4A, a partir de códigos Gerber

FPGA controller applied to the axes in a RX-4A chip components mounting machine, taking as reference Gerber codes

César Barrón Romero[2], Roque Alfredo Osornio Ríos[1], Luis Morales Velázquez[1], Néstor Velázquez Neyra[3].

[1]Profesores de la maestría en mecatrónica de la Universidad Autónoma de Querétaro, [2]Estudiante de la maestría en mecatrónica de la Universidad Autónoma de Querétaro, [3]Estudiante de la maestría en mecatrónica de la Universidad Autónoma de Querétaro

RESUMEN. La industria electrónica nacional es un pilar central de nuestra economía. México es un país líder en la manufactura de productos electrónicos de gran demanda a nivel mundial, como teléfonos celulares, aparatos de video juegos, computadoras, electrodomésticos y automotrices, los cuales utilizan la tecnología SMD para el ensamble de componentes electrónicos chip sobre tableta PCB. Actualmente existen más de 730 plantas manufactureras relacionadas con la industria electrónica en México y es uno de los sectores de más rápido crecimiento en el país en términos de potencial exportador. En este trabajo se presenta el diseño e implementación en software y hardware de un controlador FPGA aplicado a los ejes x-y de una máquina de montaje de componentes electrónicos SMD. El resultado obtenido fue un sistema que incluye un el uso de IP cores propios basados en FPGA y hardware de control, que permiten el uso de tecnología de nueva generación de arquitectura abierta y es compatible con el estándar industrial extendido RS-274X de EIA, utilizado en el ensamble de componentes electrónicos sobre PCB en la industria electrónica.

Palabras clave: Controlador, FPGA, ejes, códigos Gerber, IP cores.

1. Introducción

Las décadas que siguieron al surgimiento del transistor en los años cuarenta han atestiguado un cambio sumamente drástico en la industria electrónica, resultado de ello la tecnología de montaje y colocación de componentes superficial ha tenido un crecimiento exponencial, esto debido a la gran demanda de productos electrónicos que invaden casi todas las actividades de la vida cotidiana. Si bien la construcción de tarjetas o placas electrónicas prototipo puede lograrse con cierta habilidad de manera manual, actualmente un proceso que es ampliamente aceptado en la industria electrónica es la tecnología SMD (Surface Mount Device, dispositivo de montaje superficial). La producción en serie de tarjetas electrónicas y la miniaturización de componentes utilizados en el ensamble de éstas, tales como transistores, resistencias, capacitores e inductores, requieren el uso de máquinas-herramientas que realicen las operaciones de selección, rotación y colocación precisa de estos componentes, para tal efecto se utilizan específicamente las máquinas de montaje chip (SMD).

En la actualidad una gran variedad de máquinas de montaje de componentes electrónicos SMD de diferentes marcas importantes a nivel mundial, pueden ser encontradas en la industria electrónica nacional, tal es el caso de EUROPLACER, FUJI, SONY, TDK, PANASONIC\PANASERT, PHILIPS\ASSEMBLEON, etc. Este tipo de máquinas utilizan para la selección, rotación y montaje de componentes electrónicos chip, el estándar RS-274X de EIA (Electronic Industries Association, Asociación de Industrias Electrónicas), el cual es un archivo en formato Gerber conteniendo las coordenadas X, Y, que generan las trayectorias de posición en las que serán fijados los componentes, además de comandos que definen el inicio, término y forma de la imagen o circuito electrónico diseñado.

Las investigaciones enfocadas a mejorar la operatividad de las máquinas de montaje de componentes SMD, han abordado mejoras en cuanto al aumento de velocidad en la operación de estos sistemas, mediante la solución algoritmos o software como el de Wang et al. (1999) el cual presenta resultados experimentales usando algoritmos genéticos para optimizar el aplicador de alta velocidad paralelo de componentes en una máquina

