Arana Ruiz Diego
Mendoza Rubio Ricardo Daniel

Controlador de Movimiento para Bancada de Dos Ejes

2011

Universidad Autónoma de Querétaro
Facultad de Ingeniería

Controlador de Movimiento para Bancada de Dos Ejes

Tesis
Que como parte de los requisitos para
obtener el grado de

INGENIERO EN AUTOMATIZACIÓN

Presentan

Arana Ruiz Diego
Mendoza Rubio Ricardo Daniel

Dirigida por

Dr. Juvenal Rodríguez Reséndiz.

C.U. Santiago de Querétaro, Qro. Septiembre 2011

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Ingeniería en Automatización

Controlador de Movimiento para Bancada de Dos Ejes

TESIS

Que como parte de los requisitos para obtener el grado de

INGENIERO EN AUTOMATIZACION

Presentan:
Arana Ruiz Diego
Mendoza Rubio Ricardo Daniel

Dirigido por:
Dr. Juvenal Rodríguez Reséndiz.

SINODALES

Dr. Juvenal Rodríguez Reséndiz Presidente	_____ Firma
Dr. Edgar Rivas Araiza Sinodal	_____ Firma
Dr. Manuel Toledano Ayala Sinodal	_____ Firma
Dr. Roberto Augusto Gómez Loenzo Sinodal	_____ Firma
MC. Andrés Antonio Acosta Osorio Sinodal	_____ Firma

_____ Dr. Gilberto Herrera Ruíz Director de la Facultad	_____ Dr. Manuel Toledano Ayala Coordinador de la Carrera

Centro Universitario
Querétaro, Qro.
Agosto 2011
México

RESUMEN

Esta investigación está enfocada a la realización e implementación de un control que sea capaz de manipular posición y velocidad en una bancada de manera más económica, e igualmente eficaz, que dispositivos especializados en esta área, empleando un microcontrolador de bajo costo y una interfaz hombre-máquina fácil de utilizar para el operador.

Inicialmente se pretende explicar las características previas al control de posición y velocidad, como lo es una correcta etapa de acondicionamiento que permita el funcionamiento adecuado del microcontrolador. Se muestran sugerencias para realizar una etapa que permita economizar y ahorrar espacio de diseño al usuario. Se explican las características que debe poseer un microcontrolador para que se pueda implementar un código adecuado para el control de esta naturaleza, como son la memoria, las instrucciones por segundo, características de voltaje de entrada, etc. De igual manera se hacen referencias para tomar la decisión del compilador adecuado que se pueda utilizar y satisfaga las necesidades de la investigación, empezando por la parte del sistema operativo y software, así como para la interfaz Hombre-Máquina.

Finalmente se intenta demostrar que los objetivos y la hipótesis planteados se cumplen satisfactoriamente, permitiendo dejar una investigación que abrirá oportunidades a futuros proyectos.

(**Palabras clave**: Control de posición, control de velocidad, microcontrolador, interfaz hombre-máquina)

SUMMARY

This research is focused on the realization and implementation of a control that is able to develop position and speed in a more economical and equally effective way than specialized devices in the area, using a Human-Machine Interface easy to use by the operator and a cheaper microcontroller.

Initially intended to explain the features of position and speed control, such as proper conditioning stage to permit the proper functioning of the microcontroller, display suggestions for a stage that allows saving and space saving design to the user. The characteristics that must have a microcontroller so that it can implement an appropriate code for this kind of control, such as memory, instructions per second, input voltage characteristics, etc. Similarly, references are made to make the decision of the appropriate compiler, which can be used and meets the needs of research, starting with the part of the operating system and software, and also for the human-machine interface.

Finally, the intention was to demonstrate that the objectives and hypotheses are being satisfactorily implemented, allowing an investigation that will leave opportunities to future projects.

(**Key words**: Position control, speed control, human-machine interface, microcontroller)

Justificación

Hoy en día, "Automatización" es un término sumamente conocido, puesto que es difícil encontrar procesos industriales donde no se utilice esta técnica.

La demanda de la Automatización cada vez es mayor, lo que provoca que los procesos de producción sean más eficaces en todos sus aspectos. Por esta razón las empresas buscan la mejor calidad tecnológica, y exigen que los procesos de producción sean más eficientes y productivos. Por lo que surge la necesidad de desarrollar tecnología capaz de optimizar y mejorar estos procesos, haciéndola a su vez, menos costosa e igual de funcional.

La tecnología, es la necesidad del hombre de poner en práctica sus conocimientos, y adaptarlos a una aplicación en específico. Siendo pues, la vida del hombre una práctica diaria, el crecimiento de ésta es exponencial. Es por esto, que hacerla más eficiente y menos costosa, ayuda al hombre a implementarla con mayor facilidad, dándole lugar a la adaptación y modificación de ésta misma para ser sometida a actualización, con el objetivo de innovar dentro del ciclo tecnológico.

La meta, no es crear nueva tecnología, sino aplicar la tecnología existente, previamente aplicada. Ahora bien, el proyecto se basa en encontrar y desarrollar una actualización para controlar una bancada de dos ejes utilizando el campo de carácter digital, implementando su instrumentación y logrando un mejor control a partir de un DSC (Digital Signal Controller, por sus siglas en inglés) de la familia MicroChip (www.microchip.com), mejor conocido como dsPIC. Logrando así la disminución del costo, pero con la misma funcionalidad.

En el párrafo pasado, se menciona el término "tecnología existente", citado por la necesidad de mencionar algunos sistemas ya aplicados, como es en los sistemas de control numérico computarizado (CNC por sus siglas en ingles, "computerized numerical control") especificados en artículos científicos (N. Arda

Erol, et al, 2000)(N. Newell, 1996) donde se utiliza un “vinculo” de procesamiento digital de señales(DSP por sus siglas en ingles, “digital signal processing”) para el control de funciones. Cabe mencionar que un DSP no es lo mismo que un dsPIC(DSC) ya antes mencionado, quien ahora será el principal encargado de este “vinculo”. Posteriormente se aportarán más detalles acerca de esta diferencia.

Cumpliendo los DSCs todos los requisitos necesarios para lograr un control digital de señales funcional, de alto rendimiento, de alta velocidad y de bajo costo, es este el seleccionado para desarrollar el proyecto. Buscando tener las mejores respuestas y una tecnología actualizada, se optó por definir al microcontrolador dsPIC33FJ128MC802 como el corazón del proyecto, ya que cumple con todos las especificaciones antes mencionadas de una mejor manera que otros microcontroladores.

Dentro de la alta gama de aplicaciones que posee el dsPIC33FJ128MC802, se encuentra el Control de Movimiento (Jacob Tal, 1989) de motores de inducción (Robert D. Lorenz et al, 1994) del cual canalizan los tres siguientes: Control de Posición, Control de Velocidad y Control de Aceleración. Sistemas de lazo cerrado para el control de servo mecanismos (M. Nakamura et al, 2004). Controles de alta prioridad para el desarrollo del proyecto, es por esto que nos concentraremos en desarrollar un tema muy bien estructurado acerca de estas técnicas.

Cabe destacar que existen otros dispositivos programables, como lo es el FPGA, que ya tiene una aplicación práctica en sistemas de Control de Movimiento (Jung Uk Cho et al, 2009) sin embargo, el costo de este tipo de microcontrolador suele ser más elevado y el tiempo de desarrollo se incrementa a casi el doble que en el caso de los DSCs.

Para concluir, el DSC nos otorga funcionalidad, alto rendimiento y costos inferiores, parámetros que dan lugar al desarrollo del presente proyecto.

OBJETIVOS

Objetivo General.

Controlar e Instrumentar una bancada de dos ejes con un microcontrolador dsPIC33FJ128MC-802.

Objetivos Particulares.

Diseñar un algoritmo de control en un DSC.
Diseñar e implementar una HMI para el sistema.
Diseñar una topología eléctrica eficiente.
Comunicar el DSC con la HMI.
Generar perfil trapezoidal para la ejecución de movimiento.
Diseñar un esquema gráfico del comportamiento dinámico de los ejes.
Generar un banco de datos que muestre las variables de posición y velocidad. De forma que estas puedan ser manipuladas por el usuario.

Dedicatorias Diego Arana

A mis papis que son tan lindos conmigo y a juve que tanto me quiere y es tan bello.

Dedicatorias Ricardo Mendoza

A mis padres y todo.

AGRADECIMIENTOS

A Dios, a mis padres y profesores.

INDICE

INDICE DE TABLAS

Tabla **Página**

ÍNDICE DE FIGURAS

1. INTRODUCCION

En la actualidad, la gran mayoría de los escenarios donde los seres humanos interactúan son del tipo inteligente, lo que quiere decir que están dotados de un sin número de funciones para generar una respuesta autónoma que satisfaga sus necesidades, su bienestar, comodidad, entre otros.

Cuando se habla de acondicionar un espacio de trabajo o de entretenimiento, incluso solo para tener una buena presentación del lugar, se pide que se maneje lo más innovador posible al menor costo. Los procesos de automatización no solo se encuentran dentro de la industria, o dentro del desarrollo de nueva tecnología, también se pueden representar de forma que cosas sencillas o cotidianas se puedan realizar de una manera más fácil, eficiente e interactiva. Hoy en día, la gran mayoría de las tareas que se desarrollan exigen el uso de un proceso cada vez más rápido y eficiente, sobre todo cuando se habla de áreas de trabajo, en donde se exige y se cuida el cumplir con una alta calidad en el proceso y en el producto resultante. Es una tarea muy compleja que requiere, en su mayoría, de costos muy elevados y, como ya se ha mencionado, una buena precisión. Pero cuando se habla de todo ello se sobreentiende que un proceso automatizado se encarga de dar la solución en casos como estos.

El desarrollo o la innovación tecnológica enfocada para pequeñas aplicaciones no necesariamente deben derivarse en grandes inversiones y costos elevados, ya que, con la tecnología actual, es posible automatizar procesos de una manera eficiente a costos mucho más bajos que adquiriendo tecnología ya desarrollada, es decir, en muchas ocasiones es más conveniente y económico el automatizar un proceso sencillo que el comprar el proceso previamente automatizado.

Hoy en día, se presenta un problema social en el que las micro, pequeñas y medianas empresas (PYMES) quiebran dentro de los primeros dos años, siendo una de las principales causas el rezago tecnológico que se presenta en las

mismas, lo que las convierte en competidores muy débiles para las empresas ya consolidadas en el mercado.

En una entrevista realizada al diputado José Alberto Benavides Castañeda, Presidente de la Comisión de Fomento Económico, mencionó que el cierre de las empresas no se debe a la crisis, sino a que micro y pequeños empresarios inician operaciones sin contar con un plan de negocios. “Abren con más entusiasmo y determinación que con conocimiento debido a la falta de cultura empresarial y programas que impulsen el uso de nuevas tecnologías. Resultado, alrededor del 80 por ciento de los negocios cierran antes del primer año de operación” (14 de Agosto 2010).

Comúnmente, una PYME comienza a laborar con equipo básico y de bajo costo, por lo mismo, si la empresa comienza a incrementar su demanda, estos equipos o máquinas resultan ineficientes para satisfacer dicha demanda. Sin embargo, adquirir equipo automatizado resulta demasiado costoso y el resultado son pérdidas de clientes o daños a la imagen de la empresa por no poder satisfacer las crecientes necesidades del negocio.

Existen algunas opciones para librar este conflicto, donde la más común es la adquisición de un crédito empresarial para obtener dichos equipos, lo cual, en muchos casos, resulta contraproducente, pues se depende mucho de la estabilidad económica del país para poder saldar la deuda más rápidamente.

Una alternativa muy viable para este tipo de empresas sería la de automatizar el equipo con el que se cuenta, lo cual resultaría o aproximaría su eficiencia a la de los equipos automáticos comerciales pero a un costo mucho más bajo que estos. Es precisamente esta opción en la que se centra el presente proyecto, particularmente hablando del proceso de mecanizado de una fresadora manual.

1.1. Antecedentes.

1.1.1. Fresadora.

Con el nombre genérico de fresado se conoce al conjunto de operaciones de mecanizado que pueden efectuarse en la máquina-herramienta denominada fresadora**.** El fresado permite mecanizar en superficies planas, ranuras, engranajes e incluso superficies curvas o alabeadas. Constituye, junto con el torneado, el grupo de operaciones mayoritariamente empleadas en el mecanizado.

El movimiento principal en el fresado es de rotación, y lo lleva la herramienta o fresa. Los movimientos de avance y penetración son generalmente rectilíneos, pudiendo llevarlos la herramienta o la pieza según el tipo de máquina-herramienta y la operación realizada.

Aunque existen diversos tipos de fresadoras que incorporan ciertas particularidades, la fresadora que se utilizará en este proyecto será del tipo vertical, mostrado en la figura 1.

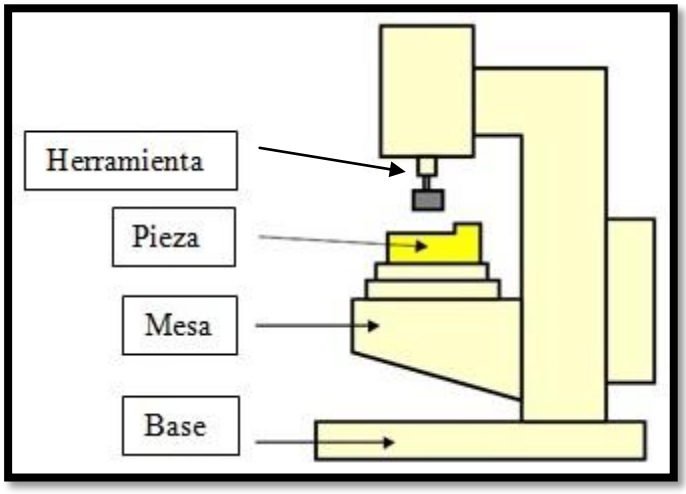


Figura 1. Esquema general de la fresadora utilizada

Como se aprecia en la figura 1, la fresadora está compuesta de cuatro partes principales, que son:

- Herramienta: También conocida como “fresa” es la que interactúa directamente con la pieza rotando alrededor del eje principal o husillo de la máquina, designado como eje Z, dándole forma y modelando la misma según los movimientos realizados en los ejes X y Y de la misma. Está montada sobre el motor de accionamiento, el cual, suministra la potencia requerida para el mecanizado. La sujeción de la herramienta a la máquina suele realizarse mediante un eje portafresas; este eje posee un extremo cónico (cono ISO, cono Morse) que se acopla al husillo de la fresadora, y al que se unen las fresas empleando para ello medios de fijación estandarizados (pinzas, chavetas, etc.).

  Existen, como se muestra en la figura 2, diversos tipos de fresas, que darán como resultado distintos tipos de fresado, según sea requerido por el tipo de material de la pieza o por la forma requerida en la misma.

Figura 2. Ejemplos de algunos tipos de fresas.

- Pieza: Se refiere al elemento que se desea moldear interactuando directamente con la herramienta o fresa. Según el material de éste será la fresa seleccionada. Los materiales empleados para el proceso son muy variados según las necesidades del producto, sin embargo, los más comúnmente utilizados son los metales y las cerámicas.

- Mesa: Es la superficie de apoyo donde se sujeta la pieza que va a ser fresada. Está constituida de un material resistente capaz de soportar los daños ocasionados por la fresa. En ella se encuentran el eje X, que es horizontal y por lo tanto paralelo a la superficie de apoyo de la pieza, y el eje Y, que es perpendicular a los otros dos formando un triedro. La fresadora permite un desplazamiento relativo entre pieza y herramienta, paralelo al husillo, además de los desplazamientos contenidos en un plano perpendicular a dicho husillo.

- Base: También conocida como bancada, permite la fijación de la máquina-herramienta al suelo y proporciona rigidez estructural a cada uno de los elementos soportando la estructura completa de la máquina-herramienta.

La bancada que se utilizará para el presente proyecto es de dos ejes del fabricante Boston Gear, donde el primero es para el movimiento de la mesa, y el segundo es para la variación de la velocidad del husillo.

Esta bancada cuenta con todos los elementos anteriormente descritos para una máquina herramienta tipo fresadora.

Como se ilustra en la figura 3, la bancada es de tipo industrial, donde ambos ejes se controlan manualmente. El primero por medio de un tornillo sinfín movido por una manivela y el segundo establecido manualmente por medio de una perilla de ajuste unida a una tarjeta de potencia provista por el fabricante.

Figura 3. Imagen de la Bancada

1.1.2. Microcontroladores.

Para entender la descripción de un microcontrolador es necesario antes conocer la de un computador, ya que la estructura y arquitectura entre estos son prácticamente lo mismo. La arquitectura de un computador se define como la organización que tienen los elementos que lo componen, al igual que el comportamiento que comprende el interactuar entre ellos al realizar operaciones normales. La arquitectura con la que está diseñado un computador o en su caso, un microcontrolador, define su comportamiento y sus posibilidades (Arnau Vicente y Orduña Manuel, 1996).

En 1945 Von Neumann definió el modelo básico de arquitectura de computador digital, el cual es utilizado actualmente en la mayoría de las computadoras. La estructura definida por Neumann se presenta en la figura 4.

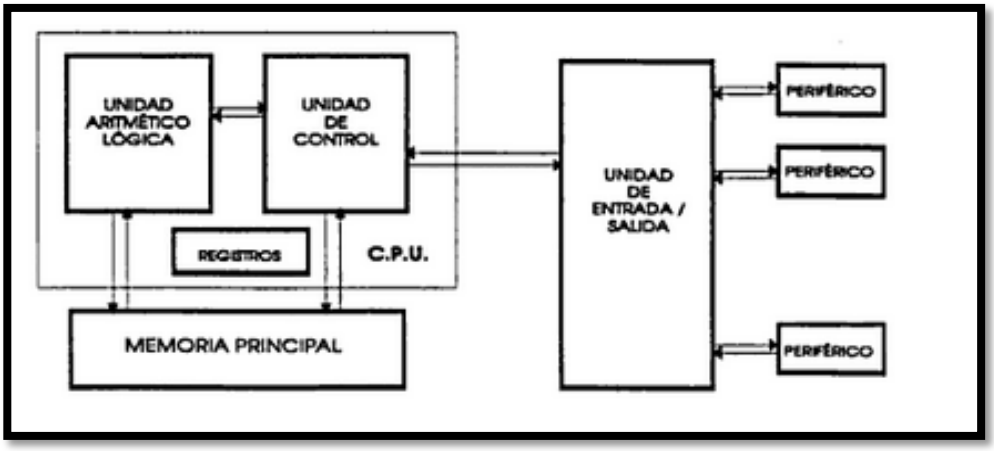


Figura 4. Arquitectura de un computador. Modelo de Von Neumann.

En conclusión, un microcontrolador combina los recursos fundamentales disponibles en un microcomputador, es decir, la unidad central de procesamiento (CPU), la memoria y los recursos de entrada y salida, en un único circuito integrado (Valdés Fernando y Pàllas Ramón. 2007).

La figura 5 muestra más detalladamente en forma de diagrama de bloques la arquitectura de un microcontrolador.

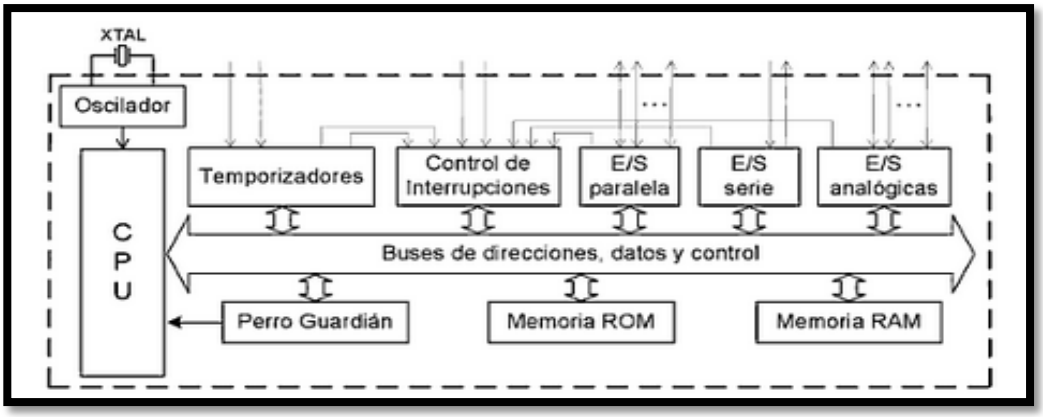


Figura 5. Arquitectura de microcontrolador.

El microcontrolador es capaz de ejecutar una serie de instrucciones que deben estar almacenadas en la memoria principal para poder ser leídas y después ejecutarlas. Las unidades fundamentales de un “micro” son las siguientes:

MEMORIA. Dispositivo de almacenamiento de información que está dividido en casillas ordenadas e identificadas mediante direcciones.

Las instrucciones se almacenan temporalmente o parcialmente en la memoria para después poder ser ejecutadas. Los datos y variables, también son almacenados en su respectiva dirección o casilla.

CPU. Es el cerebro del dispositivo, quien gobierna y ejecuta todos los cambios. Está compuesta de dos unidades fundamentales: Unidad Aritmética-Lógica, donde se realizan todas las operaciones aritméticas y lógicas elementales tales como la suma, resta, AND, OR…etc, y Unidad de Control, encargada de la lectura, decodificación e interpretación de instrucciones almacenadas en la memoria. El CPU contiene varios registros para almacenamiento temporal de datos y resultados, así como para el control del estado del microcontrolador.

UNIDADES DE ENTRADA/SALIDA. Unidad encargada de la transferencia de de información, ya sea hacia el interior o el exterior, por medio de los periféricos.

RELOJ. Parte fundamental del dispositivo. El reloj es el encargado de la sincronización de todas las operaciones en los componentes del microcontrolador. La frecuencia de reloj define la velocidad de ejecución de instrucciones. Este puede ser interno o externo.

En la actualidad, las tecnologías empleadas en la fabricación de microcontroladores son CMOS (Metal-Oxide Semiconductor, por sus siglas en inglés) y TTL (Transistor-Transistor Logic, por sus siglas en inglés). Sus características difieren en la velocidad y el consumo de energía (Arnau Vicente y Orduña Manuel, 1996). Poniendo a los dispositivos CMOS en el nivel de bajo

consumo, pero colocando a la tecnología TTL como la más rápida. Para nuestro caso en particular, la tecnología CMOS es inherente, puesto que el microcontrolador seleccionado para este proyecto fue un dsPIC con características de alta frecuencia y bajo consumo.

Entre la amplia gama de microcontroladores, el dsPIC33FJ128MC802 es el candidato perfecto para la implementación del controlador de velocidad y movimiento, por su flexibilidad al programar; bajo costo; alta frecuencia de oscilación y mínimo consumo de energía.

1.1.2.1. Microcontroladores aplicados al control de movimiento.

Existe una gran variedad de microcontroladores enfocados al control de movimiento para motores, los cuales proveen características y módulos especiales, como lo son:

- Modulador de Ancho de Pulso (PWM, por sus siglas en inglés).
- Interfaz de Cuadratura de Codificador (QEI, por sus siglas en inglés).

Dentro de esta categoría podemos encontrar a la familia dsPIC30F/dsPIC33F. Cuyas características favorecen y facilitan el trabajo de aplicación.

En la figura 6 se muestra el diagrama de bloques para esta familia, donde podemos identificar con facilidad los módulos especiales antes mencionados.

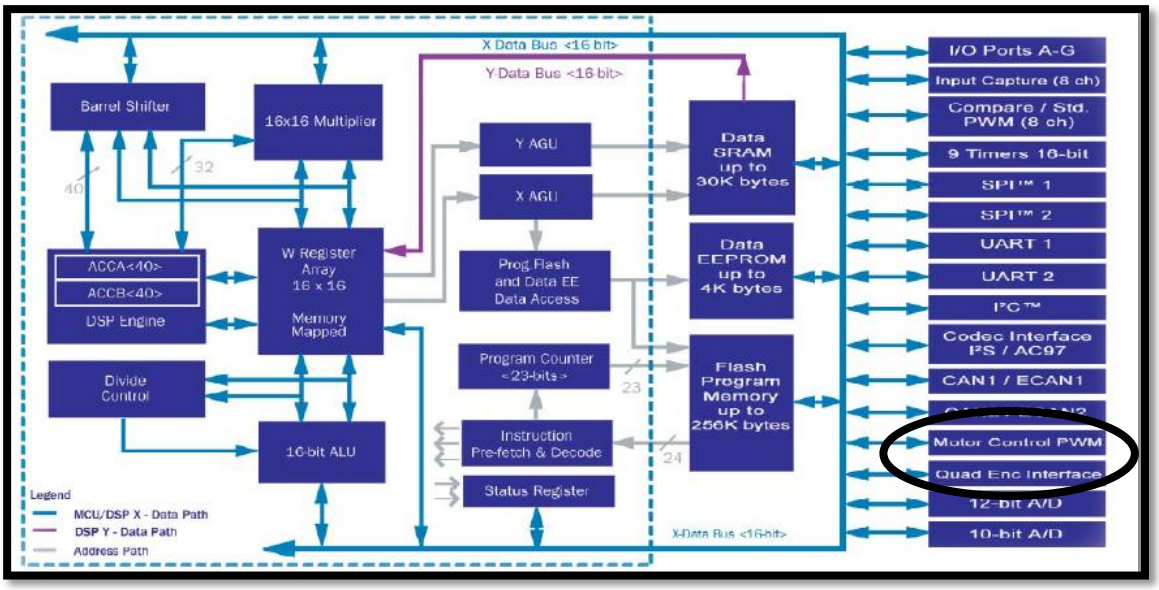


Figura 6. Diagrama de bloques, Familia dsPIC30F/dsPIC33F.

1.1.3. HMI.

El término HMI (Human Machine Interface, por sus siglas en inglés) ha sido acuñado en los últimos años para calificar una de las características más importantes de los sistemas de supervisión, control, diseño y simulación. El deseo de acercar el ordenador al hombre y de conseguir un mayor diálogo y ergonomía con los procesos de control ha fructificado en la aparición de entornos gráficos que mediante representaciones de las plantas y procesos permiten hacerse una idea bastante exacta de la ubicación de los operadores técnicos implantados así como los flujos de información en los procesos. Estas interfaces incorporan objetos gráficos a los que se les asocia. Los atributos de un objeto gráfico pueden ser: tamaño, color, movimiento, etc. (Manuel Ortega, 2001).

Con los últimos avances en visualización de datos se ha llegado a la máxima expresión del Panel Sinóptico con los ordenadores y las pantallas de visualización como estrella indiscutible de la función de diálogo entre el operador y el sistema.

La interface HMI se ha centrado principalmente en la interacción entre el operario y el ordenador, punto de contacto entre la persona y la tecnología. (Aquilino Rodríguez, 2007).

A nivel internacional no ha habido hasta ahora una línea clara a seguir acerca del diseño de las interfaces HMI. Han ido apareciendo múltiples iniciativas que pretendían cubrir necesidades concretas de diseño:

- ANSI American National Standards Institute
- CENELEC European Committee for Electrotechnical Standardization
- CEPT European Conference of Postal and Telecommunications
- ETSI European telecommunications Standards Institute
- IEEE Institute of Electronic and Electrical Engineers
- ISO International Standards Organization
- SAE Society of Automotive Engineers

2. CONTROL

Los sistemas de control son parte integral de la sociedad moderna y sus numerosas aplicaciones están alrededor de nosotros.

Un sistema de control está formado por subsistemas y procesos unidos con el fin de controlar la salida de los procesos. En su forma más sencilla un sistema de control produce una salida o respuesta para una entrada o estimulo dado, como se ilustra en la figura xx.

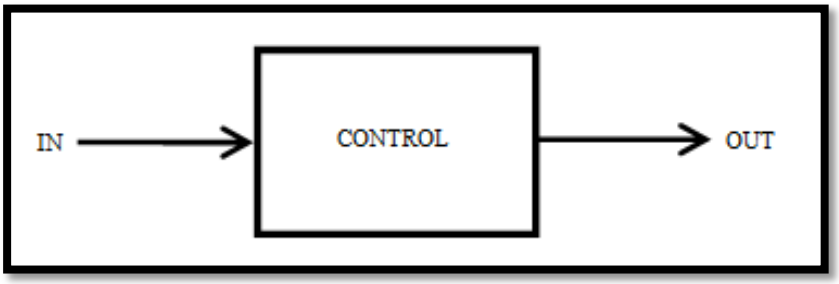


Figura 7. Sistema de Control

El diseño de sistemas de control es un promisorio campo de acción, en el cual es posible aplicar el talento en ingeniería, porque abarca numerosas disciplinas y gran cantidad de funciones dentro de esas disciplinas. El ingeniero de control se puede hallar en los puestos de mayor jerarquía de grandes proyectos, contratado para la fase conceptual a fin de determinar o estipular los requisitos de los sistemas generales. Estos requisitos comprenden especificaciones sobre la operación total de un sistema, funciones de subsistemas y la interconexión entre estas funciones, que incluye los requisitos de interface, el diseño de hardware y software, así como planes y procedimientos de prueba.

Los sistemas de control son dinámicos, responden a una entrada al experimentar una respuesta transitoria antes de llegar a una respuesta en estado estable, como se muestra en la figura, que, por lo general, se asemeja a la entrada. Existen tres objetivos principales del análisis y diseño de sistemas: producir la respuesta transitoria deseada, reducir el error en estado estable y alcanzar la estabilidad.

La respuesta transitoria es importante. En el caso de un elevador, una respuesta transitoria lenta impacienta a los pasajeros, mientras que una respuesta demasiado rápida los incomoda; si el elevador oscila en el piso por más de un segundo, puede resultar en una sensación desconcertante. Además una respuesta transitoria demasiado rápida podría ocasionar lesiones físicas a los pasajeros y a la estructura.

Primero se deben establecer definiciones cuantitativas para la respuesta transitoria. A continuación analizar el sistema por su respuesta transitoria existente. Por último, ajustar parámetros o diseñar componentes para obtener la respuesta transitoria deseada.

Otra meta del análisis y diseño se concentra en la respuesta en estado estable. Esta respuesta se asemeja a la entrada y suele ser lo que queda después de que las respuestas transitorias han decaído a cero. Primero se define

cuantitativamente el error en estado estable, después se analiza y, por último, se diseña una acción correctiva para reducirlo.

El estudio de respuesta transitoria y error en estado estable es discutible si el sistema no tiene estabilidad. Para explicar la estabilidad, se comienza desde el hecho de que la respuesta total de un sistema es la suma de la respuesta libre y la respuesta forzada. Una respuesta libre describe la forma en que el sistema disipa o adquiere energía; por otra parte, la forma o naturaleza de la respuesta forzada depende de la entrada.

Respuesta total = respuesta libre + respuesta forzada

Los sistemas de control deben ser diseñados para ser estables, esto es, su respuesta libre debe decaer a cero a medida que el tiempo se aproxima al infinito, u oscila.

Para el presente proyecto se tienen dos variables por controlar, que son, la velocidad de giro del husillo y la posición de la mesa, las cuales se muestran en la figura.

La velocidad será un control de lazo abierto en donde, por medio de la interfaz generada en LabVIEW®, se establece el “setpoint” de la velocidad requerida en RPM (revoluciones por minuto) y se enviará dicha información al microcontrolador, el cual, a su vez, convertirá esta señal en una señal de voltaje positivo por medio de PWM, moviendo de esta manera el motor a la velocidad deseada.

