ESCUELA POLITÉCNICA DEL EJÉRCITO

FACULTAD DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO EN INGENIERÍA ELECTRÓNICA

"Diseño e Implementación de un Sistema SCADA Para los Cañones Autopropulsados de 155mm pertenecientes al GAAP-11"

Ana Raquel Díaz Armijos

José Miguel Rueda Sosa

SANGOLQUÍ – ECUADOR

Marzo del 2005

CERTIFICACIÓN

Por medio de la presente certificamos que el proyecto de grado, previo a la obtención del título en Ingeniería Electrónica, titulado “Diseño e Implementación de un Sistema SCADA Para los Cañones Autopropulsados de 155mm pertenecientes al GAAP-11”, fue desarrollado en su totalidad por los señores ANA RAQUEL DIAZ ARMIJOS Y JOSÉ MIGUEL RUEDA SOSA.

ATENTAMENTE,

Ing. Danny Sotomayor, MBA
DIRECTOR

Dr. Nikolai Espinosa, Ph. D
CODIRECTOR

AGRADECIMIENTO

En todos estos años de estudios, aprendí que siempre hay alguien que te da la mano y ayuda a cumplir con tus objetivos, la vida no es fácil pero es mejor si se la recibe con una sonrisa y un palabra, Gracias.

Principalmente a Dios, gracias a él, a su amor y sus bendiciones puedo disfrutar de la vida y los triunfos que he alcanzado.

A nuestros padres, por su inmenso cariño y dedicación, a pesar de los problemas siempre nos sorprendieron con un gesto de apoyo y alegría.

A nuestros hermanos, por ser nuestros compañeros en todo momento y enseñarnos varias lecciones importantes en la vida.

A nuestros amigos que siempre estuvieron presentes en todo momento brindándome su apoyo sincero.

Al CICTE, Mayor Byron Sierra, Danny, Nikolai, Diego, Raúl y Tatiana, que siempre estuvieron dispuestos a entregar su ayuda incondicional y nos acogieron con compañerismo y amistad.

Al Ejército Ecuatoriano por la colaboración para el desarrollo del presente proyecto.

Muchas Gracias.

RAQUEL y JOSE MIGUEL

DEDICATORIA

Con mucho amor y cariño dedico todos los logros de mi carrera universitaria a nuestros padres, que con sus consejos y amor nos han brindado su apoyo incondicional para cumplir mis sueños.

RAQUEL y JOSE MIGUEL

PRÓLOGO

El Trabajo realizado en este proyecto busca brindar mejores prestaciones de la artillería mediana autopropulsada con la que cuenta el Ejército Ecuatoriano, con lo que se realiza una modernización y automatización para el proceso de disparo en los cañones de 155 mm.

Para la ejecución de un disparo en este tipo de cañones de la forma tradicional, es necesario un determinado personal de artillería encargado de calcular datos de tiro utilizando referencias del Sistema de Puntería, posicionar manualmente al tubo tanto en elevación, como en deflexión y cargar la munición, todo este trabajo está propenso a errores humanos ya que depende del criterio y experiencia del equipo humano que lo manipula.

Para la modernización se realiza un reemplazo de los métodos tradicionales de posicionamiento y comunicación, mediante un control automático e inalámbrico, el que cuenta con un PLC como controlador principal del sistema, comunicación mediante radio módems y servomotores como elementos finales de control, además se complementan con el Sistema de Puntería Digital anteriormente desarrollado, creando un verdadero Sistema autónomo, confiable que asegure un tiro efectivo en el menor tiempo posible reduciendo riesgos para el equipo completo.

El método tradicional de posicionamiento del tubo del cañón se realiza mediante 2 manivelas en este proyecto este método manual ha sido sustituido por la ubicación de servomotores que realicen el posicionamiento según datos dados por sensores y por el Centro Director de Tiro

El monitoreo de sensores y cálculos de tiro se realizan en un computador portátil remoto comunicado al cañón vía radio módem, el sistema brinda posibilidades de ser manejado localmente o en algún caso volver a instalar las manivelas y trabajar con métodos anteriores.

Todo el proyecto ha sido desarrollado con el afán de aportar a la modernización y actualización tecnológica del País.

INDICE GENERAL

CAPITULO I

FUNCIONAMIENTO DEL SISTEMA DE PUNTERÍA DIGITAL DEL CAÑÓN AUTOPROPULSADO DE 155mm

CAPITULO II

SOLUCIÓN PROPUESTA Y DESCRIPCIÓN DEL SISTEMA SCADA

CAPITULO III

DISEÑO DEL SOFTWARE DE AUTOMATIZACIÓN

CAPITULO IV

HARDWARE

CAPITULO V

PRUEBAS DE CAMPO Y RESULTADOS

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

BIBLIOGRAFIA

ANEXOS

INTRODUCCIÓN

En la actualidad se observa un continuo avance tecnológico - científico y gran competitividad de nuevas tecnologías por lo que, se hace posible y necesario, modernizar los sistemas para mejorar la confiabilidad y eficiencia en los procesos.

El material bélico mundial se desarrolla en forma impresionante, comprobándose que por medio de herramientas electrónicas se puede reducir el tiempo de operación, desperdicio de municiones y la pérdida del elemento humano.

El Centro de Investigaciones Científicas y Tecnológicas del Ejército (C.I.C.T.E.), dentro de su plan de prestación de servicios a la Fuerza Terrestre y su afán de aprovechar el desarrollo tecnológico, financia y desarrolla proyectos destinados a la modernización de equipos y sistemas militares, contribuyendo a la continua actualización del Ejército Ecuatoriano para ir acorde con los avances científicos y tecnológicos a nivel mundial.

Los proyectos desarrollados constan básicamente de la implementación de prototipos que integran material electrónico moderno en anteriores sistemas existentes, con la finalidad de optimizar, mejorar el funcionamiento y beneficio de dichos sistemas, equipos; además de reducir la intervención de personal para evitar errores y pérdidas irreparables.

El cañón autopropulsado de 155mm es un sistema de artillería mediana perteneciente al GAAP-11 Brigada Blindada Galápagos en Riobamba. Este cañón actualmente es posicionado mediante manivelas y la puntería se la realiza con ayuda de un Sistema de Puntería Digital, proyecto anteriormente realizado para los cañones autopropulsados de 155 mm con el objetivo de reemplazar los equipos tradicionales y mejorar la operatividad del sistema de disparo con la creación de un Centro Director de

Tiro (CDT) computarizado que calcule sin errores los datos de tiro; al ver la efectividad de dicho sistema se consideró importante que junto con la modernización de la puntería, es necesario y óptimo automatizar el posicionamiento del tubo del cañón, creando un sistema totalmente automático e independiente que aumente notablemente los beneficios en un disparo reduciendo al mínimo la posibilidad de errores.

Tomando en cuenta todos los enunciados anteriormente descritos y el Sistema de Puntería Digital (SPD) prototipo implementado en el cañón Autopropulsado de 155mm se propone el proyecto: "Diseño e Implementación de un Sistema SCADA para los cañones Autopropulsados de 155mm pertenecientes al GAAP – 11.

El sistema propuesto, busca automatizar totalmente el proceso actual utilizado para realizar un tiro, eliminando el cableado entre el cañón Autopropulsado de 155mm y el C.D.T. utilizado en el Sistema de Puntería digital ya implementado, mediante equipos que permitan una comunicación inalámbrica; instalando un controlador individual en cada pieza que se encargue del posicionamiento del tubo de cañón para independizar al sistema de control de un computador y adquiriendo características de un Sistema S. C. A. D. A (Supervisión, Control y Adquisición de Datos), asegurando que el prototipo a implementar sea una solución para la modernización de este tipo de material bélico.

PROYECTOS RELACIONADOS

El proyecto propuesto es la segunda fase del proyecto denominado: "Diseño e implementación de un Sistema de Puntería Digital y Factibilidad para la Automatización de los Cañones Autopropulsados de 155mm pertenecientes al GAAP-11.".

JUSTIFICACION

El C.I.C.T.E. dentro del plan anual de desarrollo de Proyectos, su afán de mejorar y optimizar el material bélico con el que cuentan el GAAP-11, busca la manera de modernizar los cañones Autopropulsados de 155mm por medio de dos proyectos de investigación.

El Sistema S.C.A.D.A. para los Cañones autopropulsados de 155 mm pertenecientes al GAAP-11 resulta necesario, ya que el Sistema actual con posicionamiento digital, aun depende del adiestramiento que tenga el personal para lograr un disparo efectivo.

La existencia de cables de comunicación, junto con la dependencia de un computador central para el control de toda la batería, y el posicionamiento manual del tubo del cañón; Entorpecen la operatividad del sistema global.

IMPORTANCIA

La implementación de un Sistema SCADA para los cañones autopropulsados de 155mm pertenecientes al GAAP-11 resulta de suma importancia, debido a que mediante dicho sistema se podrá obtener una respuesta exacta, rápida y eficiente del proceso de disparo.

Además se reduce el recurso humano y tiempo necesario en la preparación y puesta a punto de la batería lo que mejora el rendimiento.

OBJETIVOS

OBJETIVO GENERAL

- Diseñar e implementar un Sistema SCADA para cañones autopropulsados de 155mm.

OBJETIVOS ESPECÍFICOS

- Investigar y analizar las características y especificaciones técnicas del funcionamiento de instrumentos de monitoreo, control, comunicación inalámbrica, motores inteligentes y acoples.

- Adquirir los dispositivos necesarios para el Sistema SCADA de los cañones autopropulsados de 155mm

- Programar el controlador (PLC) de cada cañón autopropulsado de 155mm para que envié, reciba y controle el tubo de cañón en base a parámetros en inclinación, dirección y posicionamiento.

- Automatizar los movimientos del ángulo de elevación y azimut del tubo del cañón, por medio de motores.

- Integrar el sistema de puntería digital ya desarrollado con el sistema SCADA para los cañones autopropulsados.

- Crear un interfaz HMI global donde se visualice todos los parámetros, tanto de cálculos de tiro como el estado del sistema

- Comprobar la funcionalidad de batería mediante las pruebas de campo respectivas.

- Documentar el proyecto.

METODOLOGÍA Y EQUIPAMIENTO QUE SE PROPONE EMPLEAR

La metodología y equipamiento que se utilizará para el desarrollo de este proyecto, se detalla a continuación:

- Análisis y estudio del proceso de disparo, dirección de tiro y artillería básica mediante la lectura de manuales, consultas a expertos en el tema y visitas de campo.

- Investigación y selección de elementos necesarios para el sistema SCADA de los cañones; costos, características, especificaciones; por medio de la red de Internet.

- Desarrollo y creación de una interfaz humano maquina (HMI) entre el cañón y el operador.

- Integración del sistema de puntería digital con el nuevo sistema SCADA propuesto utilizando LabView 7.0 de National Instruments.

- Análisis comparativo de precisión y tiempo del sistema implementado respecto con el sistema actual mediante pruebas de campo.

- El equipamiento que se propone utilizar es el siguiente:
  Sistema de puntería digital, inclinómetro, brújula, GPS, PLC, Motores, radio MODEM y acoples mecánicos.

DESCRIPCION DEL TEMARIO

CAPITULO I

Funcionamiento del sistema de puntería digital del cañón autopropulsado de 155mm

En este capítulo se describe las características principales del Sistema de Puntería Digital implementado en el proyecto anterior, una breve explicación del funcionamiento del sistema y las características técnicas del hardware y software.

Observando los beneficios y los problemas que nos pueden ocasionar a nuestro sistema se buscó las opciones óptimas, para que el sistema anterior nos de las mayores prestaciones posibles para el desarrollo del proyecto.

CAPITULO II

Solución Propuesta y Descripción del Sistema SCADA

En esta sección se presenta los criterios que se utilizaron para dar una solución en la implementación del sistema SCADA además se presentan propuestas para mejorar el sistema de puntería digital del cañón autopropulsado de 155mm.

Una vez desarrollado lo anteriormente mencionado, se entrega la mejor solución encontrada para la automatización del posicionamiento y las ventajas que se obtuvo con el proyecto.

En este capitulo están definidas los parámetros y las variables principales a ser manipuladas en el software, al tener una idea clara de estos parámetros anteriormente mencionados, se desarrolla un descripción del diagrama del bloques del software seguido de la descripción del programa principal.

CAPITULO III

Diseño de software de Automatización

En este capitulo están definidas los parámetros y las variables principales a ser manipuladas en el software, al tener una idea clara de estos parámetros anteriormente mencionados, se desarrolla un descripción del diagrama del bloques del software seguido de la descripción del programa principal.
Con la ayuda del programa principal buscaremos la manera de realizar una interfase amigable y de fácil uso para el usuario y describiremos brevemente el software.

CAPITULO IV

Hardware

En esta sección daremos los diferentes criterios para la selección del equipo, como las especificaciones técnicas, el costo, etc.

Con lo mencionado anteriormente daremos una breve explicación de los criterios que se utilizaron para escoger el equipo y los beneficios que dan a nuestro sistema. En complementación adicional realizamos una definición breve del funcionamiento de los motores y la descripción del algoritmo de control y el motivo por el cual fue escogido.

CAPITULO V

Pruebas de campo y resultados

En este capítulo se realiza una descripción de la funcionalidad y la operatividad del sistema, y se puede hacer una comparación de resultados de precisión, exactitud, tiempo de retardo en el posicionamiento del tubo del cañón, y el tiempo de disparo con respecto al sistema manual.

CAPITULO VI

Conclusiones y recomendaciones

En este capítulo se muestra las conclusiones obtenidas durante la elaboración del proyecto, además se presenta las recomendaciones para elaboración de sistemas similares tanto en las Fuerza Armadas o para la elaboración de tesis similares.

CAPITULO I

FUNCIONAMIENTO DEL SISTEMA DE PUNTERÍA DIGITAL DEL CAÑÓN AUTOPROPULSADO DE 155mm

1.1 OPERACIÓN DEL SISTEMA ÓPTICO DIGITAL DE LOS CAÑONES DE 155MM.

El proyecto denominado "Diseño e Implementación de un Sistema SCADA para los cañones autopropulsados de 155 mm pertenecientes al GAAP - 11", utiliza y complementa el proyecto: "Diseño e Implementación de un Sistema de Puntería Digital y Factibilidad de Automatización para los Cañones Autopropulsados de 155mm pertenecientes al GAAP – 11" por lo que a continuación se presenta las características de operación mas relevantes de dicho proyecto.

1.1.1 Características

- El Sistema de Puntería Digital (SPD) es un prototipo desarrollado en el C.I.C.T.E., implementado en el cañón Autopropulsado de 155mm y fue diseñado para realizar cálculos de posicionamiento con un alcance máximo de 20 kilómetros.

- Reemplaza los equipos ópticos tradicionales pertenecientes a los cañones de 155 mm, por sensores digitales (brújula, inclinómetro y GPS), los cuales además de

brindar un sistema de orientación preciso, integran sus funciones con el CDT, para optimizar el procedimiento de disparo.

- Reemplaza los métodos manuales de cálculo del CDT, con un programa el cual se desarrollo con el objetivo de, calcular datos de tiro y realizar correcciones a los valores normalizados establecidos en las Tablas de Tiro (valores de las tablas están dados para condiciones ideales).

En este capitulo se presenta el hardware y software desarrollados para el Sistema de Puntería Digital:

1.2 HARDWARE

El Sistema de Puntería digital de los cañones de 155mm consta de los siguientes dispositivos electrónicos y equipos.

- Brújula
- Inclinómetro
- GPS
- Adaptador USB a Serial
- PC portátil

1.2.1 Brújula

En el proyecto del Sistema de Puntería Digital se utiliza a la brújula HMR3000 como elemento para posicionar geográficamente el tubo del cañón. Este dispositivo realiza su medición en base a sensores magnéticos los cuales permiten leer, la dirección exacta en la cual el encoger ubica la pieza.

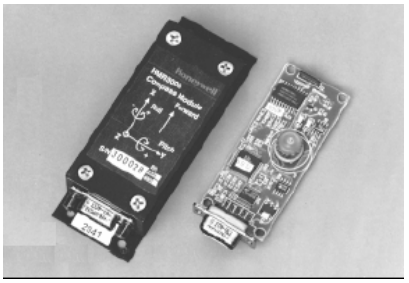


Figura 1.1 Brújula HMR3000

Las características más relevantes por la cual se usó esta brújula es que cumple las normas militares como encapsulado, precisión y lectura de los ángulos en milésimas.

1.2.1.1 Características

La brújula tiene un protocolo de comunicación serial RS – 232, está configurada para trabajar a una velocidad de transmisión de 9600 baudios con una resolución en milésimas de grado. Todos estos parámetros se los puede configurar mediante el software (PC INTERFACE), propio del dispositivo, el cual además permite calibrar y sintonizar la brújula HMR3000 para eliminar al máximo la interferencia electromagnética. Necesita una fuente de alimentación de 12VDC.

La cadena de caracteres que se transmite, está dada de la siguiente manera:

$TNTHPR,Heading,a,Pitch,a,Roll

De Heading se obtiene la dirección, y “Pitch”, “Roll”, son las inclinaciones en el eje x, eje y, respectivamente.

1.2.1.2 Montaje

Para la ubicación de la brújula se diseño un soporte de material aislante para evitar efectos de campos magnéticos producidos por la masa metálica de toda la pieza, acoplada en la base inferior del tubo del cañón tomando en cuenta el centro de giro y la dirección del tubo.

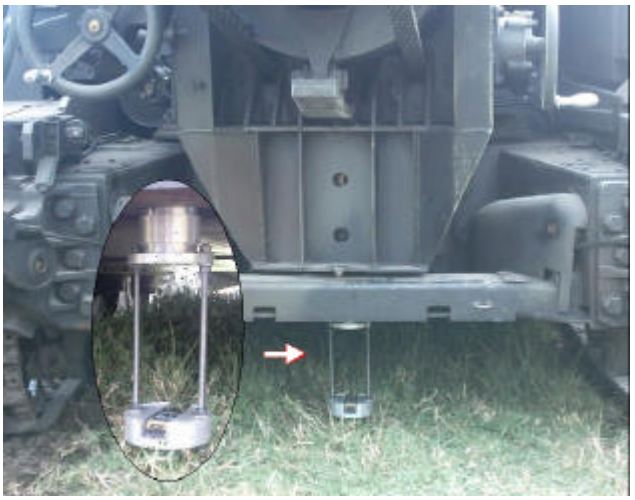


Figura 1.2 Ubicación de la brújula en su respectivo soporte

1.2.2 Inclinómetro

Para conocer el ángulo de elevación del tubo del cañón, se utiliza un sensor de inclinación modelo RDI Series de la casa Rieker, el cual fue seleccionado por tener la resolución y tiempo de respuesta adecuados, además de cumplir estándares militares ya que posee un encapsulado de protección de aluminio recubierto de plástico.

Figura 1.3 Inclinómetro RDI Series

1.2.2.1 Características

Este inclinómetro presenta visualmente en un display el grado de inclinación lateral y longitudinal con respecto a superficie horizontal con una precisión de < 0.01° en un rango de ±72° , y la transmite mediante un protocolo de comunicación serial RS-232 con una velocidad de 9600 baudios. Es necesario un voltaje de alimentación de 12 VDC.

1.2.2.2 Montaje

Para la ubicación y montaje del inclinómetro se tomó en cuenta la inclinación del tubo del cañón, por lo que se diseñó un soporte de aluminio; ubicado entre la burbuja indicadora de la elevación del tubo y el lente panorámico.

Figura 1. 4 Inclinómetro ubicado en el cañón de 155 mm.

1.2.3 G.P.S. (Sistema de Posicionamiento Global)

El GPS se lo utiliza para obtener las coordenadas exactas de la Pieza, ubicación de la Batería que ayudan a reducir errores causados por la localización, Posee un altímetro digital para la ubicación de la altura respecto al nivel del mar, es utilizado en aplicaciones militares por tener alta precisión entre equipos de localización.

Para el Sistema de Puntería Digital se adquirió el GPS marca GARMIN modelo eTrex Vista.

Figura. 1.5 Altímetro eTrex Vista y GPS

1.2.3.1 Características

El GPS de marca GARMIN se utiliza principalmente para conocer los datos de posición y altura con una precisión de < 15m. Transmite la información mediante un protocolo serial RS-232 con un Standard NMEA 0183. Tiene un tamaño parecido al Mouse de un computador, se alimenta con un voltaje de 3VDC (2 baterías AA).

1.2.4 Adaptador USB a Serial

El adaptador ESU100 de QUATECH permite obtener con un puerto USB, ocho puertos seriales independientes. Esta instalado en el Sistema de Puntería Digital ya que

todos los elementos anteriormente mencionados poseen comunicación serial, y en las computadoras portátiles este puerto ha sido reemplazado por el puerto USB, por lo que este adaptador resuelve el problema de la necesidad de varios puertos seriales (Protocolo Serial RS-232).

Figura. 1.6 Adaptador ESU100

1.2.4.1 Características

Como características principales se puede resaltar el bus de interfase compatible con USB 1.1 y USB 2.0, trabaja en un Sistema Operativo Windows 98/Me/NT/2000/XP, no necesita fuente de alimentación, tiene un protocolo de transmisión serial con una tasa de transmisión máximo de 480.6 Kbps.

1.2.5 PC Portátil

La computadora portátil es la que contiene el software desarrollado encargado de realizar: la lectura, monitoreo de los sensores, cálculos del CDT y correcciones para realizar el tiro, además de presentar una interfaz amigable al usuario para facilitar el procedimiento de disparo.

1.2.5.1 Características

Por ser un prototipo se utilizó únicamente un computador personal sin caracteristicas robustas de funcionamiento, especialmente por costos, pero se debe mencionar que la

computadora debe cumplir los requerimientos militares para un funcionamiento confiable del sistema, en caso de utilizarlo en disparos reales, debe por lo menos contar con una memoria RAM de 256 MB, ROM de 10 GB, una velocidad de 2.4 GHz y un sistema Operativo Windows 98/Me/2000/NT/XP.

1.2.6 Integración completa del Hardware SPD-155

A continuación se muestran en las figura el esquema completo de conexión del Hardware y el equipo completo utilizado en el Sistema de Puntería SPD – 155.