multiestación de ensamble SMT. Tirpak et al. (2002) trabajó sobre un simulador cuya plataforma de desarrollo fue C++, que permite determinar aspectos de una línea de producción, tales como la estimación de tiempo de manufactura y la evaluación del equipo de montaje de componentes, mediante la optimización y reducción del tiempo de producción de PCBs (Printed Circuit Boards, tarjetas de circuito impreso). Cabe mencionar que el juego de herramientas del programa puede ser configurado para realizar la simulación de una máquina en particular. Kumar y Luo (2003) utilizaron un modelo TSP sobre una máquina de montaje de componentes FUJI FCP-IV para la optimización en tiempo de ensamble. La formulación da una descripción matemática del tiempo de ensamble a ser optimizado como un problema completamente de programación. Su simulación ha sido ejecutada en tarjetas de Lexmark, Inc., Lexington, KY. Wang y Zhang (2006) por su parte trabajaron sobre un algoritmo de alta velocidad para centrar y rotar el ángulo de posicionamiento de componentes electrónicos, utilizando el área mínima de sensado sobre la geometría de los componentes y aumentando la precisión y velocidad de colocación. El algoritmo computacional y procesamiento de imagen fue desarrollado en lenguaje de programación Visual C++ 6.0. Recientemente Ho y Ji (2009) propusieron 2 modelos matemáticos cuya solución con algoritmos genéticos minimizan el tiempo de ensamble de componentes electrónicos realizados por una máquina de inserción automática del tipo mesa movible x-y, alimentadores y torreta giratoria múltiple de componentes electrónicos.

Como se observa las contribuciones se basan en la simulación mediante software, algoritmos o modelos matemáticos cuya solución reducen el tiempo de optimización en la colocación de componentes, y aumentan la velocidad del proceso. Acorde con ello sería deseable contar con un sistema de control en hardware x-y, de ensamble de componentes que tuviera la flexibilidad de una arquitectura abierta, reconfigurable, con capacidad de emulación y de bajo costo.

Este trabajo presenta el diseño e implementación de un controlador mediante IP cores basados en FPGA (Field Programmable Gate Array, arreglo de compuertas programables en campo), para el sistema de posicionamiento de una bancada o mesa de coordenadas x-y, sobre una máquina de inserción de componentes tipo chip, a partir del formato de datos estándar extendido RS-274X regulado por la EIA.

2 Metodología

El sistema de posicionamiento propuesto para este trabajo consiste de una interfaz de usuario, la cual genera las trayectorias de posicionamiento y rotación de componentes para su colocación, una interfaz usb que permite la comunicación entre el controlador y la interfaz, un controlador PID que utiliza para su implementación IP cores basados en FPGA, una etapa de potencia y retroalimentación para el movimiento y monitoreo de la bancada, en la figura 1, se muestra el sistema desarrollado.

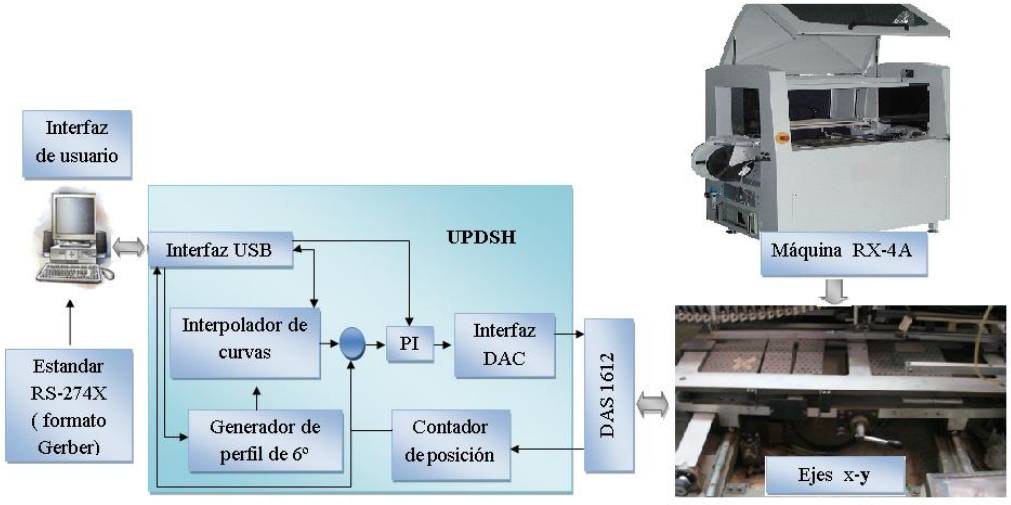


Figura 1. Arquitectura interna de controlador basado en IP cores FPGA.