La posición de la mesa, así mismo, será introducida de la misma manera a través de la interfaz de LabVIEW® y seguirá el mismo proceso descrito para la velocidad, excepto que, esta vez, el “setpoint” de la posición es introducido por medio de valores enteros positivos representativos de la posición de la mesa en el eje de coordenadas X, que son enviados al microcontrolador vía usb-serial, el

cual, convierte la señal en valores de referencia, según la dirección de movimiento correspondiente para la mesa.

Figura 8. Variables de Control (Velocidad y Posición).

2.1. Generador de perfiles.

"Los perfiles de movimiento definen la trayectoria del vector posición, velocidad o aceleración en función del tiempo. El uso de perfiles surge de la necesidad de asegurar que las aceleraciones impuestas al conjunto motor-carga exijan un par de carga menor que el par eléctrico máximo del motor" (González, 2006).

Diferentes tipos de perfiles han transcurrido en el paso de los tiempos. Históricamente el perfil exponencial abarcó una amplia aplicación debido a que la referencia exponencial era fácil de obtener desde un simple circuito RC (González, 2006). Luego, con el advenimiento de los sistemas digitales, perfiles de "fácil calculo", perfiles lineales, como los triangulares y trapezoidales dominaron entre

los más usados (Fredriksen, 1974). Sin embargo, tanto en los perfiles trapezoidales como los exponenciales, las discontinuidades producidas en la aceleración provocan reducción en la vida útil del motor y en el sistema mecánico de acoplamiento (Geiger, 1980). En la actualidad, debido a la caracterización de la aceleración y su derivada, los perfiles con curvas suaves como el parabólico, el sinusoidal y el perfil basado en curvas-S son implementados en los sistemas de control de movimientos, aunque los mismos presenten desventajas, por lentitud y dificultad en la implementación (Hyung-Min y Seung-Ki, 2002) (M.C.Tsai et al., 2004).

2.1.1. Perfil trapezoidal.

Para asegurar un movimiento suave, evitar tirones o vibraciones del motor es necesario implementar un perfil de trayectoria, en nuestro caso se implementó un perfil trapezoidal, que además de proporcionar suavidad, restringe el tiempo de ejecución, dicha característica permite tener un mejor control y coordinación del motor (Barrera et al, 2000). Esto significa que el comienzo de movimiento es descrito por una determinada aceleración constante hasta alcanzar una velocidad máxima predefinida para luego permanecer constante y, finalmente, dar inicio a un frenado con desaceleración constante hasta lograr la posición deseada. La aceleración, la velocidad máxima y la posición final son enviados como comandos al microcontrolador.

Por lo tanto, el perfil trapezoidal se divide en tres partes importantes:

- Tramo 1(Ta): Aceleración constante, velocidad variable.
- Tramo 2(Ts): Aceleración cero y velocidad constante.
- Tramo 3(Td): Desaceleración constante y velocidad variable.

El perfil de velocidad trapezoidal se muestra en la figura.

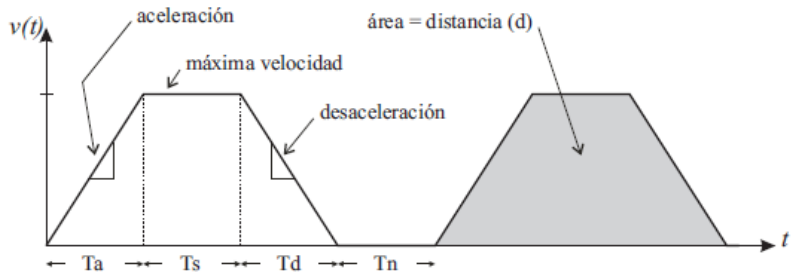


Figura 9. Perfil de velocidad trapezoidal.

$$v_{(t)} = \begin{cases} \frac{V_{máx}t}{T_a} & ,0 < t < T_a \\ V_{máx} & ,T_a < t < T_a + T_s \\ V_{máx}\left(1 - \frac{t-(T_s+T_a)}{T_d}\right) & ,T_a + T_s < t < T \end{cases} \quad (3.1)$$

donde

T_a tiempo de aceleración,

T_s tiempo de velocidad máxima,

T_d tiempo de desaceleración,

T tiempo total,

$V_{máx}$ velocidad máxima.

El desplazamiento total recorrido por el motor, D, surge de $\int_0^T v_{(t)}\,dt$, y resulta en (3.2)

$$D = V_{máx}\left(\frac{T_a}{2} + T_s + \frac{T_d}{2}\right) \quad (3.2)$$

La aceleración instantánea es la derivada de la velocidad y se muestra en (3.3):

$$a_{(t)} = \begin{cases} V_{máx}\frac{1}{T_a} & ,0 < t < T_a \\ 0 & ,T_a < t < T_a + T_s \\ -\frac{V_{máx}}{T_d} & ,T_a + T_s < t < T \end{cases} \quad (3.3)$$

Por lo apreciado en (3.3) se arriba a la aceleración máxima que se expresa en (3.4)

$$|a_{máx}| = \begin{cases} \frac{V_{máx}}{T_a} & ,T_d < T_a \\ \frac{V_{máx}}{T_d} & ,T_a < T_d \end{cases} \quad (3.4)$$

En el anexo xx se presenta la simulación y programación para un perfil de velocidad trapezoidal.

2.2. Control PID.

El control automático nace de la consideración del principio de realimentación el cual es bastante simple y muy poderoso. A lo largo de su historia, ha tenido una fuerte influencia en la evolución de la tecnología. Las aplicaciones del principio de realimentación han tenido éxito en los campos del control, comunicaciones e instrumentación. Para entender el concepto, asuma que el proceso es tal que cuando el valor de la variable manipulada se incrementa, entonces se incrementan los valores de las variables del proceso. Bajo este concepto simple, el principio de realimentación puede ser expresado como sigue:

“Incrementar la variable manipulada cuando la variable del proceso sea más pequeña que la referencia y disminuirla cuando ésta sea más grande.”

Control Proporcional.

La respuesta proporcional es la base de los tres modos de control PID, si los otros dos, control integral y control derivativo, están presentes, éstos son sumados a la respuesta proporcional. “Proporcional” significa que el cambio presente en la salida del controlador es algún múltiplo del porcentaje del cambio en la medición.

Este múltiplo es llamado “ganancia” del controlador. Para algunos controladores, la acción proporcional es establecida por medio de un ajuste de ganancia, mientras que para otros se usa una “banda proporcional”. Ambos tienen los mismos propósitos y efectos.

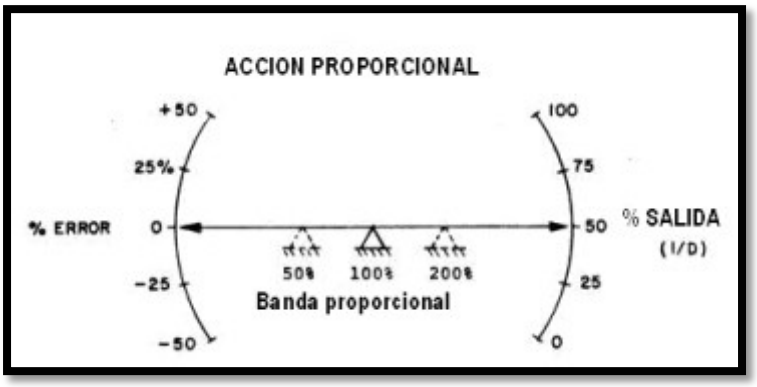


Figura 10. Acción Proporcional

La figura muestra la respuesta de un controlador proporcional por medio de un indicador de entrada/salida pivotando en una de estas posiciones. Con el pívot en el centro entre la entrada y la salida dentro del gráfico, un cambio del 100% en la medición es requerido para obtener un 100% de cambio en la salida, o un desplazamiento completo de la válvula. Un controlador ajustado para responder de ésta manera se dice que tiene una banda proporcional del 100%. Cuando el pívot es hacia la derecha, la medición de la entrada debería tener un cambio del 200% para poder obtener un cambio de salida completo desde el 0% al 100%, esto es una banda proporcional del 200%. Finalmente, si el pívot estuviera en la posición izquierda y si la medición se moviera sólo cerca del 50% de la escala, la salida cambiaría 100% en la escala. Esto es un valor de banda proporcional del 50%. Por lo tanto, cuanto más chica sea la banda proporcional, menor será la cantidad que la medición deba cambiar para el mismo tamaño de cambio en la medición. O, en otras palabras, menor banda proporcional implica mayor cambio de salida para el mismo tamaño de medición. Esta misma relación está representada por la figura. El gráfico muestra cómo la salida del controlador responde a medida que la medición se desvía del valor de consigna. Cada línea sobre el gráfico representa un ajuste particular de la banda proporcional. Dos propiedades básicas del control proporcional pueden ser observadas a partir de éste gráfico: Por cada valor de la banda proporcional, cada vez que la medición se iguala al valor de consigna, la salida es del 50%.

Cada valor de la banda proporcional define una relación única entre la medición y la salida. Por cada valor de medición existe un valor específico de salida. Por ejemplo, usando una línea de banda proporcional del 100%, cuando la medición está 25% por encima del valor de consigna, la salida del controlador deberá ser del 25%. La salida del controlador puede ser del 25% sólo si la medición esta 25% por encima del valor de consigna. De la misma manera, cuando la salida del controlador es del 25%, la medición será del 25% por encima del valor de consigna. En otras palabras, existe un valor específico de salida por cada valor de medición.

Idealmente, la banda proporcional correcta producirá una amortiguación de amplitud de un cuarto de ciclo en cada ciclo, en el cual cada medio ciclo es la mitad de la amplitud del medio ciclo previo. La banda proporcional que causará una amortiguación de onda de un cuarto de ciclo será menor, y por lo tanto alcanzará un control más ajustado sobre la variable medida, a medida que el tiempo muerto en el proceso decrece y la capacidad se incrementa .

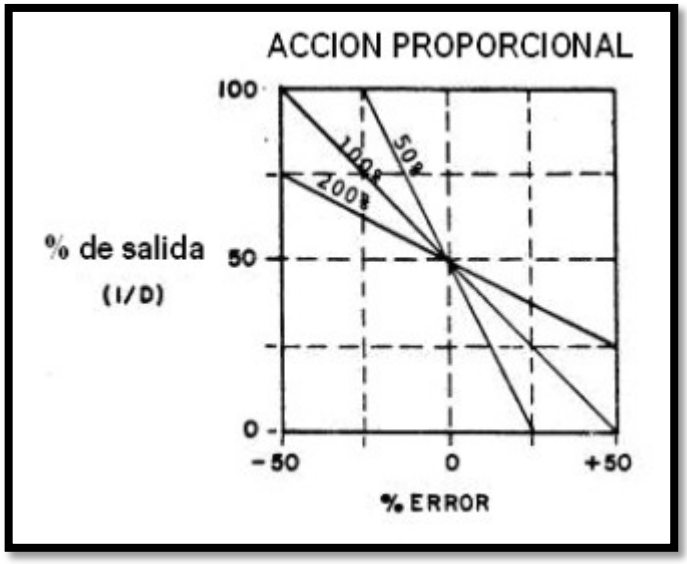


Figura 11. Gráfica Error VS Salida.

Una consecuencia de la aplicación del control proporcional al lazo básico de control es el “offset”, que significa que el controlador mantendrá la medida a un

valor diferente del valor de consigna. Esto es más fácilmente visto al observar la figura.

Note que si la válvula de carga es abierta, el caudal se incrementará a través de la válvula y el nivel comenzará a caer, de manera de mantener el nivel, la válvula de suministro debería abrirse, pero teniendo en cuenta la acción proporcional del lazo el incremento en la posición de apertura puede sólo ser alcanzado a un nivel menor.

En otras palabras, para restaurar el balance entre el caudal de entrada y el de salida, el nivel se debe estabilizar a un valor debajo del valor de consigna (o setpoint). Esta diferencia, que será mantenida por el lazo de control, es llamada offset, y es característica de la aplicación del control proporcional único en los lazos de realimentación. La aceptabilidad de los controles sólo-proporcionales dependen de si este valor de offset será o no tolerado, ya que el error necesario para producir cualquier salida disminuye con la banda proporcional, cuanto menor sea la banda proporcional, menor será el offset. Para grandes capacidades, aplicaciones de tiempo muerto pequeñas que acepten una banda proporcional muy estrecha, el control sólo proporcional será probablemente satisfactorio dado que la medición se mantendrá a una banda de un pequeño porcentaje alrededor del valor de consigna.

Control Integral

Cuando es necesario que no haya una diferencia de estado estable entre la medición y el valor de consigna bajo todas las condiciones de carga, una función adicional deberá ser agregada al controlador proporcional, esta función es llamada acción integral. La respuesta del lazo abierto del modo integral es mostrada en la figura, que indica un escalón de cambio en algún instante en el tiempo. En tanto que la medición estuviera en su valor de consigna, no existiría ningún cambio en la salida debido al modo integral del controlador.

Sin embargo, cuando cualquier error exista entre la medición y el valor de consigna, la acción integral hace que la salida comience a cambiar y continúe

cambiando en tanto el error exista. Esta función, entonces, actúa sobre la salida para que cambie hasta un valor correcto necesario para mantener la medición en el valor de consigna. Esta respuesta es agregada a la banda proporcional del controlador según se muestra en la figura.

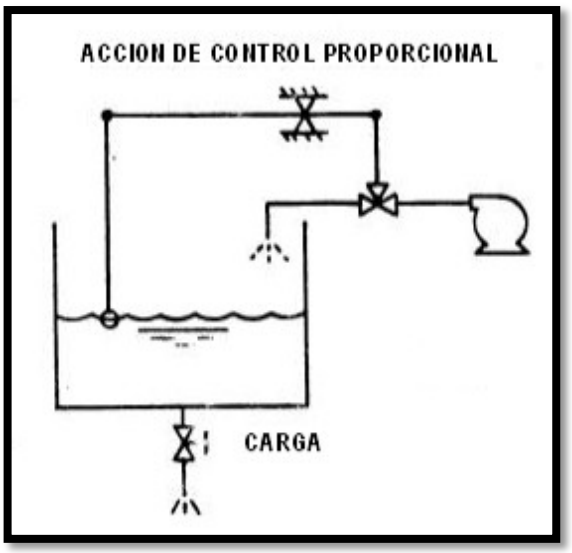


Figura 12. Control Proporcional de Nivel

El escalón de cambio en la medición primero produce una respuesta proporcional, y luego una respuesta integral es agregada a la proporcional. Cuanta más acción integral exista en el controlador, mas rápido cambia la salida en función del tiempo

Acción Derivativa.

La tercera respuesta encontrada en controladores es la acción derivativa. Así como la respuesta proporcional responde al tamaño del error y el Integral responde al tamaño y duración del error, el modo derivativo responde a cuán rápido cambia el error. En la figura 10, dos respuestas derivativas son mostradas.

La primera es una respuesta a un corte en la medición alejada del valor de consigna. Para un escalón, la medición cambia en forma infinitamente rápida, y el modo derivativo del controlador produce un cambio muy grande y repentino en la

salida, que muere inmediatamente debido a que la medición ha dejado de cambiar luego del escalón. La segunda respuesta muestra la respuesta del modo derivativo a una medición que está cambiando a un régimen constante. La salida derivativa es proporcional al régimen de cambio de éste error. Cuanto mayor sea el cambio, mayor será la salida debido a la acción derivativa. La acción derivativa mantiene ésta salida mientras la medición esté cambiando. Tan pronto como la medición deja de cambiar, esté o no en el valor de consigna, la respuesta debido a la acción derivativa cesará.

Entre todas las marcas de controladores, la respuesta derivativa es comúnmente medida en minutos como se indica en la figura. El tiempo derivativo en minutos es el tiempo que la respuesta proporcional del lazo abierto mas la respuesta derivativa está delante de la respuesta resultante del valor proporcional solamente. Así, cuanto más grande sea el número derivativo mayor será la respuesta derivativa. Los cambios en el error son un resultado de los cambios tanto en el valor de consigna como en la medición o en ambos. Para evitar un gran pico causado por las escalones de cambio en el valor de consigna, la mayoría de los controladores modernos aplican la acción derivativo sólo a cambios en la medición.

La acción derivativa en los controladores ayuda a controlar procesos con constantes de tiempo especialmente grandes y tiempo muerto significativo, la acción derivativa es innecesaria en aquellos procesos que responden rápidamente al movimiento de la válvula de control, y no puede ser usado en absoluto en procesos con ruido en la señal de medición, tales como caudal, ya que la acción derivativa en el controlador responderá a los cambios bruscos en la medición que el mismo observa en el ruido. Esto causará variaciones rápidas y grandes en la salida del controlador, lo que hará que la válvula esté constantemente moviéndose hacia arriba o hacia abajo, produciendo un desgaste innecesario en la misma.

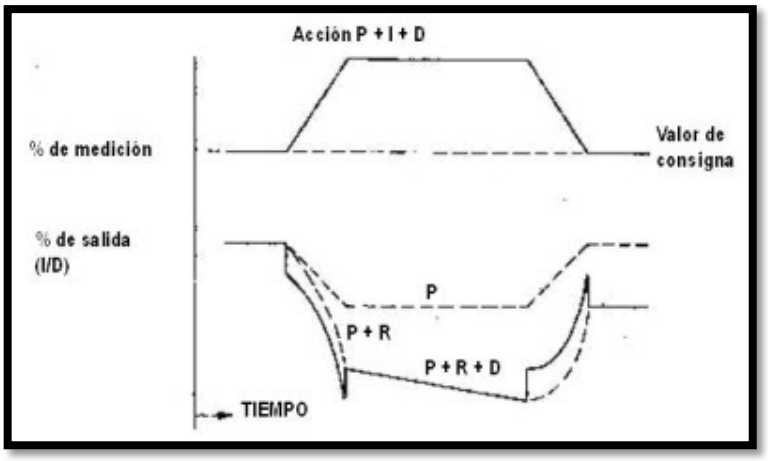


Figura 13. Control PID.

La figura 13 muestra una acción combinada de respuesta proporcional, integral y acción derivativa para la medición de temperatura de un intercambiador de calor simulado que se desvía del valor de consigna debido a un cambio de carga. Cuando la medición comienza a desviarse del valor de consigna, la primera respuesta del controlador es una respuesta derivativa proporcional al régimen de variación de la medición que se opone al movimiento de la medición al alejarse del valor de consigna.

La respuesta derivativa es combinada con la respuesta proporcional agregada, a medida que la integral en el controlador ve el error incrementarse, el mismo controla la válvula más fuerte aún. La acción continua hasta que la medición deja de cambiar, entonces la acción derivativa se detiene. Dado que existe aún un error, la medición continúa cambiando debido a la integral, hasta que la medición comienza a retornar hacia el valor de consigna.

Tan pronto como la medición comienza a moverse retornando hacia el valor de consigna, aparece una acción derivativa proporcional al régimen de cambio en la variación oponiéndose al retorno de la medición hacia el valor de consigna. La acción integral continúa debido a que aún existe un error, a pesar de que su contribución disminuye con este. Además, la salida debido al valor proporcional está cambiando. Así, la medición retorna hacia el valor de consigna. Tan pronto

como la medición alcanza el valor de consigna y deja de cambiar, la acción derivativa cesa nuevamente y la salida proporcional vuelve al 50%. Con la medición nuevamente en su valor de consigna, no existen más respuestas a variaciones debidas al valor integral. Sin embargo, la salida está ahora a un nuevo valor. El nuevo valor es el resultado de la acción integradora durante el tiempo en que la medición se alejó del valor de consigna, y compensa el cambio de carga que fue causado por la alteración original.

Finalmente se puede indicar que pese a que se ha indicado la forma de operar de tres modos de control se deberá tener un claro concepto de los siguientes puntos.

1. Para alcanzar el control automático, el lazo de con trol deberá estar cerrado.
2. Para tener una lazo realimentado de control estable, el ajuste más importante del controlador es la selección de la acción correcta, sea directa o inversa.
3. El valor correcto de los ajustes de banda proporcional, Integral, y tiempo derivativo dependen de las características del proceso, cabe consignar que en los controladores actuales dichos valores se pueden detectar en forma automática, ya que el controlador dispone de un modo en que produce alteraciones controladas, y dentro de ciertos límites establecidos previamente por el operario, en la salida se miden los resultados del proceso para una cierta cantidad de ciclos de alteración, en base a éste comportamiento puede detectar cuál es el mejor conjunto de ajustes para controlar un proceso mediante el software interno del aparato .
4. La función del modo integral es para eliminar el offset. Si mucho valor de offset es usado, el resultado será una oscilación de la medición. Si un valor muy bajo de reset es usado, el resultado será que la medición retorna al valor de consigna más lentamente que lo posible.
5. El modo derivativo se opone a cualquier cambio en la medición. Una acción derivativa muy pequeña no tiene efecto significativo, una acción con

valores muy altos provoca una respuesta excesiva del controlador y un ciclo en la medición.

Para el presente proyecto, se han calculado las variables kp, kd y ki del controlador PID, utilizado para el control de posición de la mesa de la bancada, por medio de un ensayo de prueba y error, como lo muestra la figura 12, donde se introducen las variables en un programa desarrollado en lenguaje C para MPLab que se detalla en el capítulo 4 en la sección apartada para este tema.

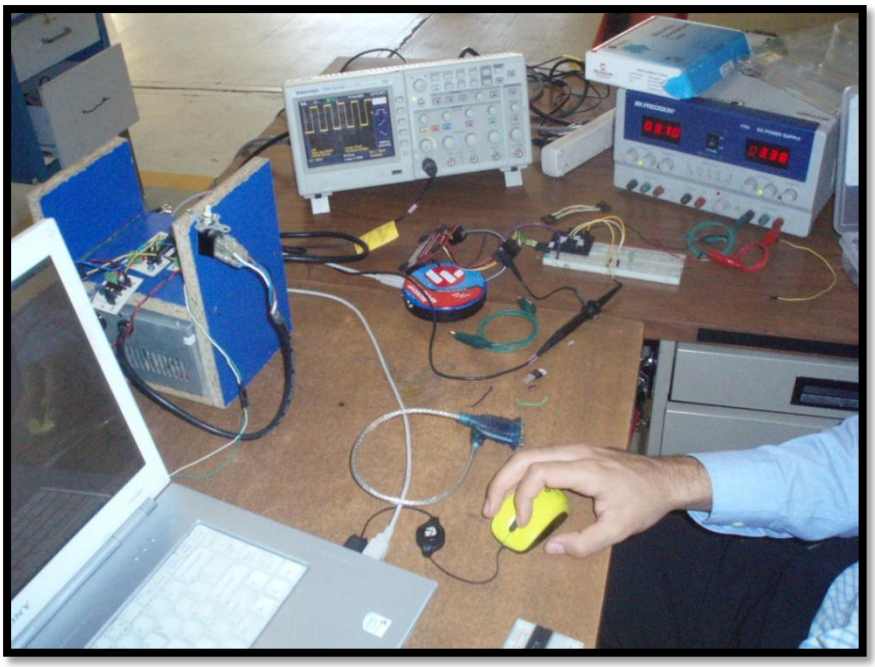


Figura 14. Estimación de los parámetros del controlador PID

2.3. Modelo Matemático

Para hacer un modelo matemático del sistema, el diseñador utiliza leyes físicas, por ejemplo, las leyes de Kirchhoff para redes eléctricas y la ley de Newton para sistemas mecánicos, junto con suposiciones de simplificación. Estas leyes son las siguientes:

- Ley de voltajes de Kirchhoff: la suma de los voltajes alrededor de una trayectoria cerrada es igual a cero.
- Ley de corrientes de Kirchhoff: la suma de corrientes que fluyen desde un nodo es igual a cero.
- Ley de Newton: la suma de fuerzas que actúan sobre un cuerpo es igual a cero; la suma de momentos sobre un cuerpo es cero.

Tales leyes de Kirchhoff y Newton llevan a modelos matemáticos que describen la relación entre la entrada y salida de sistemas dinámicos. Uno de estos modelos es la ecuación diferencial lineal e invariante con el tiempo.

Las suposiciones de simplificación hechas en el proceso de obtener un modelo matemático suelen llevar a una forma de bajo orden. Sin las suposiciones, el modelo del sistema podría ser de orden superior o descrito con ecuaciones diferenciales no lineales, variantes con el tiempo o en derivadas parciales. Estas ecuaciones complican el proceso de diseño y reducen la agudeza del diseñador. Desde luego, todas las suposiciones deben verificarse y todas las simplificaciones justificarse por medio de análisis y prueba.

Además de la ecuación diferencial la función de transferencia es otra forma de hacer un modelo matemático de un sistema. El modelo se deduce de la ecuación diferencial lineal e invariante en el tiempo si se usa lo que llamamos transformada de Laplace. Aun cuando la función de transferencia se puede emplear solo para sistemas lineales, produce una información más intuitiva que la ecuación diferencial. La función de transferencia también es útil para hacer un modelo para la interconexión de subsistemas, al formar un diagrama de bloques con una función matemática dentro de cada bloque.

Otro modelo mas es la representación en el espacio de estados. Una ventaja de los métodos en el espacio de estados es que también se pueden usar para sistemas que no se pueden describir por medio de ecuaciones diferenciales.

Además, estos métodos se emplean para modelar sistemas para simulación en computadora. Básicamente, esta representación convierte una ecuación diferencial de orden n en un sistema de n ecuaciones diferenciales de primer orden.

Por último debemos mencionar que para obtener el modelo matemático para un sistema requerimos el conocimiento de los valores de los parámetros como resistencia equivalente, inductancia, masa y amortiguamiento, que con frecuencia no son fáciles de obtener. El análisis, las mediciones o las especificaciones proporcionadas por vendedores son fuentes que el ingeniero de sistema de control puede usar para obtener los parámetros.

2.4. Función de Transferencia

La función de transferencia es la forma básica de describir modelos de sistemas lineales y está basada en la transformada de Laplace permitiendo obtener la respuesta temporal, la respuesta estática y la respuesta en frecuencia. El análisis de distintas descomposiciones de la respuesta temporal permite adquirir útiles ideas cualitativas, y definir importantes conceptos: efectos de las condiciones iniciales, respuestas libre y forzada, regímenes permanente y transitorio. También permite definir el concepto central de estabilidad, y establecer un primer criterio para su investigación.

Esta función permite la separación de la entrada, el sistema y la salida en tres partes separadas y distintas, a diferencia de la ecuación diferencial. La función también nos permitirá algebraicamente combinar representaciones matemáticas de los subsistemas para obtener una representación total del sistema.

La función de transferencia se puede representar como un diagrama de bloques con la entrada en la izquierda, la salida a la derecha y la función de transferencia del sistema dentro del bloque.

Figura 15. Diagrama de Bloques de la FT.

El denominador de la función de transferencia es idéntico al polinomio característico de la ecuación diferencial. Del mismo modo, podemos hallar la salida, C(s), usando:

$$C(s)=E(s)G(s)$$

La función de transferencia de un sistema descrito mediante una ecuación diferencial lineal invariente con el tiempo, se define como:

Cociente entre la tranformada de Laplace de la salida (función de respuesta) y la transformada de la Laplace de la entrada (función de excitación), considerando condiciones iniciales nulas.

$$G(s) = \frac{L[salida]}{L[entrada]}\Bigg]_{condiciones\ iniciales\ nulas}$$

$$= \frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + ... + b_{m-1} s + b_m}{a_0 s^n + a_1 s^{n-1} + ... + a_{n-1} s + a_n}$$

Figura 16. Función de Transferencia

Consideraciones

1.- La función de transferencia de un sistema, es un modelo matemático; es un método para expresar la ecuación diferencial que relaciona la variable de salida con la variable de entrada.

2.- Es una propiedad de un sistema, independiente de su magnitud y naturaleza de la función de entrada.

3.- Incluye las unidades necesarias para relacionar la entrada con la salida; pero no proporciona información de la estructura física del sistema. (Las funciones de transferencia de muchos sistemas físicamente diferente pueden ser idénticas).

4.- Si se conoce la función de transferencia de un sistema, se estudia la salida o respuesta para varias formas de entrada, con la intención de comprender la naturaleza del sistema.

5.- Si se desconoce la función de transferencia de un sistema, puede establecerse experimentalmente, introduciendo entradas conocidas y estudiando la salida del sistema.

6.- Una vez obtenida la función de transferencia, tendremos una descripción completa de las características dinámicas del sistema, a diferencia de su descripción física.

2.4.1. Estabilidad.

La estabilidad es la especificación más importante de un sistema. Si un sistema es inestable la respuesta transitoria y los errores en estado estable son puntos debatibles.

No se puede diseñar un sistema inestable para un requerimiento específico de respuesta transitoria o de error en estado estable.

1.- Un sistema lineal e invariante con el tiempo es estable si la respuesta libre tiende a cero conforme el tiempo tiende al infinito.

2.- Un sistema lineal e invariante con el tiempo es inestable si la respuesta libre crece sin límite conforme el tiempo tiende al infinito.

3.- Un sistema lineal e invariante con el tiempo es marginalmente estable si la respuesta libre no decae ni crece, sino que permanece constante o varia a medida que el tiempo tiende al infinito.

Entonces, la definición de estabilidad implica que solo la respuesta forzada permanece a medida que la respuesta libre tiende a cero.

Estas definiciones se apoyan solo en una descripción de la respuesta libre. Cuando se ve una respuesta total, puede ser difícil separar la respuesta libre de la respuesta forzada, pero nos damos cuenta que si la entrada es acotada y la respuesta total no tiende al infinito a medida que el tiempo tiende al infinito, entonces la repuesta libre obviamente no tiende al infinito. Si la entrada no es acotada, vemos una respuesta total no acotada y no es posible llegar a una conclusión sobre la estabilidad del sistema: no podemos saber si la respuesta total no es acotada porque la respuesta forzada no es acotada o por que la respuesta libre no es acotada. Entonces, nuestra definición alterna de estabilidad, aquella que considera la respuesta total e implica la primera definición basada en la respuesta libre, es la siguiente:

"Un sistema es estable si toda entrada acotada produce una salida acotada."

A este enunciado se le da el nombre de definición de estabilidad de entrada acotada, salida acotada (BIBO siglas en inglés de bounded-input, bounded-output).

Entonces, nuestra definición alterna de inestabilidad, aquella que considera la respeta total, es la siguiente:

"Un sistema es inestable si cualquier entrada acotada
produce una salida no acotada."

Estas definiciones ayudan a aclarar la definición previa de estabilidad marginal: la estabilidad marginal realmente significa que el sistema es estable para algunas entradas acotadas, e inestable para otras.

Físicamente, un sistema inestable cuya respuesta libre crece sin límite puede causar daños al sistema, a una propiedad adyacente o a las personas.

En la figura 17 se muestran los gráficos representativos del comportamiento de las variables en los sistemas estable, inestable y marginalmente estable.