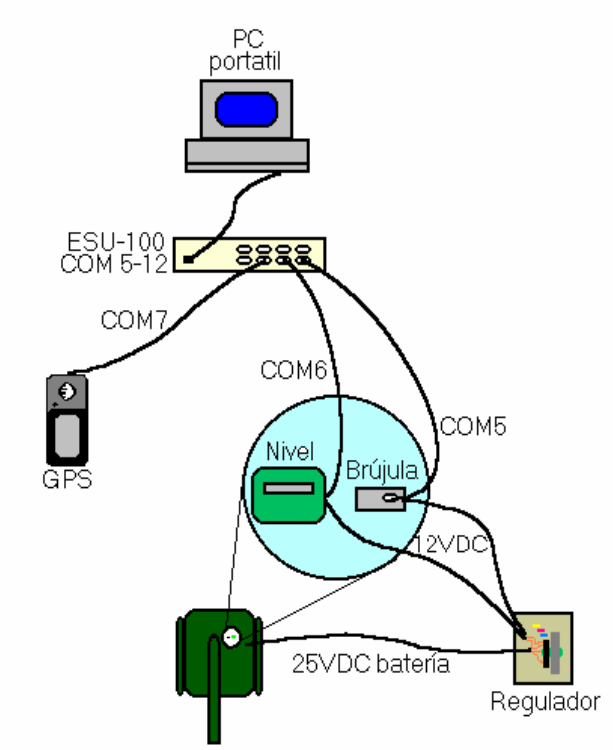


Figura 1.7 Esquema de Conexión de Hardware del SPD

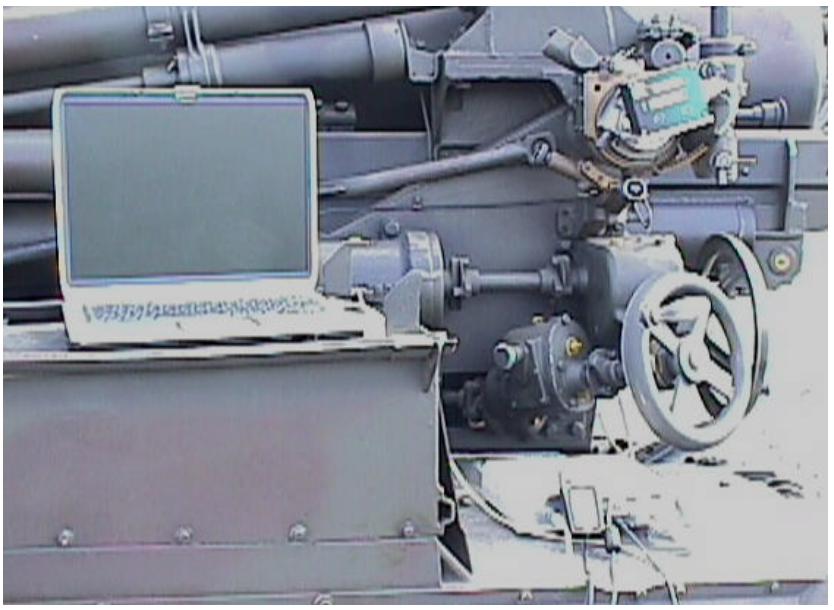


Figura 1.8 Equipo completo del SPD - 155

1.3 SOFTWARE

El Sistema de Puntería Digital se basa principalmente en un programa desarrollado en LabView Express 7.

1.3.1 Estructura.

El software se estructura en varios subprogramas que son ejecutados según un orden determinado. Existen subprogramas principales los que contienen subprogramas secundarios, a continuación se dará una breve descripción de los mismos.

1.3.1.1 Carátula.

Despliega una ventana de inicio al programa total, espera unos segundos antes de cerrarse y ejecutar la siguiente pantalla

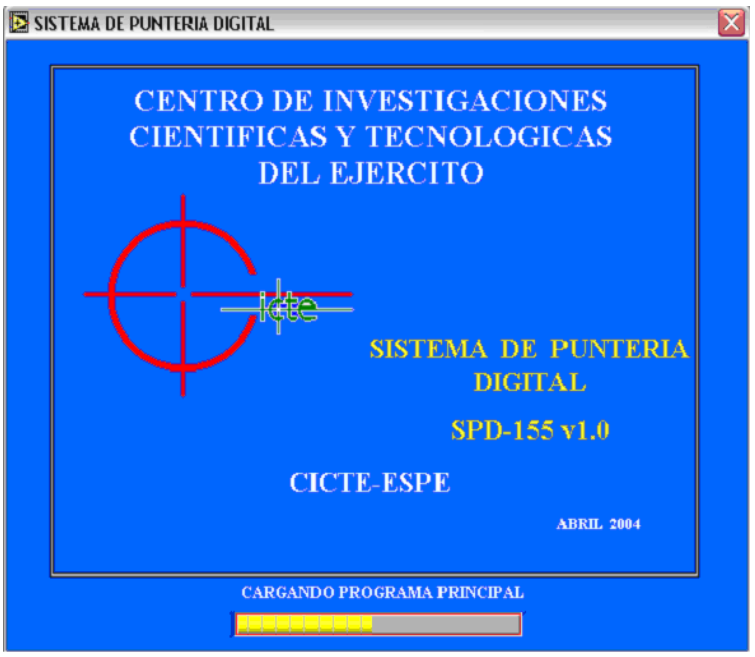


Figura 1.9. Ventana de Inicio

1.3.1.2 Lista de Cañones.

Al cerrarse la ventana de inicio se despliega una nueva pantalla con la lista de los cañones, donde se muestran que cañones están activos y listos para realizar el disparo. Una vez reconocidos los cañones se debe presionar aceptar.

Figura. 1.10. Pantalla de la Lista de Cañones

1.3.1.3 Principal HMI.

Se despliega esta pantalla que es en la que se desarrolla la mayoría del programa, tiene subpaneles en los cuales se muestra continuamente las lecturas del inclinómetro y la brújula de los cañones activos, además en un subpanel intermedio se ejecuta el CDT, tomando en cuenta los datos de las lecturas de los sensores (Inclinómetro, brújula y gps).

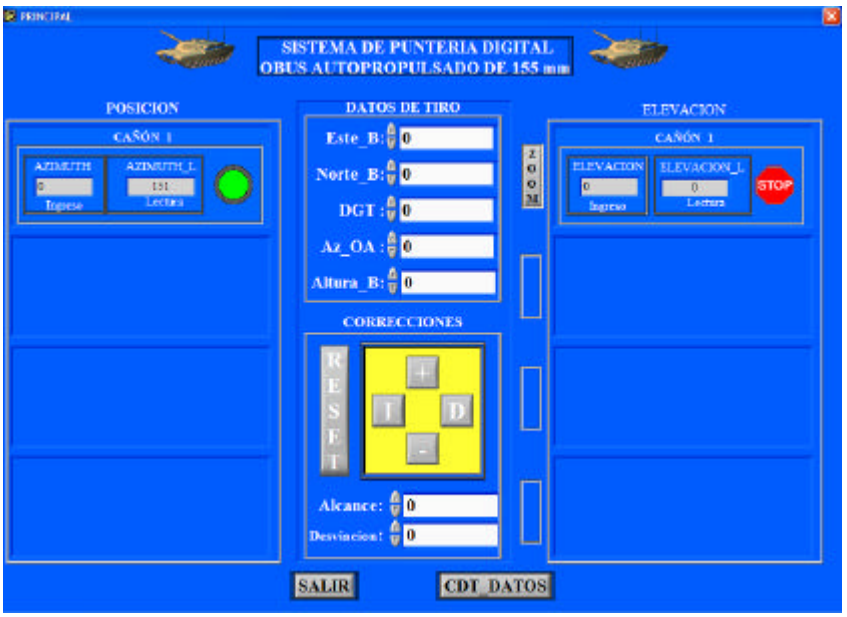


Figura. 1.11. Pantalla Principal HMI del Software SPD - 155

1.3.1.4 Subprogramas Secundarios

Se crearon varios subprogramas secundarios, para servir de ayuda en las pantallas de los subprogramas principales, entre ellos están: CDT, lectura de sensores y pantallas de ZOOM las que permitirán ampliar la visualización de los datos adquiridos por los sensores y los datos ingresados por el personal de artillería.

Figura. 1.12. Subprograma ZOOM

1.3.2 Características

El software del Sistema de Puntería Digital denominado SPD-155 es, indispensable al realizar un disparo.

Las principales funciones del programa son:

- Monitoreo continuo en forma automática de los sensores (Brújula, Inclinómetro, GPS)
- Procesamiento y cálculos del CDT, así como, las correcciones de tiro.
- Presentación de resultados precisos necesarios para realizar el tiro.
- Detección de errores en la lectura de sensores.
- Interfaz clara y de fácil manejo para los usuarios destinada a mejorar y optimizar la operación de disparo.

1.3.3 Evaluación y actualización del software SPD-155.

1.3.3.1 Evaluación del Software SPD-155

Para el desarrollo del presente proyecto "Diseño e Implementación de un Sistema SCADA para los cañones Autopropulsados de 155mm pertenecientes al GAAP-11" fue importante realizar pruebas de funcionamiento del software SPD-155, específicamente cálculos del CDT, con lo que se obtuvieron las siguientes observaciones:

- El Software de Puntería Digital SPD-155, realiza correctamente los cálculos completos de CDT, con la posibilidad de realizar correcciones de tiro, aunque únicamente para tiros con carga 9 bis.

- Para cálculos de CDT, es necesario que por lo menos estén activos y conectados los sensores de un cañón, presentando resultados solo para las

lecturas de los mismos, eliminando la posibilidad de poder realizar ejercicios de tiro con distintos datos de prueba.

- La pantalla de CDT se muestra en conjunto con las lecturas de los sensores (inclinómetro y brújula) para los 4 cañones que componen la batería.

- No se permiten realizar traslados de las coordenadas balísticas cuando se logra un tiro efectivo.

1.3.3.2 Actualización del Software SPD-155

El Software desarrollado para el Prototipo propuesto, monitorea y controla sensores y actuadotes instalados en la pieza, brindándole la posibilidad de realizar ejercicios de tiro sin la necesidad de que el sistema autom,ático esté conectado o funcionando, mostrando resultados de los cálculos a partir de datos ingresados por el operador; por lo que fue indispensable hacer una actualización al software SPD-155.

Se presenta a continuación los cambios realizados:

- Actualización y aumento de la base de datos de las Tablas de Tiro, incorporando los datos de las cargas 6, 7 y 8, brindando la posibilidad de que el operador pueda escoger la carga de disparo.

- Adaptación y mejora de la interfaz gráfica del CDT, para permitir ingresar datos para cálculos por el usuario o utilizar lecturas entregadas por los sensores, en una única pantalla.

- Brindar la posibilidad de trabajar con el CDT como parte independiente del monitoreo de sensores, reemplazando la calculadora utilizada como método tradicional de cálculo y permitiendo realizar ejercicios de tiro en cualquier lugar sin la necesidad de contar con el cañón o el equipo.

- Posibilidad de trasladar coordenadas balísticas cuando se realice un tiro efectivo.

A partir de los cambios realizados utilizando como base el Software del Sistema de Puntería Digital se diseñó la interfaz gráfica mostrada en la figura 1.13.

Figura 1.13. Pantalla actualizada del CDT.

Para dichas pruebas se calcularon datos del CDT mediante el método manual, en el que se utilizo la calculadora HP y las tablas de tiro; y los resultados se compararon con el SPD-155 actualizado. Las comparaciones demostraron que los cálculos matemáticos del CDT se realizan sin errores, entregando los resultados en un tiempo inferior al del método tradicional. Estas mejoras serán mas detalladas en el capitulo V del Pruebas y Resultados

Resultados obtenidos mediante el método tradicional de cálculo del CDT

Se resolvieron varios ejercicios de tiro, junto con el grupo de artillería encargado de los cálculos del CDT, para analizar la validez del software.

A continuación se presentan los resultados de un ejercicio de tiro a partir de datos reales, resuelto mediante el método tradicional de cálculo mediante la calculadora HP y las tablas de tiro.

Datos

COORDENADAS	**ESTE**		**NORTE**
Batería:	29500	-	18500
Observador:	21550	-	25850

Azimut: 4030 milésimas
DGT: 5390 milésimas
Distancia: 1950 alargar
CARGA: 7

Resultados

Alcance: 11132
Deflexión Pieza: 5370
Angulo de Elevación (- corrección altura): 378

1era. Corrección:

Derecha: 200 Acortar: 100

Resultados

Alcance: 11303
Deflexión Pieza: 5383
Angulo de Elevación (- corrección altura): 388

2da. Corrección:

Izquierda: 100 Acortar: 50

Resultados

Alcance:	11194
Deflexión Pieza:	5386
Angulo de Elevación (- corrección altura):	381

Resultados obtenidos mediante el SPD - 155

En el software se ingresaron los mismos datos del ejercicio anterior, para hacer una comparación de resultados, los que se presentan marcados a continuación.

1era. Corrección:

2da. Corrección:

Según los resultados obtenidos se pudo comprobar que el software funciona correctamente.

1.4 FUNCIONAMIENTO DEL SISTEMA

Una vez considerados los elementos necesarios para el Sistema de Puntería Digital se puede comprender el funcionamiento básico del sistema, el que se resume en lo siguiente

- Recopilación de los datos entregados por los sensores de posición y ubicación geográfica del cañón.

- Entrega de la información al cerebro principal (Computador Portátil) para realizar los cálculos del CDT y obtener los resultados necesarios para realizar el tiro.

- Posicionamiento manual del tubo del cañón, el que es verificado por el computador, por medio de las señales del inclinómetro (sensores utilizados para registrar la posición en azimut y deflexión) y la brújula.

CAPITULO II

SOLUCIÓN PROPUESTA Y DESCRIPCIÓN DEL SISTEMA SCADA

2.1. PRINCIPALES FACTORES DE OPTIMIZACIÓN

El Centro de Investigaciones Tecnológicas y Científicas del Ejército (C.I.C.T.E) en su afán de aprovechar el desarrollo tecnológico, se propuso realizar prototipos que pueden ser implementados en la artillería del Ejército Ecuatoriano, los que buscan principalmente la modernización, optimización, digitalización y alargamiento de vida útil del material bélico.

Se presenta a continuación los principales factores que se tomaron en cuenta para mejorar al Sistema Actual utilizado en el proceso de disparo en los cañones de 155mm y similares.

2.1.1. Cableado en el Sistema de Puntería Digital SPD – 155

Como se explico en el capitulo anterior el Sistema de Puntería Digital SPD – 155, consta de varios sensores, todos con un protocolo de comunicación serial, por lo que es necesario un concentrador de los mismos para ser conectados al computador, lo que causa dificultad por el extenso cableado, tanto de los sensores, el concentrador, como del computador; obligando además que el mismo este muy cerca del cañón durante el proceso el disparo.

Además en la realización de un disparo real, el cableado se convierte en un problema que puede demorar el tiempo de operación, ya que en un ambiente hostil es necesario tener la facilidad de preparar el equipo, realizar el disparo efectivo y movilizarse lo más pronto posible.

.

2.1.2. Posicionamiento del tubo de los Cañones Autopropulsados

El Posicionamiento de la pieza se lo realiza con ayuda de la brújula digital. Una vez que se ha entrado en posición con la respectiva DGT, es necesario esperar los datos del CDT para poder ubicar al tubo del cañón en, deflexión y elevación, respectivamente.

El Posicionamiento del tubo del Cañón es de tipo manual y se lo realiza con las manivelas que posee la pieza, lo que hace que este proceso sea lento y susceptible a errores, ya que depende del criterio personal y habilidad de los apuntadores.

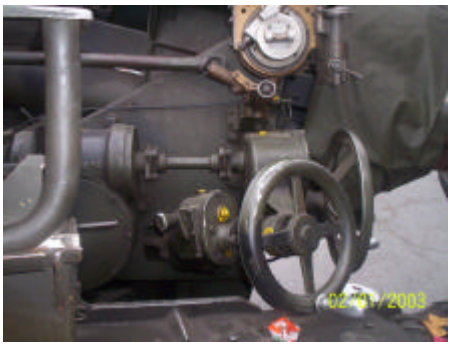


Figura. 2.1. Manivelas del tubo del cañón Autopropulsado de 155mm

2.1.2. Tiempo de respuesta

Tomando en cuenta que todo el sistema es manual, el tiempo necesario para la ejecución de un disparo depende de la experiencia del personal de artillería y el ajuste

correcto del equipo, lo que no garantiza que se pueda llegar al blanco en el menor tiempo posible.

Igualmente, el tiempo de respuesta se prolonga ya que es necesario tomar en cuenta el tiempo de cálculo del CDT, la transmisión y confirmación por radio de los datos, el posicionamiento manual del tubo de las piezas y la demora por cualquier problema que pudiera surgir como:

1. interferencia por ruido,
2. inexperiencia y
3. cansancio del equipo de artillería.

Los factores humanos y tecnológicos, antes mencionados, generan errores que pueden aumentar el tiempo de posicionamiento y corrección para un tiro efectivo, a más de producir un desperdicio de munición y un estado de inseguridad al alertar al enemigo y darle tiempo para que reaccione.

Todos los factores antes descritos han sido tomados en cuenta para plantear una propuesta técnico-científica que a futuro optimizará al sistema de artillería de los Autopropulsados de 155 mm.

2.2. PROPUESTA TECNICO-CIENTIFICA

Está propuesta, que es de tipo investigativo, busca dar mejoras al Sistema de Puntería Digital mediante la implementación de un Sistema Automático Inalámbrico de Posicionamiento para los Cañones Autopropulsado de 155mm y así dejar un legado que a

futuro permita modernizar el sistema de artillería, necesario para la defensa del país; al mismo tiempo que el Ejército contará con sistemas modernos acorde con los avances tecnológicos actuales. Por este motivo se propone desarrollar el proyecto ***"Diseño e Implementación de un Sistema SCADA para los cañones Autopropulsados de 155mm pertenecientes al GAAP – 11".***

2.2.1 Características de la Propuesta

Las características principales con las que cuenta el nuevo prototipo son las siguientes:

a) Automatiza el posicionamiento, del tubo del cañón instalando servomotores controlables que se guíen por las lecturas de los sensores instalados por el Sistema de Puntería Digital.

b) Elimina el cableado y concentrador de los sensores de la Pieza y del Computador colocados por el Sistema de Puntería Digital, utilizando un Controlador Lógico Programable (PLC), el que se encargará de recibir todas las señales y realizar el control de los motores, logrando una independencia del computador.

c) Monitorea todo el Proceso de disparo de la Batería, compuesta por cuatro piezas, con un computador que estará conectado al sistema a través de Radio Modems, un esclavo para cada pieza y un maestro para toda la batería, además que el computador realizará las funciones de CDT y enviara los datos al PLC eliminando la necesidad de comunicación por radio para confirmación de datos con el personal de artillería.

2.2.2 OBJETIVOS.

- Reducir el tiempo de posicionamiento del tubo del cañón y del proceso de disparo

- Tener un conocimiento exacto de la posición y ubicación geográfica de cada cañón.

- Prolongar la vida útil de la artillería mediana con la que cuenta actualmente la Fuerza Terrestre Ecuatoriana.

- Eliminar errores humanos.

- Disminuir estado de peligro frente al enemigo para el personal de artillería y el equipo en combate.

- Aumentar la efectividad de disparo

2.2.3. ESQUEMA.

En la figura 2.2 se muestra un esquema de los componentes principales con que cuenta el Sistema Propuesto denominado: Sistema SCADA para los Cañones Autopropulsados de 155mm y/o similares.

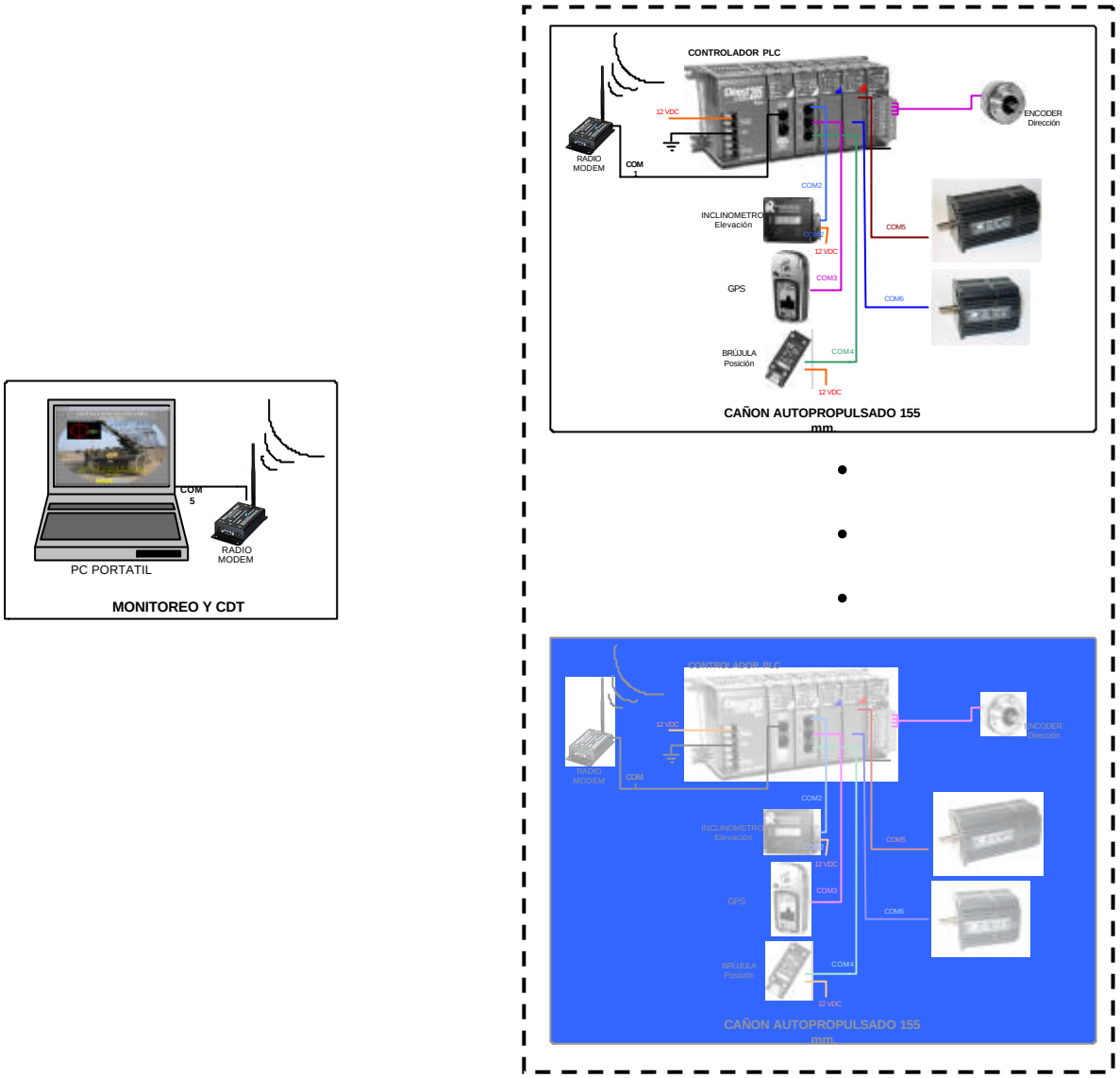


Figura 2.2. Esquema de Conexión de Hardware del Sistema SCADA

2.3.CRITERIOS DE AUTOMATIZACION

Al momento de buscar propuestas de solución al Sistema Actual fue necesario encontrar una alternativa que vaya acorde con los criterios de Automatización, Control y las necesidades requeridas por el Ejército Ecuatoriano para modernizar los Cañones de 155mm.