2.1 Esquemático formato Gerber

Usualmente las máquinas de montaje de componentes parten de un código generado por algún software de diseño de PCB, es decir los datos más relevantes son presentados como listas de componentes con sus respectivos atributos, estos datos son enviados hacia el controlador bajo el estándar RS-274X, la figura 2, muestra una pantalla conteniendo una lista usual de componentes a ser montados sobre una PCB (1-7), en la cual se observan las características más significativas y necesarias para poder generar la secuencia de colocación y rotación de componentes sobre el sistema de posicionamiento x-y.

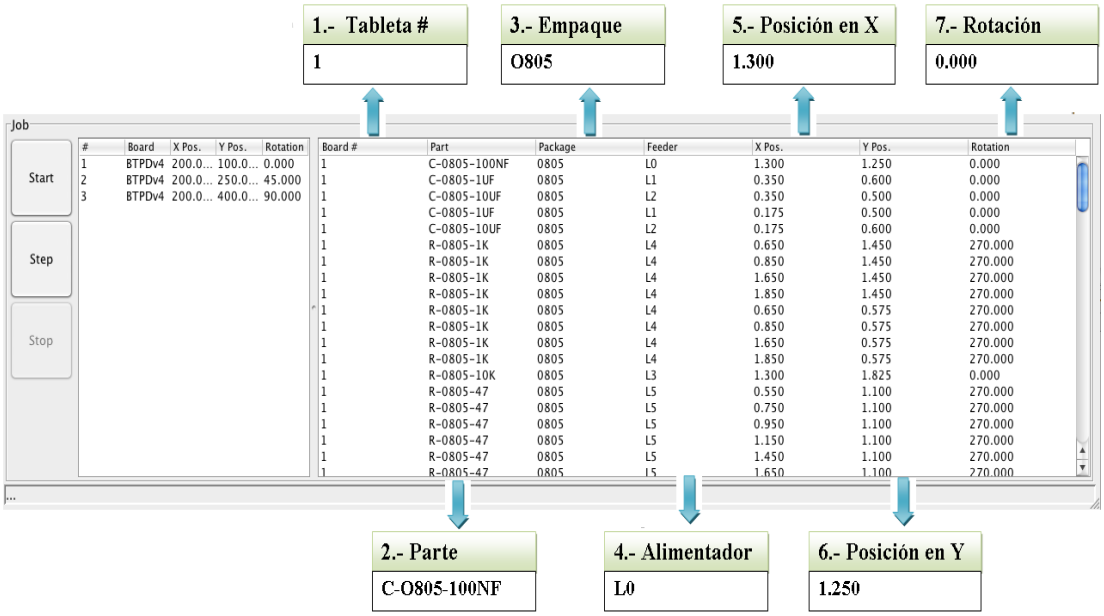


Figura 2. Lista de componentes a ensamblar solo tipo chip.

2.1.1 Interfaz USB

Este bloque implementa el protocolo de comunicación USB de acuerdo a Morales et al. (2009) y es el medio de comunicación entre el controlador y la interfaz gráfica de usuario, sirve para enviar por medio de la interfaz de usuario las constantes requeridas por los bloques del controlador y enviar del controlador la señal de encoder y de referencia a la interfaz para lograr el movimiento en la bancada x-y.

2.1.2 Generador de perfil de posición de sexto grado

En este bloque se implementa el algoritmo desarrollado por Jaen (2011) para un perfil de posición de sexto grado , el cual permite tener control en los parámetros de posición, velocidad y aceleración y por el grado del perfil tener movimientos más suaves con menos sobrepaso, el perfil esta normalizado a la unidad por lo que tiene una salida que va de 0 al iniciar el movimiento a 1 al completarlo, este dato es enviado al interpolador de curvas el cual lo utiliza para la reconstrucción de la curva, los parámetros de diseño del perfil son enviados al controlador mediante el módulo USB.