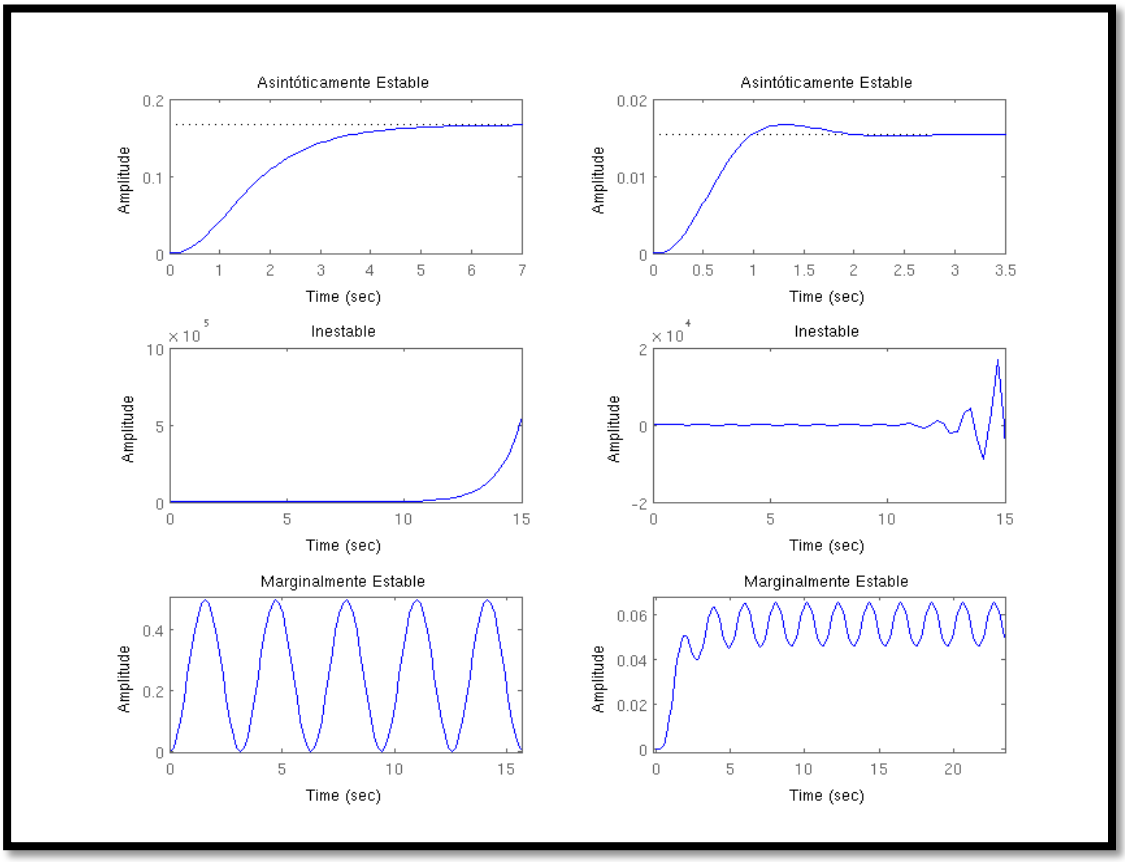


Figura 17. Estabilidad e Inestabilidad.

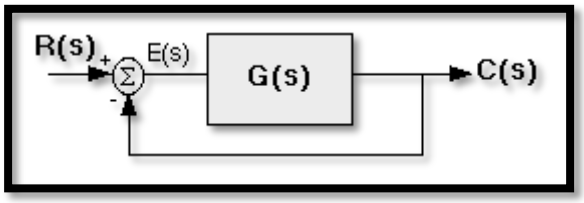
2.4.2. Error.

El error en estado estacionario es una medida de la exactitud de un sistema de control para seguir una entrada dada, después de desaparecer la respuesta transitoria.

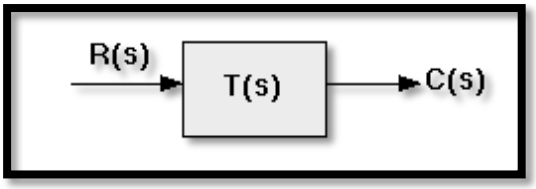
El que un sistema dado presente o no un error en estado estacionario ante determinado tipo de señal de entrada, depende del tipo de función de transferencia de lazo abierto del sistema.

El error de estado estacionario se define como la diferencia entre la entrada y la salida de un sistema en el límite cuando el tiempo tiende a infinito (e.d. cuando la respuesta ha alcanzado el estado estacionario). El error de estado estacionario dependerá del tipo de entrada (escalón, rampa, etc.) y de (tipo del sistema) que el sistema sea del tipo 0, I, II,...

Antes de exponer acerca de las relaciones entre error de estado estacionario y tipo del sistema, se mostrará cómo calcular el error sin importar el tipo del sistema o la entrada empleada. Entonces, derivaremos las fórmulas a aplicar en el análisis de error de estado estacionario. El error de estado estacionario puede calcularse de la función de transferencia a lazo cerrado o abierto para sistemas con realimentación unitaria. Por ejemplo, digamos que tenemos el siguiente sistema:



…el cual es equivalente al siguiente sistema:



Podemos calcular el error de estado estacionario para este sistema ya sea de la función de transferencia a lazo cerrado o abierto mediante el teorema del

valor final (recuerde que este teorema solo puede aplicarse si el denominador no tiene polos en el semiplano derecho):

$$e(\infty) = \lim_{s \to 0} \frac{sR(s)}{1 + G(s)}$$

$$e(\infty) = \lim_{s \to 0} sR(s)[1 - T(s)]$$

Ahora, introduzcamos las transformadas de Laplace de las diferentes entradas para hallar las ecuaciones que nos permitan calcular los errores de estado estacionario a partir de las funciones de transferencia a lazo abierto frente a diferentes entradas:

- Entrada Escalón (R(s) = 1/s):

$$e(\infty) = \frac{1}{1 + \lim_{s \to 0} G(s)} = \frac{1}{1 + K_p} \Rightarrow K_p = \lim_{s \to 0} G(s)$$

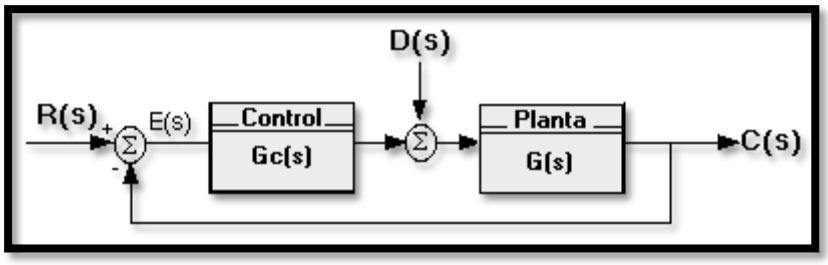
- Entrada Rampa (R(s) = 1/s^2):

$$e(\infty) = \frac{1}{1 + \lim_{s \to 0} G(s)} = \frac{1}{K_r} \Rightarrow K_r = \lim_{s \to 0} sG(s)$$

- Entrada Parabólica (R(s) = 1/s^3):

$$e(\infty) = \frac{1}{\lim_{s \to 0} s^2 G(s)} = \frac{1}{K_\alpha} \Rightarrow K_\alpha = \lim_{s \to 0} s^2 G(s)$$

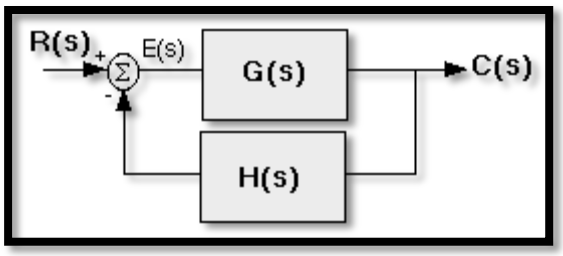
Cuando se diseña un controlador, normalmente se quiere compensar el sistema frente a perturbaciones. Digamos que tenemos el siguiente sistema con una perturbación:



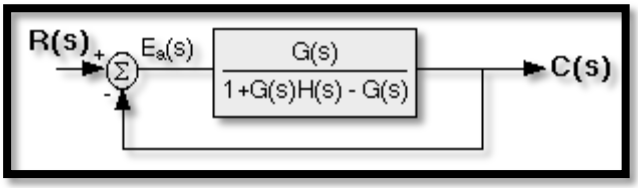
Podemos encontrar el error de estado estacionario para una entrada perturbación de un escalón con la siguiente ecuación:

$$e = \frac{1}{\lim_{s\to 0}\frac{1}{G(s)} + \lim_{s\to 0} G_\delta(s)}$$

Finalmente, podemos calcular el error de estado estacionario para sistemas con realimentación no unitaria:



Manipulando los bloques, podemos modelar el sistema como sigue:



Ahora, simplemente aplique las ecuaciones que mencionáramos arriba.

Si se refiere de nuevo a las ecuaciones para el cálculo de errores de estado estacionario para sistemas con realimentación unitaria, hallará que tenemos definidas ciertas constantes (conocidas como las constantes estáticas de error). Estas constantes son la constante de posición (Kp), la constante de velocidad

(Kv), y la constante de aceleración (Ka). Sabiendo el valor de estas constantes además del tipo del sistema, podemos predecir si el sistema va a tener un error de estado estacionario finito.

Primero, hablemos del tipo sistema. Se define como la cantidad de integradores puros en un sistema. Esto es, el tipo del sistema es igual al valor de n cuando el sistema se representa de la siguiente forma:

Por lo tanto, un sistema puede ser de tipo 0, de tipo 1, etc. A continuación se muestra cómo se relaciona un error de estado estacionario con el tipo de los sistemas:

Sistemas de tipo 0	***Entrada Escalón***	***Entrada Rampa***	***Entrada Parabólica***
Formula de error de estado estacionario	1/(1+Kp)	1/Kv	1/Ka
Constante Estática del Error	Kp = constante	Kv = 0	Ka = 0
Error	1/(1+Kp)	infinito	infinito

Tabla 1. Sistemas de Tipo 0.

Sistemas de tipo 1	***Entrada Escalón***	***Entrada Rampa***	***Entrada Parabólica***
Formula de error de estado	1/(1+Kp)	1/Kv	1/Ka

estacionario			
Constante Estática del Error	Kp = infinito	Kv = constante	Ka = 0
Error	0	1/Kv	infinito

Tabla 2. Sistemas de Tipo 1.

Sistemas de tipo 2	***Entrada Escalón***	***Entrada Rampa***	***Entrada Parabólica***
Formula de error de estado estacionario	1/(1+Kp)	1/Kv	1/Ka
Constante Estática del Error	Kp = infinito	Kv = infinito	Ka = constante
Error	0	0	1/Ka

Tabla 3. Sistemas de Tipo 2.

2.5. Modelado de un Motor de DC.

Un actuador mecánico muy difundido es el motor de DC. Provee directamente movimiento rotacional y, adecuadamente acondicionado, movimiento traslacional.

El circuito eléctrico de armadura y el diagrama mecánico rotacional, se muestran en la figura xx:

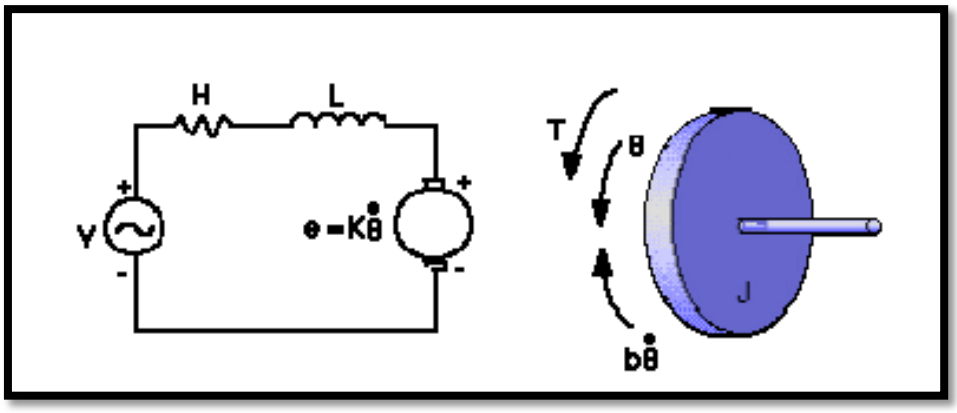


Figura 18. Modelado de un Motor de DC.

Para el ejemplo se consideraron los siguientes parámetros:

* Momento de inercia del sistema (J) = 0.01 kg.m^2/s^2
* Coeficiente de roce (b) = 0.1 Nms
* Constante de fuerza electromotriz (FEM) (K = Ke = Kt) = 0.01 Nm/Amp
* Resistencia de armadura (R) = 1 ohm
* Inductancia de armadura (L) = 0.5 H
* Entrada (V): Fuente de Tensión
* Posición del eje: Q
* Se supone rotor y eje rígidos.

La cupla (T) está relacionada con la corriente de armadura y la fem (e) con la velocidad de rotación, según las ecuaciones:

$$T = K_t \cdot i$$

$$e = K_e \cdot \dot{\theta}$$

…siendo ambas constantes iguales (Kt=Ke=K).

En base a la ley de Newton y la ley de Kirchoff, resultan las siguientes ecuaciones diferenciales que describen la dinámica del sistema:

$$J \cdot \ddot{\theta} + b \cdot \dot{\theta} = K \cdot i$$

$$L \cdot \frac{di}{dt} + R \cdot i = V - K \cdot \dot{\theta}$$

Función de Transferencia

Aplicando la Transformada de Laplace y haciendo cero las condiciones iniciales, las ecuaciones del sistema quedan expresadas en el dominio de s:

$$s \cdot (J \cdot s + b) \cdot \vartheta(s) = K \cdot I(s)$$

$$(L \cdot s + R) \cdot I(s) = V - K \cdot s \cdot \vartheta(s)$$

Eliminando I(s) se obtiene la transferencia entre la entrada de tensión de armadura V y la velocidad de rotación Θ como salida:

$$\frac{\dot{\vartheta}}{V} = \frac{K}{(J \cdot s + b) \cdot (L \cdot s + R) + K^2}$$

Espacio de Estados

La descripción del sistema de estados en el dominio temporal puede obtenerse definiendo las variables físicas velocidad de rotación Θ(t) y corriente de armadura i(t), como variables de estado, la tensión de armadura v(t) como entrada y la velocidad de rotación como salida:

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & -\frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix} \cdot \begin{bmatrix}\dot{\theta} \\ i\end{bmatrix} + \begin{bmatrix}0 \\ 1/L\end{bmatrix} \cdot v$$

Requerimientos de Diseño

El motor sin compensar puede rotar solamente a 0,1rad/s con una entrada de 1 Volt (ver simulación de la planta a lazo abierto). Uno de los requerimientos es que en estado estacionario presente un error respecto de la velocidad deseada menor que el 1%. Dinámicamente se espera un tiempo de establecimiento de 2 seg y un sobrepaso menor que el 5% para evitar daños en la máquina. Es decir:

tiempo de establecimiento de 2 seg
sobrepaso menor que el 5%
Error de estado estacionario 1%

3. Instrumentación.

3.1. Servomotores.

Los sistemas automatizados como son los equipos de control numérico por computadora (CNC) se basan en acciones de control de posición y velocidad, para lo cual emplean servomotores, los servomotores requieren de un driver para poder ser operados desde el computador. Los servomotores por sus características son ideales para el control de posición y velocidad estos son mucho más caros que un motor normal, se fabrican para DC y AC, en el caso de DC existen con y sin carbones, de estos tres tipos los más económicos son los de DC con carbones, estos tienen la desventaja de requerir mantenimiento por el desgaste natural de los carbones.

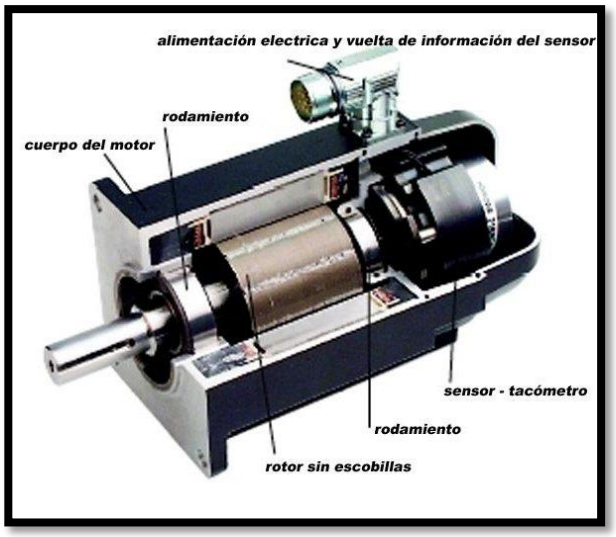


Figura 19. Estructura interna de un Servomotor.

Los motores de DC convierten la energía eléctrica en energía mecánica, o mejor dicho, convierten corriente en torque. Los parámetros de un motor dc son: torque constante, Kt, la resistencia de la armadura, r, el momento de la inercia, Jm, y los niveles máximos del torque. La constante Kt está representada en unidades Nm/A o oz-in/A, e indica la cantidad de torque que el motor genera por cada Ampere.

La resistencia de la armadura está representada en Ohms. El momento de inercia es la suma de los momentos de inercia de las partes rotatorias del motor y se expresa en unidades de Kg-m2 o oz-in-s2.

Los motores están clasificados por el nivel de torque que pueden producir. La capacidad de torque se expresa en dos parámetros: torque continuo y torque pico. El torque continuo es el torque que el motor puede producir continuamente, a valores nominales, sin presentar sobrecalentamiento. En cambio, el torque pico, es el torque máximo que puede ser generado en un periodo corto de tiempo sin causar daños mecánicos.

La corriente con la que el motor es alimentado, es generada por un amplificador de potencia. El cual es descrito en el siguiente tema.

Las especificaciones características de los motores implementados se muestran a continuación.

Para la obtención de los parámetros del motor se pueden usar distintos métodos. La primera y más sencilla, es buscar la hoja característica que provee fabricante. La segunda es por medio de pruebas eléctricas y mediciones de parámetros con instrumentación especializada.

Para la adquisición de las señales es importante conocer las características del “encoder”. Motivo por el cual se analizó con un osciloscopio cada una de las terminales del “encoder”. En la figura 20 se muestra la caracterización del “encoder”.

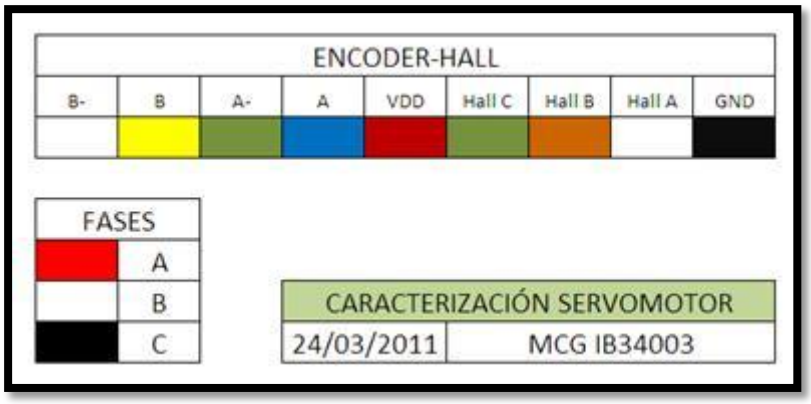


Figura 20. Caracterización del Motor.

3.2. Microcontroladores dsPIC.

Un microcontrolador dsPIC es un DSC de la familia de microcontroladores de 16-bits, manufacturado por “Microchip technology Inc.”. Es un dispositivo electrónico, que integra a la perfección los atributos de control de un microcontrolador (MCU) con las capacidades de cómputo y el rendimiento de un procesador de señal digital (con siglas en inglés, DSP) en un solo núcleo, adquiriendo de esta forma cualidades de ambas tecnologías (Figura 21).

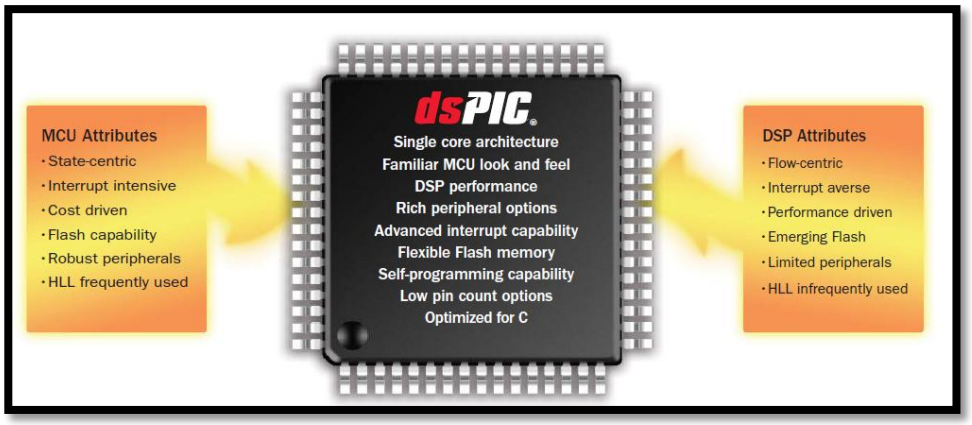


Figura 21. Características especiales para un dsPIC.

DsPIC de Microchip, ofrece todos los atributos que se pueden esperan de un poderoso microcontrolador de 16-bit MCU: manejo rápido y flexible de la interrupciones; una amplia gama defunciones periféricas analógicas y digitales; gestión de energía; flexibilidad de programación; opciones flexibles de reloj; “power-on-reset”; “brown-out protection”; temporizador “watchdog”; seguridad de código; simulación y funcionamiento en tiempo real; y la máxima velocidad en depuración entre otras.

Es importante mencionar, que el DSC siendo la combinación de tecnología entre MCU y DSP mantiene un nivel intermedio de costo, otro punto a favor para justificar su uso en este proyecto. En la Figura 22 se muestra gráficamente lo antes mencionado.

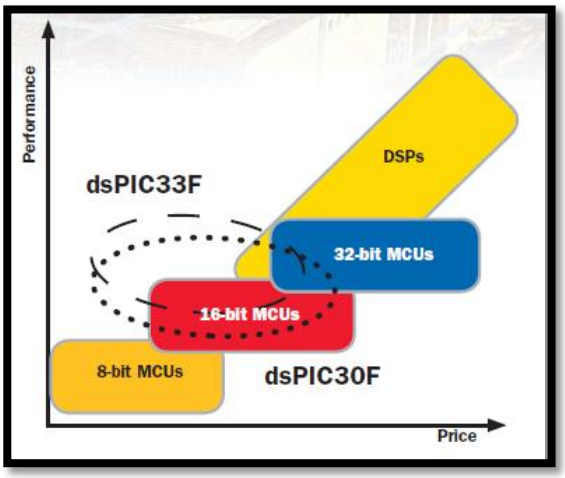


Figura 22. Grafica costo-rendimiento de tecnología “Microchip”

3.2.1. dsPIC33FJ128MC802.

Como ya se mencionó anteriormente, el dsPIC33FJ128MC802 fue el candidato perfecto para el desenlace del proyecto, pero aún no se han mencionado los detalles justificando su elección.

En la tabla 4 se presenta una comparativa de características entre algunos dispositivos de la familia dsPIC33F dedicados al control de motores. Se puede observar que el dsPIC33FJ128MC802 contiene los módulos indispensables para lograr el control.

Device	Pins	Program Flash Memory (Kbyte)	RAM (Kbyte)(1)	Remappable Peripheral										RTCC	I²C™	CRC Generator	10-bit/12-bit ADC (Channels)	6-pin 16-bit DAC	Analog Comparator (2 Channels/Voltage Regulator)	8-bit Parallel Master Port (Address Lines)	I/O Pins	Packages
				Remappable Pins	16-bit Timer(2)	Input Capture	Output Compare Standard PWM	Motor Control PWM (Channels)(3)	Quadrature Encoder Interface	UART	SPI	ECAN™	External Interrupts(4)									
dsPIC33FJ128MC804	44	128	16	26	5	4	4	6, 2	2	2	2	1	3	1	1	1	9	1	1/1	11	35	QFN TQFP
dsPIC33FJ128MC802	28	128	16	16	5	4	4	6, 2	2	2	2	1	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ128MC204	44	128	8	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ128MC202	28	128	8	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ64MC804	44	64	16	26	5	4	4	6, 2	2	2	2	1	3	1	1	1	9	1	1/1	11	35	QFN TQFP
dsPIC33FJ64MC802	28	64	16	16	5	4	4	6, 2	2	2	2	1	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ64MC204	44	64	8	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ64MC202	28	64	8	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S
dsPIC33FJ32MC304	44	32	4	26	5	4	4	6, 2	2	2	2	0	3	1	1	1	9	0	1/1	11	35	QFN TQFP
dsPIC33FJ32MC302	28	32	4	16	5	4	4	6, 2	2	2	2	0	3	1	1	1	6	0	1/0	2	21	SDIP SOIC QFN-S

Note 1: RAM size is inclusive of 2 Kbytes of DMA RAM for all devices except dsPIC33FJ32MC302/304, which include 1 Kbyte of DMA RAM.
2: Only four out of five timers are remappable.
3: Only PWM fault pins are remappable.
4: Only two out of three interrupts are remappable.

Tabla 4. Características especiales de algunos microcontroladores.

Es muy importante para el usuario conocer el hardware del dispositivo, así como sus especificaciones electrónicas. Para lograr este acometido, se sugiere revisar la figura 23 (recordando que no es suficiente, para mayores detalles consultar la hoja de datos proovista por microchip).

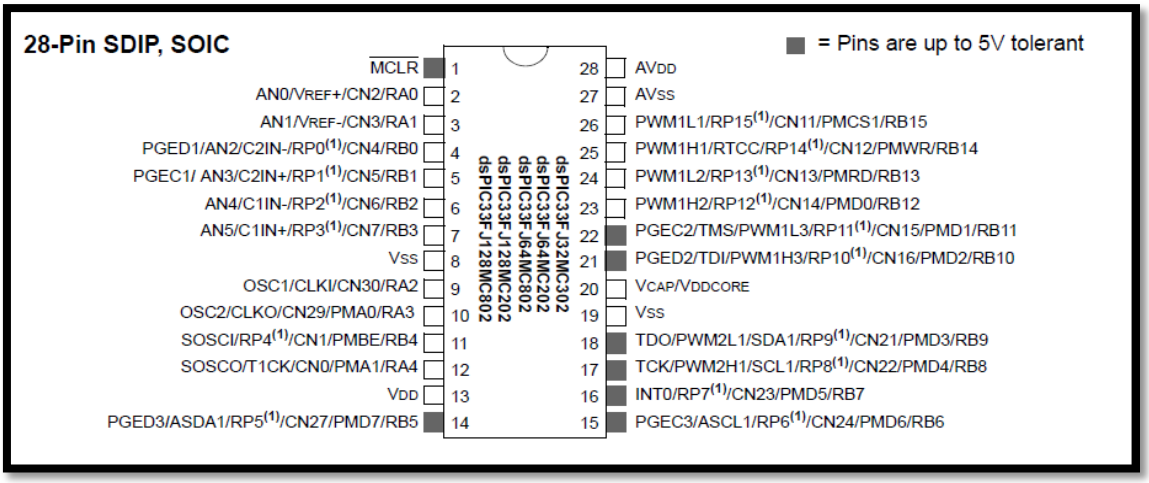


Figura 23. Diagrama de pines.

A continuación se muestra en forma general el diagrama de bloques del circuito integrado. Es ampliamente recomendado entender este diagrama, pues este describe el comportamiento interno del dispositivo: intercomunicación entre sus componentes, flujo y dirección de datos, bits relacionados a cada registro y periféricos disponibles.

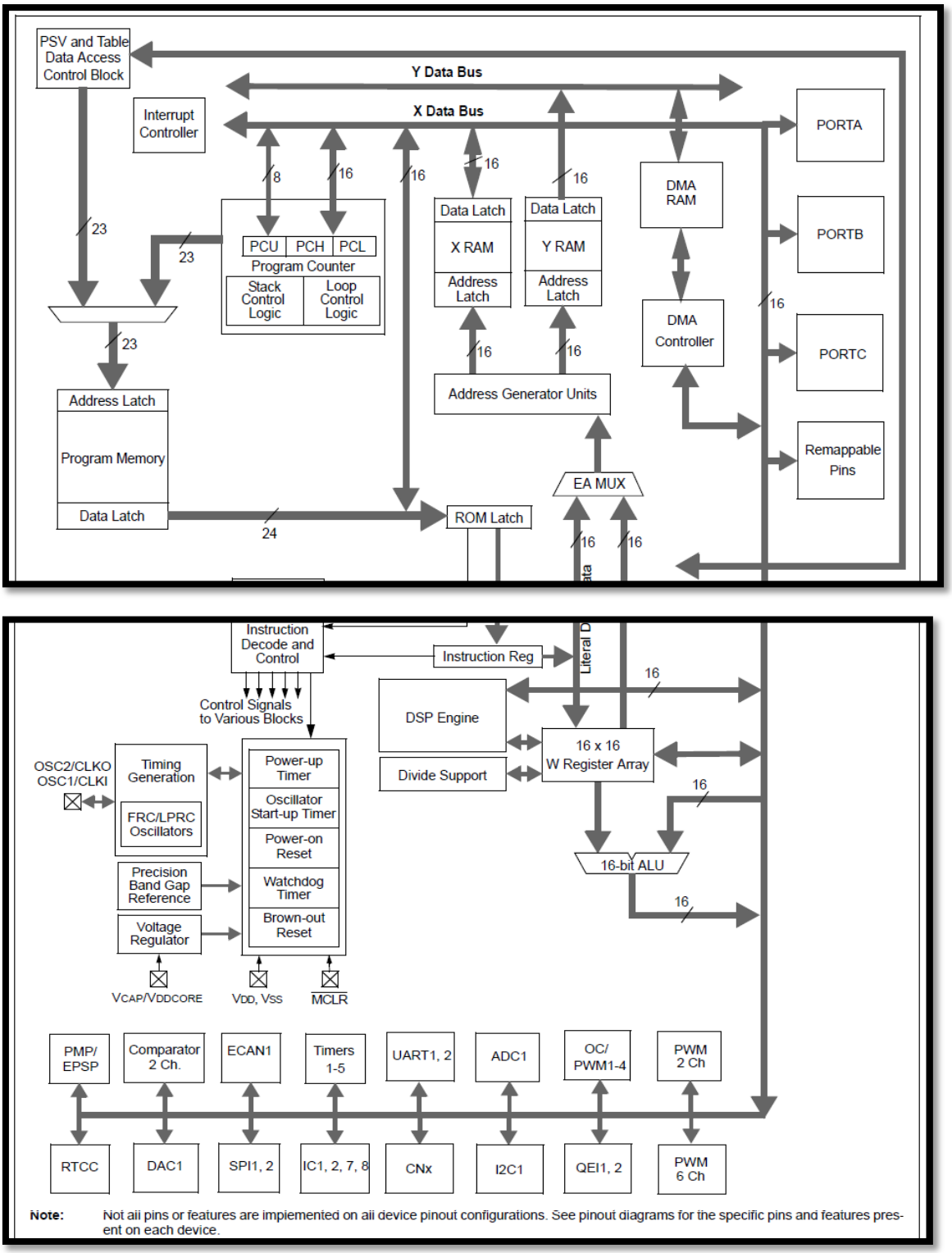


Figura 24. Diagrama de bloques, familia dsPIC30F/dsPIC33F.

Como todo dispositivo programable, el dsPIC contiene en su memoria una serie de registros ordenados sistemáticamente, de forma tal que a partir de

direcciones de memoria se tiene acceso (Anexos). Los registros, para este caso especial de controlador, son pilas de 16 bits, cada registro puede contener bits de solo lectura, solo escritura o lectura y escritura. Por lo tanto, se puede acceder a los registros para su configuración y/o para extracción de información.