Se denominó al Proyecto Sistema de Supervisión, Control y Adquisición de Datos SCADA (Supervisory Control And Data Acquisition), considerando los siguientes enunciados.

2.3.1. SISTEMA S.C.A.D.A. (SUPERVISORY CONTROL AND DATA ACQUISITION)

Un SCADA es un sistema basado en una aplicación de software para trabajar sobre computadores que permite supervisar y controlar a distancia una instalación de cualquier tipo por parte de un operador. A diferencia de los Sistemas de Control Distribuido, el lazo de control es generalmente cerrado por el operador. Los Sistemas de Control Distribuido se caracterizan por realizar las acciones de control en forma automática.

En la tabla 2.1 se muestra un cuadro comparativo de las principales características de los sistemas SCADA y los sistemas de Control Distribuido (DCS)

ASPECTO	SCADA	DCS
TIPO DE ARQUITECTURA	CENTRALIZADA	DISTRIBUÍDA
TIPO DE CONTROL PREDOMINANTE	SUPERVISORIO: Lazos de control cerrados por el operador. Adicionalmente: control secuencial y regulatorio.	REGULATORIO: Lazos de control cerrados automáticamente por el sistema. Adicionalmente: control secuencial algoritmos avanzados, etc.
TIPOS DE VARIABLES	DESACOPLADAS	ACOPLADAS
ÁREA DE ACCIÓN	Áreas geográficamente distribuidas.	Área de la planta.
UNIDADES DE ADQUISICIÓN DE DATOS Y CONTROL	Remotas, PLCs.	Controladores de lazo, PLCs.
MEDIOS DE COMUNICACIÓN	Radio, satélite, líneas telefónicas, conexión directa, LAN, WAN.	Redes de área local, conexión directa.
BASE DE DATOS	CENTRALIZADA	DISTRIBUÍDA

Tabla 2.1. Diferencias entre sistemas SCADA y DCS.

2.3.1.1.SISTEMA DE ADQUISICIÓN DE DATOS

Un sistema de adquisición de datos es un equipo que nos permite tomar señales físicas y transformarlas en datos que posteriormente podrán ser procesados o presentados.

En algunos casos el sistema de adquisición es un componente de un sistema de control, y por tanto la información recibida se procesa para obtener una serie de señales de control.

Estructura de un sistema de adquisición de datos se muestra en la figura 2.3. En este diagrama podemos observar los bloques que componen un sistema de adquisición de datos:

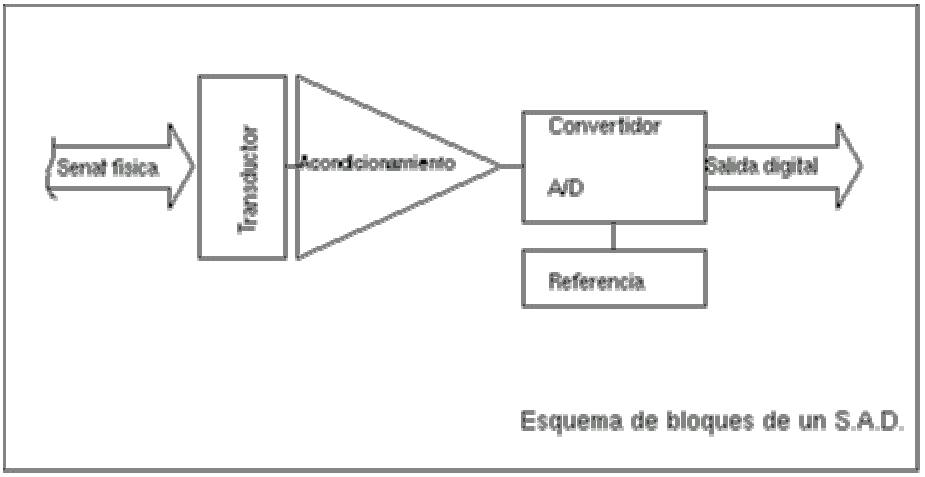


Figura 2.3 Estructura de un Sistema de Adquisición de Datos

Los bloques principales de un sistema de adquisición de datos son: El transductor, el acondicionamiento de señal, el convertidor analógico-digital, la etapa de salida.

El transductor es un elemento que convierte una magnitud física en una señal de salida (normalmente tensión o corriente) que puede ser procesada. El transductor debe tomar poca energía del sistema bajo observación, para no alterar la medida.

El acondicionamiento de señal es la encargada de realizar el filtraje y la adaptación de la señal proveniente del transductor a la entrada del convertidor A/D. Esta adaptación se encarga de:

- Adaptar el rango de salida del transductor al rango de entrada del convertidor. Que tiene como objetivo el aprovechar el margen dinámico del convertidor.

- Acoplar la impedancia de salida de uno con la impedancia de entrada del otro.

- El convertidor Analógico / Digital es un sistema que presenta en su salida una señal digital a partir de una señal analógica de entrada realizando las funciones de cuantificación y codificación.

La etapa de salida es el conjunto de elementos que permiten conectar el sistema de adquisición de datos con el resto del equipo, y puede ser desde una serie de buffers digitales incluidos en el circuito convertidor, hasta un interfaz serial RS-232, RS, para conectar a un ordenador o estación de trabajo, en el caso de sistemas de adquisición de datos comerciales, en nuestro proyecto los sensores con estas características son: Brújula digital, inclinómetro, GPS y encoder.

2.3.1.2. Necesidad de un sistema SCADA.

Para analizar si un sistema SCADA es necesario para manejar una instalación dada, el proceso a controlar debe cumplir las siguientes características:

Normas:

a) El número de variables del proceso que se necesita monitorear es alto.

b) El proceso está geográficamente distribuido. Esta condición no es limitativa, ya que puede instalarse un SCADA para la supervisión y control de un proceso concentrado en una localidad.

c) Las información del proceso se necesita en el momento en que los cambios se producen en el mismo, o en otras palabras, la información se requiere en tiempo real

.

d) La necesidad de optimizar y facilitar las operaciones de la planta, así como la toma de decisiones, tanto gerenciales como operativas.

e) Los beneficios obtenidos en el proceso justifican la inversión en un sistema SCADA. Estos beneficios pueden reflejarse como aumento de la efectividad de la producción, de los niveles de seguridad, etc.

f) La complejidad y velocidad del proceso permiten que la mayoría de las acciones de control sean iniciadas por un operador. En caso contrario, se requerirá de un Sistema de Control Automático, el cual lo puede constituir un Sistema de Control Distribuido, PLC's, Controladores a Lazo Cerrado o una combinación de ellos.

Funciones:

Dentro de las funciones básicas realizadas por un sistema SCADA están las siguientes:

a) Recabar, almacenar y mostrar información, en forma continua y confiable, correspondiente a la señalización de campo: estados de dispositivos, mediciones, alarmas, etc.

b) Ejecutar acciones de control iniciadas por el operador, tales como: abrir o cerrar válvulas, arrancar o parar bombas, etc.

c) Alertar al operador de cambios detectados en la planta, tanto aquellos que no se consideren normales (alarmas) como cambios que se produzcan en la operación diaria de la planta (eventos). Estos cambios son almacenados en el sistema para su posterior análisis.

d) Aplicaciones en general, basadas en la información obtenida por el sistema, tales como: reportes, gráficos de tendencia, historia de variables, cálculos, predicciones, detección de fugas, etc.

2.3.1.2. JUSTIFICACIÓN DE UN SISTEMA SCADA EN LA SOLUCIÓN PROPUESTA

a) El prototipo esta diseñado para controlar una batería que esta constituida por cuatro cañones autopropulsados de 155mm, cada tanque tiene que monitorear 13 señales, lo que implica que nuestro sistema tiene que realizar una lectura constante de 52 datos en tiempo real en ambientes hostiles, dichas lecturas deben tener una alta

confiabilidad para evitar cualquier error que cause perdidas materiales y/o humanas.

b) Todas las señales van destinadas a un controlador lógico programable por cada pieza de la batería, toda esa información debe ser monitoreada por un solo operador principal que estará ubicado en un computador a por lo menos 100 m. de distancia mediante una interfaz clara y de fácil uso para el personal militar de artillería

c) Por ser un material usado para conflictos bélicos, en donde el tiempo de posicionamiento es un factor predominante para el accionamiento de estrategias de batalla; se busca realizar una reducción del mismo, esto implica una automatización completa del proceso existente de disparo.

d) El prototipo obtendrá lo siguientes beneficios: Reducir el tiempo de disparo de la batería de cañones, reducir los costos de operación, aumentar confiabilidad y eficiencia, y alargar la vida útil para la artillería mediana.

En igual forma se va a describir las funciones del prototipo:

a) El Sistema SCADA para el autopropulsado de 155mm, recaba, almacena y muestra las señales emitidas por los sensores de ubicación y posición. (Inclinómetro, Brújula, encode y GPS), además de otras señales discretas para procesarlas en el PLC e indicarles en un HMI en el computador central.

b) En el prototipo el PLC realiza el control de servomotores, así como el procesamiento de las señales de los sensores para su uso en el CDT.

c) El Sistema SCADA tiene cuatro sensores fines de carrera para el tubo de cada cañón, encargados de dar avisos de alerta de que se ha alcanzado la deflexión y elevación máxima.

d) El prototipo tiene una interfaz gráfica que presenta: CDT, estado y señal de los sensores, coordenadas balísticas, tipo de munición y posibilita al operador a accionar o detener el posicionamiento dado por los motores.

Una vez revisadas las normas y funciones de un Sistema SCADA y analizando las características y necesidades del Proyecto se puede asegurar que el prototipo propuesto se lo puede denominar "Sistema SCADA para los cañones Autopropulsados de 155mm".

2.4. ANÁLISIS COMPARATIVO DEL SISTEMA ACTUAL CON EL SISTEMA PROPUESTO.

El Prototipo que se propone en este proyecto, optimiza las características que posee el Sistema de Puntería Digital anteriormente desarrollado, además que automatiza el proceso.

No será necesario que el operador del CDT se encuentre cerca de la Pieza para poder controlar y monitorear el desarrollo del tiro, logrando disminuir cualquier tipo de error causado por fallas humanas o interferencias por ruido, etc.

Es importante recalcar que el Sistema Propuesto deja abierta la posibilidad de trabajar en el tiro de manera manual y tradicional, hasta que el personal se adapte a los nuevos métodos y en el caso que pudiera aparecer cualquier imprevisto.

Se puede demostrar que el desarrollo de un proceso siempre que este sea manual, dará opción a demoras y pérdidas lo que se trata de eliminar en el proyecto propuesto a la vez que se aprovecha el desarrollo tecnológico actual.

CAPITULO III

DISEÑO DEL SOFTWARE DE AUTOMATIZACIÓN

3.1. ESTRUCTURA DEL SOFTWARE

El Software utilizado en el desarrollo del proyecto denominado: “Diseño e Implementación de un Sistema SCADA para los cañones autopropulsados de 155mm” consta de dos partes indispensable: el software de cálculo y monitoreo y el software de control.

En este capítulo se explica el protocolo de comunicación para el intercambio de información entre el Controlador y el computador.

3.2. SOFTWARE DE CONTROL

El Controlador WinPLC DL205 trabaja conjuntamente con el paquete de programación “Think & Do Live from Entivity” adquirido para el proyecto, el mismo que cuenta con varias herramientas que facilitan la creación de programas para el control del equipo utilizado. Entre las partes mas sobresaliente es la forma de programación mediante diagramas que flujo, **FlowView**, que permite realizar todo tipo de aplicaciones de manera fácil y eficiente, ya que presenta la posibilidad de organizar dichos diagramas en varias rutinas y subrutinas.

Este software de programación cuenta con módulos especiales de comunicación, control Proporcional, Integral y Derivativo (PID), cálculos matemáticos, manipulación y distribución de datos como arreglos, matrices y cadenas de caracteres.

Para la realización del programa fue importante utilizar herramientas de investigación y adiestramiento en el software adquirido.

Figura. 3.1. Software Think & Do Live

3.2.1 Algoritmo de Control

Existen varias técnicas de control las cuales deben ser seleccionadas cuidadosamente dependiendo del proceso a ser controlado. El diseñador debe buscar la forma más sencilla que le preste mejores rangos de confiabilidad y eficiencia, ya que no siempre el control más complejo proporciona las mejores condiciones de trabajo.

3.2.1.1. Requerimientos de control del proceso.

Para definir un algoritmo de control fue necesario analizar los requerimientos principales del proceso de disparo, los mismos que se reducen a los siguientes:

- Se debe posicionar el tubo del cañón en dirección y elevación con un error máximo de ± 1 milésima.

- Los tiempos de posicionamiento deben ser menores a los conseguidos en el método manual.

- Los valores de referencia para el control serán enviados inalámbricamente desde un computador maestro, ubicado a una distancia no menor a los 200m.

- El sistema debe estar preparado para realizar el control de posición de manera independiente al computador y a la mano del personal militar.

- El posicionamiento del tubo del cañón, en elevación y dirección se realizará con dos servomotores como elementos finales de control, los mismos que pueden ser controlados según variables de velocidad, aceleración y torque..

Según los requerimientos descritos y el equipo con el que se cuenta es importante definir el Sistema de Control y las variables a utilizarse.

3.2.1.2. Sistemas De Control

Un Sistema de Control consiste en un controlador automático, un actuador, una planta y un elemento de medición, su principal función es la de administrar la respuesta de una planta, sin la intervención directa del operador sobre sus elementos de salida.

Existen dos tipos de sistemas de control:

- Sistemas de control de Lazo Abierto
- Sistema de control de Lazo Cerrado

Los sistemas de control de lazo abierto son aquellos que no reciben ningún tipo de información del comportamiento de la planta.

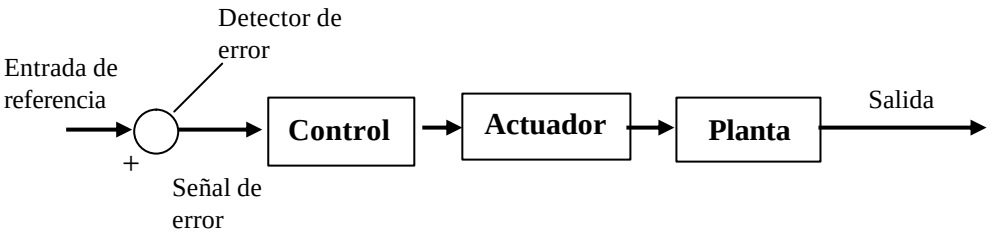


Figura. 3.2. Diagrama de bloques de un sistema de control de lazo abierto

Los sistemas de control de lazo cerrado tiene la característica de tener una señal de retroalimentación enviada de la salida del proceso de control para reducir la señal de error.

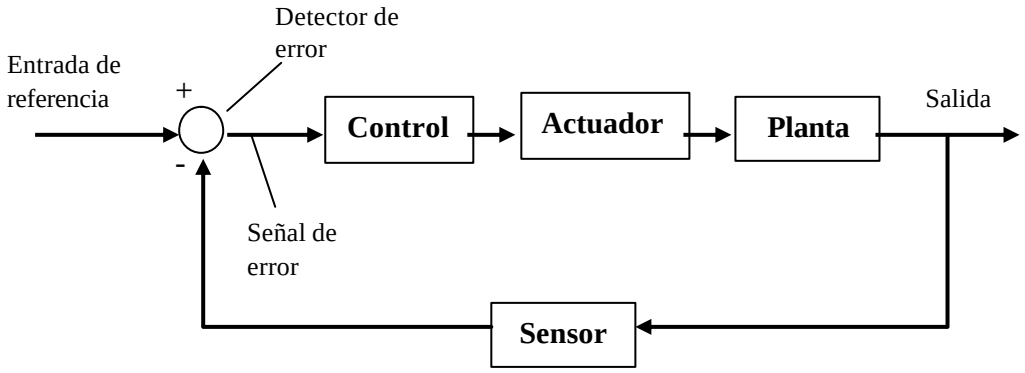


Figura. 3.3. Diagrama de bloques de un sistema de control de lazo cerrado

Dependiendo de los procesos a ser controlados, los sistemas de control pueden ser analizados como:

- Sistemas Analógicos
- Sistemas Digitales
- Sistemas Híbridos (analógico-digitales)

Los sistemas analógicos son los que trabajan con señales continuas, con un determinado rango de variación, las cuales representan señales físicas mediante tensión o corriente proporcionales a su valor.

Los sistemas digitales tienen la característica de trabajar con sistemas binarios, estos niveles suelen representarse por variable lógicas o bits, cuyo valor puede ser de 1 ó 0.

Los sistemas híbridos son usados en sistemas que tenga un mayor grado de complejidad, ya que en muchos casos la utilización de sensores analógicos es imprescindible.

Para el prototipo realizado en este proyecto se define al Sistema de Control con las siguientes características:

- **Sistema de Control en Lazo Cerrado.**
- **Sistema Digital.**

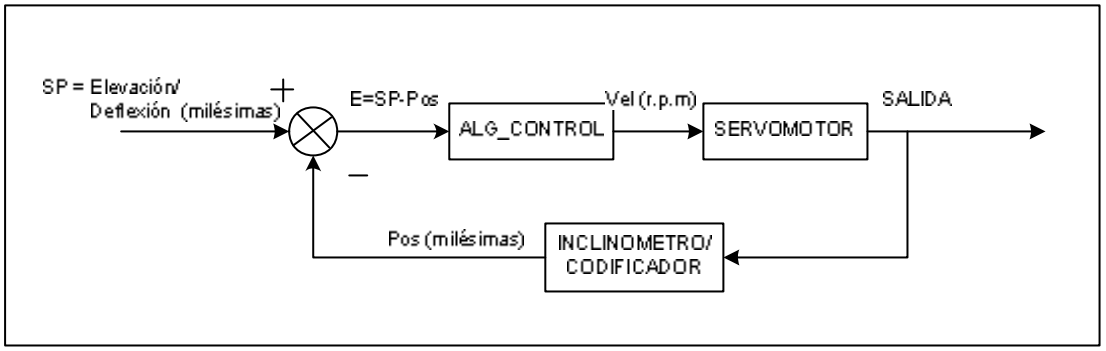


Figura. 3.4. Sistema de Control para el Posicionamiento

3.2.1.3. Parámetros y Variables.

Al definir los parámetros y variables que se toman en cuenta en el desarrollo del Software de Control, debemos analizar que parámetros y variables son medidos en el proceso, teniendo presente los términos que se usan para el control automático de procesos, los cuales son los siguientes:

Variable Controlada.- Es la variable que se debe controlar dentro de un valor requerido.

Variable Manipulada.- Es la variable que se utiliza para mantener a la variable controlada en el punto de control o referencia.

Punto de referencia.- Este valor es el que se desea alcanzar por la variable controlada, llamada también SET POINT.

Perturbación.- Variables que ocasiones desviación del punto de control

Este Software principalmente se encarga del control de posición del tubo del cañón a partir de datos calculados por el CDT y entregados por los sensores. Es necesario recalcar que el posicionamiento debe realizarse en el mínimo tiempo y con la mayor presición posible, de acuerdo al rendimiento de los sensores y actuadotes.

Se puede definir entonces como parámetros y variables a tomar en cuenta a los siguientes:

- Variables controladas: Valor medido por los sensores de posición: codificador incremental (Número de pulsos) e inclinómetro digital (grados)..

- Variable manipulada: Velocidad del motor dada en revoluciones por minuto (R.P.M.).

- Punto de referencia: Valor transmitido por el CDT (Controlador principal)

- Perturbaciones: Desgaste de los engranes del tubo del cañón y acoples de los servomotores.

3.2.1.4 Funcionamiento de los Servomotores

Una vez analizadas las características y requerimientos del proceso, fue necesario adaptar dichas especificaciones al modo de operación de los servomotores como elementos finales de control, por lo que se presenta a continuación las principales consideraciones tomadas en cuenta respecto al actuador del proceso.

Los servomotores SM3420D y SM3450D poseen varias herramientas de funcionamiento gracias al controlador incorporado que nos permite trabajar en varios modos de operación según la necesidad de la aplicación.

Dichos modos son principalmente tres:

Modo Velocidad MV: Según este modo se tiene un constante control en la velocidad, dependiendo de valores de aceleración y velocidad máximos. El Torque depende de las curvas del Servomotor.

Modo Posición MP: Alcanza una posición dada a una velocidad constante.