2.1.3 Interpolador de curvas paramétricas

En este bloque se implementa un algoritmo para la construcción de curvas Bspline y NURBS, lo cual permite desarrollar trayectorias curvas de manera suave, para el cálculo de los puntos de la curva se requieren de puntos de control y knots, en este caso en particular se hace uso de movimientos lineales o sea control punto a punto, sin embargo, el módulo será útil para una mayor precisión del movimiento, este bloque proporciona la referencia variable que debe hacer seguir el controlador PID.

2.1.4 PID

El bloque controlador PID implementado por Osornio (2008) se encarga de aproximar a cero el error entre la referencia proporcionada por el interpolador de curvas paramétricas y la posición real proporcionada por el contador de posición, las constantes de la ecuación de diferencias del PID son enviadas al controlador mediante el modulo USB, con una previa sintonización de la planta, la salida del PID es enviada al bloque interfaz DAC.

2.1.5 Interfaz DAC

Este bloque se encarga de manejar las señales para el funcionamiento del DAC 7565 de Texas Instruments en la tarjeta DAS1612 y de esta manera transformar el dato digital de salida del módulo PID a una señal analógica enviada al servoamplificador.

2.1.6 Contador de posición

Este bloque procesa los pulsos del encoder para determinar la posición absoluta en que se encuentra la máquina durante la ejecución de una secuencia completa. Este IP core basado en FPGA, consiste básicamente en un bloque contador ascendente-descendente, un detector de giro y una máquina de estados, para así obtener el número máximo de cuentas por revolución (CPR) o resolución en un factor x4 con la detección de margen.

2.1.7 DAS 1612

Es el sistema de adquisición de datos del controlador, fue desarrollado por el grupo HSP digital de la UAQ (2007)**,** cuenta con un Convertidor Digital Analógico (DAC ,Digital to Analog Convertir,) y un ADC (Analog to Digital Convertir, Convertidor Analógico Digital) de 12 bits, en este trabajo solo se hace uso de el DAC el cual transforma la señal digital del controlador a una señal analógica la cual es amplificada para obtener una señal de +/- 10 volts que se envía a los amplificadores, la DAS 1612 alimenta los encoder y recibe las señales generadas por éstos, también recibe las señales de los interruptores de limite.

3. Pruebas y Resultados

Para probar la funcionalidad del sistema de control y posicionamiento diseñado e implementado, diferentes pruebas fueron ejecutadas realizando movimientos punto a punto de una secuencia de montaje de componentes electrónicos chip sobre una tableta, a partir de un circuito electrónico genérico diseñado en el software de diseño FABMASTER, la referencia de comparación se realizó contra un controlador comercial Galil DMC-1832, estableciendo las mismas condiciones para seguimiento de sus trayectorias. En el caso del sistema comercial se hizo uso del algoritmo de autosintonización (Auto-Crossover-Frequency) desarrollado por el fabricante. En la figura 3, se muestras las trayectorias y secuencias que debe seguir el sistema de control, para la colocación de componentes SMD.