Para el caso en particular del proyecto, se describe de forma general (para más detalle, consultar anexos) las características del dsPIC utilizadas en la programación del control de los motores:

TIMER1.

El módulo Timer1 es un temporizador de 16 bits, puede servir como un contador de tiempo real, o como un temporizador de intervalos de libres. El modulo Timer1 se muestra en forma de diagrama de bloques en la figura 25.

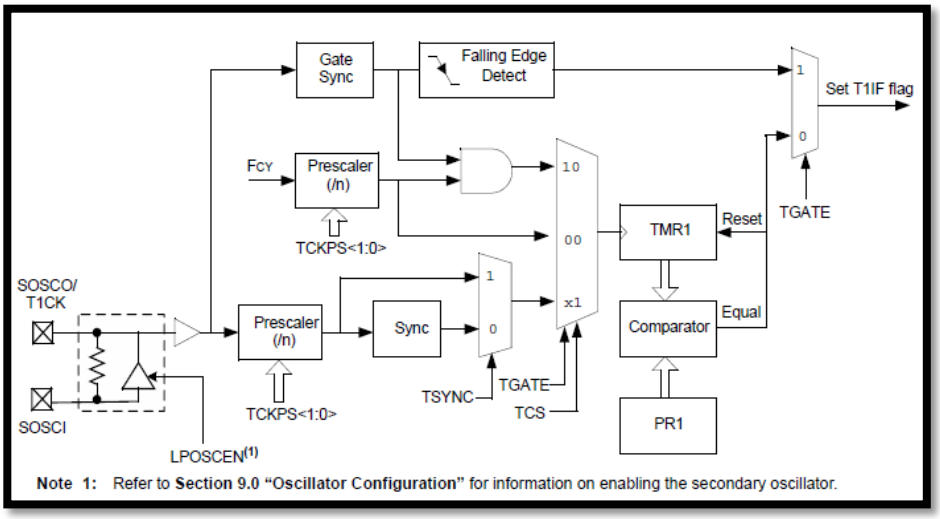


Figura 25. Diagrama de bloques del modulo Timer1.

El módulo Timer1 puede funcionar en los siguientes modos:

- Temporizador.
- Temporizador cerrado.
- Contador síncrono.
- Contador asíncrono.

Los modos de funcionamiento para el temporizador “Timer1” se determinan en los siguientes bits:

Para definir el la fuente del reloj, (TCS): T1CON <1>

Para definir si es síncrono o no, (TSYNC):T1CON<2>

Para definir si es de tipo cerrado, (TGATE): T1CON<6>

La tabla 5 muestra la forma de configurar los modos disponibles.

Mode	TCS	TGATE	TSYNC
Timer	0	0	x
Gated timer	0	1	x
Synchronous counter	1	x	1
Asynchronous counter	1	x	0

Tabla 5. Bits para el módulo Timer1.

QUADRATURE ENCODER INTERFACE (QEI)

El modulo de interfaz de cuadratura del codificador (QEI, por sus siglas en ingles) proporciona la interfaz para la obtención de los datos de la posición mecánica de los codificadores incrementales. La figura xx muestra el diagrama de bloques para el modulo QEI.

Las características operativas de la QEI incluyen:

• Tres canales de entrada. Dos para señales de fase y uno para el pulso índice.

• Contador de 16 bits ascendente/ descendente.

• Contador para el cambio de dirección.

• Multiplicadores para la resolución (x2 y x4).

• Generador de filtros digitales en las entradas.

• Alternador entre el modo temporizador / contador de 16 bits.

• Interrupción en el desbordamiento del contador o en cada pulso generado por el canal índice.

Estos modos de funcionamiento se determinan mediante el establecimiento de la bits apropiados, QEIM <02:00> en (QEIxCON <10:08>).

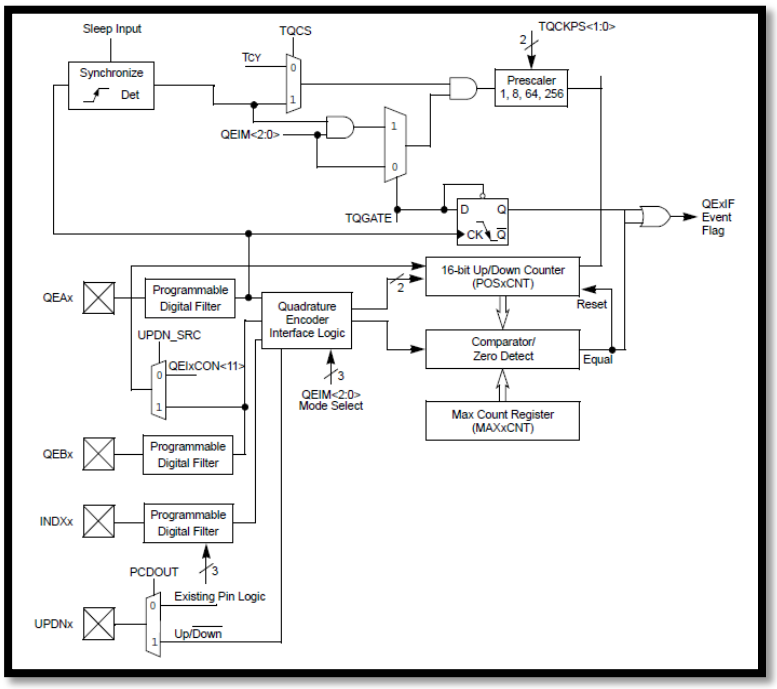


Figura 26. Diagrama de bloques para el modulo QEI.

UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART).

En este apartado solo se toman en cuenta los registros asociados al protocolo de comunicación UART y no a las características de este. Para aprender más acerca del protocolo de comunicación habrá que consultar el capítulo 3.

El modulo UART depende de tres elementos clave:

Generador de tasa de baudios.

Transmisor asíncrono.

Receptor asíncrono.

En la figura 27 se muestra un diagrama simplificado del modulo UART.

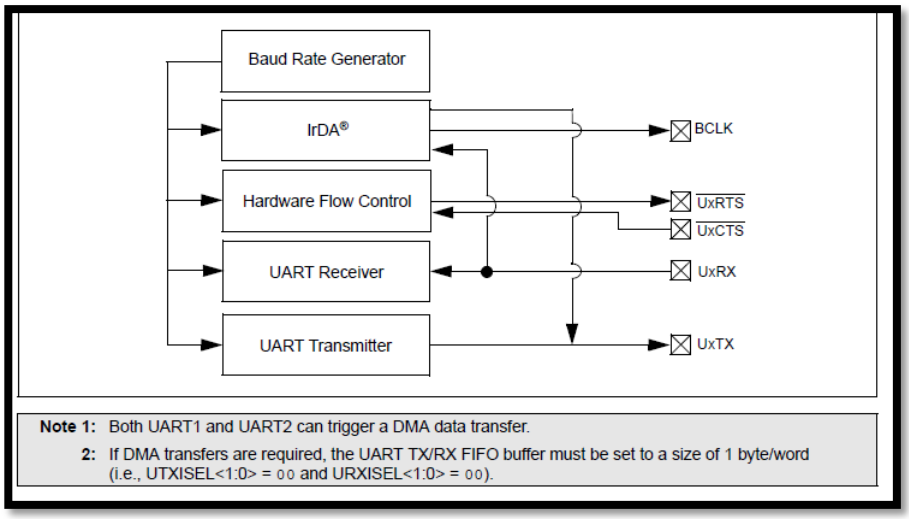


Figura 27. Modulo UART.

Los registros asociados al módulo son:

UxMODE, que define el modo de operación y configuración del UART. (Anexos)

UxSTA, que define el estado. (Anexos)

PERIPHERAL PIN SELECT (PPS).

Entre de las características especiales que el dsPIC33FJ128MC802 nos ofrece, está la Selección de Pines Periféricos (PPS por sus siglas en ingles), el cual nos aporta la facilidad de reasignación pines. Esto nos permite asignar periféricos a los pines con designación “RPn”(Anexos), donde "RP" define que es un pin de reasignación y "n" es el número pin reasignable. La selección de pines incluye un rango amplio de hasta 26 pines. El número de pines disponibles depende del dispositivo en particular y su número pines. Esta asignación es posible tanto para entradas, como para salidas.

Entradas PPS.

Las entradas del PPS se asignan en función de los periféricos y se configuran en el registro RPINRx (Anexos), Cada registro contiene conjuntos de 5 campos de bits, cada uno de estos esta asociado con los periféricos reasignables.

Para cualquier dispositivo determinado, el rango de valores válidos para cualquiera de los campos de bits corresponde con el número máximo de pines de reasignación del dispositivo.

La figura 28 ilustra la selección de pines reasignables para la entrada del registro U1RX que corresponde al receptor del protocolo de comunicación RS232.

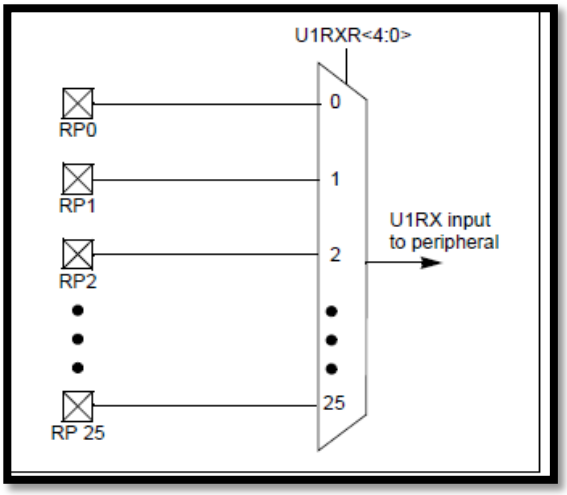


Figura 28. Multiplexor de selección para U1RX.

Salidas PPS.

En contraste con las entradas, las salidas son asignadas hacia los pines. En este caso un registro de control asociado a un pin en particular determina el periférico de salida a reasignar. El registro RPORx se usa para controlar las salidas reasignadas (Anexos), Al igual que RPINx, cada registro contiene conjuntos de 5 campos de bits y cada bit está asociado a un pin RPn. La figura 27 ilustra la selección de periféricos a un único pin de salida.

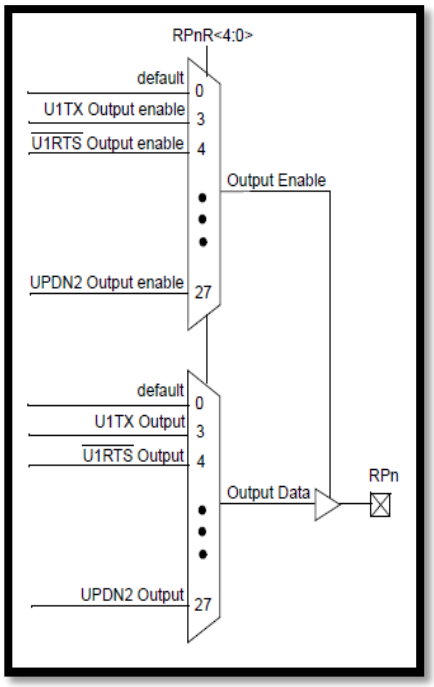


Figura 29. Multiplexor de selección de periféricos para pines de salida.

Los periféricos utilizados para este proyecto se muestran en la tabla 6 con su descripción y el registro al que corresponden.

QEI1 Phase A	QEA1	RPINR14	QEAIR<4:0>
QEI1 Phase B	QEB1	RPINR14	QEBIR<4:0>
QEI1 Index	INDX1	RPINR15	INDXIR<4:0>
UART1 Receive	U1RX	RPINR18	U1RXR<4:0>
UART1 Clear To Send	$\overline{\text{U1CTS}}$	RPINR18	U1CTSR<4:0>

Tabla 6. Bits correspondientes para la selección de pines.

3.3. Etapa de servo amplificación.

La etapa de servo amplificación consta de la entrada de una señal de referencia y la amplificación de esta por medio de un amplificador de potencia.

El propósito del amplificador de potencia es proporcionar una tensión de salida con máxima excursión simétrica sin distorsión a una baja resistencia de carga. En la práctica, un sistema puede consistir en varias etapas de amplificación.

Los factores del amplificador de potencia que mayor interés presentan son:

- Eficiencia en potencia del circuito (rendimiento).
- Máxima cantidad de potencia que el circuito es capaz de manejar.
- Acoplamiento de impedancia en relación con el dispositivo de salida.

El amplificador de potencia recibe una señal (normalmente una señal analógica de -10 a +10V) y la amplifica a un nivel de corriente requerido. El amplificador puede ser configurado en modo de corriente o modo de velocidad. El modo de velocidad (Figura 30) es de preferencia solo cuando se ocupa la retroalimentación de la velocidad.

En el modo de corriente, el amplificador produce una corriente que es directamente proporcional a la entrada de voltaje. Los amplificadores de corriente se caracterizan por la ganancia de corriente, Ka, la cual es indicada por el amplificador.

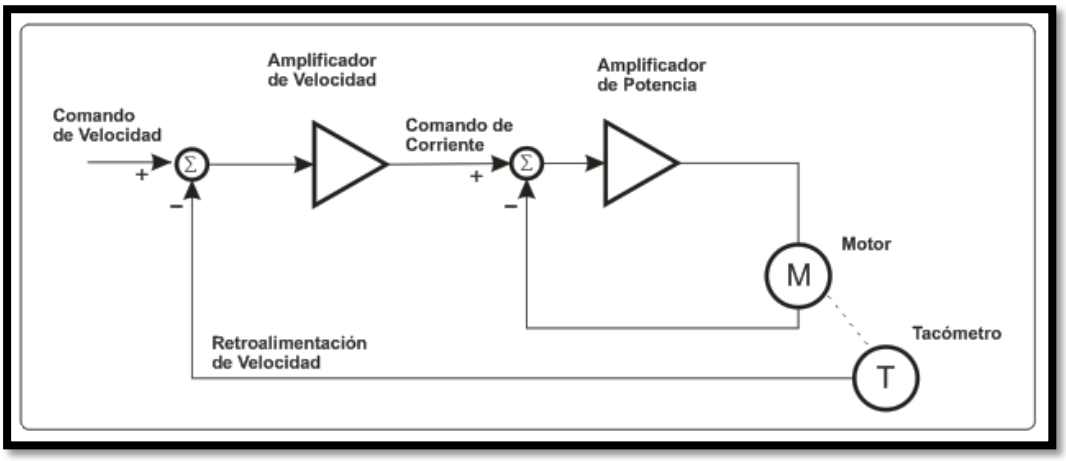


Figura 30. Elementos del Amplificador en el modo de velocidad

Para la instrumentación de este sistema, se utilizaron dos servoamplificadores con distintas características.

3.4. Codificador/Decodificador.

Se compone de dos vías y dos sensores de cuyos resultados se denominan canales A y B. Cuando el eje gira, trenes de pulsos se producen en estos canales con una frecuencia proporcional a la velocidad del eje, y la relación de fase entre los rendimientos de las señales de la dirección de rotación. El modelo de disco de códigos y señales de salida A y B se ilustran en la Figura 31. Al contar el número de pulsos y conociendo la resolución de la disco, el movimiento angular se puede medir. Los canales A y B se utilizan para determinar el sentido de rotación mediante la evaluación de los canales que "conduce" el otro. Las señales de los dos canales son de 1 / 4 ciclo fuera de fase entre sí y se conocen como señales de cuadratura. A menudo, un tercer canal de salida, llamado índice, el rendimiento de un pulso por revolución, lo cual es útil en el conteo de vueltas completas. También es útil como referencia para definir una base de operaciones o la posición cero.

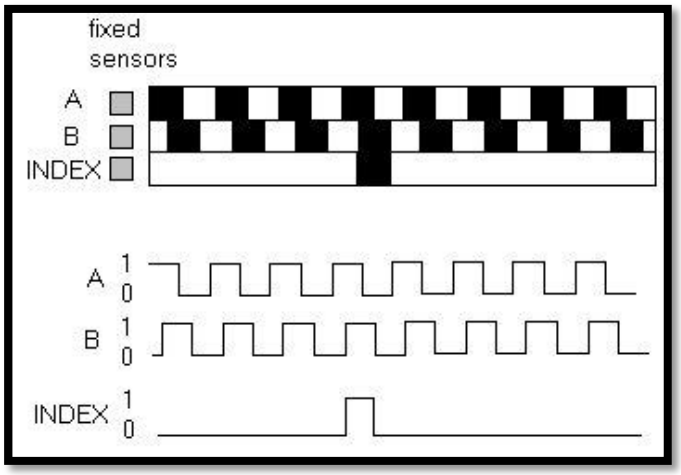


Figura 31. Señales de salida de un “encoder” Incremental

La figura 31 ilustra dos pistas separadas para los canales A y B, pero una configuración más común utiliza una sola vía con la A y B sensores compensar un ciclo de 1 / 4 en la pista para obtener el patrón de la misma señal. Un disco de código de un solo tema es más simple y más barato de fabricar. Al comparar las señales de salidas, se puede describir la dirección que lleva el sensor como se muestra en la Figura 32.

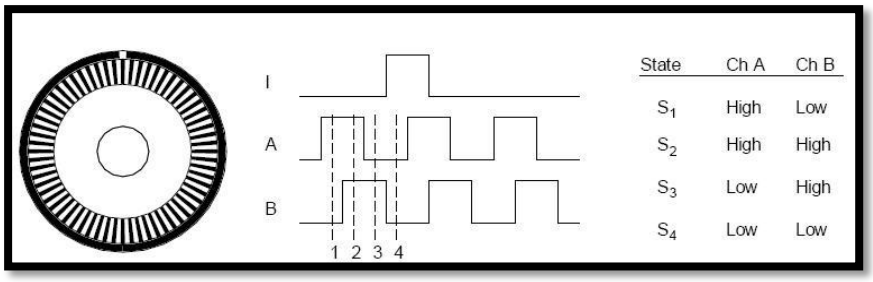


Figura 33. Determinación del sentido de rotación.

3.5. Diseño PCB.

3.5.1. Tarjeta para dsPIC.

3.5.1.1. Circuito mínimo.

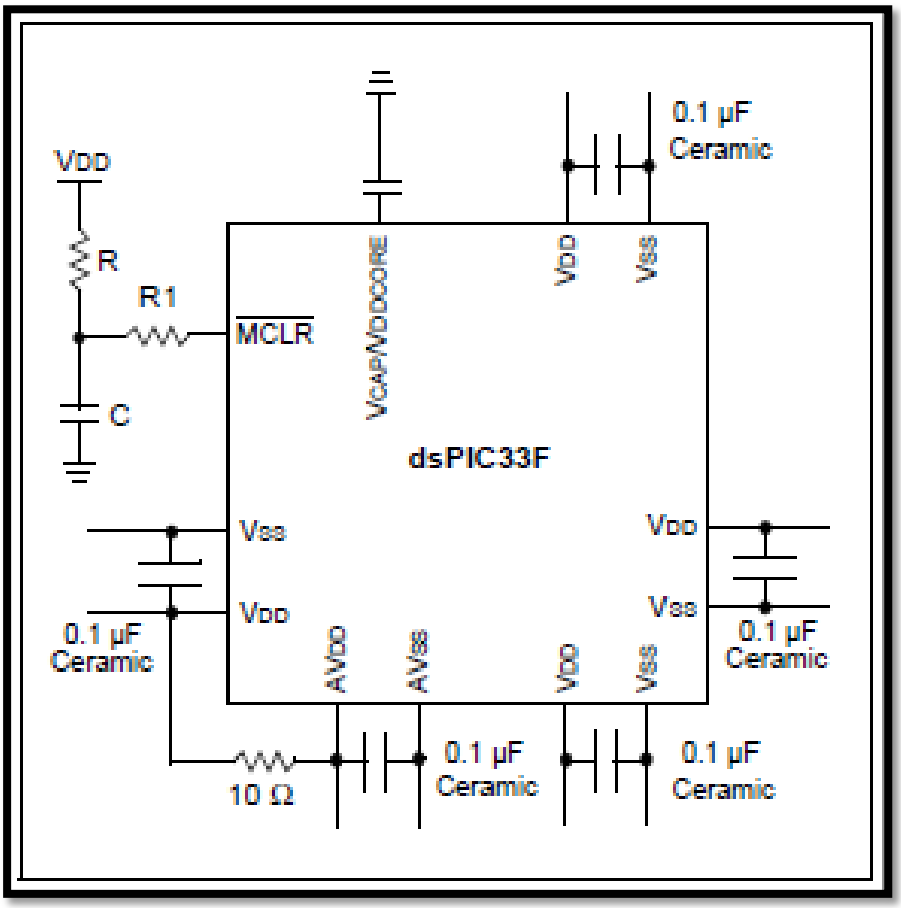


Figura 34. Circuito Mínimo dsPIC33F.

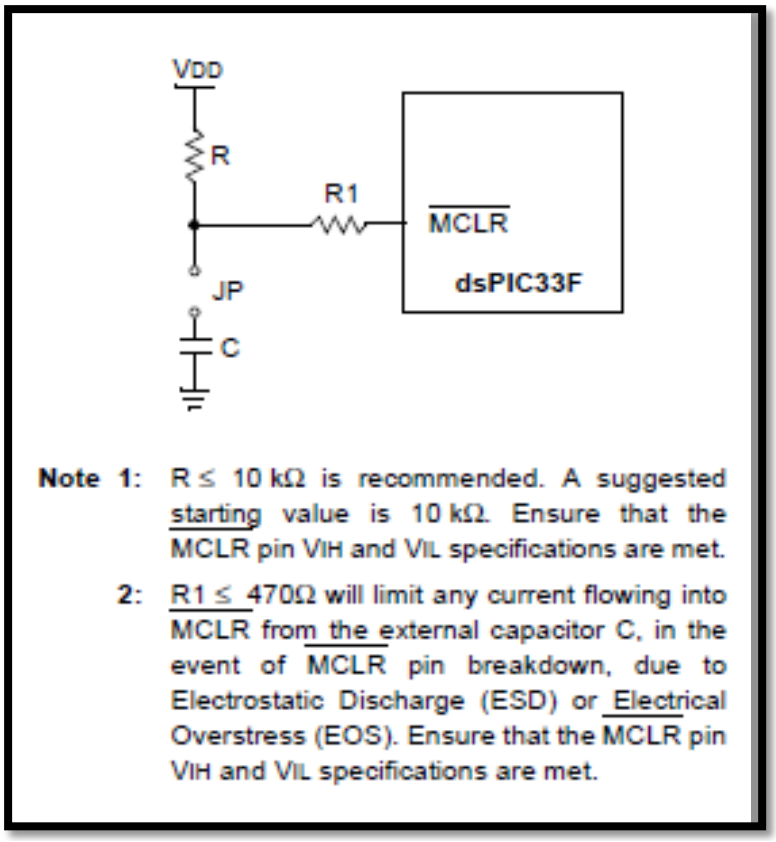


Figura 35. Master Clear dsPIC33F.

3.5.2. Etapa de Amplificación.

3.5.2.1. Acondicionamiento de señales.

3.5.2.2. Aislamiento óptico.

3.6. Interfaz UART (RS232).

RS232 es un estándar para la comunicación serial entre DTE (Data Terminal Equipment) y DCE (Data Communication Equipment).

El UART es el protocolo de comunicación asíncrono que trabaja en un rango de velocidades de 110 a 115,200 baudios y soporta comunicaciones de tipo “Simplex”, “Half-Duplex” y “Full Duplex”. Suele tener un alcance de hasta 18 metros.

La trama de datos se muestra en la figura 36. Esta empieza con un pulso en bajo y termina con un pulso en alto.

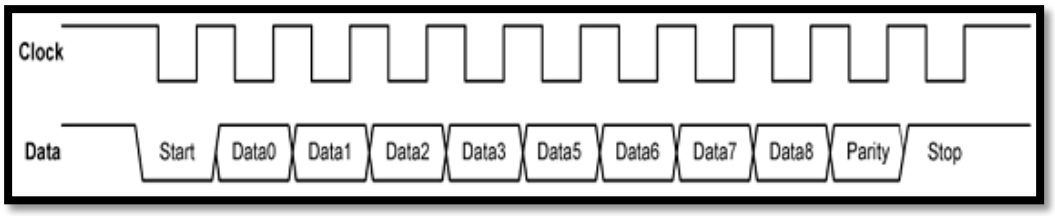
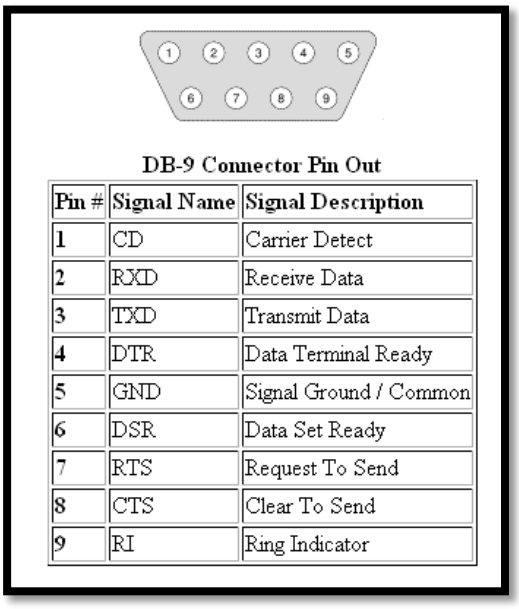


Figura 36. Trama de datos del protocolo de comunicación UART.

Niveles de voltaje:

- Pulso en Alto →1 lógico → -3 a -18 V (Marca)
- Pulso en bajo→0 lógico → +3 a +18 V (Espacio)

La figura 37 ilustra la descripción del conector DB-9 utilizado en la comunicación RS232.



DB-9 Connector Pin Out

Pin #	Signal Name	Signal Description
1	CD	Carrier Detect
2	RXD	Receive Data
3	TXD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	Signal Ground / Common
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	RI	Ring Indicator

Figura 37. Descripción de pines y conector para comunicación RS232.

3.6.1. Protocolo de comunicación RS232-USB

4. Software y Firmware.

4.1. MPLab.

4.2. LabView.

LabVIEW, software desarrollado por National Instruments, ofrece una de las mejores integraciones entre hardware y software de programación gráfica. LabVIEW ofrece cientos de características, incluyendo gráficas en 2D y 3D en el panel frontal y funciones de escritura de datos de alta velocidad. (Christian G. Quintero, 2011).

LabVIEW es un programa para el desarrollo de aplicaciones de propósito general que National Instruments ha creado para facilitar la programación de instrumentos virtuales (VI's). LabVIEW se encarga de gestionar los recursos del ordenador a través de un entorno sencillo, rápido y eficiente. De esta manera, se reducen enormemente los tiempos de desarrollo a la hora de realizar los programas. (C. Ramos et al; Monitorización y Control Distribuido a través de Internet).

Las principales características de LabVIEW son las siguientes:

- Entorno de desarrollo gráfico; desaparece el código en formato testo que estamos acostumbrados a utilizar. Con esto se consigue una forma de programación más intuitiva.
- Diseño de la interfaz gráfica del instrumento virtual, utilizando elementos prediseñados (controles numéricos, gráficas, etc.).
- Gestión automática en la creación de hilos de ejecución.
- Herramientas convencionales para la depuración de los programas (VI's): ejecución paso a paso, puntos de ruptura, flujos de datos, etc.
- Programación modular.
- Potente conjunto de librerías para: adquisición de datos, control de instrumentos a través de bus GPIB, VXI o serie, análisis, presentación y almacenamiento de datos, etc.

La programación de una aplicación en LAbVIEW es muy diferente a la programación en un lenguaje de alto nivel como el C o el Basic. LabVIEW utiliza los símbolos gráficos (íconos) para describir el programa de acciones. El flujo de los datos se realiza a través de los conductores en un diagrama de bloques. Puesto que LabVIEW es gráfico y basado en un sistema de ventanas, se le denomina Lenguaje de Programación Gráfica o Lenguaje G y su uso es a menudo mucho más sencillo que un lenguaje tradicional. (Información Tecnológica 2001).

4.3. Firmware.

4.3.1. Código MPLab.

Para el proyecto se utiliza programación por maquina de estados. La figura 38 muestra en diagramas la máquina de estados incluida en la función principal del código.

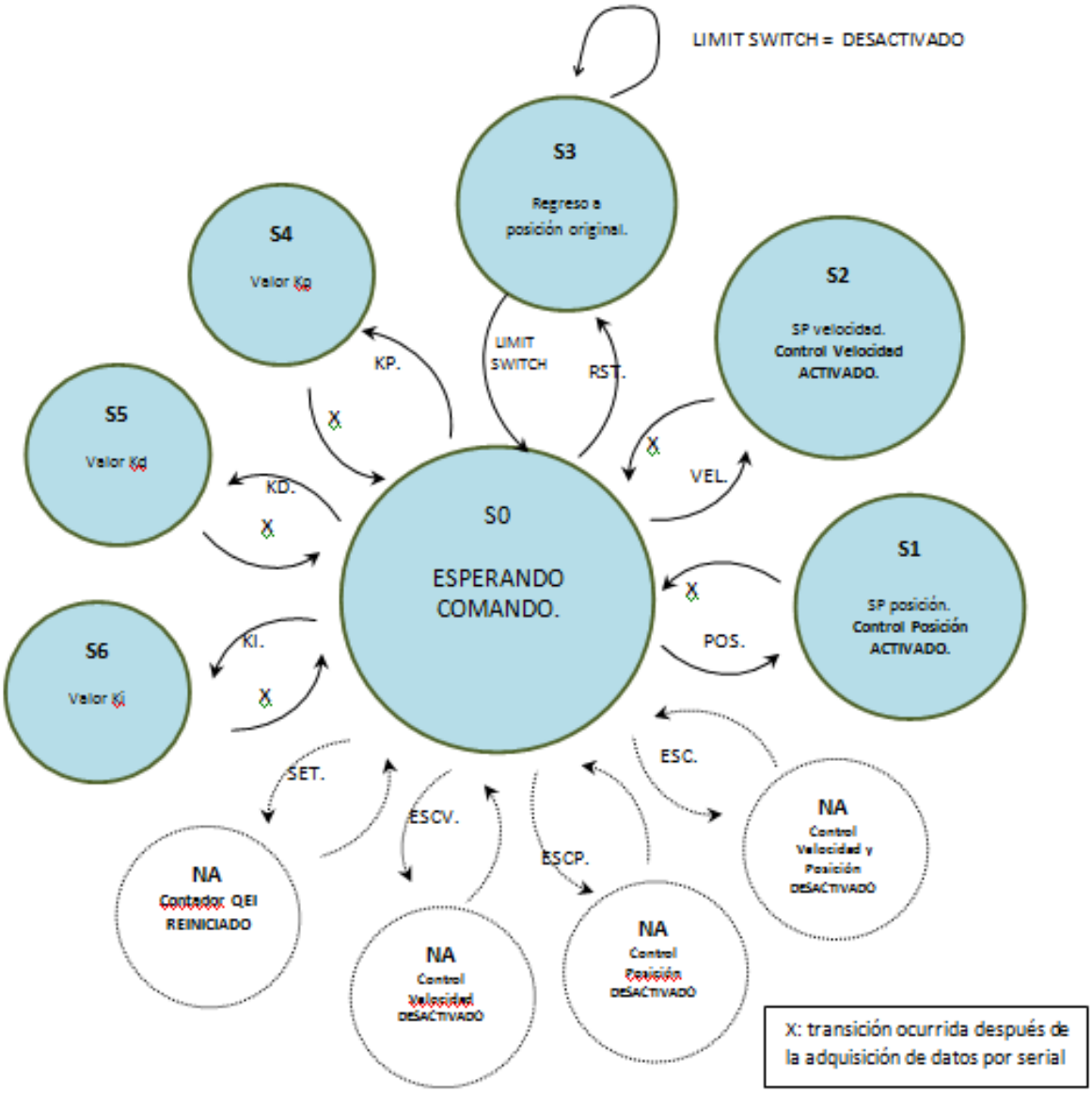


Figura 38. Maquina de estados. Software “Control de Bancada”

Es importante tratar de entender el diagrama, ya que la programación contenida en el ciclo infinito del dsPIC se basa en ésta estructura.