Modo Torque MT: Trabaja según el valor de Torque y velocidad dadas.

Las curvas de comportamiento Torque vs. Velocidad mostradas en las figuras 4.14 y 4.15, muestran que el torque se mantiene estable y relativamente constante hasta alcanzar una velocidad de 960 r.p.m, a partir de este valor se empieza a perder torque significativamente a cambio de ganar velocidad, lo que nos dio un rango máximo de trabajo de velocidad. Por especificaciones propias del motor, la velocidad mínima del mismo es de 60 r.p.m.

Por tanto, según lo enunciado en el párrafo anterior, la importancia de no desgastar al motor y tener un sobredimensionamiento en el torque para corregir cualquier pérdida por acoples mecánicos, el rango de velocidad a utilizarse es de:

Servomotor 3420D y 3450D utilizado en el posicionamiento en elevación:

Velocidad: 60 r.p.m. hasta 480 r.p.m.

Aceleración cte: 720 r.p.m^2.

Los valores máximos de velocidad y aceleración han sido dimensionados según resultados de pruebas realizadas en el cañón con los servomotores en lazo abierto.

Cálculo de errores: Para garantizar un error máximo de posicionamiento de los motores de ± 1 milésima, es necesario analizar los principales parámetros tanto de sensores como de acoples mecánicos de motores.

- **Inclinómetro,** trabaja con una resolución de ± 0.01 º ,repetibilidad menor a 0.05 º y su tiempo de respuesta es menor 0.5 segundos y su equivalente es:

Resolución	0.177 milésimas.
Repetibilidad	< 0.885 milésimas
Tiempo de Respuesta	< 0.5 seg

- **Codificador Incremental,** Tiene una resolución de 2500 pulsos por vuelta, según el acople del mismo, se cuenta un promedio de 14391 pulsos con ± 4 pulsos de error, para el recorrido total en dirección (800 milésimas), por lo que, se entrega una resolución equivalente a:

Resolución	0.056 milésimas.
Repetibilidad	< 0.5 milésimas
Tiempo de Respuesta	10 µseg

  Se debe aclarar que la Repetibilidad del encoger se lo puede reducir si los acoples mecánicos son perfeccionados.

- **Acoples del Volante en dirección,** La relación entre los acoples del volante en dirección y el eje del motor es de 146/23, equivalente a 6,384. Cada vuelta del volante en dirección, de acuerdo a los manuales técnicos del tanque, es proporcional a 4 milésimas de movimiento. Por lo tanto 1.587 vueltas del motor representan una milésima de movimiento.

- **Acoples del Volante en elevación,** La relación entre los acoples del volante en elevación y el eje del motor es de 144/23, equivalente a 6,261. Cada vuelta del

volante es proporcional a 6 milésimas de movimiento. Por lo tanto 1.043 vueltas del motor representan una milésima de movimiento.

Según los datos presentados se puede garantizar que el error en el posicionamiento no será mayor a 1 milésima, es importante tomar en cuenta esta condición, ya que 1 milésima en un tiro de 20 Km. equivale a 20 m de error.

3.2.1.5. Algoritmo Realizado

Una vez analizados las necesidades en el prototipo del Sistema de Posicionamiento de los tubos del Cañón, las condicionantes de operación de los servomotores, y los sensores se tiene que para dar solución al Problema automático de Posicionamiento de los motores se realiza lo siguiente:

Control forma trapezoidal: Se busca obtener una curva de respuesta de los servomotores Velocidad vs. Posición en forma trapezoidal, en la que la velocidad del motor máxima dependa de la posición que necesite alcanzarse, y el tiempo de aceleración y desaceleración sean suaves para evitar desgastes en los servomotores.

Para realizar cálculos de velocidad máxima, tiempos de aceleración y desaceleración según la posición; se utilizaron ecuaciones de Movimiento Rectilíneo Uniformemente Variado (M.R.U.V.)

$$V_F = V_O \pm a \cdot t \qquad \text{Ec. 3.1.}$$

$$X = V_O \cdot t \pm \frac{1}{2} a \cdot t^2 \qquad \text{Ec. 3.2.}$$

$$V_F^2 = V_O^2 \pm 2 \cdot a \cdot X \qquad \text{Ec. 3.3.}$$

Para la aceleración de los servomotores se tomó en consideración el tiempo que toma el motor en alcanzar la velocidad máxima, con lo que, según los resultados obtenidos para optimizar tiempos se trabaja con un valor constante de aceleración de 2 r.p.m^2.

El tiempo de desaceleración será el mismo de aceleración, por lo cual se calcula el desplazamiento, X_F, del servomotor en ese intervalo, cuando al motor le falte un 60% del desplazamiento calculado, X_F, se inicia el cambio de velocidad, dejando un 20% de desplazamiento, X_F, más allá en caso cualquier perturbación que demore al motor.

Este tipo de servomotores brinda la característica de dar una parada inmediata a los mismos, esta facilidad se utiliza cuando el error de posición es menor a 0.5 milésimas. En la figura 3.5. se muestra el comportamiento deseado del algoritmo.

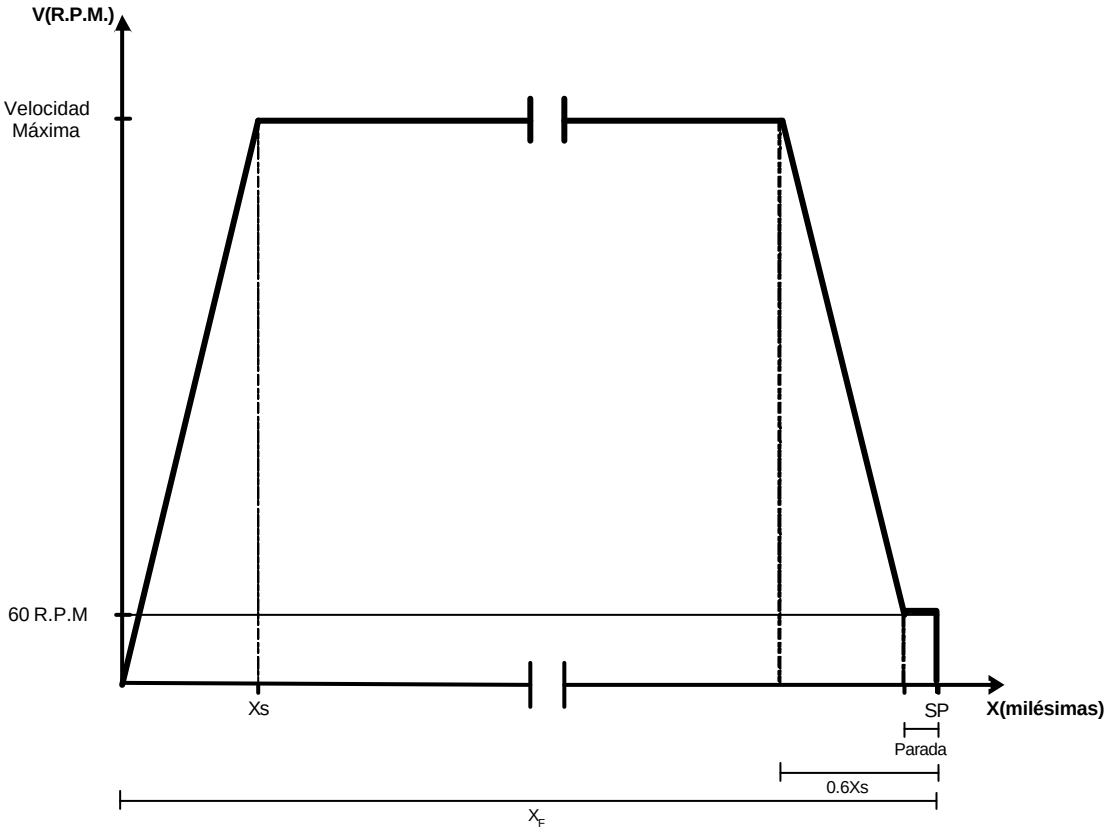


Figura. 3.5 Curva de respuesta Velocidad vs. Posición del Algoritmo de control

Cálculos realizados: Se presenta un ejemplo de los cálculos realizados para una Velocidad máxima de 60 r.p.m. en el motor de dirección.

Tiempo de Subida (Ts)

$$60r.p.m \approx 1r.p.s$$
$$V_O = 0$$
$$a = 2rps$$

$$V_F = V_O \pm a \cdot Ts \Rightarrow Ts = \frac{V_F}{a}$$
$$Ts = \frac{1}{2} = 0.5seg.$$

Intervalo de desplazamiento Xs en el tiempo de Subida

$$V_O = 0$$
$$a = 2rps$$

$$Xs = V_O \cdot t \pm \frac{1}{2} a \cdot Ts^2 \Rightarrow Xs = \frac{a \cdot Ts^2}{2}$$
$$Xs = \frac{1 \cdot 0.5^2}{2} = 0.25rev.$$

Si en dirección 1 milésima equivale a 1.587 revoluciones del motor entonces,

0.25 revoluciones equivale 0.156 milésimas

Si consideramos que Xs= X_F el 80 % de 0.156 milésimas es 0.13 milésimas, por lo tanto el motor será configurado a una velocidad de 60 r.p.m.

		Ts	Xs	mil=h/1.587	80% Xs	Redon(80%)
rpm	a / rps	2	2	2	2	2
60	1	0,5	0,25	0,156	0,13	0
120	2	1	1	0,625	0,50	0
180	3	1,5	2,25	1,406	1,13	1
240	4	2	4	2,500	2,00	2
300	5	2,5	6,25	3,906	3,13	3
360	6	3	9	5,625	4,50	4
420	7	3,5	12,25	7,656	6,13	6
480	8	4	16	10,000	8,00	8
540	9	4,5	20.25	12.76	10,2	10
600	10	5	25	15.75	12.6	13

Tabla. 3.1 Cálculos para el algoritmo de control del motor en deflexión

Se debe recordar que la relación de acoples es distinta en ambos servomotores, en elevación una vuelta en el volante equivale a 6 milésimas, por tanto 1 milésima corresponde a 1.04 vueltas del servomotor de elevación.

		Ts	Xs	mil=h/1.04	80% Xs	Redon(80%)
rpm	a / rps	2	2	2	2	2
60	1	0,5	0,25	0,240	0,19	0
120	2	1	1	0,962	0,77	0
180	3	1,5	2,25	2,163	1,73	1
240	4	2	4	3,846	3,08	3
300	5	2,5	6,25	6,010	4,81	4
360	6	3	9	8,654	6,92	6
420	7	3,5	12,25	11,779	9,42	9
480	8	4	16	15,385	12,31	12

Tabla. 3.2 Cálculos para el algoritmo de control del motor en elevación

Según resultados de cálculos de tiempo de posicionamiento realizado en tablas se establecieron rangos de velocidad máxima según la cantidad de desplazamiento, además se presenta el número de milésimas mínimo para cambiar la velocidad del servomotor, dichas tablas se presentan en el Anexo 2 y los rangos se presentan a continuación

Desplazamiento (Milésimas)	**Velocidad Máxima (R. P. M)**	**Punto de Cambio de Velocidad (Milésimas)**
1 – 6	120	1
7 – 11	240	3
12 – 16	300	5
17 – 25	360	6
26 – 32	420	8
33 – 800	480	12

Tabla. 3.3 Velocidad Máxima según rangos de desplazamiento en el Motor de Dirección

Desplazamiento (Milésimas)	**Velocidad Máxima (R. P. M)**	**Inicio de Desaceleración (Milésimas)**
1 – 6	120	1
7 – 10	240	2
11 – 15	300	3
16 – 24	360	4
25 – 31	420	6
33 – 1000	480	8

Tabla. 3.4 Velocidad Máxima según rangos de desplazamiento en el Motor De Elevación

3.2.2. Diagrama De Flujo del Software de control.

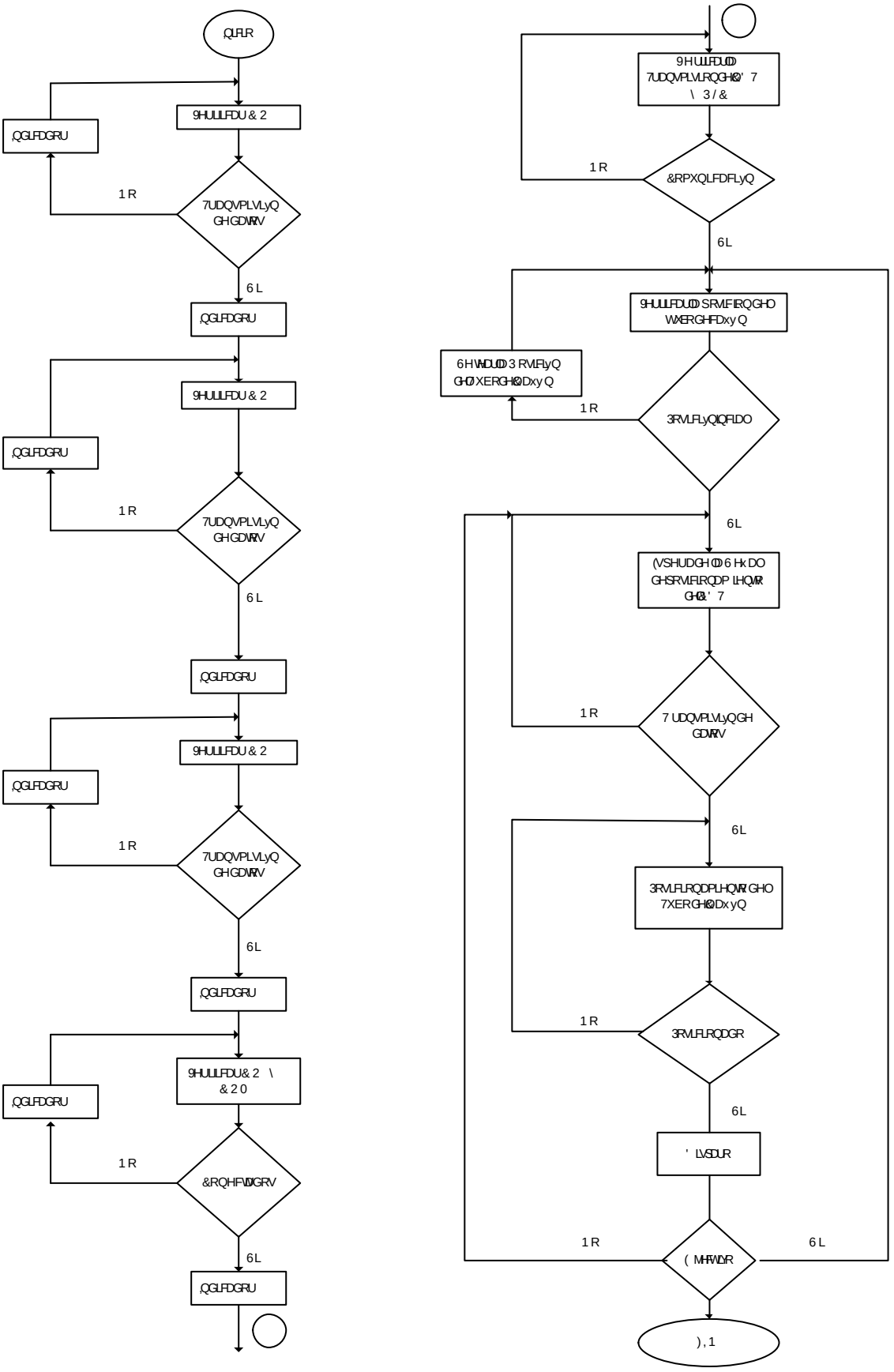


Figura. 3.6. Flujograma para el Posicionamiento del tubo del Cañón

3.2.3. Desarrollo Del Programa Principal

Como se menciono anteriormente, el software de programación del controlador se lo hace mediante rutinas y subrutinas en diagramas de flujo. Las rutinas principales se ejecutan continuamente, mientras que, las subrutinas son ejecutadas solamente cuando son llamadas desde una rutina principal.

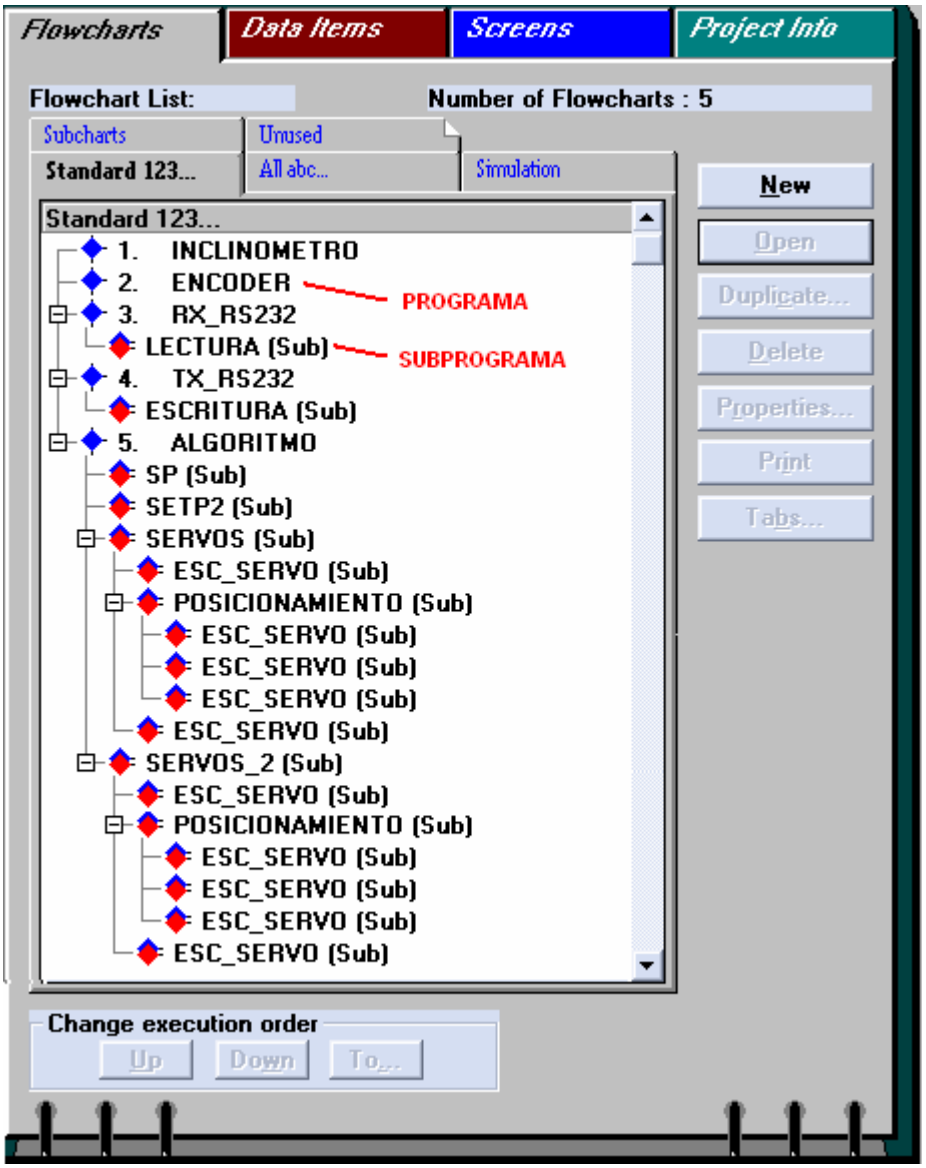
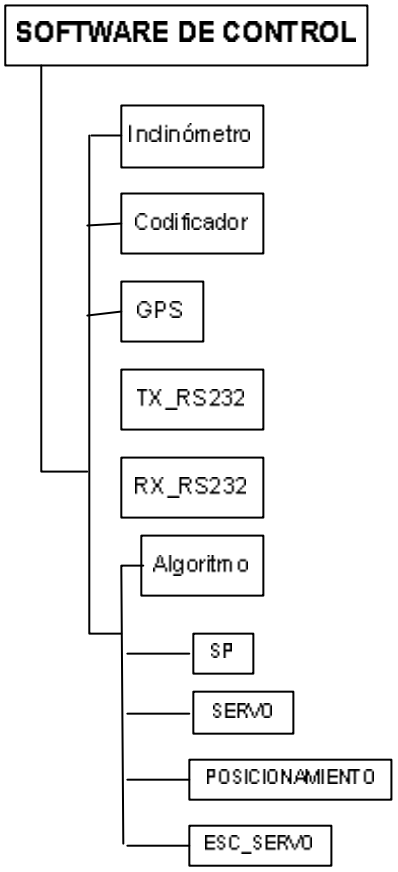
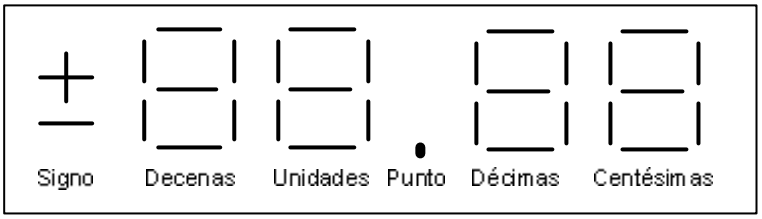


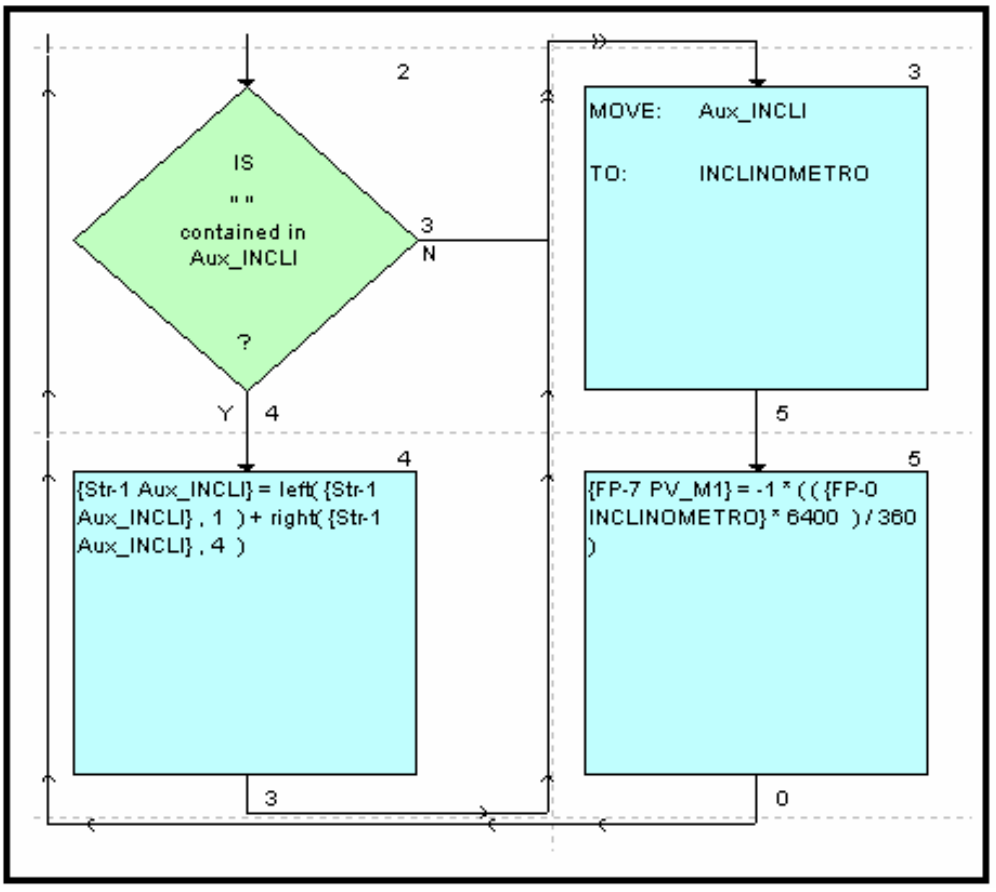
Figura. 3.7. Distribución de programas y subprogramas mediante software

Se presenta a continuación las rutinas y subrutinas primordiales creadas en el controlador WinPLC DL205. para el monitoreo de los sensores, control de posición de los

servomotores y el protocolo de comunicación entre el computador y el PLC, mediante los radio módems.