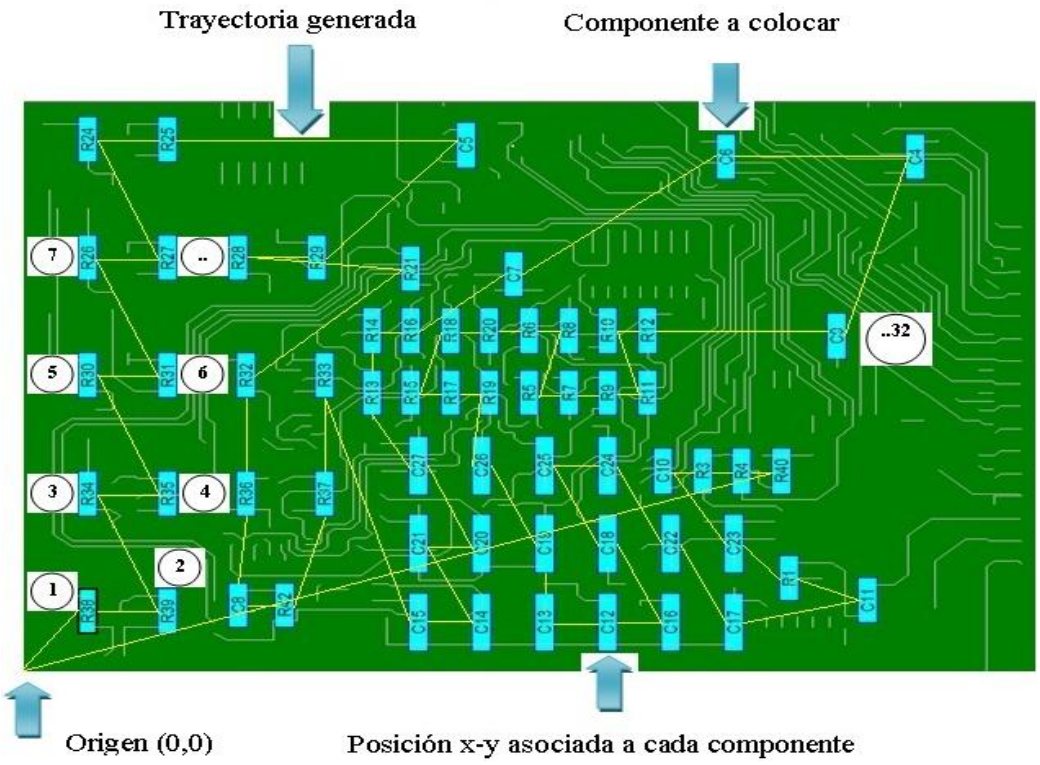


Figura 3. Trayectorias de colocación de componentes

Para realizar la validación de los datos obtenidos en la experimentación, debido a que no se cuenta con un patrón de referencia se hace un comparativo entre el controlador comercial Galil DMC-1832 y el controlador propio utilizando IP cores basados en FPGA, siguiendo la trayectoria de la figura 4, de ambos ejes x-y. En color azul se muestra la trayectoria de referencia y color rojo la trayectoria de seguimiento de ambos controladores figuras 4 y 5.

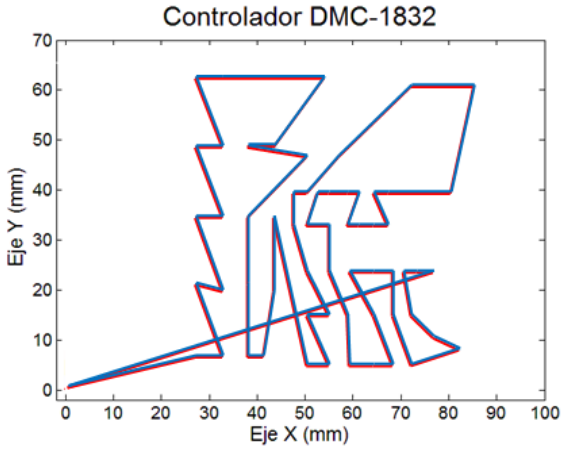


Figura 4. Trayectorias de ejes coordenados de controlador DMC-1832

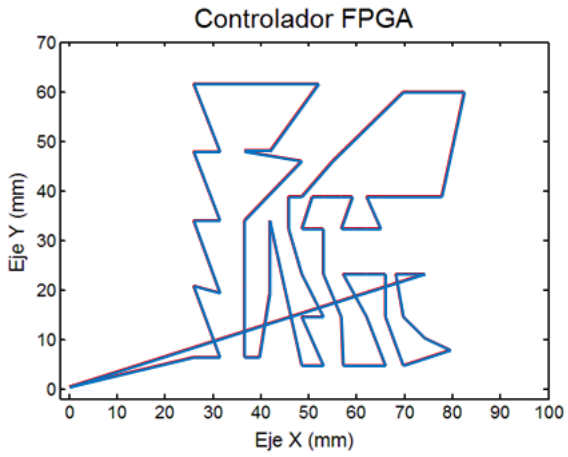


Figura 5. Trayectorias de ejes coordenados de controlador FPGA

Para tener una evaluación más acertada, sobre el error generado por los controladores se calcularon sobre el error los siguientes parámetros: $\bar{x}$ la media del error, y la desviación estándar σ. Sobre un tamaño de muestras de 100 000 datos. Los cuales dieron como resultado los datos de la tabla 2 y sus gráficas correspondientes en las figuras 6 y 7.