A continuación se muestra de forma explicada el código programado en el entorno MPLAB IDE y compilado con la herramienta C30 Compiler.

```c
#include "p33FJ128MC802.h"      //Libreria de Procesador
#include <string.h>             //Funciones String. ej. strcpy, strcmp
#include "Mystdlib.h"           //Funciones StdLib. ej. itoa, atoi
#include "FCY.h"                //Configuracion del Reloj.
#include "Timer1.h"             //Configuracion Timer1. ej. setup_T1
#include "IO.h"                 //Configuracion de Pines y Pines reasignables.
#include "UART.h"               //Funciones UART. ej. setup_UART, WriteUART...
#include "QEI.h"                //Funciones QEI. ej. setup_QEI, ReadQEI, ResetQEI...
#include "PWM.h"                //Funciones PWM.ej. setup_PWM, OpenMCPWM1, SetDCMCPWM1
#include "DAC_AD7390.h"         //Funciones DAC.ej. WriteDAC
#include "PERFIL.h"             //Funciones Perfiles de velocidad. ej. PerfilVelocidad

/*********************************** DEFINICIONES *********************************** */
#define buf_length 32           //Longitud de Buffer

/*************************** CONFIGURACION DE BITS. FUSIBLES ****************************************************** */
_FBS(BWRP_WRPROTECT_OFF )                                   //Proteccion de escritura desabilitada
_FSS(SWRP_WRPROTECT_OFF )                                   //Proteccion de escritura desabilitada
_FGS(GSS_OFF & GCP_OFF )                                    //Proteccion de codigo desabilitada
_FWDT(FWDTEN_OFF)                                           //Watchdog desabilitado
_FOSC(FCKSM_CSDCMD & OSCIOFNC_ON & POSCMD_XT)               //Oscilador XT, Pin OSC2 como entrada/salida digital
_FOSCSEL( FNOSC_PRIPLL & IESO_OFF   )                       //Oscilador primario (XT, HS, EC) con PLL
_FPOR( PWMPIN_ON & HPOL_ON &  LPOL_ON & ALTI2C_OFF  )       //Modo de PWM como registro, Polaridades PWM activadas en alto
_FICD( BKBUG_OFF & COE_OFF  & JTAGEN_OFF & ICS_PGD1 )       //No debug, JTAG desabilitado, Comunicacion del ICD en Pines PG(C/D)1

/*********************************************VARIABLES GLOBALES************************************************************/
char buf[buf_length],str[buf_length],strAux[buf_length];    //Cadenas buffer de caracteres.
int str_flag=0;     //Bandera de fin de cadena, Activo detecta el caracter de terminacion de 1 cadena
int i=0, tb=0;      //Contadores, Bases de tiempo.

//............................................. VARIABLES PID ....................................................//
int control = 0;                    //Señal de salida PID para DAC
unsigned long setPosition = 0;      //Referencia de Posicion
int setVelocity = 0;                //Referencia de Velocidad

float ts = 0.001;                   //Tiempo de Muestreo = 0.001   --->>> corroborar tiempo de ejecucion de codigo en Timer1
unsigned long VP;                   //Variable de Proceso. Cuentas del encoder
unsigned long SP;                   //Set Point.
unsigned long e;                    //Error. e= SP-VP
unsigned long e_p = 0;              //Error Pasado
float u;                            //Salida. Señal de Control
float acum = 0;                     //Acumulador. Area bajo la curva
float Kp, Ki, Kd;                   //Ganancias PID

int flagPosition = 0;               //Bandera. Activar control de posicion//>> checar con forti. que sea un solo bit o boleano.
int flagVelocity = 0;               //Bandera. Activar control de velocidad

//...............................................  COMANDOS   ...................................................//
char rst [4] = {'R','S','T', 0};        //Resetea contador del encoder. Regresa la bancada a su origen
char pos [4] = {'P','O','S', 0};        //Establece la referencia de posicion
char vel [4] = {'V','E','L', 0};        //Establece la referencia de velocidad
char esc [4] = {'E','S','C', 0};        //Salida. Desactiva los controladores
char set [4] = {'S','E','T', 0};        //Establece la posicion actual como origen. Resetea contador de encoder

char escp [5] = {'E','S','C','P', 0};   //Desactiva control de posicion
char escv [5] = {'E','S','C','V', 0};   //Desactiva control de velocidad

char kp  [3] = {'K','P', 0};            //Establece la ganacia proporcional (Kp)
char kd  [3] = {'K','D', 0};            //Establece la ganacia derivativa (Kd)
char ki  [3] = {'K','I', 0};            //Establece la ganacia integral (Ki)

//char cero[4]={'0','0','0',0};                               --> ELIMINAR SI NUNCA SE USA

//*************************************************  INTERRUPCIONES   ******************************************/

//..................................TIMER 1..........................................................//
/* Frecuencia de interrupcion a 20Khz. Todos los parametros de configuracion se establecen en la funcion
Setup_T1() que se encuentra declarada en la libreria "Timer1.h". Para lograr esa frecuencia es necesario
primero configurar la frecuencia de oscilacion del  procesador(FCY) configurada en  "FCY.h"*/
void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void)
    {
    if(flagVelocity)                                //Control de velocidad activado en "1" logico
    {
    //SetDCMCPWM1(1,setVelocity&0xFFF,0);
    PerfilVelocidad(setVelocity);                   //Funcion que permite aceleraciones y desaceleraciones constantes para el motor
    }
    if(flagPosition)                                //Control de posicion activado en "1" logico
    {

      VP =(unsigned long) ReadQEI();                // VP toma el valor del contador del QEI
      e= SP - VP;                                   //Calculo del error
```

```
84          acum =acum + (ts*e);                          //Cálculo para término integral. Área bajo la curva
85          u =((Kp*e)+ (Ki*acum) + (Kd*((e-e_p)/ts)));   //Cálculo de la salida PID
86
87          control = (0xFFF/2) + u;                      /*Transferencia de salida PID.!!! Recordar que una referencia de 0...
88                                                        para el servodriver implica la escritura de 0xFFF/2 para el DAC !!!!*/
89            if(control>0xfff)                           //Limite superior de escritura en DAC
90                control=0xfff;
91            else if (control<0)                         //Limite inferior de escritura en DAC
92                control=0;
93
94          Write_DAC(control);                           //Escritura de control para DAC
95          e_p = e;                                      //Error pasado toma el valor del error presente
96
97         if(tb==50)                                     //Tiempo base de transferencia de datos por el serial
98         {
99            tb=0;
100           putsUART1(" *");
101           SendUARTInt32(POSCNT);                      //Funcion para enviar datos de tipo entero de 4bytes(32 bits) por UART. ej. 0xFFFFFFFF >> 25
102           putsUART1("* ");
103        }else
104          tb++;
105       }
106   IFS0bits.T1IF = 0;                                  //Bandera de interrupcion borrada por software. Necesario para generar la sig. interrupcion
107   }
108
109   //.................................UART RX ..........................................................//
110   /* Cada que es recibido un caracter por el serial se ejecuta esta interrupcion*/
111   void __attribute__((__interrupt__, no_auto_psv)) _U1RXInterrupt(void)
112   {
113      buf[i]=getcUART1();                              //Recibe cada trama de 8 bits y almacena el caracter un un buffer
114      _LATA0 = _LATA0^1;                               //Indicador de recepcion de datos
115      if(buf[i]=='.')                                  //Identificador del caracter de fin de cadena('.')
116        {
117         str_flag=1;
118         buf[i]=0;                                     //Caracter nulo necesario para todo dato tipo cadena.
119         strcpy(str,buf);                              //Copia la cadena recibida en un buffer auxiliar(str) para posteriormente ser comparado
120         i=0;
121        }
122         else
123             i++;                                      //Contador para apuntador de la cadena "buf"
124   IFS0bits.U1RXIF = 0;                                //Bandera de interrupcion borrada por software. Necesario para generar la sig. interrupcion
125   }
126
127
128   //*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  FUNCION PRINCIPAL (main)  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
129   /*Codigo principal sistematizado en MAQUINA DE ESTADOS, cada estado pertenece a un dato tipo enumerador (currentState) que
130   rigue el estado actual de la maquina*/
131   int main(void)
132   {
133           enum state{STMST0, STMST1,STMST2,STMST3,STMST4,STMST5,STMST6}currentState;   //Declaracion de Estados
134
135           setup_IO();                                       //Configuracion de pines como E/S y pines reasignables. Ver "IO.h"
136           setup_FCY();                                      //Configuracion de Frecuencia del Procesador. Ver "FCY.h"
137           setup_T1();                                       //Configuracion del Modulo Timer1. Ver "Timer1.h"
138           setup_UART();                                     //Configuracion del Modulo UART. Ver "UART.h"
139           setup_QEI();                                      //Configuracion del Modulo QEI. Ver "QEI.h"
140           setup_PWM();                                      //Configuracion del Modulo PWM. Ver "PWM.h"
141
142           /*!!!! Ganancias estimadas por metodo de sintonizacion. Ver capitulo xx !!!*/
143           Kp=120.88;
144           Kd=1684.50;
145           Ki=12.99;
146
147           putsUART1("COMUNICACION ESTABLECIDA...");
148           _LATA0=1;
149           Write_DAC(0xFFF/2);                               //Referencia en 0 para servodriver
150           SetDCMCPWM1(1,0,0);                               //PWM 0% duty cycle
151           currentState= STMST0;                             //Estado Actual, Estado Cero
152
153   while(1)                                                  //Bucle infinito.
154   {
155       switch(currentState)
156       {
157    //############################### STATE 0 ########################################
158         case STMST0:
159           if(str_flag)
160           {
161            if (strcmp(str,pos)==0){
```

```
160             {
161              if (strcmp(str,pos)==0){
162                 currentState= STMST1;
163             }else if (strcmp(str,vel)==0){
164                 currentState= STMST2;
165             }else if (strcmp(str,rst)==0){
166                 putsUART1(" #RST# ");
167                 currentState= STMST3;
168             }else if (strcmp(str,kp)==0){
169                 putsUART1(" #KP# ");
170                 currentState= STMST4;
171              }else if (strcmp(str,kd)==0){
172                 putsUART1(" #KD# ");
173                 currentState= STMST5;
174              }else if (strcmp(str,ki)==0){
175                 putsUART1(" #KI# ");
176                 currentState= STMST6;
177              }else if (strcmp(str,set)==0){
178                 putsUART1(" #SET CERO# ");
179                 flagPosition=0;
180                 ResetQEI();
181                 Write_DAC(0xFFF/2);
182                 currentState= STMST0;
183              }else if (strcmp(str,escp)==0){
184                 putsUART1(" #EXIT POS. ");
185                 flagPosition=0;
186                 currentState= STMST0;
187              }else if (strcmp(str,escv)==0){
188                 putsUART1(" #EXIT VEL. ");
189                 flagVelocity=0;
190                 currentState= STMST0;
191              }else if(strcmp(str,esc)==0){
192                 putsUART1(" #ESC. ");
193                 flagPosition=0;
194                 flagVelocity=0;
195                 SetDCMCPWM1(1,0,0);
196                 Write_DAC(0xFFF/2);
197                 currentState= STMST0;
198              }
199             str_flag=0;
200             }
```

```
202         break;
203
204     //############################## STATE 1 ########################################
205         case STMST1:  //POSICION
206
207             if(str_flag)
208             {
209                setPosition =atol(str);
210                SP=setPosition;
211                 putsUART1("POS SP: ");
212                 SendUARTInt32(setPosition);
213                 str_flag=0;
214                 flagPosition=1;
215                 currentState=STMST0;
216             }
217
218
219                     break;
220
221     //############################## STATE 2 ########################################
222
223         case STMST2:  //VELOCIDAD
224
225          if(str_flag)
226          {
227              setVelocity = atoi(str);
228              itoa(setVelocity,strAux);
229              putsUART1("VEL SP: ");
230              putsUART1(strAux);
231              str_flag=0;
232              flagVelocity=1;
233              currentState=STMST0;
234          }
235
236                  break;
```

```
238     //############################## STATE 3 ########################################
239           case STMST3: // RST
240
241
242                 flagPosition=0;
243                 flagVelocity=0;
244                 setPosition=0;
245                 setVelocity=0;
246                 SetDCMCPWM1(1,0,0);
247                 Write_DAC(0xFFF/2);
248
249                 //MOVERSE EN SENTIDO CONTRARIO HASTA ALCANZAR A PULSAR EL LIMIT SWITCH
250                 while(_RA1!=0)
251                 {
252                     Write_DAC(0xFFF/4);
253                 }
254
255                 Write_DAC(0xFFF/2);
256                 ResetQEI();
257                 str_flag=0;
258
259                 currentState= STMST0;
260
261                         break;
262
263      //############################## STATE 4 ########################################
264           case STMST4: //KP
265                 if(str_flag)
266                 {
267                     Kp =atof(str);
268                     itoa(Kp,strAux);
269                     putsUART1("kp: ");
270                     putsUART1(strAux);
271                     str_flag=0;
272
273                     currentState=STMST0;
274                 }
275
276                         break;
277
278      //############################## STATE 5 ########################################
```

```
279             case STMST5: //KD
280                 if(str_flag)
281                 {
282                     Kd =atof(str);
283                     itoa(Kd,strAux);
284                     putsUART1("kd: ");
285                     putsUART1(strAux);
286                     str_flag=0;
287 
288                     currentState=STMST0;
289                 }
290 
291                         break;
292 
293       //############################## STATE 6 ########################################
294             case STMST6: //KI
295                 if(str_flag)
296                 {
297                     Ki =atof(str);
298                     itoa(Ki,strAux);
299                     putsUART1("ki: ");
300                     putsUART1(strAux);
301                     str_flag=0;
302 
303                     currentState=STMST0;
304                 }
305 
306                         break;
307 
308       //############################ Default ########################################
309             default:
310                 currentState=STMST0;
311                         break;
312 
313         }
314     }
315     return 0;
316     }
317     //Salto de linea. Ultima linea de codigo en blanco por peticion del Compilador.
```

4.3.2. Código LabView.

Como hemos visto anteriormente, el código de LabVIEW es un código de tipo gráfico, por lo tanto, se mostrará a continuación el programa desarrollado para el proyecto en forma de segmentos de imágenes, en donde se irán detallando cada una de las secciones del código o programa.

Imagen 39. Interfaz HMI desarrollada en LabVIEW.

- Primera parte: Configuración del Puerto Serial

En la imagen 40 se muestran todos los elementos necesarios para una correcta configuración del puerto serial. Para el presente proyecto se suponen constantes la mayoría de estos elementos, ya que, si se llegasen a cambiar estos parámetros, se debería modificar también el código en el microcontrolador. Por lo tanto, estos elementos se mantienen ocultos en la interfaz del usuario operador, a excepción de la selección del puerto en la PC, ya que ésta puede cambiar según el sistema operativo o el puerto en que se conecte el cable de comunicación serial-USB.

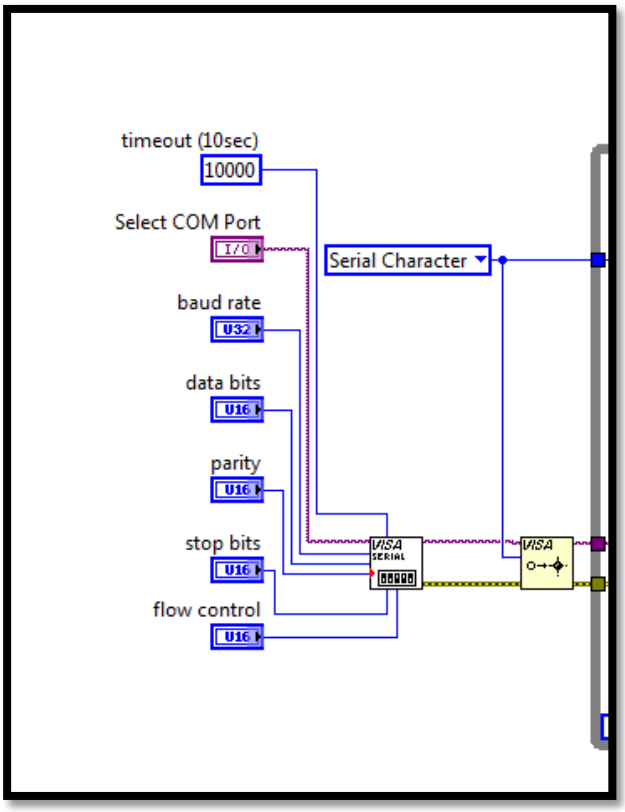


Figura 40. Configuración Serial.

1. Timeout. Especifica el tiempo, en milisegundos, para las operaciones de lectura y escritura del puerto. Se ha utilizado el tiempo predeterminado de LabVIEW de 10 segundos.

2. Select COM Port. Especifica el puerto a abrir. Este parámetro también especifica la sesión y la clase. Éste es el único parámetro de configuración visible para el operador.

3. Baud rate. Es la tasa de transmisión que representa la cantidad de veces que cambia el estado de una señal en un periodo de tiempo (en este caso coincide con los bits por segundo). Se ha especificado un Baud rate de 115200.

4. Data bits. Es el número de bits en los datos de entrada. Estos valores están entre 5 y 8. Se especificó un valor de 8.

5. Parity. Especifica la paridad usada para cada trama a ser transmitida o recibida. Los valores que acepta se muestran en la tabla XX. Se especificó un valor de 0.

6. Stop bits. Especifica el número de bits de parada usados para indicar el final de una trama. Esta entrada acepta los valores mostrados en la tabla xx. Se especificó un valor de 1.0.

7. Flow control. Establece el tipo de control usado por el mecanismo de transferencia. Esta entrada acepta los valores mostrados en la tabla xx. Se especificó un valor de 0.

0	no parity (default)
1	odd parity
2	even parity
3	mark parity
4	space parity

Tabla 8. Paridad

1.0	**1 stop bit**
1.5	1.5 stop bits
2.0	2 stop bits

Tabla 9. Stop bits.

0	None **(default)**
1	**XON/XOFF**
2	**RTS/CTS**
3	**XON/XOFF and RTS/CTS**
4	**DTR/DSR**
5	**XON/XOFF and DTR/DSR**

Tabla 10. Control de Flujo.

- Set Point de la velocidad del husillo.

Para establecer la velocidad por medio de la interfaz se utiliza un instrumento tipo perilla “Knob” que simula un tacómetro, como se aprecia en la figura XX, con lo que se crea la sensación de advertencia en el operador al momento de incrementar demasiado la velocidad. Ésta herramienta se puede manipular manualmente en la intefaz moviendo con el mouse la flecha indicadora de color azul que se muestra en la fugura hasta obtener el valor de la velocidad deseado; o, en su defecto, también se puede escribir la velocidad deseada en el recuadro inferior central, el cual, automáticamente actualizará la flecha a la posición deseada.

Figura 41. Tacómetro de Velocidad.

A continuación se detalla la secuencia a seguir para una correcta determinación de la velocidad con la interfaz creada.

Primero, el operador establece en el tacómetro la velocidad del husillo deseada. Una vez seleccionada, presionar el botón llamado “Set Vel”. Éste, como se muestra en la fugura 41, enviará una señal positiva a un par de “Case Structure” donde, la primera, enviará al microcontrolador la palabra “VEL.”, indicándole que recibirá un valor de velocidad, seguido de un delay que permite al microcontrolador procesar la información recibida. Acto seguido, el segundo “Case Structure” procesa la información del tacómetro y la convierte en un valor equivalente que interpretará el microcontrolador como el valor de velocidad.

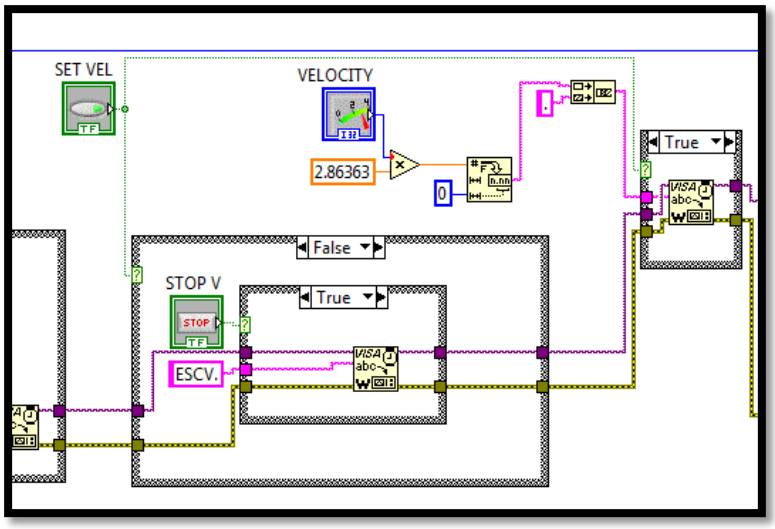


Figura 42. Configuración de la Velocidad.

Por último, si se desea detener el husillo en cualquier momento sin tener que llevar la posición de la flecha del tacómetro a cero, se puede presionar el botón de Stop, el cual, enviará una señal a un “Case Structure”, como se muestra en la figura 42, que enviará el código “ESCV.” al microcontrolador, el cual, interpretará la información para interrumpir la alimentación del motor.

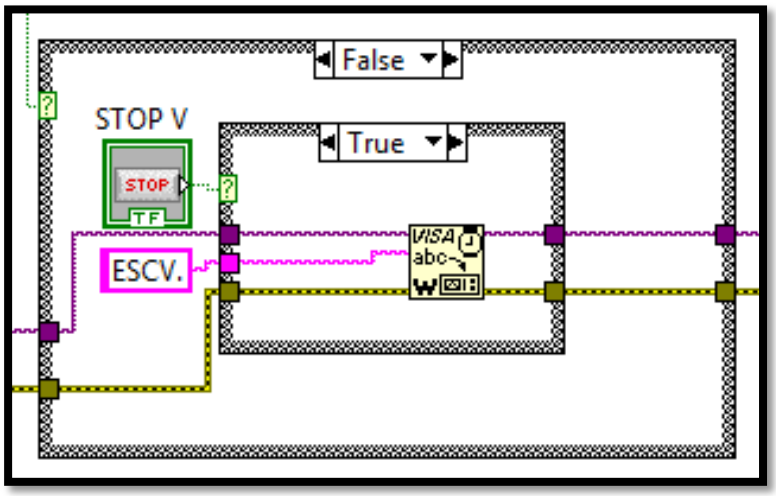


Figura 43. Paro del Motor de Velocidad.

- Set Point de la Posición de la mesa.

Para establecer la posición de la mesa se deben tener en cuenta varios parámetros a configurar para la obtención de un correcto funcionamiento del

control PID. A continuación se muestran y se detallan cada uno de éstos parámetros, los cuales, se muestran de manera gráfica en la figura 43.

1. Slide Position. Establece de manera gráfica la posición deseada expresada en un valor entero positivo representativo de la posición de la mesa en el eje de coordenadas X. Éste valor va desde 0 a xxx. Cuenta además con un recuadro en el que el operador puede introducir de forma numérica el valor deseado.

2. SET POS. Botón que envía la información establecida en el Slide Position al microcontrolador.

3. SET ZERO. Botón que envía un código al microcontrolador indicándole que la posición de la mesa actual será el punto cero en el eje de coordenadas X.

4. RESET. Botón que regresará a la mesa a la posición cero original (extrema izquierda).

5. SET Kp. Botón que envía el código de configuración de la ganancia Kp del controlador PID.

6. SET Kd. Botón que envía el código de configuración de la ganancia Kd del controlador PID.

7. SET Ki. Botón que envía el código de configuración de la ganancia Ki del controlador PID.

8. STOP READ. Botón que interrumpe la lectura de información proveniente del microcontrolador.

9. STOP. Botón que detiene el movimiento de la mesa y al mismo tiempo la acción del controlador PID.

Figura 44. Interfaz del controlador PID de posición.

La estructura del programa se muestra en la figura 45, en donde, la secuencia inicia al presionar el botón SET POS, el cual activa un par de case structure, en donde, en primer lugar, se envía un código para indicar al microcontrolador que se enviara un valor de posición. Después de un retardo se envía el valor de posición deseado.

Los valores de las ganancias Kp, Kd y Ki, se introducen, después de acoplar la señal para el microcontrolador, de manera semejante.

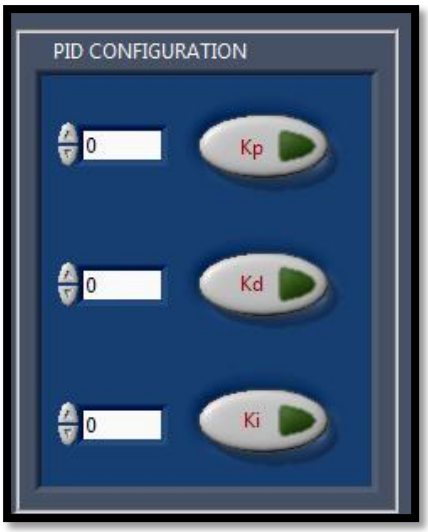


Figura 45. Configuración de los parámetros del controlador PID.

La estructura de los botones Kp, Kd y Ki, como se muestra en la figura 46, es semejante a las estructuras anteriores, a excepción de los botones SET ZERO y RESET, que incluyen, además, una estructura que regresa la posición de la barra del “slide position” a un valor de cero.

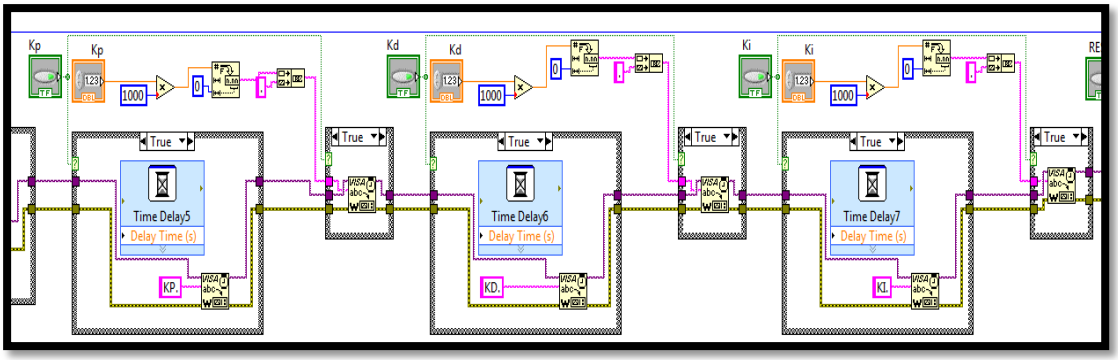


Figura 46. Estructura de los botones Kp, Kd, y Ki.

- Monitoreo de la variable de proceso.

La interfaz de monitoreo se ilustra en la figura 47, en donde se muestra un grafico variable con el tiempo que traza la señal de la variable de entrada junto con la variable de proceso, mostrando, de esta manera, el “set point” introducido por el operador y la respuesta obtenida por el movimiento de la mesa hasta alcanzar el valor de “set point”. En el grafico se aprecia, por lo tanto, la respuesta transitoria y la respuesta en estado estable del controlador PID.

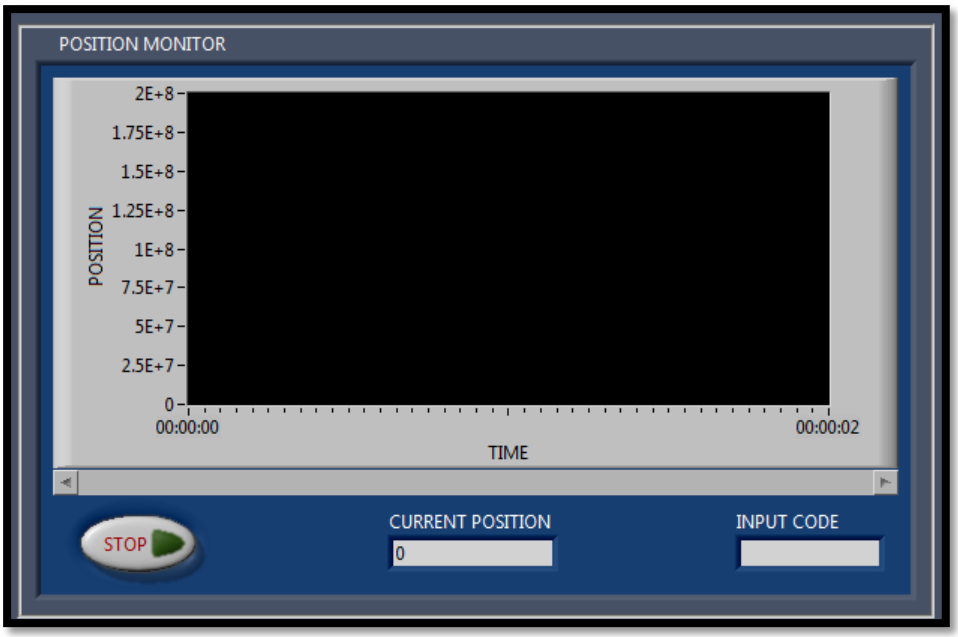


Figura 47. Monitoreo de la variable de entrada y de proceso.

La lectura de los valores captados por el microcontrolador de la variable de proceso se realiza a través de una interrupción, dentro de la cual, el proceso de lectura se cicla hasta que se detecta un byte de fin de la trama. La trama es recibida en forma codificada, ya que se manejan datos enteros de 32 bits, por lo que es procesada para decodificarla y poder enviarla al grafico correspondiente de forma que el operario pueda usarla e interpretarla fácilmente. Lo anterior se ilustra en la figura 48.

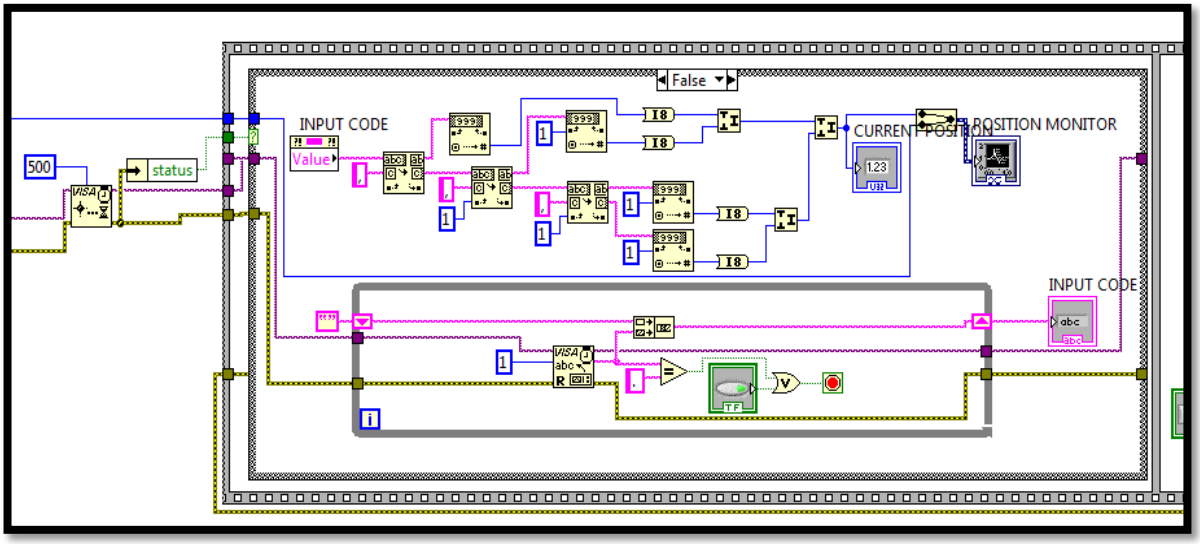


Figura 48. Estructura de la interrupción para la captura de datos.