Signo
Decenas
Unidades
Punto
Décimas
Centésimas


2
IS
" "
contained in
Aux_INCLI
?
3
N
Y
4
3
MOVE: Aux_INCLI
TO: INCLINOMETRO
5
4
{Str-1 Aux_INCLI} = left({Str-1 Aux_INCLI} , 1) + right({Str-1 Aux_INCLI} , 4)
3
5
{FP-7 PV_M1} = -1 * (({FP-0 INCLINOMETRO} * 6400) / 360)
0

3.2.3.2 Codificador

Al igual que la rutina anterior, realiza una lectura continua del codificador, almacenando la información en una variable de tipo entera, para luego ser escalada y utilizada en el control de posición en deflexión del tubo del cañón, se utiliza el módulo H2-CTRIO, por lo que es necesario un correcto mapeo de entradas en el módulo.

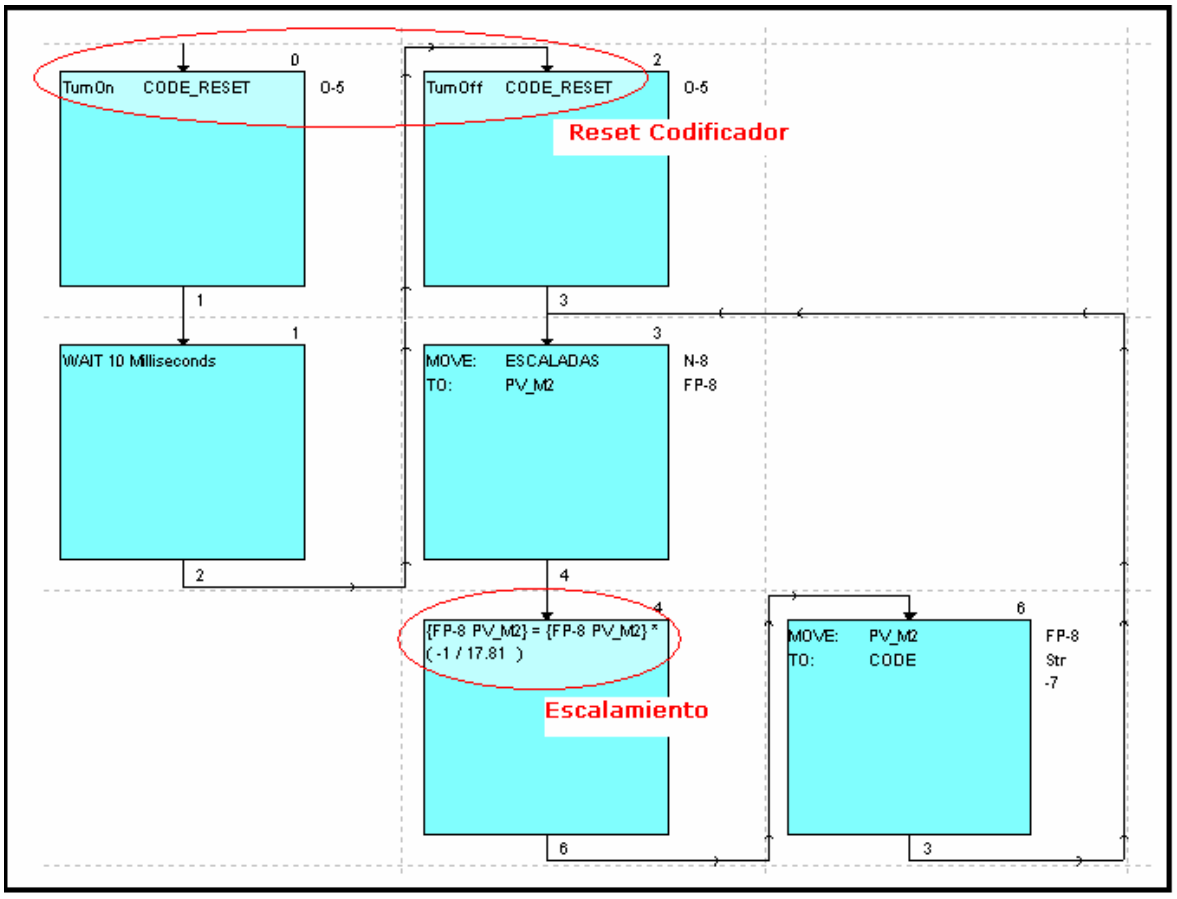


Figura. 3.11. Diagrama de Flujo de lectura del Codificador Incremental

La resolución del codificador luego de los acoples mecánico es de 0.056 milésimas por pulso, por lo tanto, una milésima equivale a 17.81 pulsos, este valor es importante para el escalamiento del conteo de pulsos y la exactitud del posicionamiento. Se debe recalcar que el contador inicia en un valor de cero

Notar que el codificador inicia en un valor cero.

3.2.3.3 GPS

Se realiza una lectura del segundo puerto serial del módulo H2-SERIO, dicha lectura es almacenada en una cadena de caracteres, para luego mediante módulos de manipulación de caracteres, separar en cadenas diferentes la información de Altura y Posición.

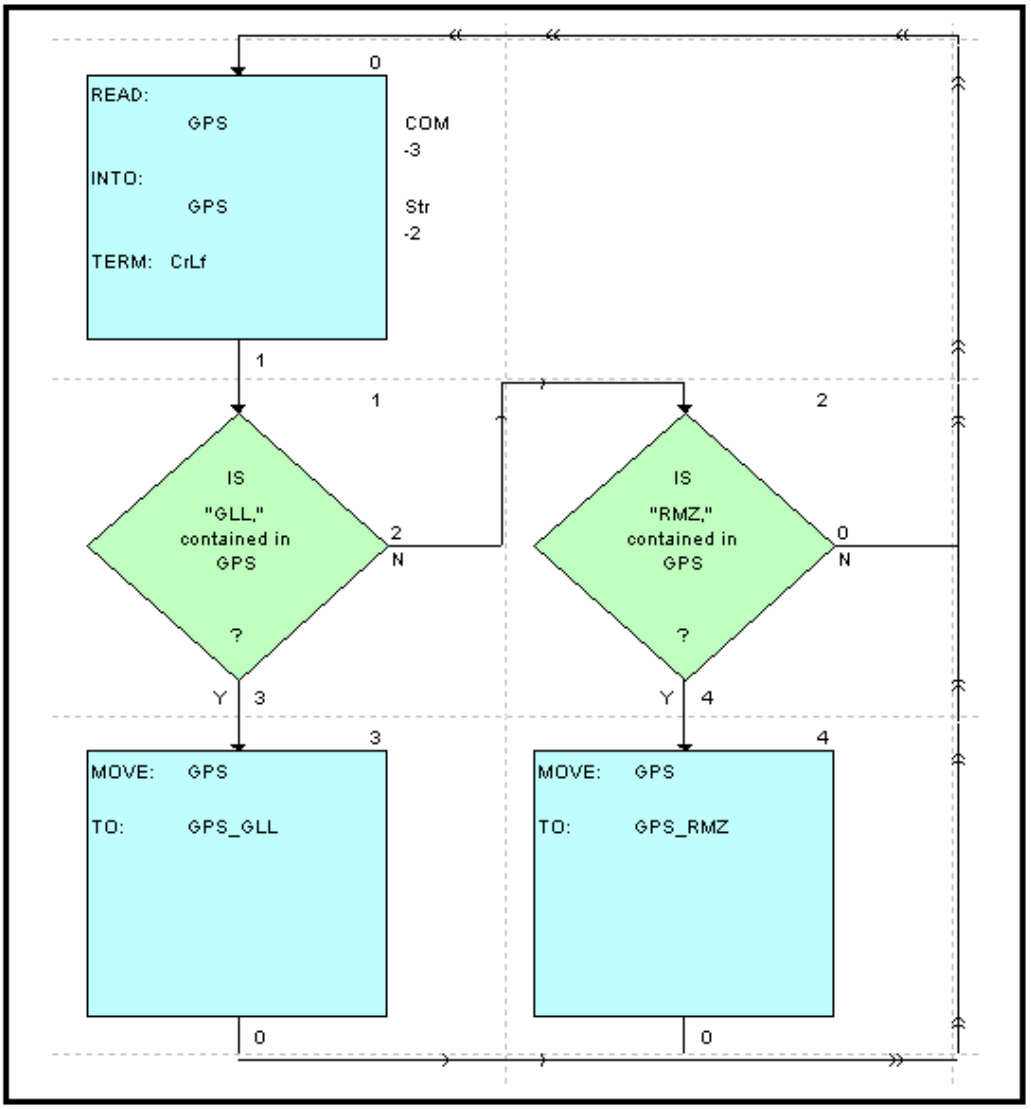


Figura. 3.12. Diagrama de Flujo de lectura del GPS

El GPS utiliza el Standard NMEA 0183 que envía una cadena completa de caracteres con toda la información de medición, por lo que es necesario identificar bs valores de Altura, Posición y almacenarlo en variables tipo String Distintas.

Según el Standard utilizado se transmite en el siguiente formato:

$GPRMZ, Altura, $GPGLL Posición Geográfica

3.2.3.4 TX_R232, RX_R232

Estas rutinas trabajan conjuntamente con el protocolo de comunicación creado en LabView, en la que, el controlador se encuentra en espera de una orden para transmisión o recepción de datos, dicha orden es una cadena de caracteres.

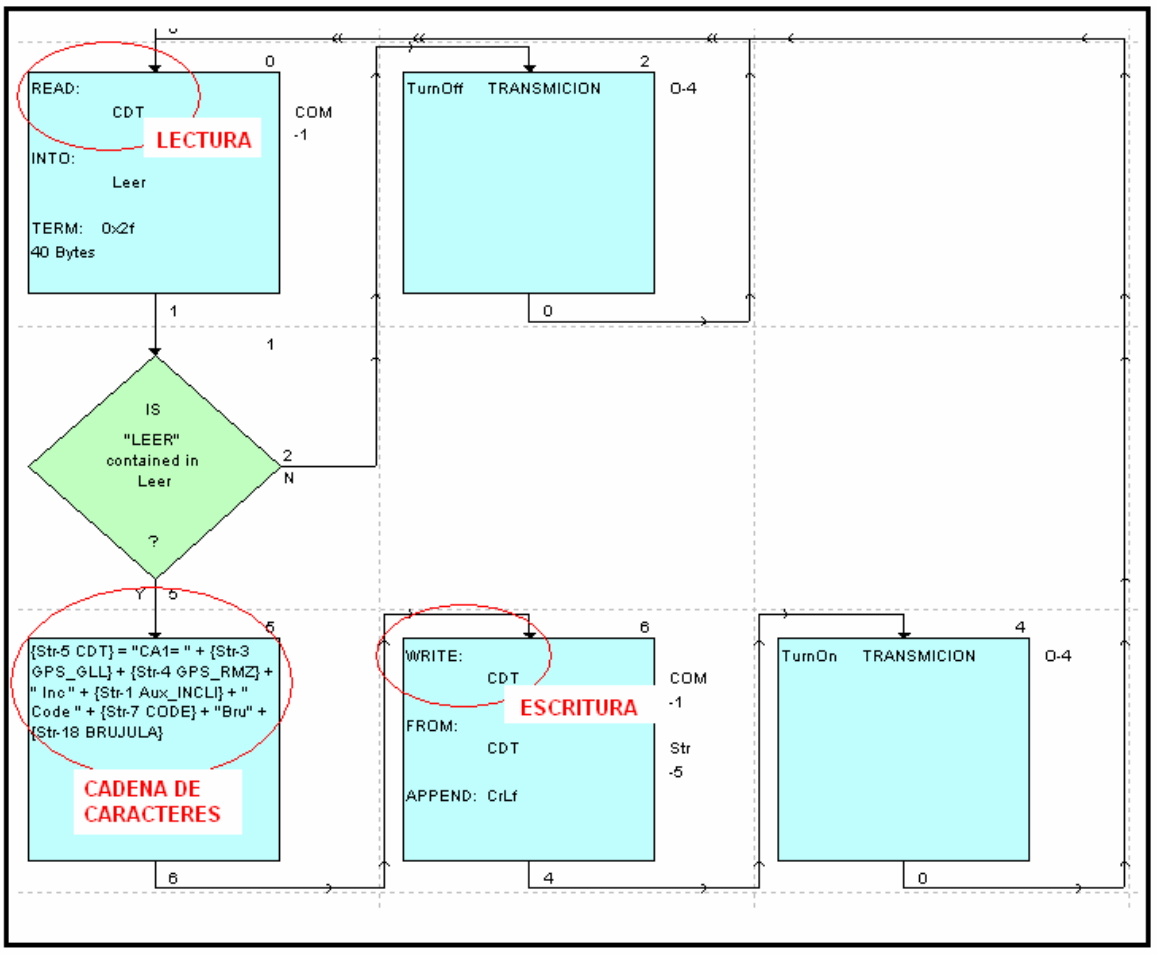


Figura. 3.13. Diagrama de Flujo del Protocolo de Comunicación

En caso de recibir “LEER/” desde LabView, el controlador pasa a modo de escritura, enviando la información más relevante e importante del cañón en una cadena de caracteres denominada CDT, y transmite la información de los sensores antecedidas de palabras que diferencien los datos

La cadena tiene el siguiente formato

“CAX=” (donde X es el número de Tanque en la batería) “Inc ” lectura del inclinómetro “Code” lectura del codificador “Bru” lectura de la brújula

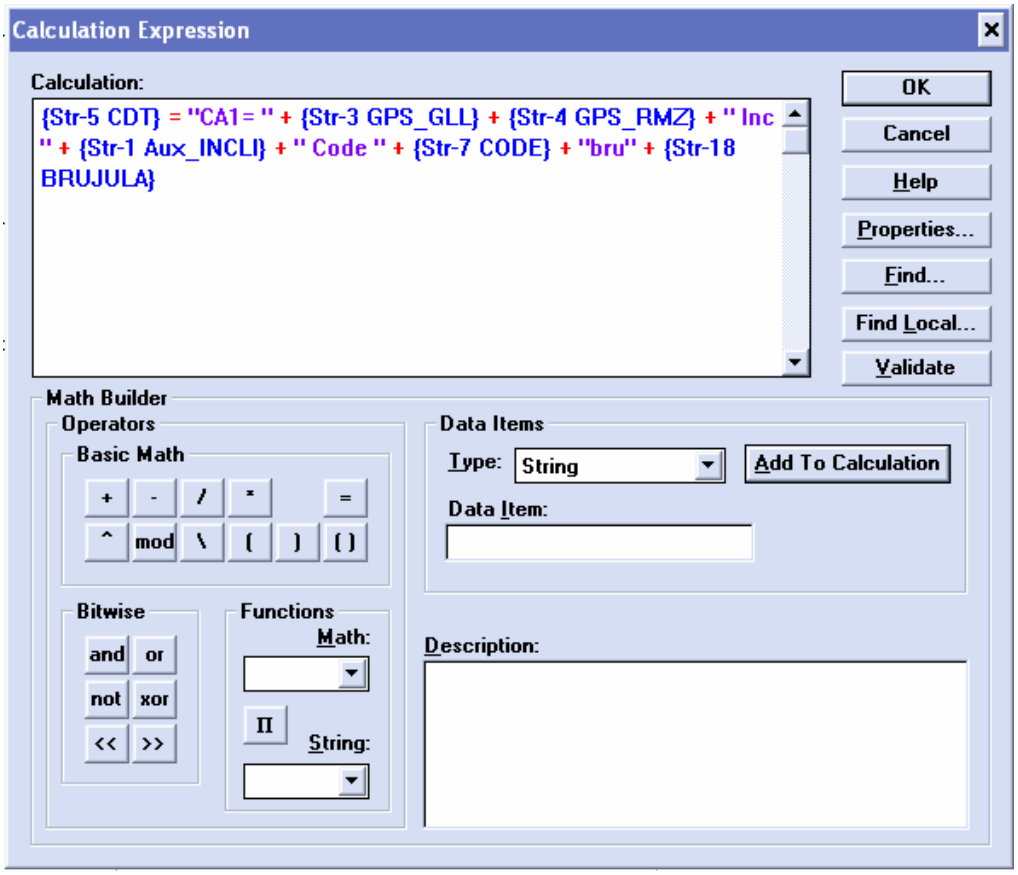


Figura. 3.14. Palabra de Caracteres transmitida desde PLC

Según sea necesario el puerto está configurado en modo de lectura y escritura, a continuación se muestra en las figuras dicha configuración en software

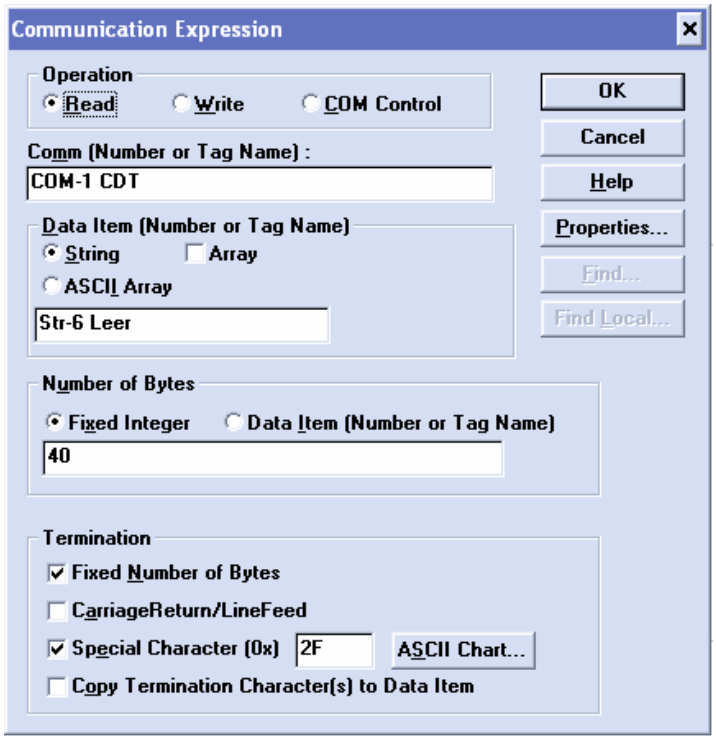


Figura. 3.15. Configuración del puerto para modo Lectura

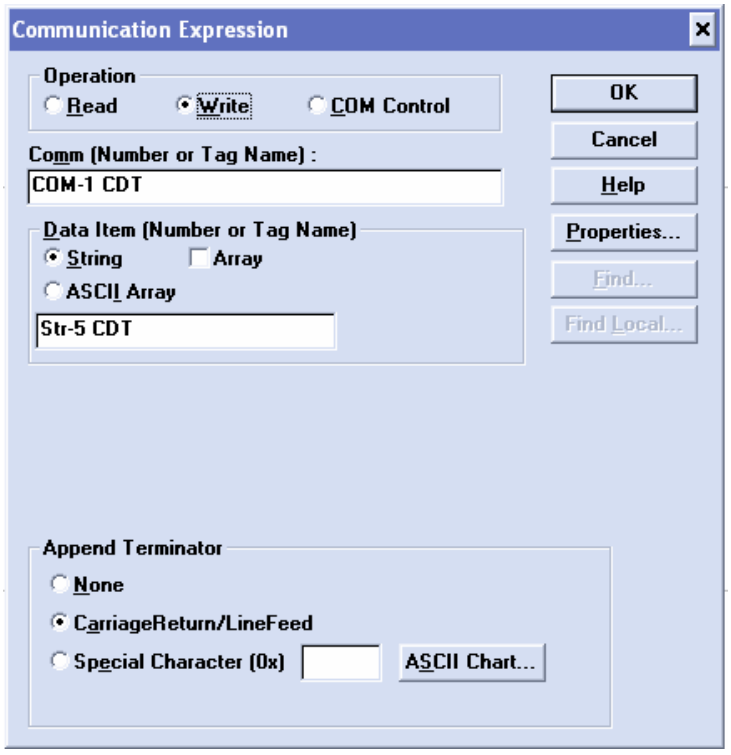


Figura. 3.16. Configuración del puerto para modo Escritura

3.2.3.5. Algoritmo

Esta rutina, ejecuta el algoritmo de control de posición de los servomotores para elevación y deflexión del tubo del cañón, por lo que se utilizan 4 subrutinas principales:

- SP, inicializa los valores de todas las variables que intervienen en el control de los servomotores, reconoce sensores fin de carrera, valores de referencia enviados desde el CDT y se prepara para el posicionamiento.