Tabla 1. Mediciones estadísticas de datos obtenidos de posicionamiento

Controlador	$\bar{x}$	σ	Error máximo
Controlador DMC-1832	0.22738	0.00819	0.26160
Controlador basado en FPGA	0.00254	0.00023	0.03600

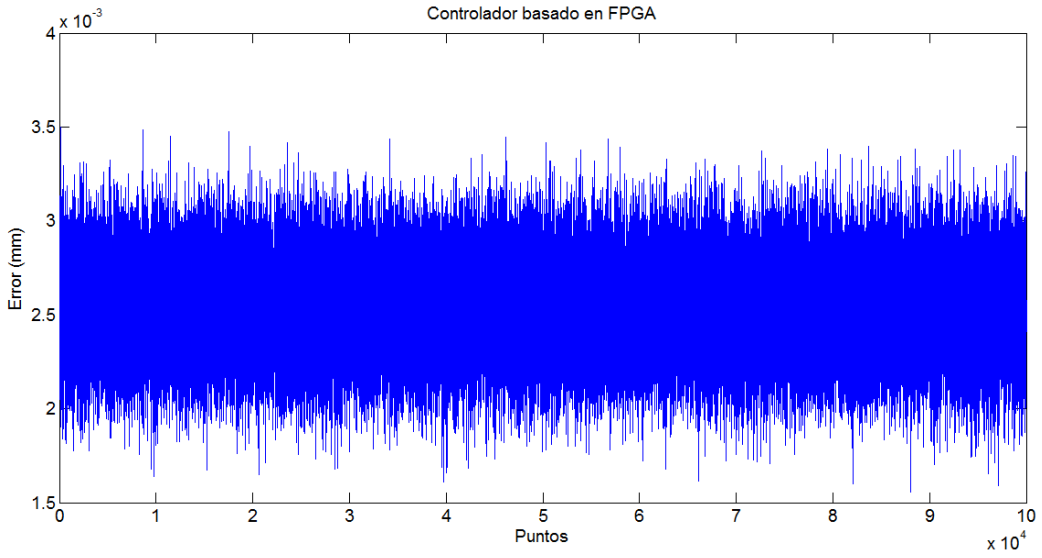


Figura 6. Gráfica del error de controlador FPGA

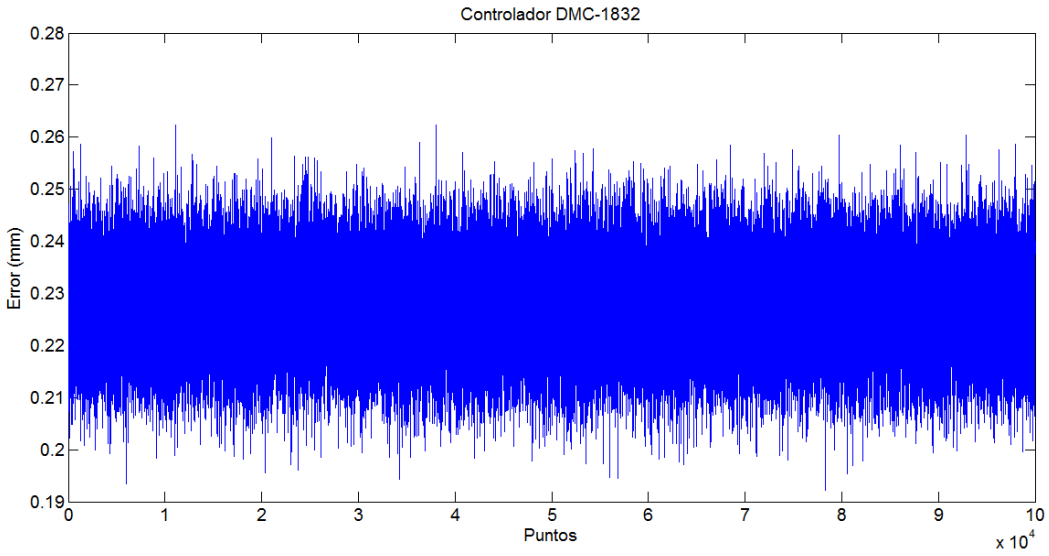


Figura 7. Gráfica del error de controlador DMC-1832

De acuerdo a los resultados observados en las figuras 6 y 7 , la tabla 1 muestra claramente que la media de los datos analizados del controlador basado en FPGA, es 100 veces más pequeña comparada con el controlador comercial Galil, respecto a la desviación estándar, se observa que los datos se concentran o presentan una mayor cercanía hacia la media y el valor de error máximo en el controlador propio, lo cual es deseable para el control de posicionamiento en el sistema x-y de la máquina de inserción de componentes.

4. Conclusiones

Este trabajo presentó el diseño e implementación de un controlador FPGA aplicado a los ejes x-y de una máquina de colocación de componentes tipo chip, el cual da como resultado que mediante el uso de IP cores propios basados en FPGA y hardware de control, permiten el uso de tecnología de nueva generación, y es compatible con el estándar industrial extendido RS-274X de EIA, utilizado en el ensamble de componentes electrónicos sobre PCB en la industria electrónica.

Con la implementación de este tipo sistemas de control, se obtienen como resultado sistemas de control de arquitectura abierta, flexibles, reconfigurables y de bajo costo que muestran una mejora en la operatividad de este tipo de máquina con respecto a su sistema del posicionamiento, lo cual también da como beneficio el contar con un sistema flexible que permita la reconversión de maquinaria de este tipo y permitir que su vida útil pueda ser extendida.

Además el interpolador de curvas paramétricas y generador de perfiles que fue implementado en el controlador, permiten tener movimientos más suaves, los cuales en algún momento pudieran afectar el ensamblado final de los componentes electrónicos sobre la tableta PCB montada en el sistema de posicionamiento x-y.