Por último, para terminar el programa se presiona el botón “END PROGRAM” el cual libera la señal del ciclo infinito terminando en la función que cierra la sesión del puerto serial y dejarlo disponible para otras aplicaciones o, en su defecto, arrojar un mensaje de error según sea el caso.

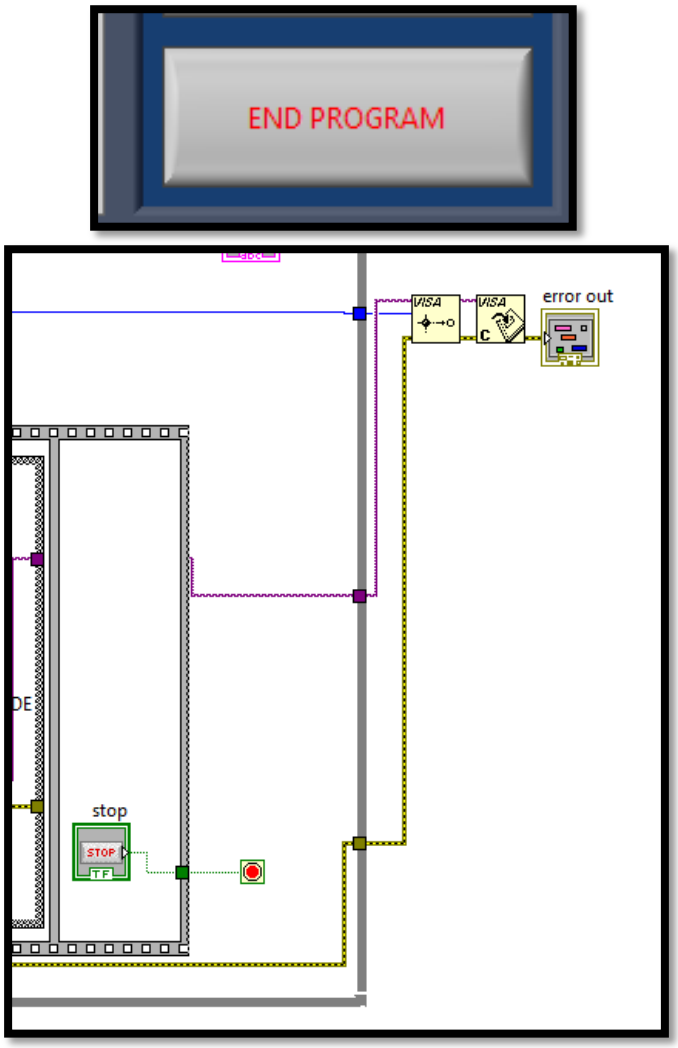


Figura 49. Finalización de la Interfaz.

5. Resultados y Conclusiones.

5.1. Resultados.

Experimento para la determinación de las RPM.

Para poder conocer la velocidad máxima medida en RPM (Revoluciones por minuto) del motor del husillo se utilizó un tacómetro digital, el cual, presenta dos modalidades para la medición de la velocidad.

La primera, mostrada en la figura 50, requiere de un adaptador montado en la flecha del tacómetro el cual hace contacto con la flecha del rotor del motor girando, por consiguiente, a la misma velocidad que éste.

Figura 50. Tacómetro Digital en prueba por contacto.

La segunda modalidad que presenta el tacómetro es por medio de un laser y cinta reflectiva, como se aprecia en la figura 51. En éste método, el tacómetro contará el número de vueltas por unidad de tiempo que da el rotor del motor por medio de la información recibida a través de la señal del laser reflejado en la cinta, convirtiendo esta información en RPM.

Figura 51. Tacómetro digital en prueba reflectiva.

Se realizaron tres pruebas con cada método a máxima velocidad y tres pruebas más con cada método al 50%. Posteriormente se obtuvieron los promedios de las RPM obtenidas y se redondeó el resultado al valor obvio más cercano. Los resultados obtenidos se muestran en la tabla 11.

Velocidad	**Prueba**						**Velocidad**	**Ajuste**
RPM	1	2	3	4	5	6	Promedio	
Máx.100% (10V)	1427	1426	1433	1434	1435	1435	1431.67	**1430**
Med. 50% (5V)	716	722	719	720	717	714	718	**715**
Min. 0% (0V)	0	0	0	0	0	0	0	**0**

Tabla 11. Experimentación con tacómetro Digital.

Experimentación con el PWM.

Para la realización de las pruebas del modulador de ancho de pulso del microcontrolador se utilizó un osciloscopio para poder ver gráficamente el control de la forma del pulso de salida, así como un voltímetro para medir el voltaje arrojado por cada una de las señales utilizadas. (Ver figura 52).

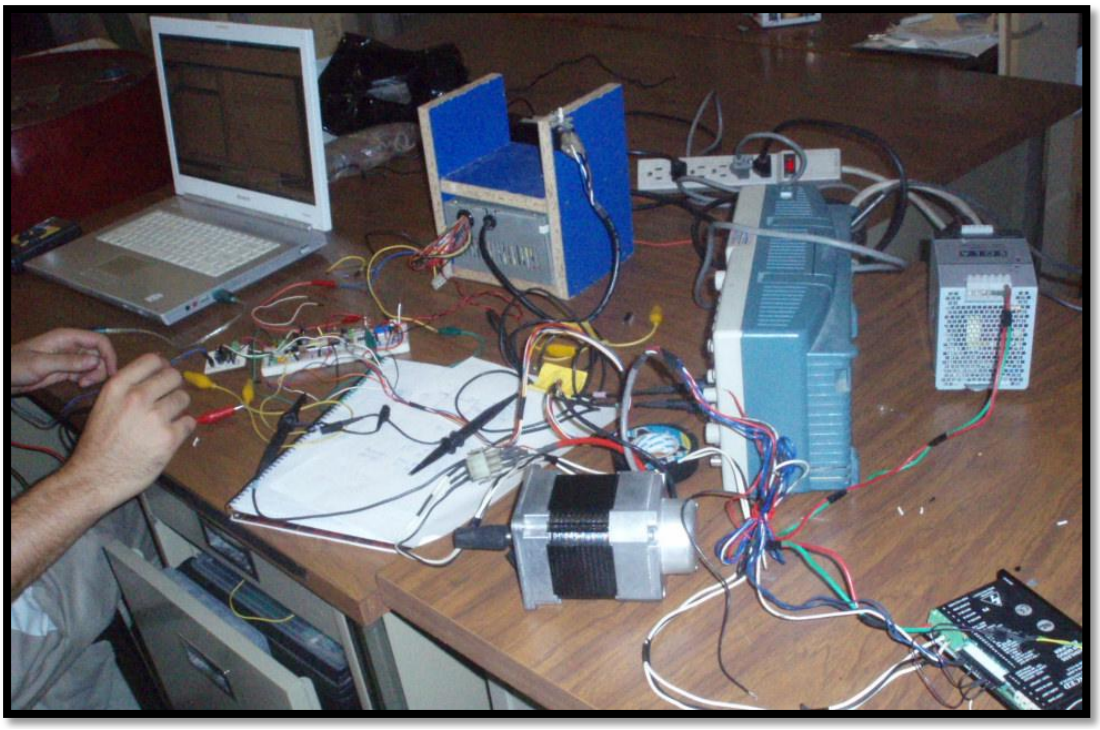


Figura 52. Experimentación con PWM.

A continuación se muestran una serie de imágenes en donde se muestran los resultados obtenidos con el osciloscopio y el multímetro digital para la determinación de la correcta funcionalidad del control de velocidad por medio del PWM del microcontrolador.

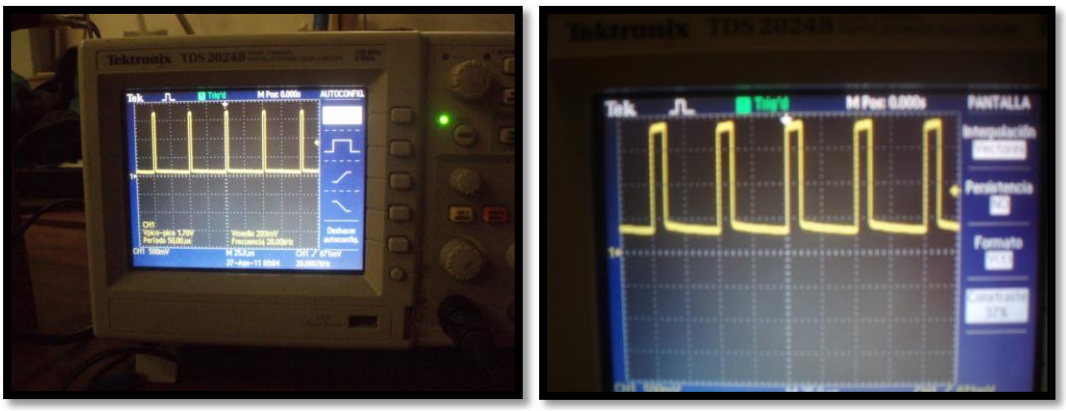


Figura 53. Serie de Imágenes de la Experimentación del PWM.

Experimentación con el DAC.

De manera semejante a las pruebas realizadas con el PWM, para la comprobación del cprrecto funcionamiento del convertidor digital-analógico se utilizó un osciloscopio y un multímetro digital para visualizar los bits enviados al DAC y medir el voltaje de salida del mismo respectivamente. Como se vio anteriormente el DAC maneja una resolución de 12bits, por lo que el voltaje medido es muy preciso. A continuación se muestran una serie de imágenes en donde se ilustran los resultados obtenidos.

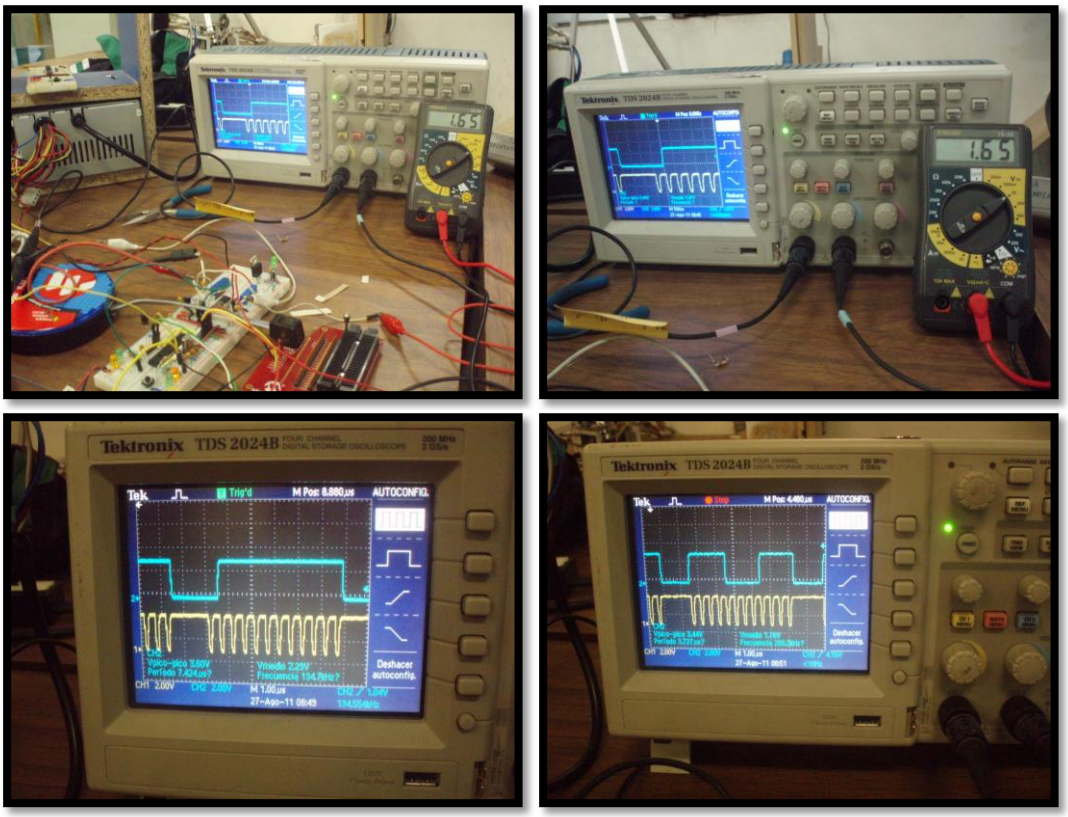


Figura 54. Experimentación con el DAC.

SINTONIZACION DEL MOTOR.

Para la sintonización del motor y la adquisición de las ganancias requeridas para el control PID, Gallil nos ofrece un software capaz de realizar esta función a

partir de distintos métodos de sintonización. En el Anexo xx se explica más acerca de esta herramienta y sus características. A continuación se presentan las fases de la experimentación que se realizó para adquirir estos parámetros a partir del software antes mencionado.

MÉTODO “AUTO CROSSOVER FREQUENCY”

Con un voltaje máximo de 24V. se sometió el motor a la primera prueba(Figura 55) sin tomar en cuenta la magnitud limitada de paso de corriente para el motor del servodriver. En la figura 55 se muestra la señal generada después de realizar la prueba durante 2 ms, al igual que las ganancias PID obtenidas.

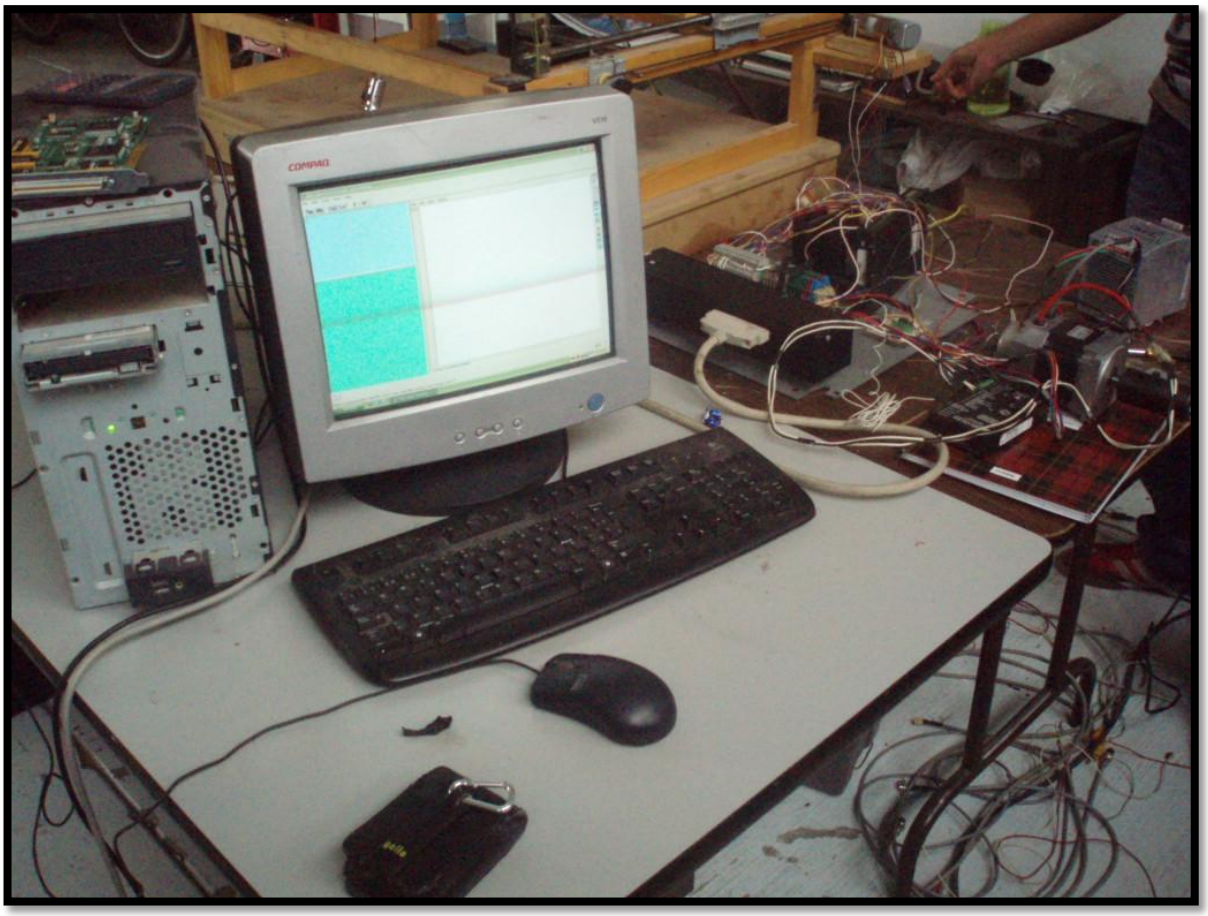


Figura 55. Pruebas con la Tarjeta Galil.

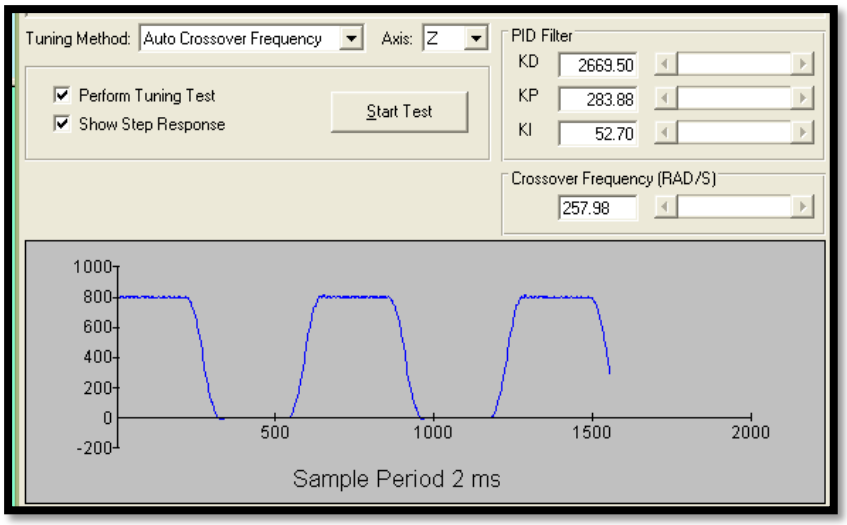


Figura 56. Primera sintonización del motor.

La siguiente prueba se realizo enseguida, sin necesidad de modificar los parámetros o características de alimentación del motor. El objetivo de ésta prueba fue el corroborar los datos obtenidos durante la prueba con los anteriores. Como se puede notar en la Figura 57, hubo una “ligera” variación de datos, pero conservaron su relación entre ellos.

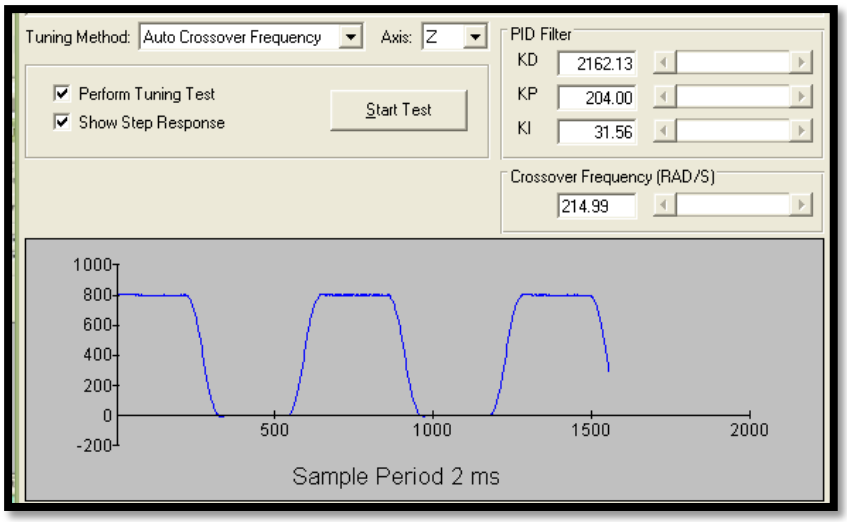


Figura 57. Segunda prueba de sintonización del motor.

Durante la tercera prueba de sintonización del motor. Con un voltaje máximo de 24V y con el mínimo de limitación de corriente del servodriver se

obtuvo la respuesta mostrada en la figura 58, la cual, después de observarla, podemos concluir que se descarta la configuración, pero primero habrá que corroborar que el error no se dio por algún problema de eléctrico o mecánico inherentes en el sistema.

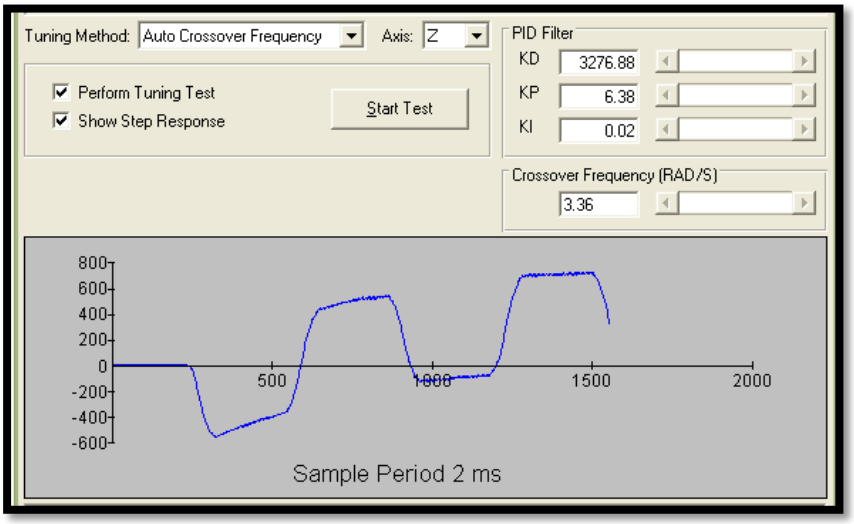


Figura 58. Tercera prueba de sintonización del motor.

Corroborando se obtuvo la cuarta y última prueba de sintonización del motor por el método de Auto Crossover Frequency. El resultado se muestra en la Figura xx. Podemos notar que la señal mejoró pero áun sigue teniendo algunas variaciones inconvenientes en comparación de las primeras pruebas.

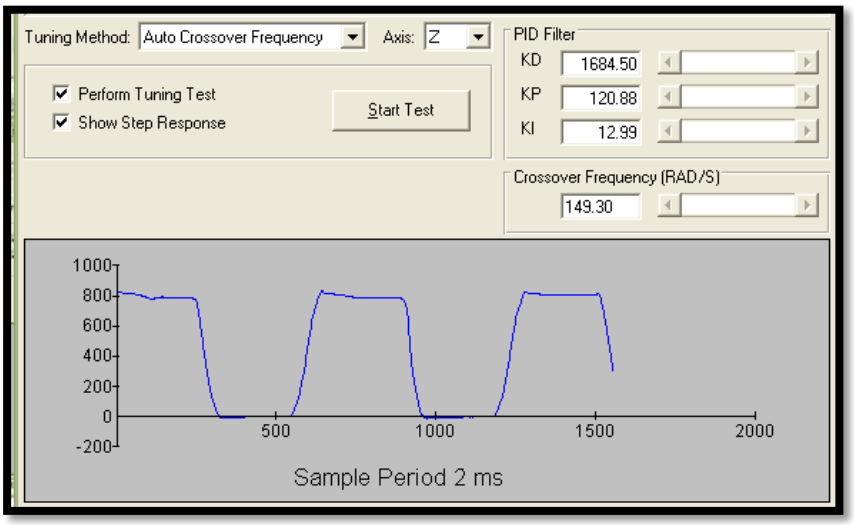


Figura 59. Cuarta prueba de sintonización del motor.

Método “CROSSOVER FREQUENCY”

Explorando las utilidades que posee el software se realizaron dos últimas pruebas pero ahora sin autoajuste, logrando obtener las respuestas mostradas en la Figura 60 y Figura 61.

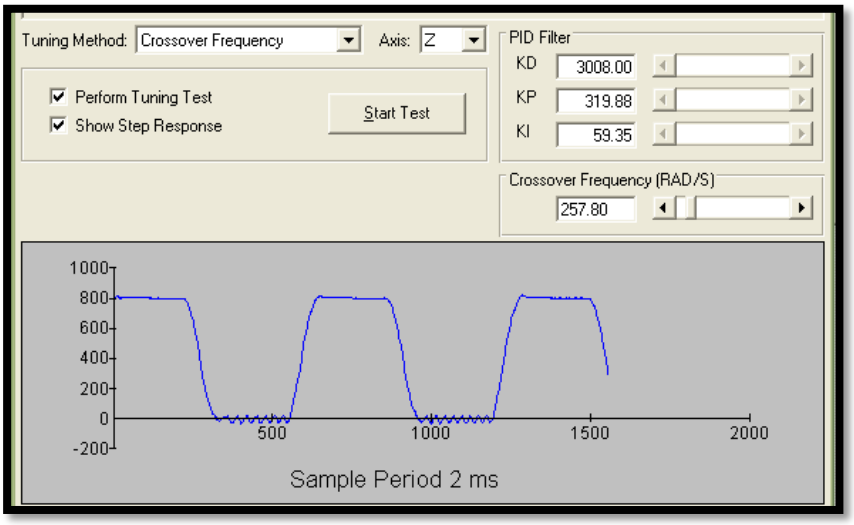


Figura 60. Primera prueba sin autoajuste.

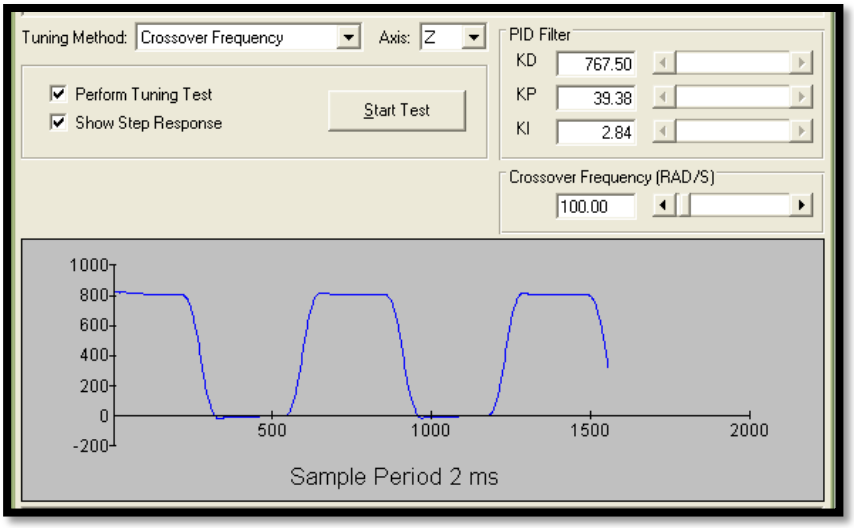


Figura 61. Segunda prueba sin autoajuste.

Cada una de las ganancias estimadas anteriormente son probadas en el software generado para la bancada, esperando permanecer con la que tenga mejor respuesta.

Experimento tipo prueba y error para el ajuste y la determinación de las variables del controlador PID de posición.

5.2. Conclusiones.

Para el presente proyecto podemos decir que los objetivos especificados fueron alcanzados, cumpliendo con la correcta implementación del control e instrumentación de los dos ejes de la bancada por medio de un microcontrolador dsPIC, manipulando las variables por medio de un HMI gráfico y fácil de utilizar para el operador de la máquina.

Además, con los resultados del proyecto, se pudo constatar que las hipótesis predichas fueron correctas, ya que el costo de la automatización de la velocidad del husillo, y la posición de la mesa es sumamente más bajo que el de adquirir una máquina ya automatizada como se vio previamente.

Así mismo, el control de la velocidad del husillo es más preciso que la manipulación de la perilla de ajuste manual, lo que convierte a este proceso en un proceso más eficiente, sumando calidad y confianza al producto final.

Sin embargo, la ventaja en la precisión y velocidad de ajuste de la posición de la mesa de la máquina, sólo puede ser asumida, ya que no se cuenta con la herramienta necesaria para el ajuste manual de la misma. Aún así, parece clara la ventaja asumida, pues la velocidad con que se ajusta dicha posición es evidentemente superior a la velocidad con la que un operador podría ajustar la posición de forma segura manualmente.

Bibliografía

- "Open System Architecture Modular Tool Kit for Motion and Machining Process Control, N. Arda Erol, Yusuf Altintas, and Mabo Robert Ito*, Member, IEEE, VOL. 5, NO. 3, SEPTEMBER 2000*"

- "Modular CNC Design for Intelligent Machining, Part 1:Design of a Hierarchical Motion Control Module for CNC Machine Tools, N. Newell, Department of Electrical Engineering, The University of British Columbia, 506 / Vol. 118, NOVEMBER 1996"

- Datasheet dsPIC33FJ128MC-804

- Dr. Jacob Tal, "Motion Control Applications"

- M. Nakamura, S. Goto, N. Kyura, "Mechatronic Servo System Control", Springer.