- SERVOS, calcula el desplazamiento, para encontrar el valor de velocidad máxima, las milésimas de parada y se prepara para enviar los siguientes comandos de transmisión por el puerto serial de los motores.

  MV (Modo de Operación Modo Velocidad)

  A = *"Valor en rps*^{2}*"* * 16 (Aceleración)

  V = "Signo de dirección" *"Valor en rps"* * 64424 (Velocidad)

  G (Inicio)

- Posicionamiento, Una vez definidos los comandos de programación, los transmite, calcula el error y cambia la velocidad del motor desacelerando según las milésimas de parada, para detener inmediatamente el motor cuando exista un error de 0.5 milésimas.

- ESC_SERVO, Es una subrutina que envía cadena de caracteres al puerto del servomotor que está en operación.

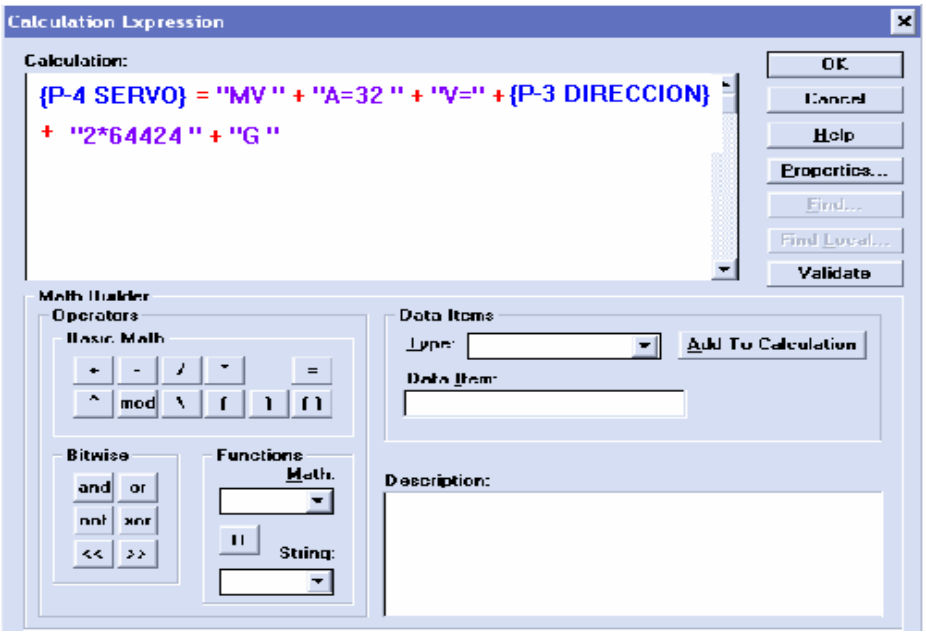


Figura. 3.17. Configuración del Comando de Programación

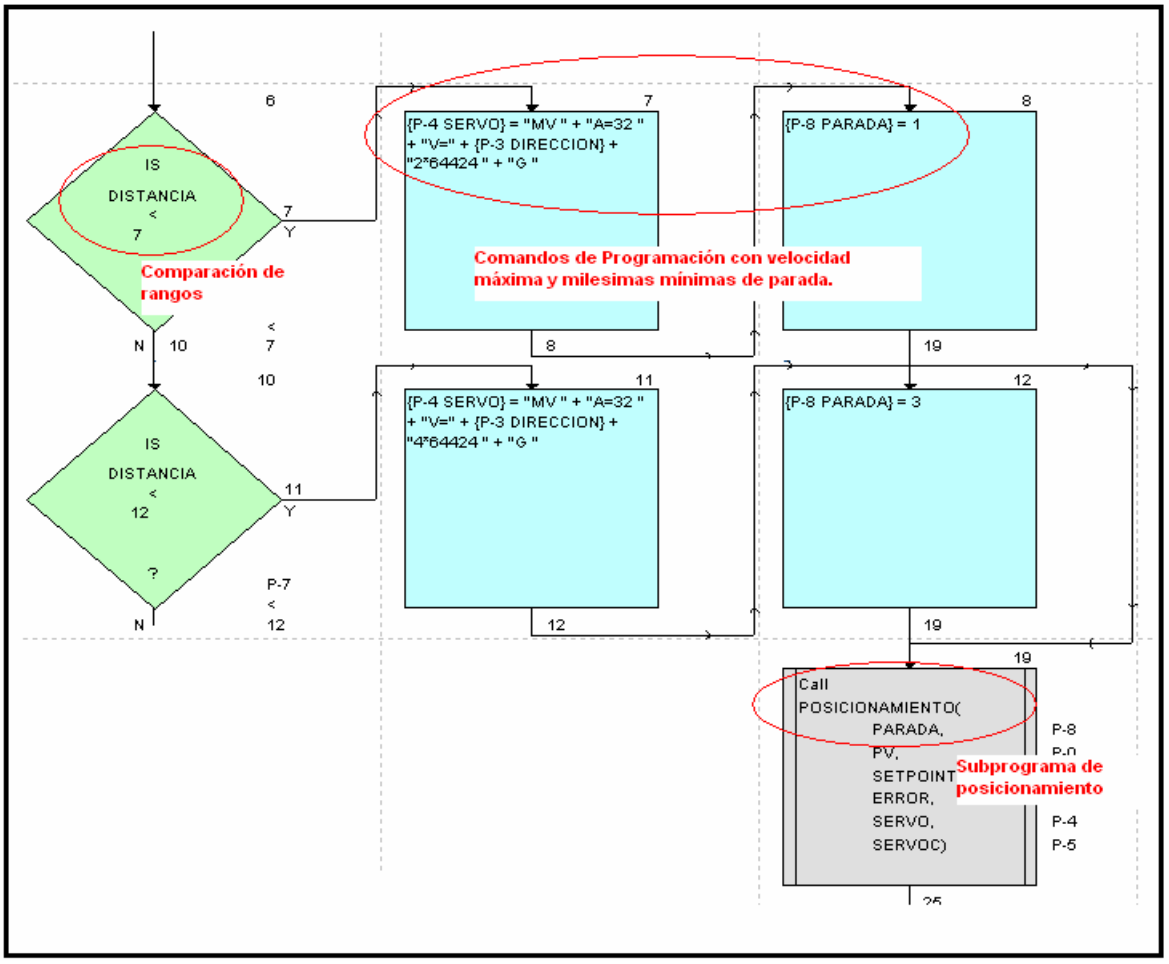


Figura. 3.18. SubDiagrama de Flujo de comparación de rangos de desplazamiento

3.3. INTERFAZ HMI

El Programa creado para la interfaz HMI (Human – Machina Interfase) se basa en el Software desarrollado en el Sistema de Puntería Digital SPD-155, al que se ha denominado como Programa inicial, con lo que fue indispensable la utilización LabView 7.0 Express.

La interfaz creada es una fusión del SPD-155 con el Sistema desarrollado para la automatización del Cañón autopropulsado de 155mm, por tanto, se conserva algunas fases del programa inicial, se complementa y actualiza con la parte del Sistema de Control Automático y comunicación SCA-155 necesaria en el Sistema propuesto.

3.3.1. Parámetros y Variables.

Los Parámetros y Variables importantes considerados en este programa son:

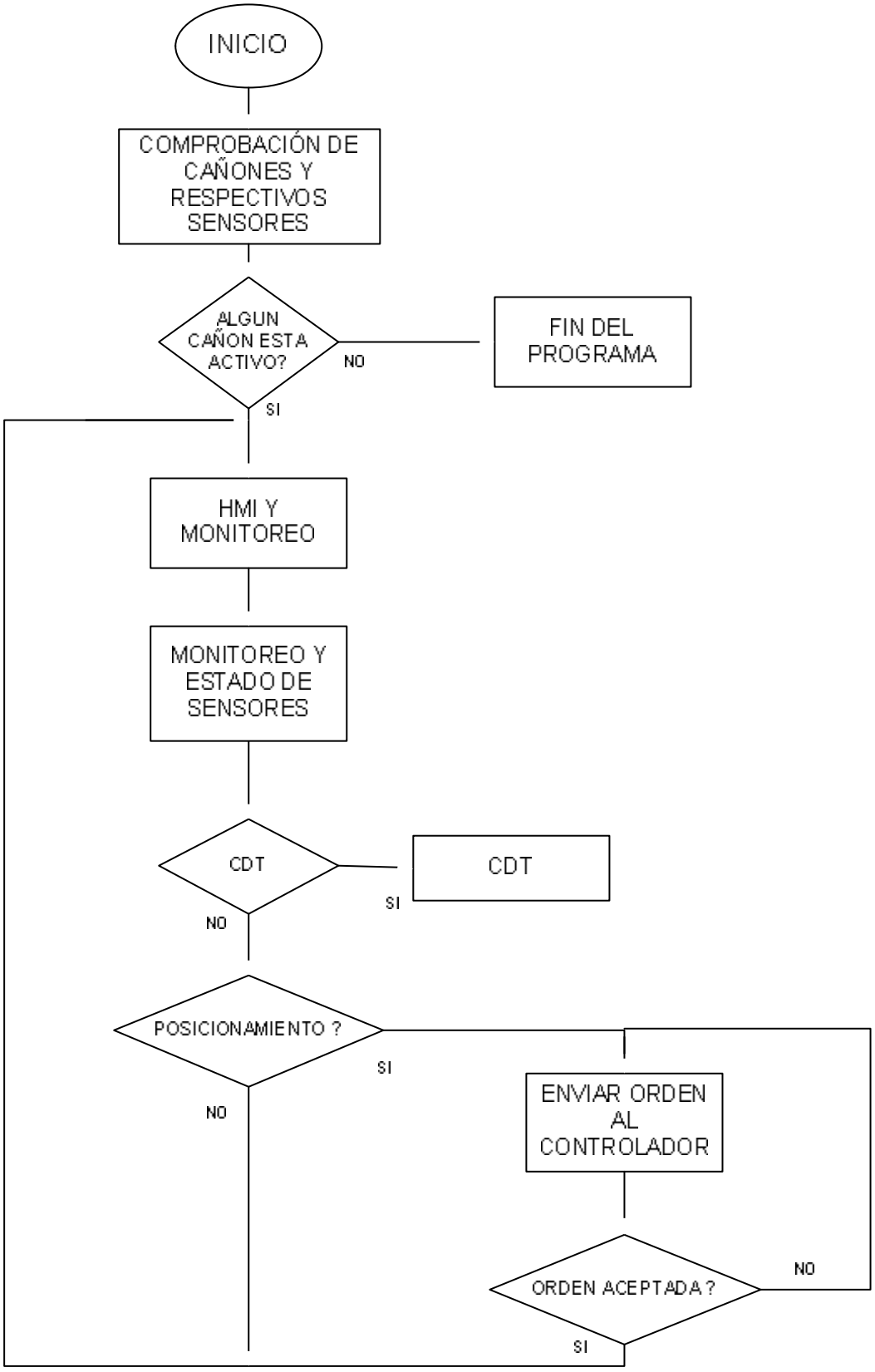
- Comunicación Serial RS – 232 entre el Computador y el PLC.
- Deflexión y elevación dada en milésimas de grado.
- Altura y alcance dada en metros

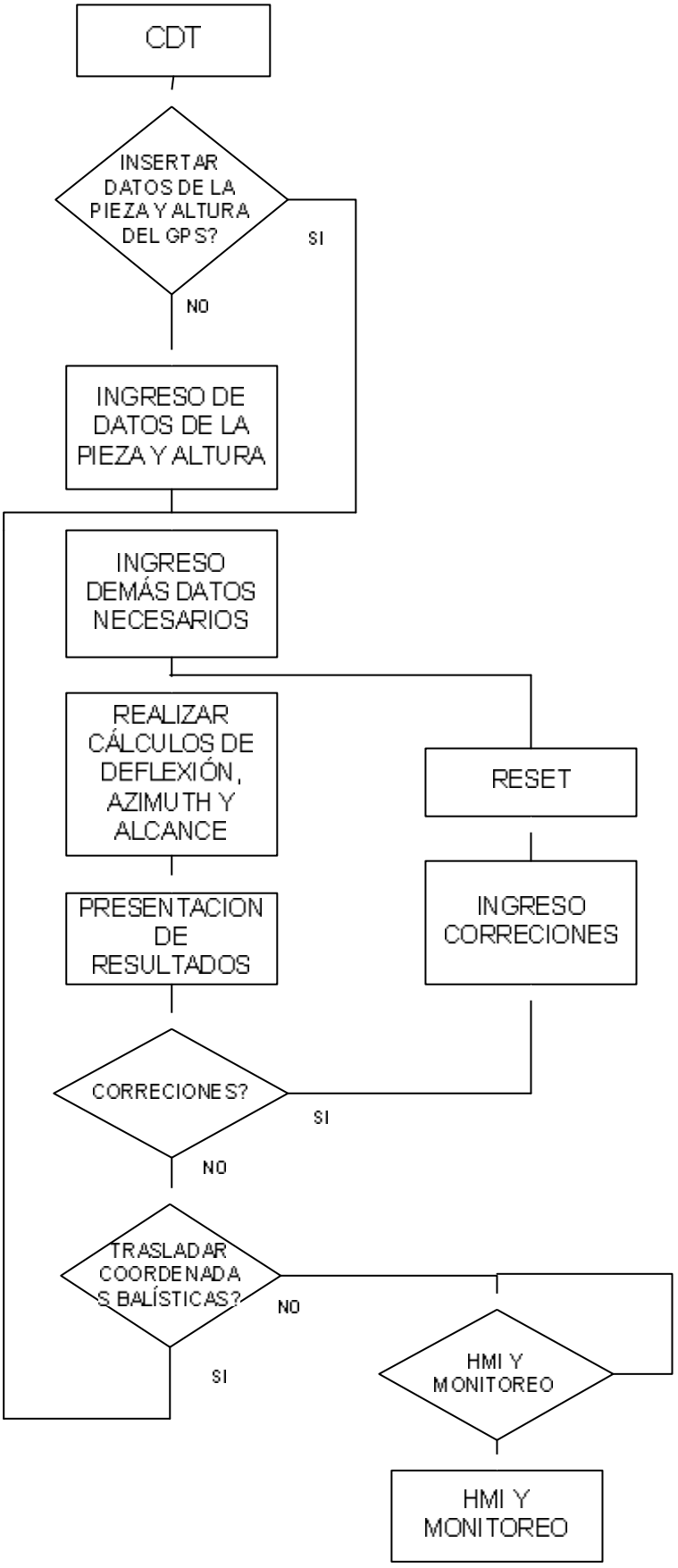
.

- Carga de la Munición.
- Datos de las Tablas de Tiro.

A partir de todos estos parámetros se realiza el reconocimiento de los sensores, cálculos matemáticos del CDT, y se envía datos de ángulos de deflexión y elevación para el posicionamiento automático en el controlador.

3.3.2. Diagramas De Flujo del Software de Cálculo y Monitoreo

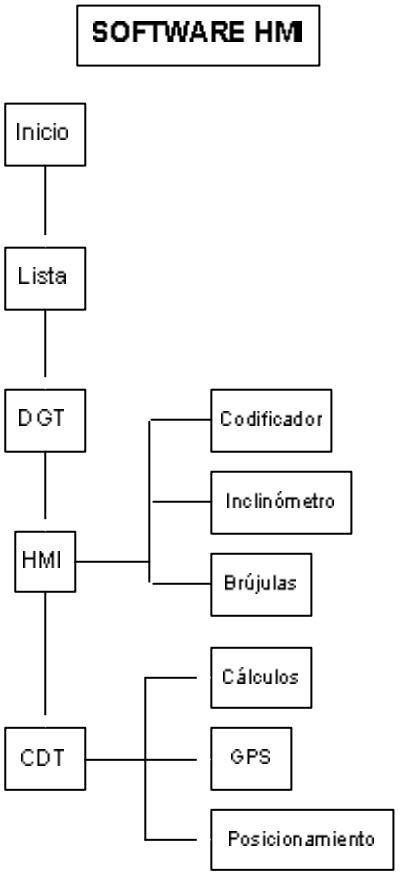



CDT
INSERTAR DATOS DE LA PIEZA Y ALTURA DEL GPS?
SI
NO
INGRESO DE DATOS DE LA PIEZA Y ALTURA
INGRESO DEMÁS DATOS NECESARIOS
REALIZAR CÁLCULOS DE DEFLEXIÓN, AZIMUTH Y ALCANCE
RESET
PRESENTACION DE RESULTADOS
INGRESO CORRECIONES
CORRECIONES?
SI
NO
TRASLADAR COORDENADAS BALÍSTICAS?
NO
SI
HMI Y MONITOREO
HMI Y MONITOREO

3.3.3. Desarrollo Del Programa Principal

El Programa para el Sistema de Posicionamiento Automático de bs Cañones de 155 mm, continúa con el mismo diseño del Sistema de Puntería Digital, presentando una secuencia de pantallas en las que se desarrolla la interfaz de comunicación con el operador.

A continuación se presenta un diagrama de la organización secuencial de cada fase del programa el orden de ejecución, cada vez que una nueva pantalla es ejecutada la anterior se cierra.



3.3.3.1. Inicio

Es una pantalla de inicio y arranque del software HMI, en el que se muestra la información principal del programa, tiene un tiempo de presentación de 10 segundos, para luego cerrarse y dar a lugar la ejecución de la siguiente pantalla en forma automática.

Permite escoger el Puerto Serial COM con el que va a trabajar el computador.

Figura. 3.22. Pantalla Inicio

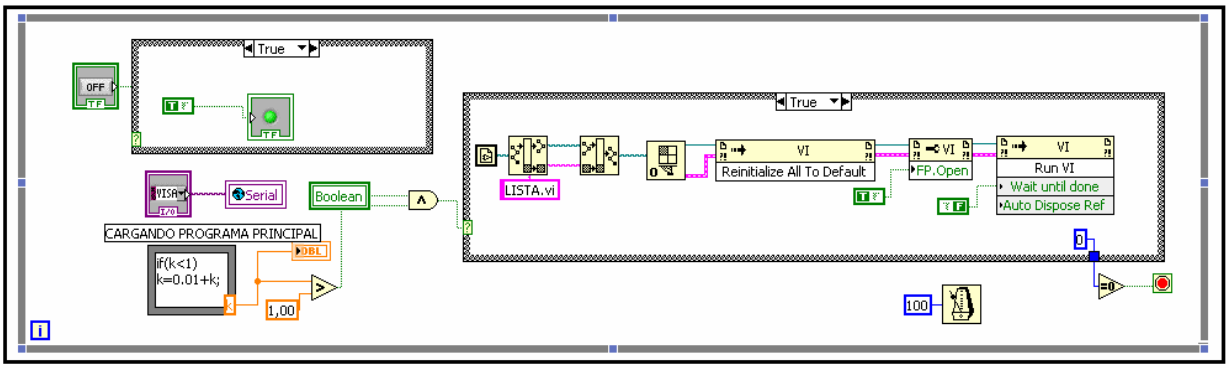


Figura. 3.23. Código LabView Inicio

3.3.3.2. Lista

Al cerrarse la pantalla de Inicio, se ejecuta la pantalla de Lista, en la cual se entrega la información visual acerca de los cañones activos de la batería, para lo que es necesario establecer comunicación con los controladores activos en cada cañón.

Figura. 3.24. Pantalla Lista

Entonces se ejecuta el protocolo de comunicación creado para el PLC y el Computador, en el que se envía una palabra indicadora de que el programa esta listo para recibir información del controlador, abriendo el puerto serial de lectura.

Una vez dado el reconocimiento se transmite la orden de cerrar el puerto en el Programa y el controlador, para evitar conflictos en la comunicación y errores de ejecución.

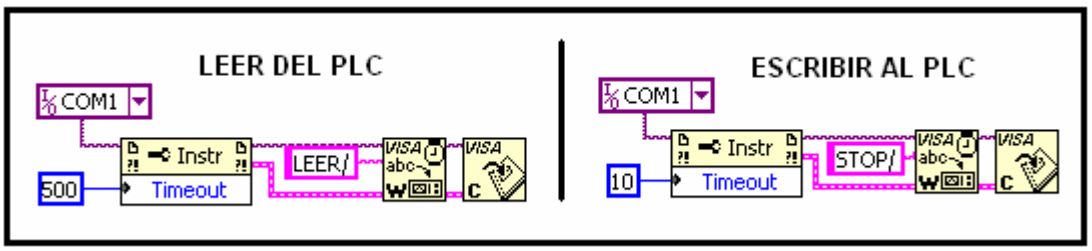


Figura. 3.25. Código LabView Comunicación

Cada cañón envía su información en una cadena de caracteres tipo String antecedida por la palabra “CAX=”, donde X, corresponde al número de posición del cañón en la batería, en el código se reconoce el valor de X para identificar los cañones que están activos.

Al reconocerse los cañones activos, la opción “ACEPTAR”, permite continuar con la secuencia del programa; si ningún cañón está presente, existe la posibilidad de cerrar automáticamente el programa o continuar solamente con el cálculo del CDT.