5. Referencias

Electronic Industries Association (EIA), RS-274X Format 1998.

Galil Motion Corporation, 2004,DMC-18X2 series Data Sheet .

Jaen Cuellar, Arturo Y., 2011. Desarrollo de perfiles polinomiales 3D basado en FPGA para control de posición en máquina fresadora CNC. Tesis Maestría. Universidad Autónoma de Querétaro. Facultad de Ingeniería.

Jun Wang y Mingzhu Zhang., 2006. A New High-Speed Algorithm for Center and Rotate Angle of Electronic Components. IEEE International Conference on Mechatronics and Automation.

Kumar, R. Zhonghui Luo. 2003., Optimizing the operation sequence of a chip placement machine using TSP model. IEEE International Conference on Mechatronics and Automation.

Morales Velázquez L., 2010, Diseñó de plataforma hardware-software para el desarrollo de aplicaciones industriales basadas en FPGA, Tesis de Doctorado, Universidad Autónoma de Querétaro, Facultad de Ingeniería.

Morales Velázquez L., Romero Troncoso R. de J., Osornio Ríos R. A., Herrera Ruiz G. y De Santiago Pérez J. J., 2009, Special purpose processor for parameter identification on CNC second order servo Systems on a low-cost FPGA platform, Journal Elsevier.

Osornio Ríos R. A., Romero Troncoso R. J., 2008, Herrera Ruiz G. y Castañeda Miranda R., FPGA implementation of higher degree polinomial acceleration profiles for peak jerk reduction on servomotors, Journal Science Direct.

Rivera Guillén, Jesús R. 2007., Perfiles Polinomiales de Movimiento para Máquinas CNC. Tesis Maestría. Universidad de Guanajuato. FIMEE.

Tirpak, T.M. Mohapatra y P.K. Nelson, P.C. Rajbhandari, R.R. 2002., A generic classification and object-oriented simulation toolkit for SMT assembly equipment. IEEE International Conference on Systems, Man and Cybernetics.

Weihsin Wang, Nelson, P.C., y Tirpak, T.M. 2002., Optimization of high-speed multistation SMT placement machines using evolutionary algorithms. IEEE International Conference on Systems, Man and Cybernetics.

William Ho y Pin Ji. 2009. Integrated component_scheduling models for chip shooter machines. Journal Science Direct.

http://www.ti.com/product/dac7565.