- Jung Uk Cho, Quy Ngoc Le, and Jae Wook Jeon. "An FPGA-Based Multiple-Axis Motion Control Chip" IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 56, NO. 3, MARCH 2009

- Robert D. Lorenz, Thomas A. Lipo, and Donald W. Novotny, "Motion Control with Induction Motors"

- Agustín Cruz Contreras, J.Carlos González Robles, "Control de Movimiento P.I.D. Digital para Servomotor de DC 1.5KW", Centro de Innovación y Desarrollo Tecnológico en Cómputo,

- "Entrevista al diputado Jose Alberto Benavides Castañeda, Presidente de la Comisión de Fomento Económico; 14 de Agosto 2010"

- www.microchip.com 15/09/10

- http://search.microchip.com/searchapp/?id=2&q=dsPIC33FJ128MC804 15/09/10

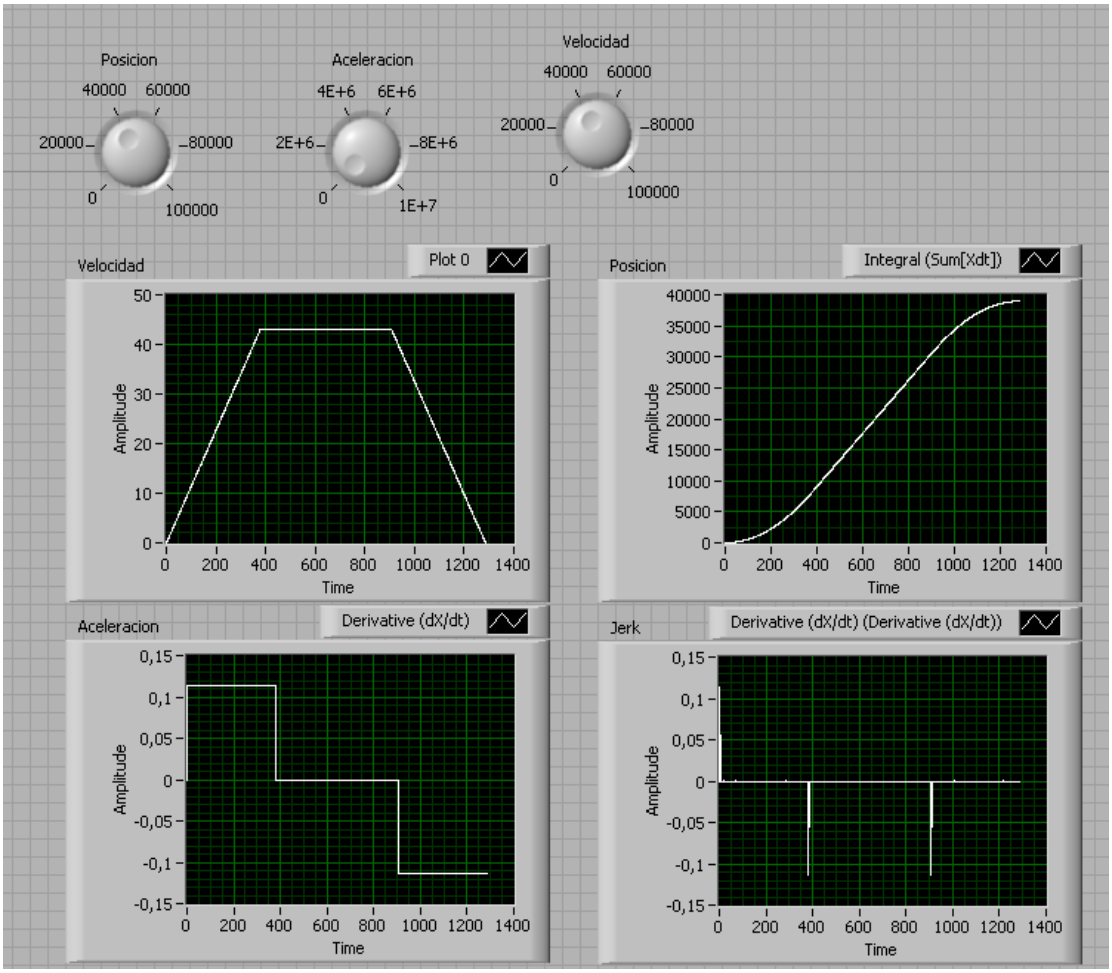
- Norman S. Nise. 2009. “Sistemas de control para ingenieria”. Grupo Editorial Patria. Tercera edición. Mexico. Pp. 2, 9 y 14-15.

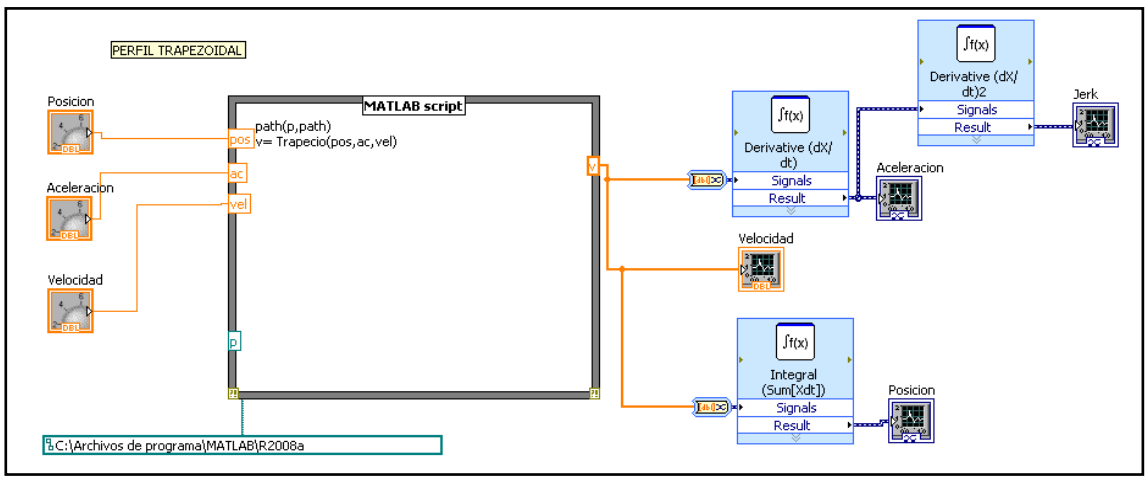
- Manuel Ortega Cantero, José Bravo; Sistemas de interacción persona-computador; Univ de Castilla La Mancha, 2001.

- Aquilino Rodríguez Penin; Sistemas Scada; Marcombo, 2007.

Apéndice.

ANEXO A.





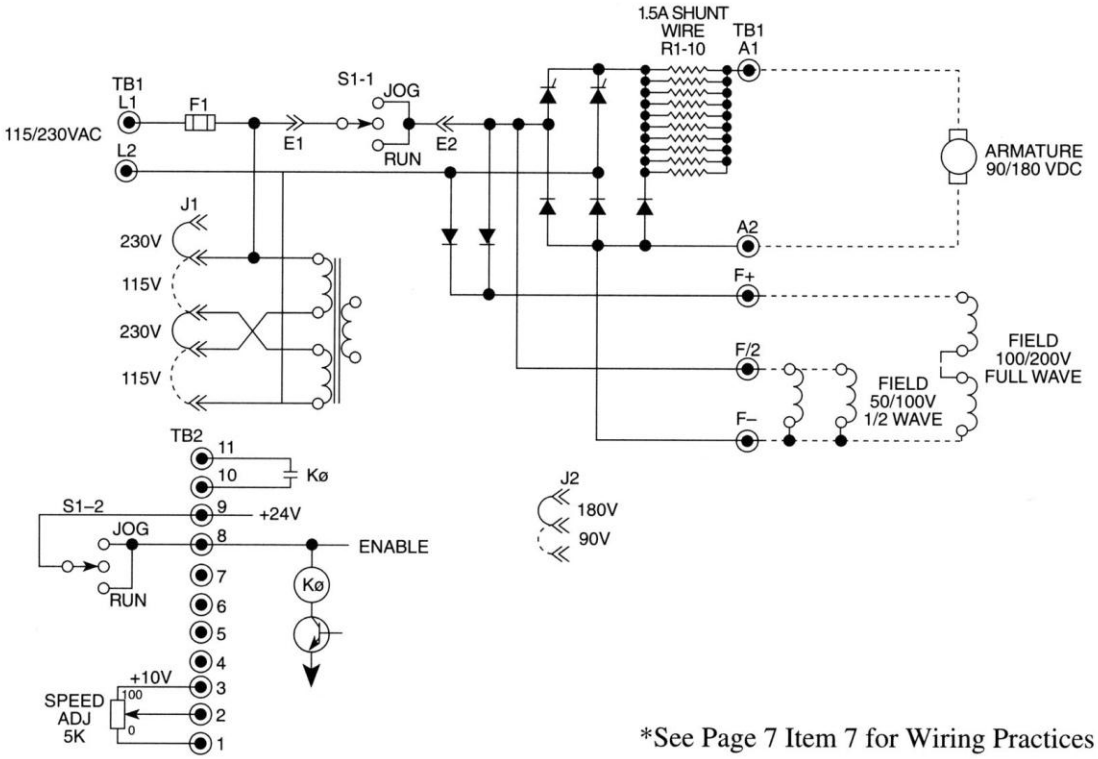

```
function [v,s]= Trapecio(pos,ac,vel)

Ts=0.001; %Mínimo 1 ms Máximo 100us
% obtener datos del perfil
```

```
pdes = pos;
amax = ac;
vmax = vel;

%Calcular tiempos e intervalos
tmax = pdes / vmax;
t1 = (vmax/amax)/Ts;
t3 = t1;
t2 = (tmax / Ts) - t1;

%Ajustar valores en cuentas por tiempo de muestreo
acel = amax * Ts * Ts;
v(1) = 0;
s(1) = 0;
k = 2;

for i = 0 : t1
    v(k) = acel+ v(k-1);
    s(k) = v(k) + s(k-1);
    k = k+1;
end

for i =0:t2
    v(k) = v(k-1);
    s(k) = v(k) + s(k-1);
    k=k+1;
end

for i=0 : t3
    v(k) = -acel+v(k-1);
    s(k)=v(k)+s(k-1);
    k=k+1;
end

subplot(2,1,1)
plot(Ts* (1:length(v)),v/Ts)
title('Perfil de velocidad trapezoidal')
xlabel('Tiempo (s)'), ylabel('Cuentas/s')
subplot(2,1,2)
plot(Ts*(1:length(s)),s)
title('Perfil de posición')
xlabel('Tiempo (s)'),ylabel('Cuentas')
```

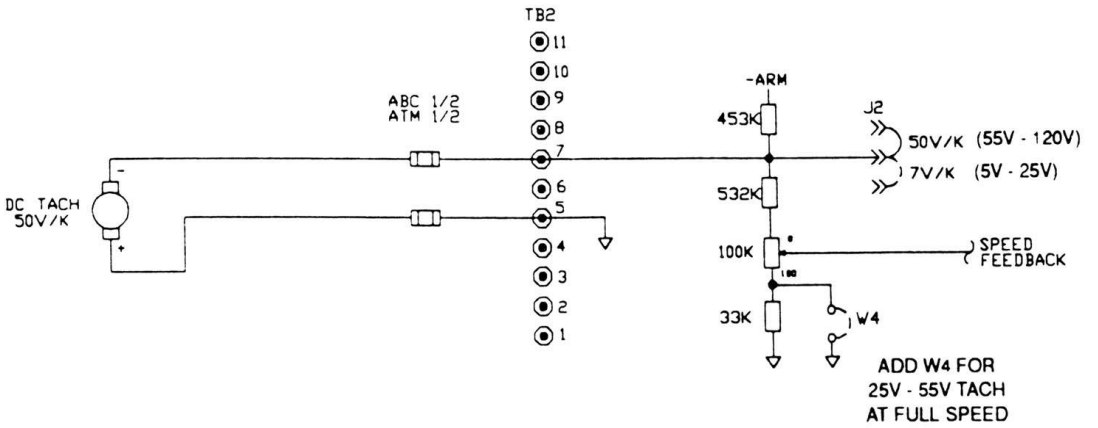
Ratiotrol®

DC Motor Speed Control

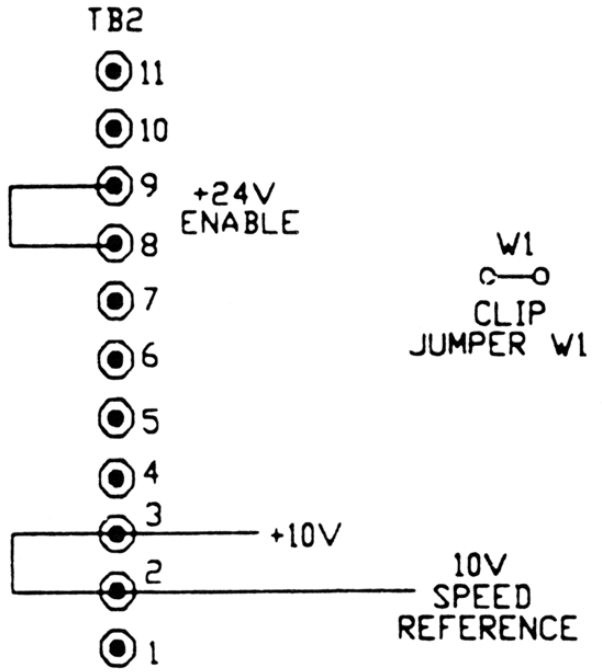
Doc. No. 57762
Beta II Series
Single Phase
1/6-3 HP



LOGIC CONNECTION DIAGRAM, RUN-STOP-JOG SWITCH, 1/6 - 2 HP



SIGNAL CONNECTION DIAGRAM, TACHOMETER FEEDBACK



SIGNAL CONNECTION DIAGRAM, LINE STARTING WITHOUT A MOTOR SPEED POTENTIOMETER

INITIAL STARTUP

1. Remove the controller cover (if used) by removing the four cover screws.

2. Be sure all wiring is correct and all connections are tightened securely.

3. Be sure the controller is calibrated correctly. See steps 4 and 5 under "Installing The Controller" on pages 7 and 8.

4. Be sure the AC line voltage to the controller agrees with the controller nameplate.

5. The potentiometers in the controller are factory adjusted as shown in Table 4. These settings will provide satisfactory operation for most applications. If different settings are required, refer to "Adjustment Instructions" starting on page 18.

TABLE 4. INITIAL POTENTIOMETER SETTINGS

POTENTIOMETER	SETTING	DESCRIPTION
CUR LMT	Fully Clockwise (100%)	150% Load
MIN SPD	Fully Counterclockwise (0%)	0% Speed
MAX SPD	3/4 Turn Clockwise	100% Speed
DECEL	2/3 Turn Clockwise	10 Seconds
ACCEL	2/3 Turn Clockwise	10 Seconds
IR/COMP	Fully Counterclockwise (0%)	0% Boost

6. Replace the controller cover, if used, and secure it with four cover screws.

7. Switch on the AC power to the controller.

8. Check motor rotation, as follows:

   a. If a MOTOR SPEED potentiometer is used, turn it fully counterclockwise. If an external signal is used for the speed reference, set it at minimum.

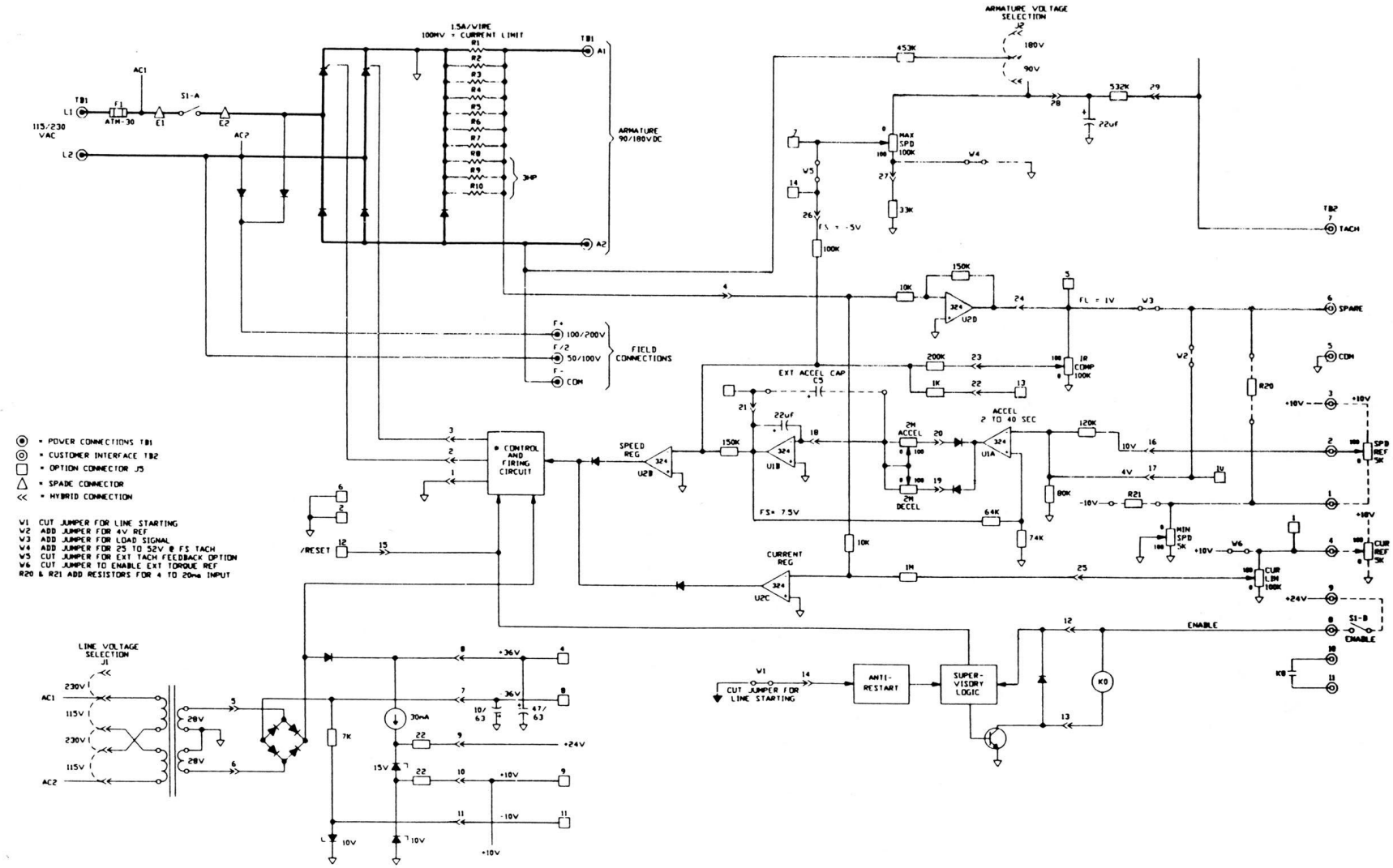
   b. If a RUN-STOP-JOG switch is used, place it in RUN position. Otherwise, initiate a RUN command.

   c. Turn the MOTOR SPEED potentiometer clockwise or increase the speed reference signal, as applicable. To stop the motor, place the switch in STOP position or initiate a STOP command, as applicable.

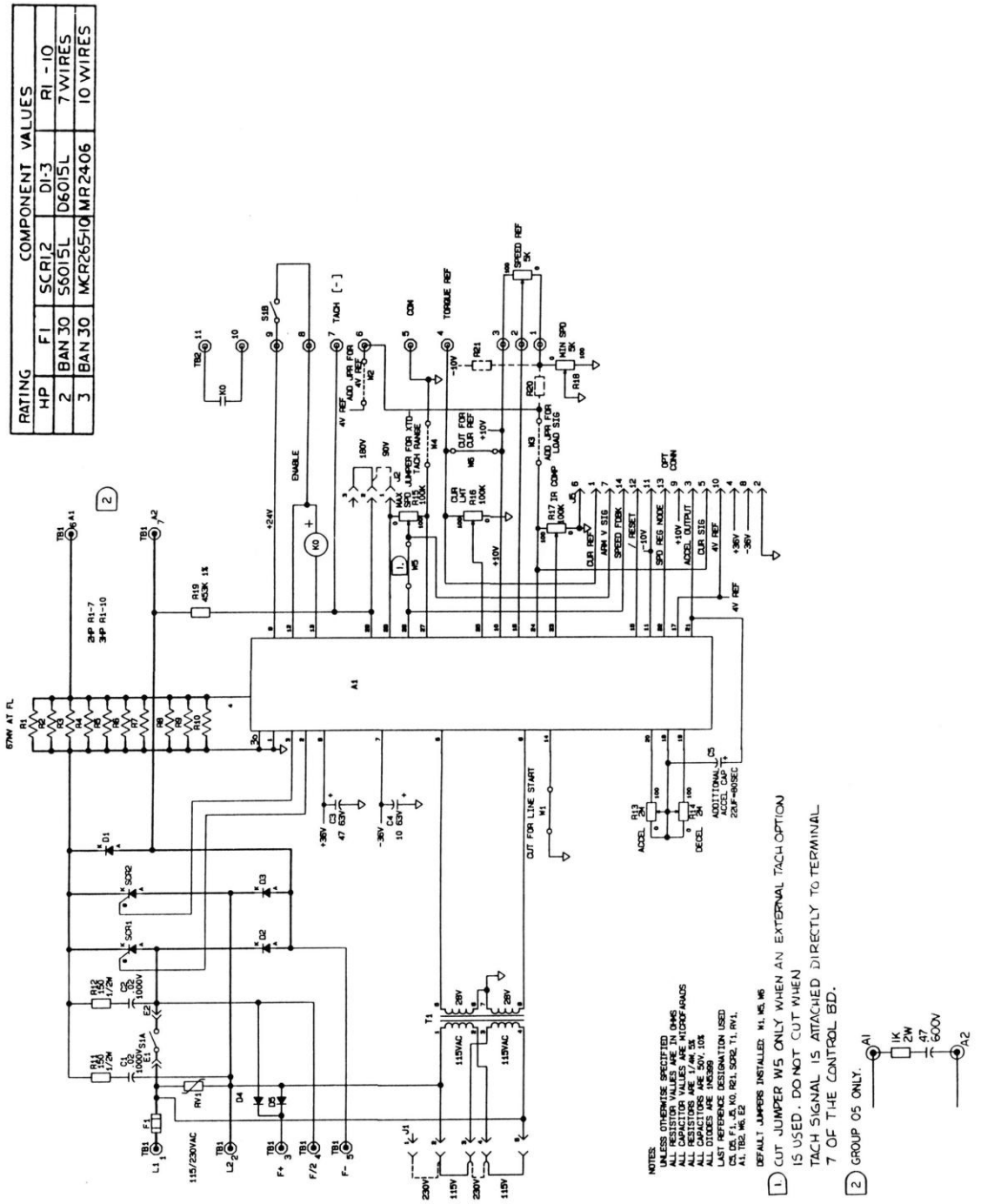
   If the motor rotates in the wrong direction, disconnect the AC power to the control, and then interchange the motor armature leads at the motor connection box or at the controller terminal board.

9. Refer to Section III, "Operation" for operating instructions.

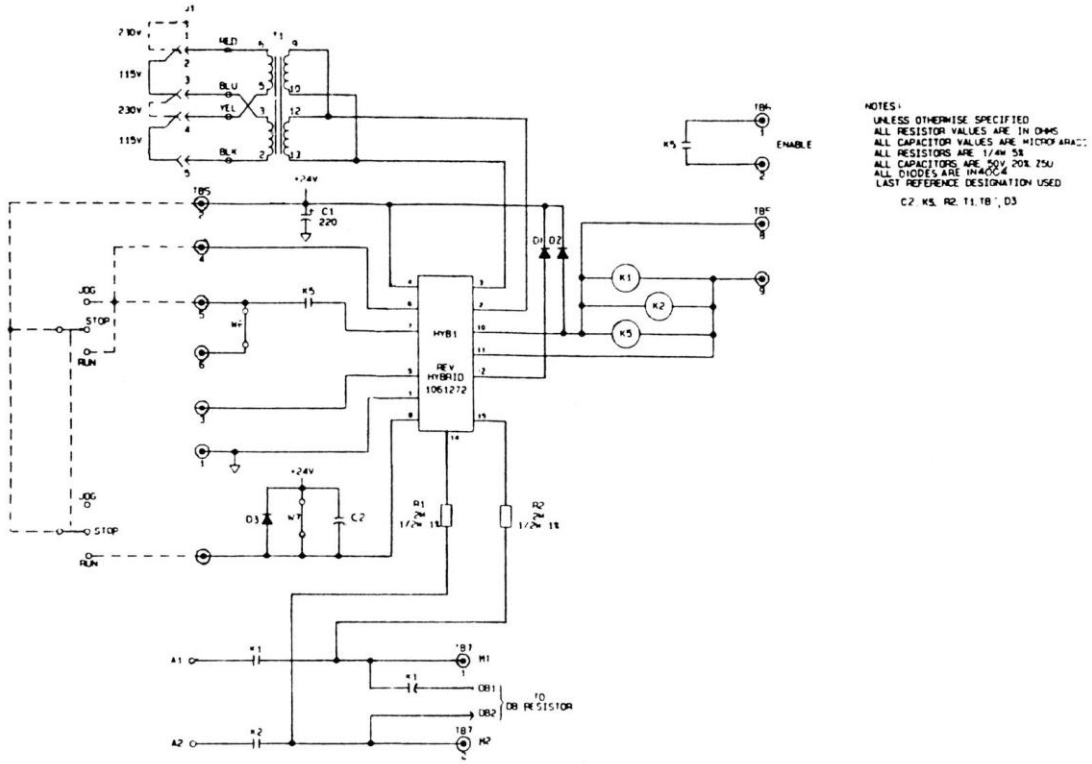
DRAWINGS


ARMATURE VOLTAGE SELECTION
1.5A/VIPE
100MV = CURRENT LIMIT
TB1
L1
L2
115/230 VAC
AC1
AC2
F1
ATM-30
E1
E2
S1-A
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
3HP
ARMATURE 90/180VDC
A1
A2
453K
180V
90V
532K
22uf
MAX SPD 100K
W4
W5
33K
FS = -5V
100K
TB2
TACH
150K
10K
U2D
FL = 1V
W3
SPARE
COM
F+
100/200V
F/2
50/100V
F-
COM
FIELD CONNECTIONS
EXT ACCEL CAP
C5
200K
IR COMP 100K
1K
W2
R20
+10V
SPD REF 5K
22uf
ACCEL 2 TO 40 SEC
2M ACCEL
2M DECEL
U1B
U1A
120K
10V
4V
80K
-10V
R21
FS= 7.5V
64K
74K
MIN SPD 5K
W6
CUR REF 5K
CUR LIM 100K
+24V
CONTROL AND FIRING CIRCUIT
SPEED REG
U2B
/RESET
CURRENT REG
U2C
1M
25
ENABLE
S1-B
K0
W1
CUT JUMPER FOR LINE STARTING
ANTI-RESTART
SUPER-VISORY LOGIC
LINE VOLTAGE SELECTION
J1
230V
115V
28V
+36V
36V
30mA
7K
+24V
15V
+10V
-10V
10V
W1 CUT JUMPER FOR LINE STARTING
W2 ADD JUMPER FOR 4V REF
W3 ADD JUMPER FOR LOAD SIGNAL
W4 ADD JUMPER FOR 25 TO 52V @ FS TACH
W5 CUT JUMPER FOR EXT TACH FEEDBACK OPTION
W6 CUT JUMPER TO ENABLE EXT TORQUE REF
R20 & R21 ADD RESISTORS FOR 4 TO 20ma INPUT
= POWER CONNECTIONS TB1
= CUSTOMER INTERFACE TB2
= OPTION CONNECTOR J5
= SPADE CONNECTOR
= HYBRID CONNECTION