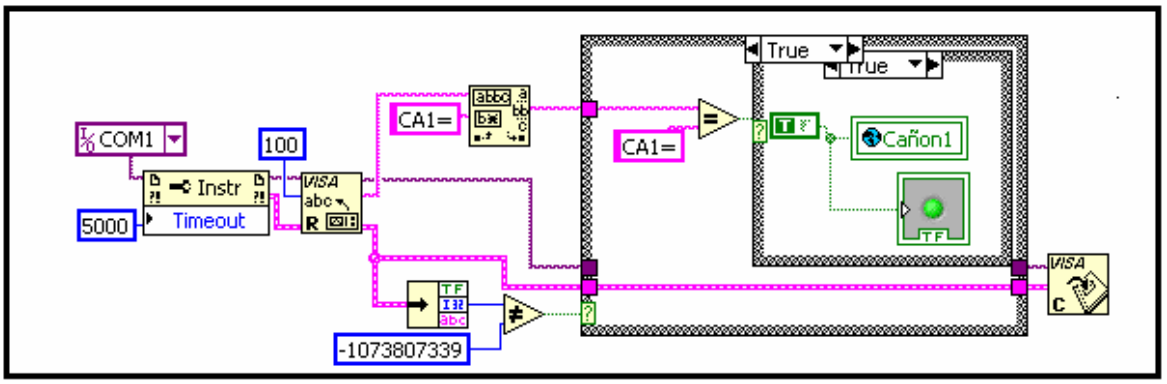


Figura. 3.26. Código LabView Reconocimiento Tanques activos

3.3.3.3 DGT

Se despliega la Pantalla en al que se realiza la corrección de la DGT, según los valores teóricos y valores reales, presenta las siguientes opciones:

“Home”, Transmite ordenes de posicionamiento inicial del tubo del cañón

Los valores iniciales son los siguientes:

Elevación: 300 milésimas

Deflexión: Corrección de la DGT.

"Corregir", Envia valores de corrección en dirección y posiciona nuevamente el tubo del cañón, pero ya en la dirección general de tiro exacto, en la que se desea realizar el tiro.

"Aceptar", Permite seguir con la Interfaz HMI.

Figura. 3.27. Pantalla DGT

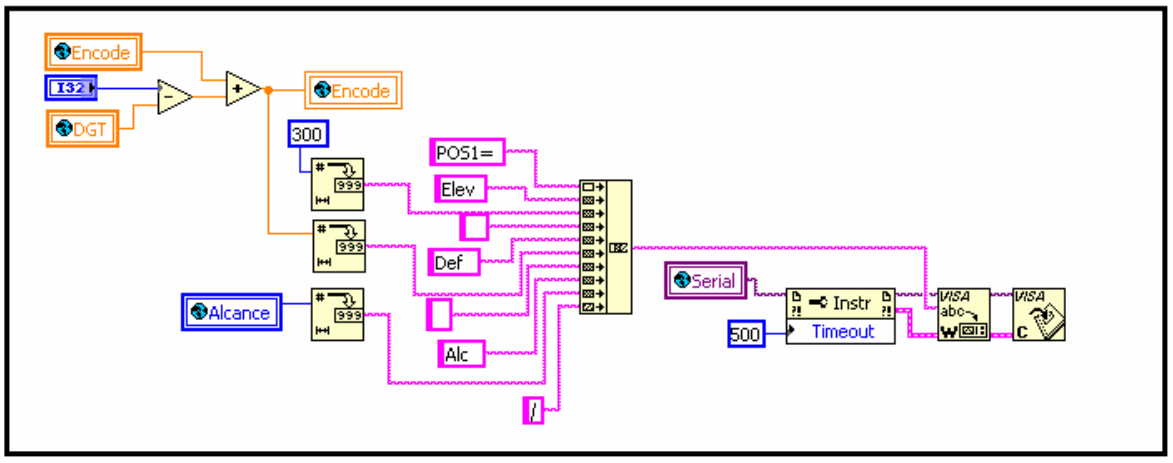


Figura. 3.28. Código LabView Transmisión de la Palabra de control de corrección

3.3.3.4. HMI

Esta sección del programa es la encargada del monitoreo del sistema, hace un barrido de todos los sensores de los cañones activos, dicha información llega en una sola cadena de caracteres desde el controlador, por lo que se requiere enviar una palabra antecediendo a la lectura de cada sensor, “Inc” en el caso del inclinómetro y “Code” en el codificador incremental, para realizar la diferenciación de datos, mediante herramientas de manipulación de caracteres para LabView, con lo que se consigue tener lecturas separadas de los equipos.

Figura. 3.29. Pantalla HMI

El monitoreo se realizará siempre que en el protocolo de comunicación se envíe la orden de abrir el puerto de lectura en LabView, para que el puerto del controlador permanezca en modo de escritura.

La información se presenta en milésimas de grado por lo que se hace la conversión de datos, donde se utiliza la siguiente relación:

360º corresponden a 6400 milésimas de grado.

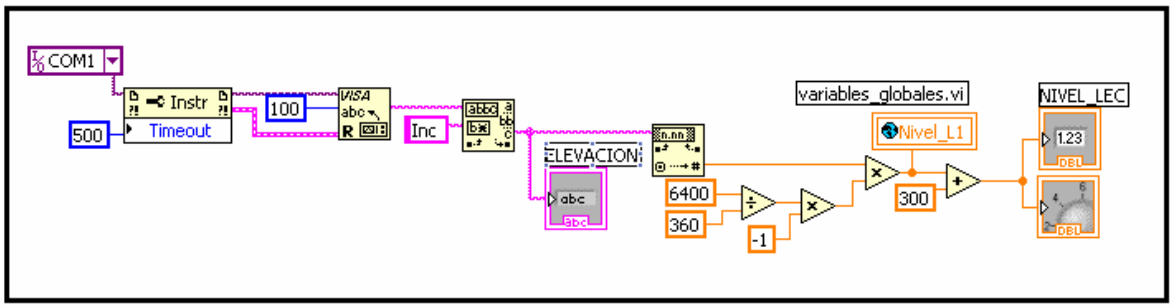


Figura. 3.30. Código LabView Monitoreo del Inclinómetro

Esta Pantalla presenta las siguientes opciones:

"CDT", permite cerrar la pantalla de monitoreo y abrir la Pantalla del CDT para la realización de cálculos.

"SALIR", cierra todo el software HMI.

3.3.3.5. CDT

En CDT se realiza el cálculo de la deflexión y elevación del tubo del cañón dependiendo de los parámetros del observador, unidad de tiro, DGT, y correcciones dadas. Utiliza las tablas de tiro según la carga de munición seleccionada. Esta parte del programa es una actualización directa del cálculo del CDT realizado en el SPD-155.

Figura. 3.31. Pantalla CDT

Además existen las siguientes opciones principales:

“POSICIONAR”, permite transmitir la orden de posicionamiento del tubo de los cañones conjuntamente con los valores de Deflexión y Alcance calculados, en una cadena de caracteres hacia el controlador, por lo que hace uso de una manipulación de caracteres y del protocolo de comunicación para que el controlador este listo a recibir la información, está opción estará disponible solamente si, por lo menos un cañón ha sido reconocido como activo.

“GPS”, permite cargar automáticamente la información de altura y Posición del tanque dada por el GPS, para el cálculo del CDT, esta opción estará activa, siempre y cuando dicho sensor este conectado al sistema y este trabajando correctamente.

“HMI”, permite regresar y ejecutar la pantalla de Monitoreo de Sensores

“SALIR”, Cierra todo el Programa SCA - 155

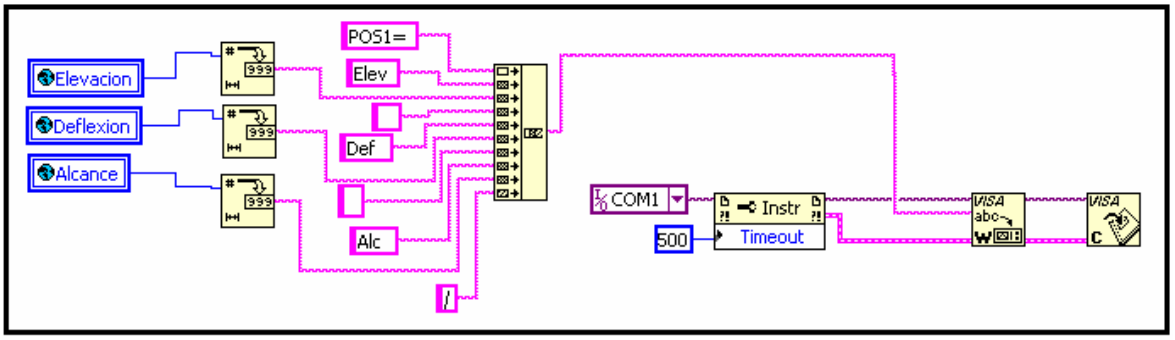


Figura. 3.32. Código LabView Envio de la orden de Posicionar

Como se menciono anteriormente se utilizaron 2 tipos de lenguajes de programación, tanto para el controlador como, para la interfaz HMI; dichos programas solo permiten trabajar a cada puerto de comunicación serial solamente en un sentido a la vez o Half duplex.

Al momento de manipular la dirección de la transmisión de datos existen funciones para abrir y cerrar los puertos tanto en, modo de escritura como, de lectura. Se debe recordar que siempre que se cambie el sentido de transmisión es indispensable cerrar dichos puertos.

El Protocolo creado se basa en lo siguiente:

El Controlador se encuentra en espera de una señal del Computador que indique en que sentido de transmisión debe trabajar, según esta señal no existen conflictos con puertos, es decir si el Computador está en modo de escritura el Controlador estará obligatoriamente en modo de lectura y viceversa.

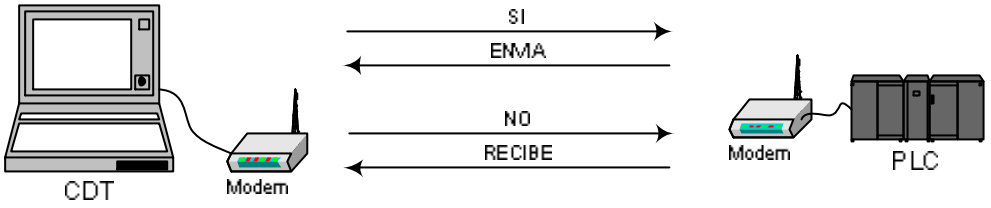


Figura 3.22. Forma de transmisión entre el Computador y el Controlador

Se tomó en cuenta que el tiempo de procesamiento del Controlador y del Computador no son los mismos, por lo que, fue necesario una sincronización mediante software, para que la transmisión de datos sea transparente en el Software de control y en la Interfaz HMI. Dicha sincronización se realiza en el controlador, que dependiendo la palabra de control enviada desde el computador, se realiza la lectura o transmisión por el Puerto.

CAPITULO IV

HARDWARE

4.1 INTRODUCCIÓN

En los Capítulos anteriores se presentó una explicación de las características principales que debe cumplir el prototipo diseñado en este proyecto, si bien es cierto las características de software son importantes no se puede descuidar el hardware utilizado, ya que cada elemento del equipo debe ser seleccionado dependiendo de las necesidades y especificaciones técnicas requeridas por el proyecto. Como hardware nos referimos a todo el material físico utilizado.

Este capitulo se refiere a todo el material a utilizarse como hardware y se presenta todos los criterios que se determino importantes para su selección

4.2 CRITERIOS DE SELECCIÓN DEL EQUIPO UTILIZADO

4.2.1 Controlador (Controlador Lógico Programable)

Un PLC es un dispositivo digital utilizado para el control de máquinas y operación de procesos. Se trata de un aparato digital electrónico con una memoria programable para

el almacenamiento de instrucciones, permitiendo la implementación de funciones específicas como: operaciones lógicas y aritméticas, secuencias, temporizaciones y conteos; con el objeto de controlar máquinas y procesos.

Respecto al software de programación, los PLCs en su mayoría se programan por medio de lógicas en escalera aunque muchos fabricantes ofrecen al usuario otras posibilidades adecuadas a sus requerimientos creando otros lenguajes de programación.

La estructura de un PLC se muestra a continuación en un sencillo diagrama en bloques. En la Figura 4.1 se muestran las tres partes fundamentales: la CPU, las entradas y las salidas.

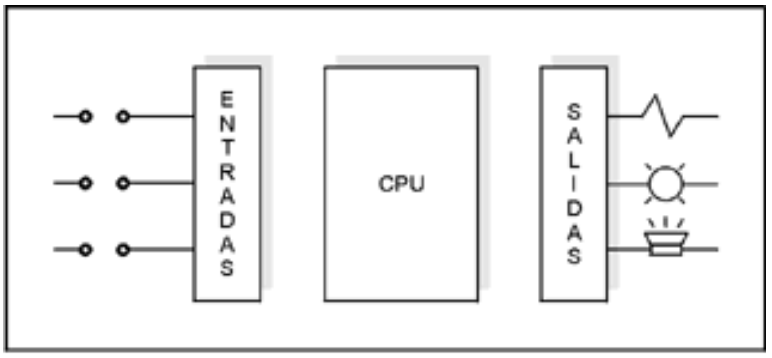


Figura. 4.1 Diagrama en bloques de los elementos básicos de un PLC

La CPU es el cerebro del PLC, responsable de la ejecución del programa desarrollado por el usuario. Estrictamente, la CPU está formada por uno o varios procesadores; en la práctica, puede abarcar también a la memoria, puertos de comunicaciones, circuitos de diagnóstico, fuentes de alimentación, etc.

Las entradas (interfases o adaptadores de entrada) se encargan de adaptar señales provenientes del exterior a niveles que la CPU pueda interpretar como información. En efecto, las señales externas o de campo pueden implicar niveles y tipos de señal eléctrica diferentes a los que maneja la CPU. En forma similar, las salidas (interfases o adaptadores de salida) comandan dispositivos de campo en función de la información enviada por la CPU.

Existen varios tipos de PLCs según su capacidad, su cantidad de I/0s, o su aplicación, entre los que tenemos PLCs modulares que constan de varios módulos insertables en el chasis del PLC con funciones distintas, para poder adaptar las necesidades de la aplicación al controlador.

En el proyecto se escogió un controlador Lógico Programable, ya que fue una alternativa conveniente en costo y rendimiento, dando las siguientes ventajas:

- Poseer una estructura modular dando diferentes funciones en cada módulo.
- Programación sencilla y fácil manejo
- Adaptabilidad a los tipos y rangos de I/O utilizados.
- Requerir poco espacio, sencillez en las conexiones y ofrecer resistencia a las vibraciones, ruido e inclemencias del tiempo.
- Ofrecer posibilidades de comunicación con PC´s.

Para este proyecto se adquirió el WinPLC DirectLOGIC 205 Koyo, debido a que cumple con las especificaciones necesarias dadas en el punto anterior, además que cuenta con varios módulos que presentan funciones directas como el módulo de comunicación RS-232 con tres puertos seriales con el que se puede adaptar hasta más

de 10 puertos seriales, el módulo contador de alta velocidad para control de motores, módulos de I/Os, etc.

Figura. 4.2 WinPLC DL205

Se detalla a continuación las características específicas del PLC según los módulos que se adquirieron para la realización del prototipo

4.2.1.1 CPU DL205

El WinPLC DL205 tiene varios tipos de CPU dependiendo de su velocidad de procesamiento y su capacidad de memoria. Posee el mismo tamaño que un módulo cualquiera para ser insertado en la base del PLC junto a los demás módulos

La CPU que se utiliza en este proyecto es **H2-WPLC3-EN** por ser el que posee mayor capacidad de memoria y velocidad de procesamiento en los procesadores de su familia, garantizando un trabajo en tiempo real, lo que es una necesidad fundamental para el control de motores, sus características específicas y técnicas se muestran más adelante.

Figura. 4.3 CPU H2 - WPLC3 - EN

Especificaciones	**H2-WPLC3-EN**
Procesador	Hitachi SH3 Series 770B Processor
Velocidad de procesamiento	100 Mhz
Software compatible	Entivity Studio o Think & Do Live
Memoria	8 Mb FLASH ROM 8 Mb RAM
Puertos	Puerto 0: RJ12, Serial Puerto 1: RJ45, Ethernet 10 MBPS
Alimentación	680 mA a 5 VDC
Peso	6 oz.
Temperatura	Operación: 0 – 60°C Almacenamiento: -20 – 70°C

Tabla. 4.1. Especificaciones del CPU DL205

4.2.1.2 Módulo H2 – SERIO

El módulo H2 – SERIO permite añadir 3 puertos seriales al sistema WinPLC, cada módulo se puede insertar en la base del DL205. Al utilizar puertos seriales en el PLC se elimina la necesidad de usar el adaptador ESU – 100 y la conexión directa del computador utilizado en el Sistema de Puntería Digital SPD – 155.

Figura. 4.4 Módulo H2 -SERIO

Especificaciones	**H2-WPLC3-EN**
Tipo	Módulo inteligente para uso con H2-WPLC
# de Puertos Seriales por módulo	3
# de módulos soportados por WinPLC	3
Protocolos	Serial ASCII y Modbus RTU
Conector	RJ12
Alimentación	210 mA a 5 VDC
Velocidad de transmisión	300 a 57.6 Kbaud
Condiciones Ambientales Temperatura de Operación: Humedad:	 0 – 60°C 5 – 95%

Tabla. 4.2. Especificaciones del H2 -SERIO

Configuración: La configuración de los puertos seriales se la hace mediante el software del Controlador, en el que se dan los valores según la aplicación y necesidades. En la herramienta IOView se presentan toda la información y configuración del hardware, de manera automática una vez conectados los módulos.

Attributes	Value
Serial Port	COM2
Device Name	WinPLC_Slot2_PortA
Serial Port Settings	
Access Type	Generic Serial Communication
Baud Rate	9600
Parity	None
Data Bits	8
Stop Bits	1
Flow Control	None
Receive Buffer Size	2048
Transmit Buffer Size	2048

Figura. 4.5 Presentación de la Configuración de un Puerto

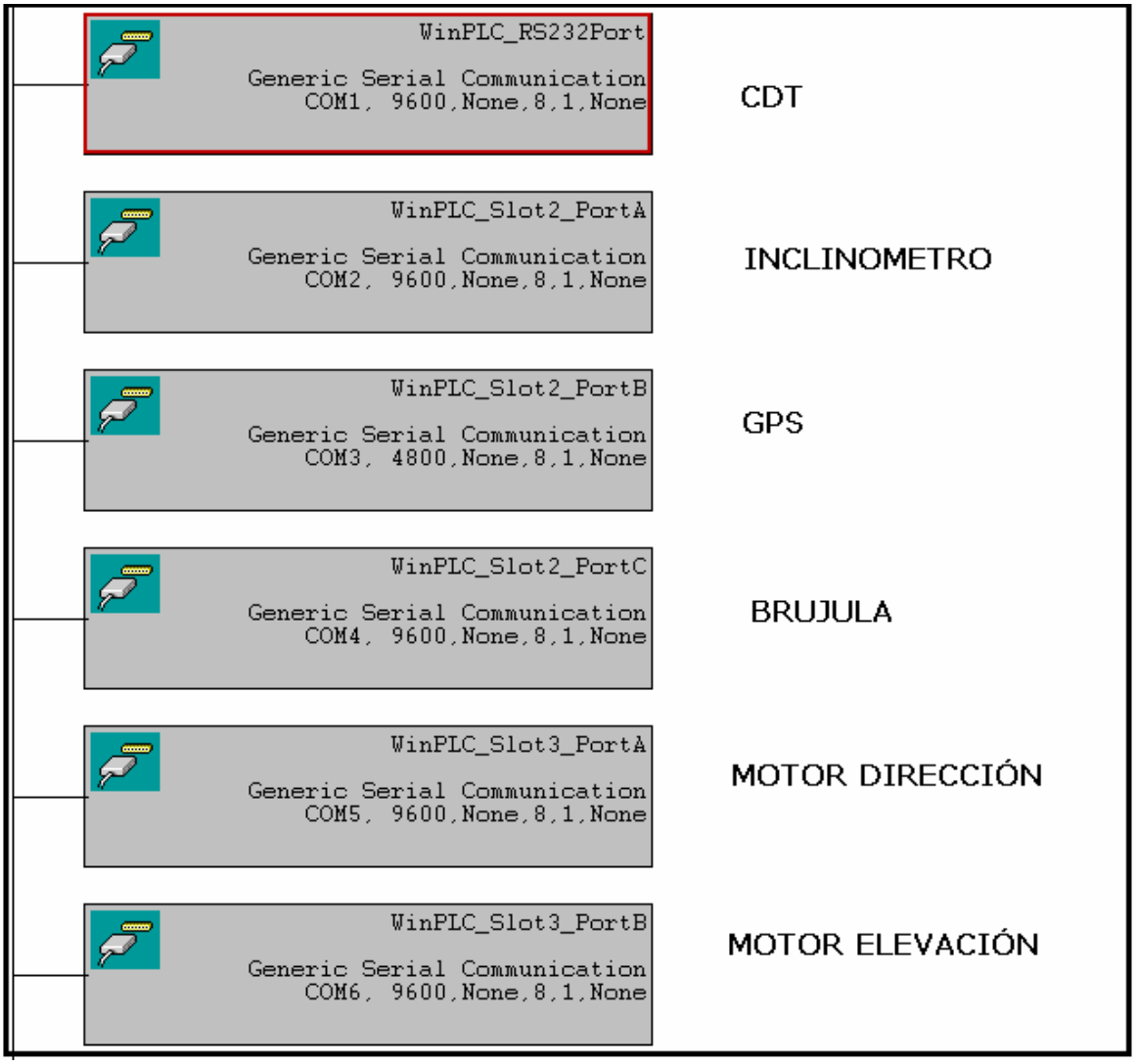


Figura. 4.6 Configuración y asignacón de puertos para el Controlador en IOView

4.2.1.3 Módulo H2 – CTRIO

El Módulo H2-CTRIO es un contador de alta velocidad que permite conteo y salida de pulsos, permite controlar servomotores y motores de pasos.

Ofrece dos contadores de 100 Khz. o dos salidas de tren de pulsos hasta 25Khz cada uno. Posee un software específico para su configuración denominada H2-CTRIO Workbench, el cual facilita el manejo de dicho módulo.

Este módulo es utilizado para la utilización del codificador incremental, sensor de posición en deflexión del tubo del cañón.