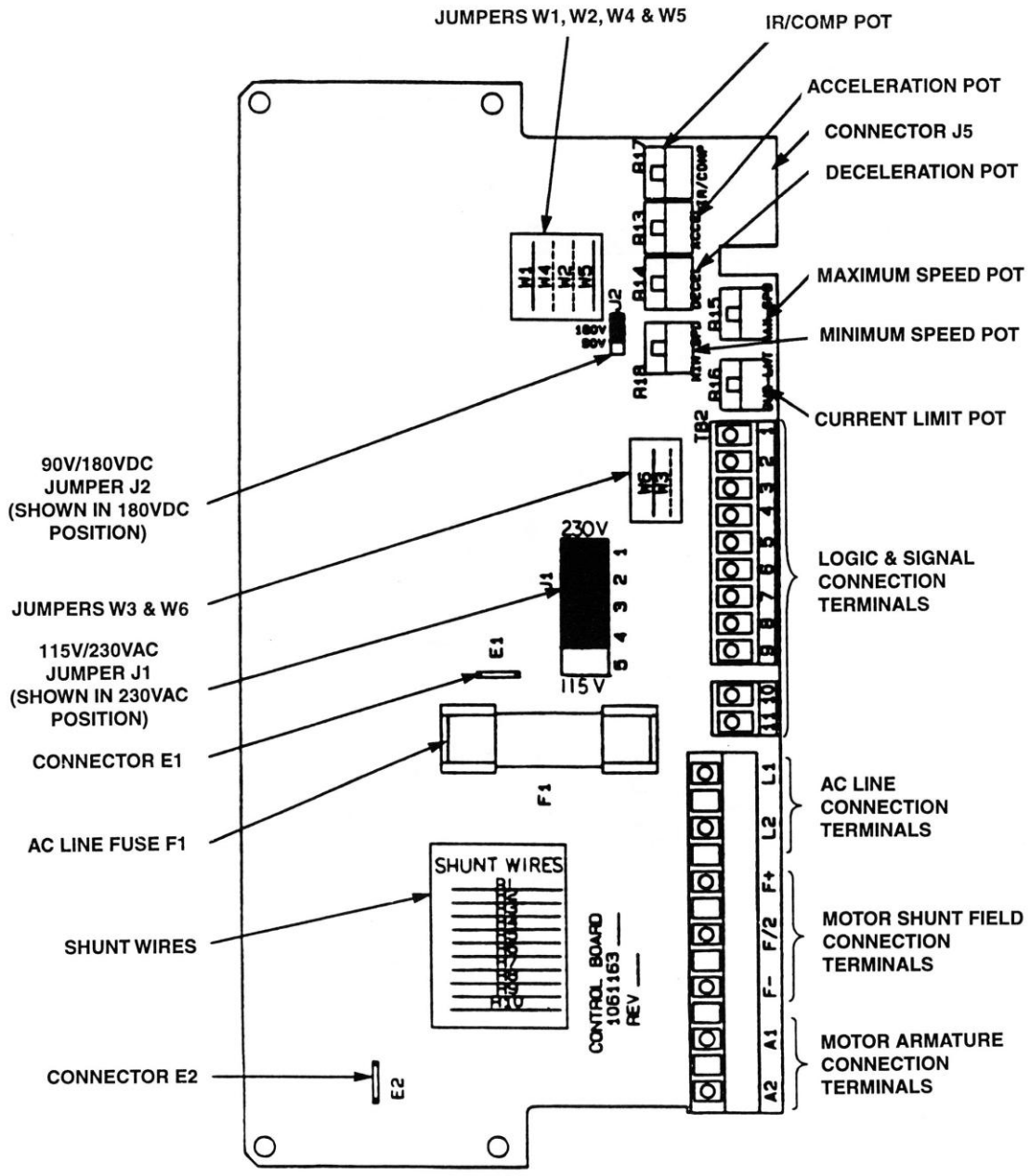
FUNCTIONAL SCHEMATIC, BETA II



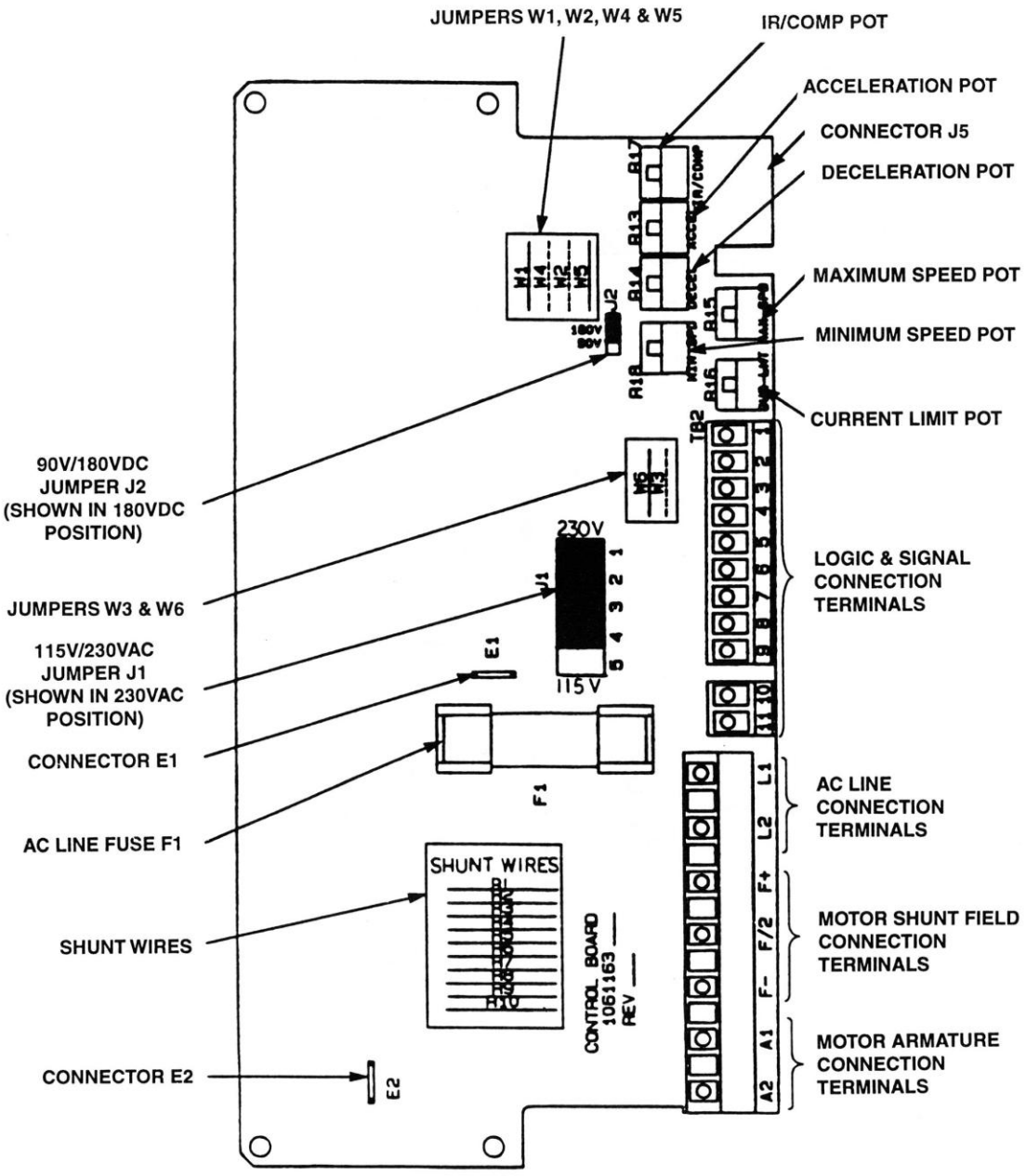
SCHEMATIC, BETA II, 1/6 - 3 HP



SCHEMATIC, UNIDIRECTIONAL ARMATURE CONTACTOR BOARD



BETA II CONTROL BOARD, 1/6 - 3 HP



BETA II CONTROL BOARD, 1/6 - 3 HP

MEMORY ORGANIZATION

TABLE 4-1: CPU CORE REGISTERS MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
WREG0	0000	Working Register 0																0000
WREG1	0002	Working Register 1																0000
WREG2	0004	Working Register 2																0000
WREG3	0006	Working Register 3																0000
WREG4	0008	Working Register 4																0000
WREG5	000A	Working Register 5																0000
WREG6	000C	Working Register 6																0000
WREG7	000E	Working Register 7																0000
WREG8	0010	Working Register 8																0000
WREG9	0012	Working Register 9																0000
WREG10	0014	Working Register 10																0000
WREG11	0016	Working Register 11																0000
WREG12	0018	Working Register 12																0000
WREG13	001A	Working Register 13																0000
WREG14	001C	Working Register 14																0000
WREG15	001E	Working Register 15																0800
SPLIM	0020	Stack Pointer Limit Register																xxxx
ACCAL	0022	ACCAL																xxxx
ACCAH	0024	ACCAH																xxxx
ACCAU	0026	ACCA<39>								ACCAU								xxxx
ACCBL	0028	ACCBL																xxxx
ACCBH	002A	ACCBH																xxxx
ACCBU	002C	ACCB<39>								ACCBU								xxxx
PCL	002E	Program Counter Low Word Register																xxxx
PCH	0030	—	—	—	—	—	—	—	—	Program Counter High Byte Register								0000
TBLPAG	0032	—	—	—	—	—	—	—	—	Table Page Address Pointer Register								0000
PSVPAG	0034	—	—	—	—	—	—	—	—	Program Memory Visibility Page Address Pointer Register								0000
RCOUNT	0036	Repeat Loop Counter Register																xxxx
DCOUNT	0038	DCOUNT<15:0>																xxxx
DOSTARTL	003A	DOSTARTL<15:1>															0	xxxx
DOSTARTH	003C	—	—	—	—	—	—	—	—	—	—	DOSTARTH<5:0>						00xx
DOENDL	003E	DOENDL<15:1>															0	xxxx
DOENDH	0040	—	—	—	—	—	—	—	—	—	—	DOENDH						00xx
SR	0042	OA	OB	SA	SB	OAB	SAB	DA	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000
CORCON	0044	—	—	—	US	EDT	DL<2:0>			SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	0020
MODCON	0046	XMODEN	YMODEN	—	—	BWM<3:0>				YWM<3:0>				XWM<3:0>				0000
XMODSRT	0048	XS<15:1>															0	xxxx
XMODEND	004A	XE<15:1>															1	xxxx
YMODSRT	004C	YS<15:1>															0	xxxx
YMODEND	004E	YE<15:1>															1	xxxx
XBREV	0050	BREN	XB<14:0>															xxxx
DISICNT	0052	—	—	Disable Interrupts Counter Register														xxxx

TABLE 4-2: CHANGE NOTIFICATION REGISTER MAP FOR dsPIC33FJ128MC202/802, dsPIC33FJ64MC202/802 AND dsPIC33FJ32MC302

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
CNEN1	0060	CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	—	—	—	CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE	0000
CNEN2	0062	—	CN30IE	CN29IE	—	CN27IE	—	—	CN24IE	CN23IE	CN22IE	CN21IE	—	—	—	—	CN16IE	0000
CNPU1	0068	CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	—	—	—	CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE	0000
CNPU2	006A	—	CN30PUE	CN29PUE	—	CN27PUE	—	—	CN24PUE	CN23PUE	CN22PUE	CN21PUE	—	—	—	—	CN16PUE	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-3: CHANGE NOTIFICATION REGISTER MAP FOR dsPIC33FJ128MC204/804, dsPIC33FJ64MC204/804 AND dsPIC33FJ32MC304

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
CNEN1	0060	CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE	CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE	0000
CNEN2	0062	—	CN30IE	CN29IE	CN28IE	CN27IE	CN26IE	CN25IE	CN24IE	CN23IE	CN22IE	CN21IE	CN20IE	CN19IE	CN18IE	CN17IE	CN16IE	0000
CNPU1	0068	CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE	CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE	0000
CNPU2	006A	—	CN30PUE	CN29PUE	CN28PUE	CN27PUE	CN26PUE	CN25PUE	CN24PUE	CN23PUE	CN22PUE	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-4: INTERRUPT CONTROLLER REGISTER MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
INTCON1	0080	NSTDIS	OVAERR	OVBERR	COVAERR	COVBERR	OVATE	OVBTE	COVTE	SFTACERR	DIV0ERR	DMACERR	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000
IFS0	0084	—	DMA1IF	AD1IF	U1TXIF	U1RXIF	SPI1IF	SPI1EIF	T3IF	T2IF	OC2IF	IC2IF	DMA0IF	T1IF	OC1IF	IC1IF	INT0IF	0000
IFS1	0086	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	DMA2IF	IC8IF	IC7IF	—	INT1IF	CNIF	CMIF	MI2C1IF	SI2C1IF	0000
IFS2	0088	—	DMA4IF	PMPIF	—	—	—	—	—	—	—	—	DMA3IF	C1IF(1)	C1RXIF(1)	SPI2IF	SPI2EIF	0000
IFS3	008A	FLTA1IF	RTCIF	DMA5IF	—	—	QEI1IF	PWM1IF	—	—	—	—	—	—	—	—	—	0000
IFS4	008C	DAC1LIF(2)	DAC1RIF(2)	—	—	QEI2IF	FLTA2IF	PWM2IF	—	—	C1TXIF(1)	DMA7IF	DMA6IF	CRCIF	U2EIF	U1EIF	—	0000
IEC0	0094	—	DMA1IE	AD1IE	U1TXIE	U1RXIE	SPI1IE	SPI1EIE	T3IE	T2IE	OC2IE	IC2IE	DMA0IE	T1IE	OC1IE	IC1IE	INT0IE	0000
IEC1	0096	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	DMA2IE	IC8IE	IC7IE	—	INT1IE	CNIE	CMIE	MI2C1IE	SI2C1IE	0000
IEC2	0098	—	DMA4IE	PMPIE	—	—	—	—	—	—	—	—	DMA3IE	C1IE(1)	C1RXIE(1)	SPI2IE	SPI2EIE	0000
IEC3	009A	FLTA1IE	RTCIE	DMA5IE	—	—	QEI1IE	PWM1IE	—	—	—	—	—	—	—	—	—	0000
IEC4	009C	DAC1LIE(2)	DAC1RIE(2)	—	—	QEI2IE	FLTA2IE	PWM2IE	—	—	C1TXIE(1)	DMA7IE	DMA6IE	CRCIE	U2EIE	U1EIE	—	0000
IPC0	00A4	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			4444
IPC1	00A6	—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			—	DMA0IP<2:0>			4444
IPC2	00A8	—	U1RXIP<2:0>			—	SPI1IP<2:0>			—	SPI1EIP<2:0>			—	T3IP<2:0>			4444
IPC3	00AA	—	—	—	—	—	DMA1IP<2:0>			—	AD1IP<2:0>			—	U1TXIP<2:0>			0444
IPC4	00AC	—	CNIP<2:0>			—	CMIP<2:0>			—	MI2C1IP<2:0>			—	SI2C1IP<2:0>			4444
IPC5	00AE	—	IC8IP<2:0>			—	IC7IP<2:0>			—	—	—	—	—	INT1IP<2:0>			4404
IPC6	00B0	—	T4IP<2:0>			—	OC4IP<2:0>			—	OC3IP<2:0>			—	DMA2IP<2:0>			4444
IPC7	00B2	—	U2TXIP<2:0>			—	U2RXIP<2:0>			—	INT2IP<2:0>			—	T5IP<2:0>			4444
IPC8	00B4	—	C1IP<2:0>(1)			—	C1RXIP<2:0>(1)			—	SPI2IP<2:0>			—	SPI2EIP<2:0>			4444
IPC9	00B6	—	—	—	—	—	—	—	—	—	—	—	—	—	DMA3IP<2:0>			0004
IPC11	00BA	—	—	—	—	—	DMA4IP<2:0>			—	PMPIP<2:0>			—	—	—	—	0440
IPC14	00C0	—	—	—	—	—	QEI1IP<2:0>			—	PWM1IP<2:0>			—	—	—	—	0440
IPC15	00C2	—	FLTA1IP<2:0>			—	RTCIP<2:0>			—	DMA5IP<2:0>			—	—	—	—	4440
IPC16	00C4	—	CRCIP<2:0>			—	U2EIP<2:0>			—	U1EIP<2:0>			—	—	—	—	4440
IPC17	00C6	—	—	—	—	—	C1TXIP<2:0>(1)			—	DMA7IP<2:0>			—	DMA6IP<2:0>			0444
IPC18	00C8	—	QEI2IP<2:0>			—	FLTA2IP<2:0>			—	PWM2IP<2:0>			—	—	—	—	4440
IPC19	00CA	—	DAC1LIP<2:0>(2)			—	DAC1RIP<2:0>(2)			—	—	—	—	—	—	—	—	4400
INTTREG	00E0	—	—	—	—	ILR<3:0>				—	VECNUM<6:0>							4444

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: Interrupts disabled on devices without ECAN™ modules.
2: Interrupts disabled on devices without DAC.

TABLE 4-5: TIMER REGISTER MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TMR1	0100	Timer1 Register																xxxx
PR1	0102	Period Register 1																FFFF
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	TSYNC	TCS	—	0000
TMR2	0106	Timer2 Register																xxxx
TMR3HLD	0108	Timer3 Holding Register (for 32-bit timer operations only)																xxxx
TMR3	010A	Timer3 Register																xxxx
PR2	010C	Period Register 2																FFFF
PR3	010E	Period Register 3																FFFF
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		T32	—	TCS	—	0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	—	TCS	—	0000
TMR4	0114	Timer4 Register																xxxx
TMR5HLD	0116	Timer5 Holding Register (for 32-bit timer operations only)																xxxx
TMR5	0118	Timer5 Register																xxxx
PR4	011A	Period Register 4																FFFF
PR5	011C	Period Register 5																FFFF
T4CON	011E	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		T32	—	TCS	—	0000
T5CON	0120	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS<1:0>		—	—	TCS	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-6: INPUT CAPTURE REGISTER MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
IC1BUF	0140	Input 1 Capture Register																xxxx
IC1CON	0142	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000
IC2BUF	0144	Input 2 Capture Register																xxxx
IC2CON	0146	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000
IC7BUF	0158	Input 7 Capture Register																xxxx
IC7CON	015A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000
IC8BUF	015C	Input 8Capture Register																xxxx
IC8CON	015E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-7: OUTPUT COMPARE REGISTER MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
OC1RS	0180	Output Compare 1 Secondary Register																xxxx
OC1R	0182	Output Compare 1 Register																xxxx
OC1CON	0184	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000
OC2RS	0186	Output Compare 2 Secondary Register																xxxx
OC2R	0188	Output Compare 2 Register																xxxx
OC2CON	018A	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000
OC3RS	018C	Output Compare 3 Secondary Register																xxxx
OC3R	018E	Output Compare 3 Register																xxxx
OC3CON	0190	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000
OC4RS	0192	Output Compare 4 Secondary Register																xxxx
OC4R	0194	Output Compare 4 Register																xxxx
OC4CON	0196	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-8: 6-OUTPUT PWM1 REGISTER MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
P1TCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000
P1TMR	01C2	PTDIR	PWM Timer Count Value Register															0000
P1TPER	01C4	—	PWM Time Base Period Register															0000
P1SECMP	01C6	SEVTDIR	PWM Special Event Compare Register															0000
PWM1CON1	01C8	—	—	—	—	—	PMOD3	PMOD2	PMOD1	—	PEN3H	PEN2H	PEN1H	—	PEN3L	PEN2L	PEN1L	00FF
PWM1CON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	IUE	OSYNC	UDIS	0000
P1DTCON1	01CC	DTBPS<1:0>		DTB<5:0>						DTAPS<1:0>		DTA<5:0>						0000
P1DTCON2	01CE	—	—	—	—	—	—	—	—	—	—	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I	0000
P1FLTACON	01D0	—	—	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	—	FAEN3	FAEN2	FAEN1	0000
P1OVDCON	01D4	—	—	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	—	—	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	FF00
P1DC1	01D6	PWM Duty Cycle #1 Register																0000
P1DC2	01D8	PWM Duty Cycle #2 Register																0000
P1DC3	01DA	PWM Duty Cycle #3 Register																0000

Legend: u = uninitialized bit, — = unimplemented, read as '0'

TABLE 4-9: 2-OUTPUT PWM2 REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
P2TCON	05C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000
P2TMR	05C2	PTDIR	PWM Timer Count Value Register															0000
P2TPER	05C4	—	PWM Time Base Period Register															0000
P2SECMP	05C6	SEVTDIR	PWM Special Event Compare Register															0000
PWM2CON1	05C8	—	—	—	—	—	—	—	PMOD1	—	—	—	PEN1H	—	—	—	PEN1L	00FF
PWM2CON2	05CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	IUE	OSYNC	UDIS	0000
P2DTCON1	05CC	DTBPS<1:0>		DTB<5:0>						DTAPS<1:0>		DTA<5:0>						0000
P2DTCON2	05CE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DTS1A	DTS1I	0000
P2FLTACON	05D0	—	—	—	—	—	—	FAOV1H	FAOV1L	FLTAM	—	—	—	—	—	—	FAEN1	0000
P2OVDCON	05D4	—	—	—	—	—	—	POVD1H	POVD1L	—	—	—	—	—	—	POUT1H	POUT1L	FF00
P2DC1	05D6	PWM Duty Cycle #1 Register																0000

Legend: u = uninitialized bit, — = unimplemented, read as '0'

TABLE 4-10: QEI1 REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
QEI1CON	01E0	CNTERR	—	QEISIDL	INDX	UPDN	QEIM<2:0>			SWPAB	PCDOUT	TQGATE	TQCKPS<1:0>		POSRES	TQCS	UPDN_SRC	0000
DFLT1CON	01E2	—	—	—	—	—	IMV<1:0>		CEID	QEOUT	QECK<2:0>			—	—	—	—	0000
POS1CNT	01E4	Position Counter<15:0>																0000
MAX1CNT	01E6	Maximum Count<15:0>																FFFF

Legend: u = uninitialized bit, — = unimplemented, read as '0'

TABLE 4-11: QEI2 REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset State
QEI2CON	01F0	CNTERR	—	QEISIDL	INDX	UPDN	QEIM<2:0>			SWPAB	PCDOUT	TQGATE	TQCKPS<1:0>		POSRES	TQCS	UPDN_SRC	0000
DFLT2CON	01F2	—	—	—	—	—	IMV<1:0>		CEID	QEOUT	QECK<2:0>			—	—	—	—	0000
POS2CNT	01F4	Position Counter<15:0>																0000
MAX2CNT	01F6	Maximum Count<15:0>																FFFF

Legend: u = uninitialized bit, — = unimplemented, read as '0'

TABLE 4-13: UART1 REGISTER MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
U1MODE	0220	UARTEN	—	USIDL	IREN	RTSMD	—	UEN1	UEN0	WAKE	LPBACK	ABAUD	URXINV	BRGH	PDSEL<1:0>		STSEL	0000
U1STA	0222	UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0110
U1TXREG	0224	—	—	—	—	—	—	—	UTX8	UART Transmit Register								xxxx
U1RXREG	0226	—	—	—	—	—	—	—	URX8	UART Received Register								0000
U1BRG	0228	Baud Rate Generator Prescaler																0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-14: UART2 REGISTER MAP

SFR Name	SFR Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
U2MODE	0230	UARTEN	—	USIDL	IREN	RTSMD	—	UEN1	UEN0	WAKE	LPBACK	ABAUD	URXINV	BRGH	PDSEL<1:0>		STSEL	0000
U2STA	0232	UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0110
U2TXREG	0234	—	—	—	—	—	—	—	UTX8	UART Transmit Register								xxxx
U2RXREG	0236	—	—	—	—	—	—	—	URX8	UART Receive Register								0000
U2BRG	0238	Baud Rate Generator Prescaler																0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-24: PERIPHERAL PIN SELECT INPUT REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
RPINR0	0680	—	—	—	INT1R<4:0>					—	—	—	—	—	—	—	—	1F00
RPINR1	0682	—	—	—	—	—	—	—	—	—	—	—	INT2R<4:0>					001F
RPINR3	0686	—	—	—	T3CKR<4:0>					—	—	—	T2CKR<4:0>					1F1F
RPINR4	0688	—	—	—	T5CKR<4:0>					—	—	—	T4CKR<4:0>					1F1F
RPINR7	068E	—	—	—	IC2R<4:0>					—	—	—	IC1R<4:0>					1F1F
RPINR10	0694	—	—	—	IC8R<4:0>					—	—	—	IC7R<4:0>					1F1F
RPINR11	0696	—	—	—	—	—	—	—	—	—	—	—	OCFAR<4:0>					001F
RPINR12	0698	—	—	—	—	—	—	—	—	—	—	—	FLTA1R<4:0>					001F
RPINR13	069A	—	—	—	—	—	—	—	—	—	—	—	FLTA2R<4:0>					001F
RPINR14	069C	—	—	—	QEB1R<4:0>					—	—	—	QEA1R<4:0>					1F1F
RPINR15	069E	—	—	—	—	—	—	—	—	—	—	—	INDX1R<4:0>					001F
RPINR16	06A0	—	—	—	QEB2R<4:0>					—	—	—	QEA2R<4:0>					1F1F
RPINR17	06A2	—	—	—	—	—	—	—	—	—	—	—	INDX2R<4:0>					001F
RPINR18	06A4	—	—	—	U1CTSR<4:0>					—	—	—	U1RXR<4:0>					1F1F
RPINR19	06A6	—	—	—	U2CTSR<4:0>					—	—	—	U2RXR<4:0>					1F1F
RPINR20	06A8	—	—	—	SCK1R<4:0>					—	—	—	SDI1R<4:0>					1F1F
RPINR21	06AA	—	—	—	—	—	—	—	—	—	—	—	SS1R<4:0>					001F
RPINR22	06AC	—	—	—	SCK2R<4:0>					—	—	—	SDI2R<4:0>					1F1F
RPINR23	06AE	—	—	—	—	—	—	—	—	—	—	—	SS2R<4:0>					001F
RPINR26(1)	06B4	—	—	—	—	—	—	—	—	—	—	—	C1RXR<4:0>					001F

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
Note 1: This register is present for dsPIC33FJ128MC802/804 and dsPIC33FJ64MC802/804 devices only.

TABLE 4-25: PERIPHERAL PIN SELECT OUTPUT REGISTER MAP FOR dsPIC33FJ128MC202/802, dsPIC33FJ64MC202/802 AND dsPIC33FJ32MC302

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
RPOR0	06C0	—	—	—	RP1R<4:0>					—	—	—	RP0R<4:0>					0000
RPOR1	06C2	—	—	—	RP3R<4:0>					—	—	—	RP2R<4:0>					0000
RPOR2	06C4	—	—	—	RP5R<4:0>					—	—	—	RP4R<4:0>					0000
RPOR3	06C6	—	—	—	RP7R<4:0>					—	—	—	RP6R<4:0>					0000
RPOR4	06C8	—	—	—	RP9R<4:0>					—	—	—	RP8R<4:0>					0000
RPOR5	06CA	—	—	—	RP11R<4:0>					—	—	—	RP10R<4:0>					0000
RPOR6	06CC	—	—	—	RP13R<4:0>					—	—	—	RP12R<4:0>					0000
RPOR7	06CE	—	—	—	RP15R<4:0>					—	—	—	RP14R<4:0>					0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-26: PERIPHERAL PIN SELECT OUTPUT REGISTER MAP FOR dsPIC33FJ128MC204/804, dsPIC33FJ64MC204/804 AND dsPIC33FJ32MC304

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
RPOR0	06C0	—	—	—	RP1R<4:0>					—	—	—	RP0R<4:0>					0000
RPOR1	06C2	—	—	—	RP3R<4:0>					—	—	—	RP2R<4:0>					0000
RPOR2	06C4	—	—	—	RP5R<4:0>					—	—	—	RP4R<4:0>					0000
RPOR3	06C6	—	—	—	RP7R<4:0>					—	—	—	RP6R<4:0>					0000
RPOR4	06C8	—	—	—	RP9R<4:0>					—	—	—	RP8R<4:0>					0000
RPOR5	06CA	—	—	—	RP11R<4:0>					—	—	—	RP10R<4:0>					0000
RPOR6	06CC	—	—	—	RP13R<4:0>					—	—	—	RP12R<4:0>					0000
RPOR7	06CE	—	—	—	RP15R<4:0>					—	—	—	RP14R<4:0>					0000
RPOR8	06D0	—	—	—	RP17R<4:0>					—	—	—	RP16R<4:0>					0000
RPOR9	06D2	—	—	—	RP19R<4:0>					—	—	—	RP18R<4:0>					0000
RPOR10	06D4	—	—	—	RP21R<4:0>					—	—	—	RP20R<4:0>					0000
RPOR11	06D6	—	—	—	RP23R<4:0>					—	—	—	RP22R<4:0>					0000
RPOR12	06D8	—	—	—	RP25R<4:0>					—	—	—	RP24R<4:0>					0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-32: PORTA REGISTER MAP FOR dsPIC33FJ128MC202/802, dsPIC33FJ64MC202/802 AND dsPIC33FJ32MC302

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISA	02C0	—	—	—	—	—	—	—	—	—	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	079F
PORTA	02C2	—	—	—	—	—	—	—	—	—	—	—	RA4	RA3	RA2	RA1	RA0	xxxx
LATA	02C4	—	—	—	—	—	—	—	—	—	—	—	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx
ODCA	02C6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-33: PORTA REGISTER MAP FOR dsPIC33FJ128MC204/804, dsPIC33FJ64MC204/804 AND dsPIC33FJ32MC304

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISA	02C0	—	—	—	—	—	TRISA10	TRISA9	TRISA8	TRISA7	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	079F
PORTA	02C2	—	—	—	—	—	RA10	RA9	RA8	RA7	—	—	RA4	RA3	RA2	RA1	RA0	xxxx
LATA	02C4	—	—	—	—	—	LATA10	LATA9	LATA8	LATA7	—	—	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx
ODCA	02C6	—	—	—	—	—	ODCA10	ODCA9	ODCA8	ODCA7	—	—	—	—	—	—	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-34: PORTB REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISB	02C8	TRISB15	TRISB14	TRISB13	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	FFFF
PORTB	02CA	RB15	RB14	RB13	RB12	RB11	RB10	RB9	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx
LATB	02CC	LATB15	LATB14	LATB13	LATB12	LATB11	LATB10	LATB9	LATB8	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx
ODCB	02CE	—	—	—	—	ODCB11	ODCB10	ODCB9	ODCB8	ODCB7	ODCB6	ODCB5	—	—	—	—	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-35: PORTC REGISTER MAP FOR dsPIC33FJ128MC204/804, dsPIC33FJ64MC204/804 AND dsPIC33FJ32MC304

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
TRISC	02D0	—	—	—	—	—	—	TRISC9	TRISC8	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	03FF
PORTC	02D2	—	—	—	—	—	—	RC9	RC8	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx
LATC	02D4	—	—	—	—	—	—	LATC9	LATC8	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	xxxx
ODCC	02D6	—	—	—	—	—	—	ODCC9	ODCC8	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	—	—	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

TABLE 4-36: SYSTEM CONTROL REGISTER MAP

File Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
RCON	0740	TRAPR	IOPUWR	—	—	—	—	CM	VREGS	EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR	xxxx(1)
OSCCON	0742	—	COSC<2:0>			—	NOSC<2:0>			CLKLOCK	IOLOCK	LOCK	—	CF	—	LPOSCEN	OSWEN	0300(2)
CLKDIV	0744	ROI	DOZE<2:0>			DOZEN	FRCDIV<2:0>			PLLPOST<1:0>		—	PLLPRE<4:0>					3040
PLLFBD	0746	—	—	—	—	—	—	—	PLLDIV<8:0>									0030
OSCTUN	0748	—	—	—	—	—	—	—	—	—	—	TUN<5:0>						0000
ACLKCON	074A	—	—	SELACLK	AOSCMD<1:0>		APSTSCLR<2:0>			ASRCSEL	—	—	—	—	—	—	—	0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.
Note 1: RCON register Reset values dependent on type of Reset.
2: OSCCON register Reset values dependent on the FOSC Configuration bits and by type of Reset.

PINES REMAPEABLES
ENTRADAS

TABLE 11-1: SELECTABLE INPUT SOURCES (MAPS INPUT TO FUNCTION)(1)

Input Name	Function Name	Register	Configuration Bits
External Interrupt 1	INT1	RPINR0	INT1R<4:0>
External Interrupt 2	INT2	RPINR1	INT2R<4:0>
Timer2 External Clock	T2CK	RPINR3	T2CKR<4:0>
Timer3 External Clock	T3CK	RPINR3	T3CKR<4:0>
Timer4 External Clock	T4CK	RPINR4	T4CKR<4:0>
Timer5 External Clock	T5CK	RPINR4	T5CKR<4:0>
Input Capture 1	IC1	RPINR7	IC1R<4:0>
Input Capture 2	IC2	RPINR7	IC2R<4:0>
Input Capture 7	IC7	RPINR10	IC7R<4:0>
Input Capture 8	IC8	RPINR10	IC8R<4:0>
Output Compare Fault A	OCFA	RPINR11	OCFAR<4:0>
PWM1 Fault	$\overline{\text{FLTA1}}$	RPINR12	FLTA1R<4:0>
PWM2 Fault	$\overline{\text{FLTA2}}$	RPINR13	FLTA2R<4:0>
QEI1 Phase A	QEA1	RPINR14	QEAIR<4:0>
QEI1 Phase B	QEB1	RPINR14	QEBIR<4:0>
QEI1 Index	INDX1	RPINR15	INDXIR<4:0>
QEI2 Phase A	QEA2	RPINR16	QEA2R<4:0>
QEI2Phase B	QEB2	RPINR16	QEB2R<4:0>
QEI2 Index	INDX2	RPINR17	INDX2R<4:0>
UART1 Receive	U1RX	RPINR18	U1RXR<4:0>
UART1 Clear To Send	$\overline{\text{U1CTS}}$	RPINR18	U1CTSR<4:0>
UART2 Receive	U2RX	RPINR19	U2RXR<4:0>
UART2 Clear To Send	$\overline{\text{U2CTS}}$	RPINR19	U2CTSR<4:0>
SPI1 Data Input	SDI1	RPINR20	SDI1R<4:0>
SPI1 Clock Input	SCK1	RPINR20	SCK1R<4:0>
SPI1 Slave Select Input	$\overline{\text{SS1}}$	RPINR21	SS1R<4:0>
SPI2 Data Input	SDI2	RPINR22	SDI2R<4:0>
SPI2 Clock Input	SCK2	RPINR22	SCK2R<4:0>
SPI2 Slave Select Input	$\overline{\text{SS2}}$	RPINR23	SS2R<4:0>
ECAN1 Receive	CIRX	RPINR26	CIRXR<4:0>

Note 1: Unless otherwise noted, all inputs use Schmitt input buffers.

SALIDAS

Function	RPnR<4:0>	Output Name
NULL	00000	RPn tied to default port pin
C1OUT	00001	RPn tied to Comparator1 Output
C2OUT	00010	RPn tied to Comparator2 Output
U1TX	00011	RPn tied to UART1 Transmit
$\overline{\text{U1RTS}}$	00100	RPn tied to UART1 Ready To Send
U2TX	00101	RPn tied to UART2 Transmit
$\overline{\text{U2RTS}}$	00110	RPn tied to UART2 Ready To Send
SDO1	00111	RPn tied to SPI1 Data Output
SCK1	01000	RPn tied to SPI1 Clock Output
$\overline{\text{SS1}}$	01001	RPn tied to SPI1 Slave Select Output
SDO2	01010	RPn tied to SPI2 Data Output
SCK2	01011	RPn tied to SPI2 Clock Output
$\overline{\text{SS2}}$	01100	RPn tied to SPI2 Slave Select Output
C1TX	10000	RPn tied to ECAN1 Transmit
OC1	10010	RPn tied to Output Compare 1
OC2	10011	RPn tied to Output Compare 2
OC3	10100	RPn tied to Output Compare 3
OC4	10101	RPn tied to Output Compare 4
UPDN1	11010	RPn tied to QEI1 direction (UPDN) status
UPDN2	11011	RPn tied to QEI2 direction (UPDN) status

TABLE 31-1: OPERATING MIPS VS. VOLTAGE

Characteristic	V_{DD} Range (in Volts)	Temp Range (in °C)	Max MIPS
			dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 and dsPIC33FJ128MCX02/X04
	3.0-3.6V	-40°C to +85°C	40
	3.0-3.6V	-40°C to +125°C	40