Figura. 4.5. Módulo H2 -CTRIO

Configuración: La configuración de las entradas y salidas del módulo H2-CTRIO, se la realiza mediante un software gratis disponible en Internet denominado **CTRIO Workbench**

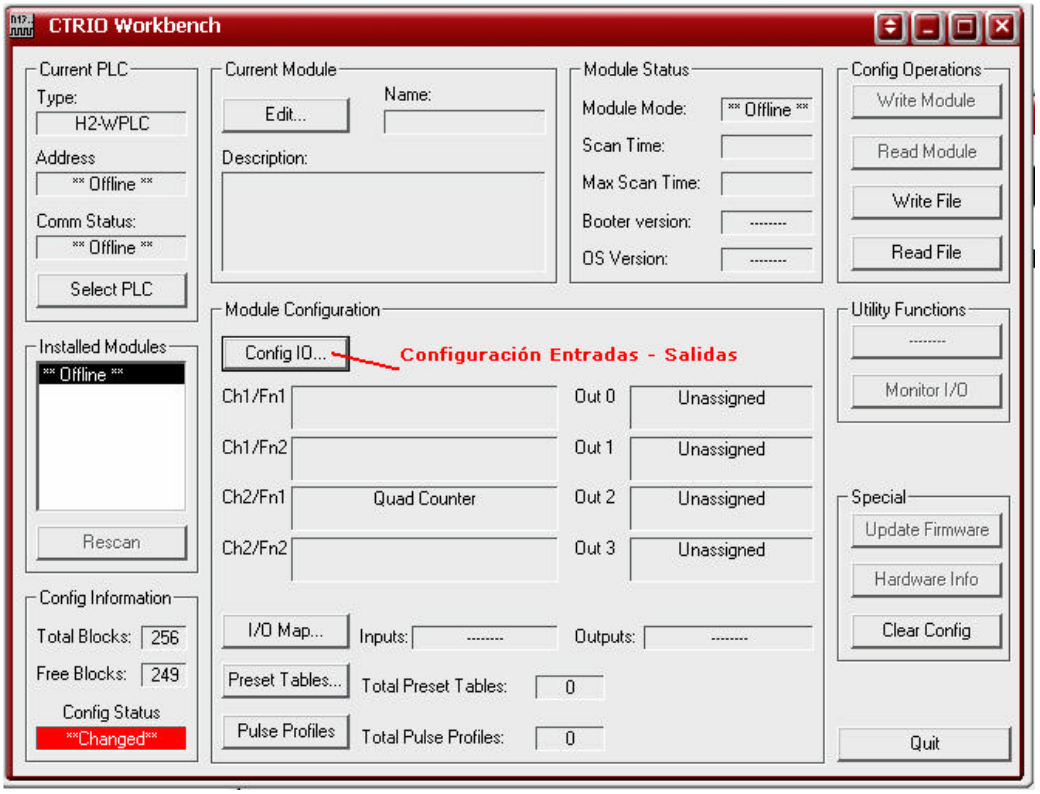


Figura. 4.6. CTRIO Workbench

Mediante este software se puede realizar pruebas de funcionamiento del módulo, ya que una vez configuradas las entradas y salidas del mismo, se puede manipularlas para analizar su respuesta.

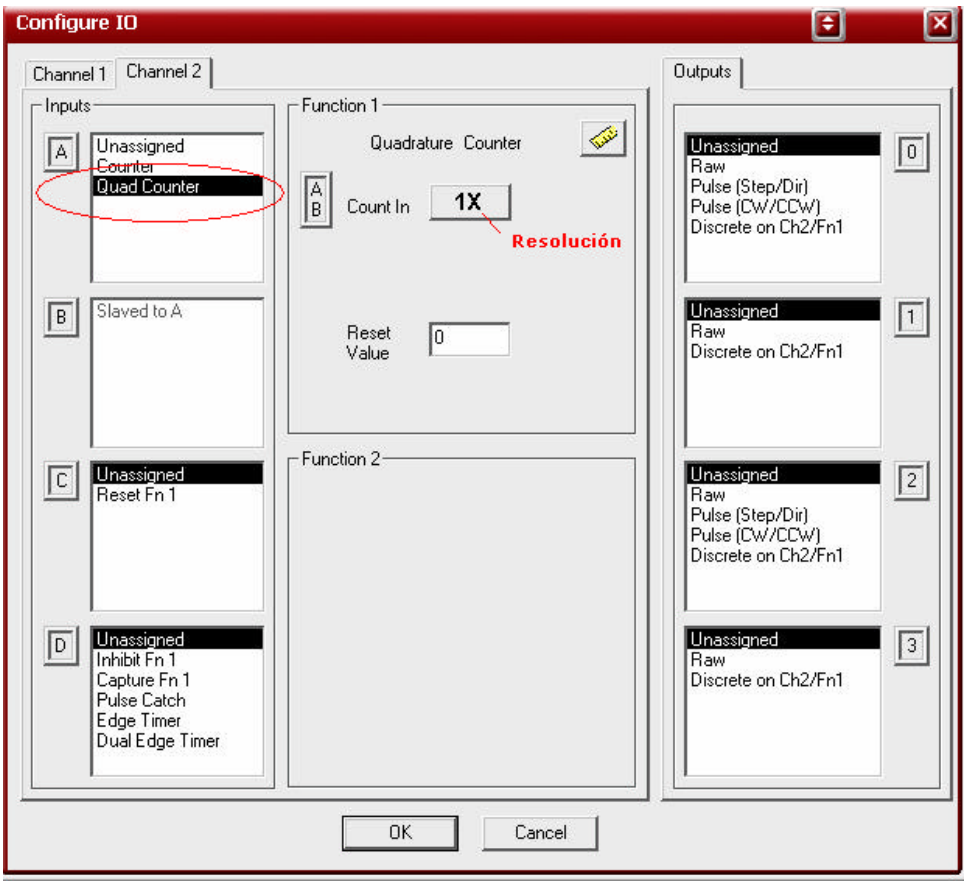


Figura. 4.7. Configuración de Entradas y Salidas

La configuración realizada es la siguiente:

- En el canal 2, las entradas A y B están seleccionadas como Quad Counter, para el funcionamiento con el codificador incremental.

- La resolución es de 2500 pulsos, por tanto, 1X.

- El valor inicial de reset es cero 0.

4.2.1.4 Módulos I/O

Para la realización del proyecto se necesitaron dos módulos:

D2-08ND3: Módulo de 8 Entradas discretas tipo relay, para recibir las señales de los sensores fines de carrera en el tubo del cañón (total 4), entradas para emergencia y arranque, y 2 entradas libres para que en un futuro se pueda ampliar las utilidades del prototipo.

Figura. 4.8. Módulo de Entradas discretas D2-08ND3

Configuración: Las entradas físicas del Controlador están configuradas de la siguiente manera:

I-0	POWER (Señal de Inicio del Sistema)
I-1	OFF_EMERG (Señal de apagado de las fuentes de los motores)
I-2	AUTO/MANUAL (Modo de trabajo)
I-3	SUPERIOR (Señal del sensor fin de carrera superior del tubo)
I-4	INFERIOR (Señal del sensor fin de carrera inferior del tubo)
I-5	DERECHA (Señal del sensor fin de carrera derecho del tubo)
I-6	IZQUIERDA (Señal del sensor fin de carrera izquierdo del tubo)

Dicha configuración esta dada en la herramienta IOView del software y se presenta como se muestra en la figura 4.9.

	I/O Device Description	Data Type	Logical ID	Tagname	Physical I/O
1	WinPLC_Slot1i_B0	Input	I-0	POWER	WinPLC_Slot1i_B0
2	WinPLC_Slot1i_B1	Input	I-1	OFF_EMERG	WinPLC_Slot1i_B1
3	WinPLC_Slot1i_B2	Input	I-2	AUTO/MANUAL	WinPLC_Slot1i_B2
4	WinPLC_Slot1i_B3	Input	I-3	SUPERIOR	WinPLC_Slot1i_B3
5	WinPLC_Slot1i_B4	Input	I-4	INFERIOR	WinPLC_Slot1i_B4
6	WinPLC_Slot1i_B5	Input	I-5	IZQUIERDA	WinPLC_Slot1i_B5
7	WinPLC_Slot1i_B6	Input	I-6	DERECHA	WinPLC_Slot1i_B6
8	WinPLC_Slot1i_B7	Input			WinPLC_Slot1i_B7

Figura. 4.9. Configuración de Entradas discretas en IOView

D2 – 08TD1: Módulo de 8 Salidas discretas para señales de seguridad, de activación de relays de ahorro de energía, ampliaciones, mejoras y cambios que se puedan dar en futuros proyectos.

Figura. 4.10. Módulo de Salidas discretas D2-08TD1

Configuración: Las salidas físicas del Controlador están configuradas de la siguiente manera:

O-0 POWER (Señal de Inicio del Sistema)

O-1 APAGADO_EMERG (Señal de apagado de las fuentes de los motores)

O-2 AUTO/MANUAL (Modo de trabajo)

O-3 POSICIONADO (Señal indicadora de la finalización del posicionamiento)

O-4	TRANSMISIÓN (Señal indicadora de transmisión)
O-6	SV1(Señal de conexión para contactor de la fuente del servomotor en elevación)
O-7	SV2 (Señal de conexión para contactor de la fuente del servomotor en dirección)

Dicha configuración esta dada en la herramienta IOView del software y se presenta como se muestra en la figura 4.11.

	I/O Device Description	Data Type	Logical ID	Tagname	Physical I/O
1	WinPLC_Slot0o_B0	Output	0-0	POWER	WinPLC_Slot0o_B0
2	WinPLC_Slot0o_B1	Output	0-1	APAGADO_EMERGENCIA	WinPLC_Slot0o_B1
3	WinPLC_Slot0o_B2	Output	0-2	AUTO/MANUAL	WinPLC_Slot0o_B2
4	WinPLC_Slot0o_B3	Output	0-3	POSICIONADO	WinPLC_Slot0o_B3
5	WinPLC_Slot0o_B4	Output	0-4	TRANSMICION	WinPLC_Slot0o_B4
6	WinPLC_Slot0o_B5	Output			WinPLC_Slot0o_B5
7	WinPLC_Slot0o_B6	Output	0-13	Contactor1	WinPLC_Slot0o_B6
8	WinPLC_Slot0o_B7	Output	0-14	Contactor2	WinPLC_Slot0o_B7

Figura. 4.11. Configuración de Salidas en IOView

4.2.1.5 Base y fuente de alimentación

El PLC DL205 posee una base para la inserción de los módulos con fuente de alimentación incluida, existen varios tipos y tamaños dependiendo el número de módulos que se vayan a utilizar y el tipo de alimentación que se requiera.

Tomando en cuenta que la alimentación los sensores del Sistema de Puntería Digital, los fines de carrera y PLC toman su energía de las baterías que tienen el tanque, la cual es de máximo 48 VDC, 12A, y que se tendrán como máximo 6 módulos, se escogió la **base D2-O6BDC1-1** que cuenta con una fuente de de 12 - 24VDC y permite hasta 6 módulos en su base, entregando la suficiente energía a cada uno para su correcto funcionamiento.

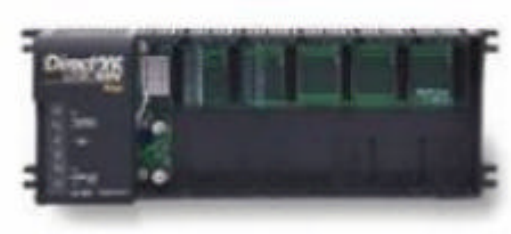


Figura. 4.12. Base y Fuente de Alimentación D2-06BDC1-1

4.2.2 Servomotores

Los motores utilizados son servomotores, debido al excelente rendimiento que brindan, analizando para su dimensionamiento el torque y velocidad máximos permitidos, tanto para deflexión como para elevación, garantizando con esto, que el sistema posicione el tubo cañón rápida y efectivamente, sin causar mayor esfuerzo al motor y por ende lograr un menor consumo de energía.

Las características iniciales requeridas fueron las siguientes:

- Torque Mínimo Continuo de 3 N-m, medido con un torquímetro en cada volante, tanto en elevación, como en dirección.

- Alimentación Máxima de 48 VDC, ya que los cañones autopropulsados de 155mm poseen 4 baterías de 12 V c/u.

- Interfase de comunicación compatible con el PLC.

Fueron escogidos los servomotores marca Animatics SmartMotor SM3420D y SM3450D por acercarse más a los requerimientos que otros fabricantes ofrecen en el mercado, además que poseen un controlador incorporado en su estructura, que da mayor robustez y facilidad de montaje de los motores a la estructura del tanque.

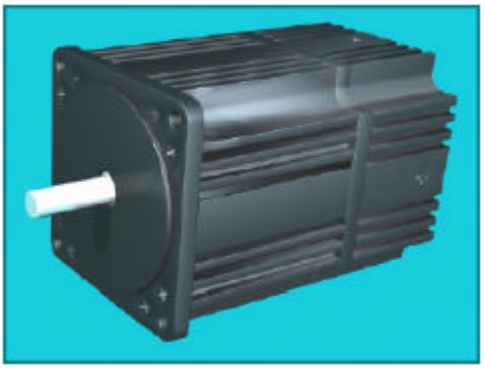


Figura. 4.13. Servomotores Animatics Smartmotors

Las características generales de los motores se presentan en la siguiente tabla:

Especificaciones	**SM3420**	**SM3450**
Torque Continuo (N.m)	0.71	1.77
Torque Pico (N.m)	3.81	5.30
Potencia Nominal (KW)	0.18	0.27
Velocidad (R.P.M)	Ver curvas torque vs. Velocidad Fig. 4.14,4.15	
Diámetro del eje (mm)	9.53	9.53
Peso (kg.)	1.59	2.95
Tamaño	105mm (largo) x 82.6 (ancho) x 82.6(altura)	155mm (largo) x 82.6 (ancho) x 82.6(altura)
Protocolo de Comunicación	Serial RS – 232	
Alimentación	24 VDC - 48 VDC	

Tabla. 4.3. Características de los Servomotores SM3450, SM3420

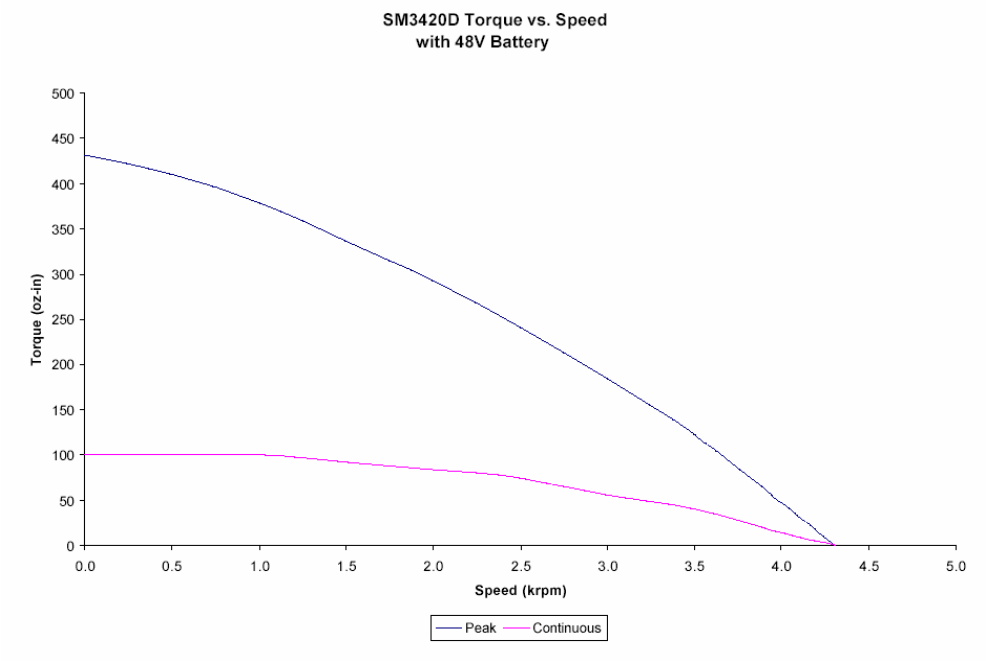


Figura. 4.14. Curvas torque/velocidad del motor SM3420

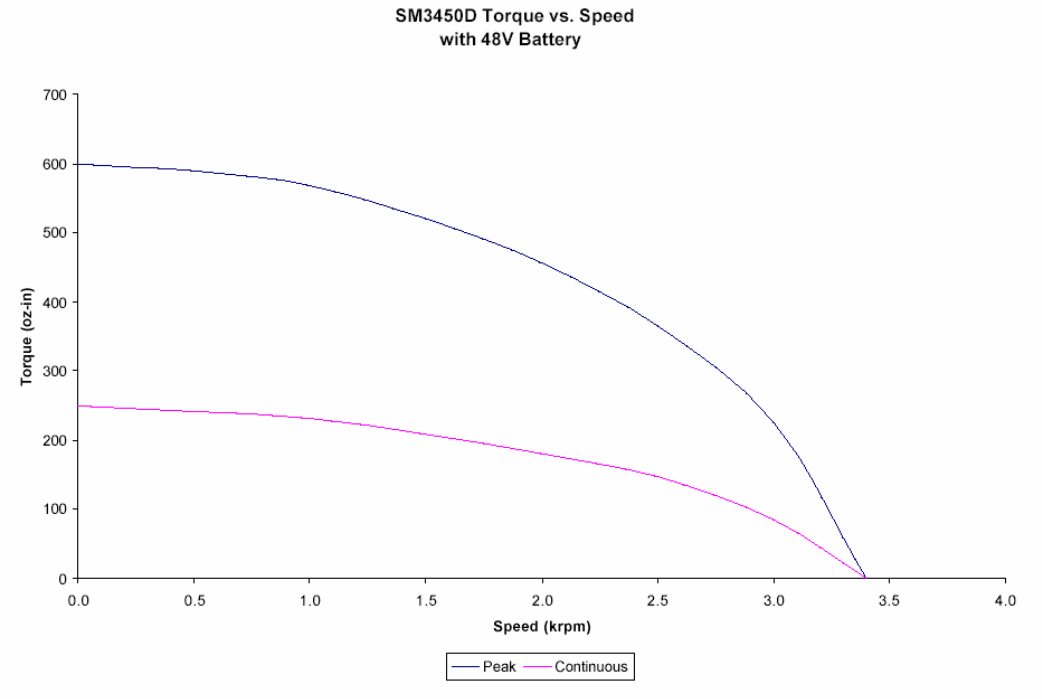


Figura. 4.15. Curvas torque/velocidad del motor SM3450

Se utilizó contactores en la fuente de alimentación de los servomotores para evitar que estén energizados innecesariamente, y se produzca un desgaste de energía. Estos contactores los que serán activados desde una salida del PLC, para que dejen pasar la energía de las fuentes de alimentación, por lo mismo estos deben soportar las siguientes condiciones mínimas:

Entrada:	12 Vdc con un máximo de 200 mA
Contacto:	48 Vdc con un mínimo de 6A

4.2.2.1 Modo de Programación.

El control de servomotores y su programación se realiza en una memoria interna incorporada, para lo cual se tiene dos conectores I/O, uno de 15 pines y otro de 7 pines para el puerto Serial.

Para la programación se debe enviar una secuencia por el puerto en la cual se indica el modo de trabajo y otras variables a tomarse en cuenta.

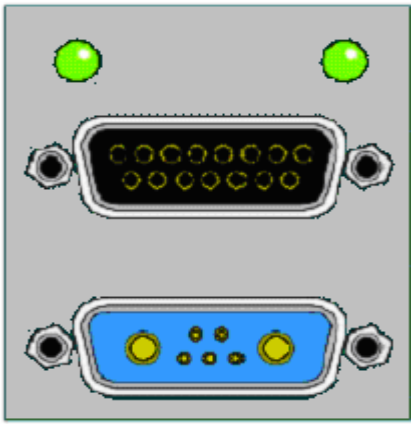


Figura. 4.16. Conectores ubicados en los Servomotores.

Para el proyecto se transmite una secuencia de modo Velocidad, donde sus parámetro principales, son la Velocidad y Aceleración.

4.2.3 Codificador

Los codificadores o encoders, denominados también transductores de posición, están encargados de convertir una señal física de posición en una señal eléctrica.

Se pueden clasificar en dos grupos principales: Analógicos y Digitales.

Dentro de los analógicos se encuentran los potenciómetros, sincros, resolver, etc.

Los transductores de posición digitales o numéricos convierten posiciones angulares y desplazamientos lineales en una representación numérica; éstos se subdividen en dos clases:

a) Transductores de posición numéricos del tipo absoluto llamados codificadores absolutos. En ellos está codificado (en código Gray, BCD o binario) cada uno de los incrementos de posición, a partir de una referencia que es cero, de modo que la posición puede ser determinada directamente.

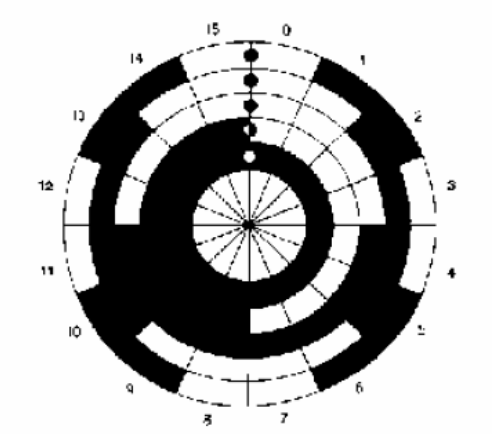


Figura. 4.17. Disco codificado en código Gray

b) Transductores de posición numéricos del tipo incremental, llamados codificadores incrementales. En estos no se tiene una referencia para todos los puntos, sino que cada posición constituye un origen para el siguiente punto. El principio de funcionamiento de los codificadores incrementales se basa en proporcionar en su

salida una serie de pulsos, donde cada pulso corresponde a un desplazamiento mecánico, que puede ser de un disco o una varilla, para desplazamientos angulares o lineales respectivamente. Tanto el disco como la varilla están divididos en sectores equidistantes y girando delante de un dispositivo de lectura fijo el cual produce una señal eléctrica en correspondencia con cada sector. La figura 4.16 muestra los elementos móviles, angulares y lineales del codificador incremental.

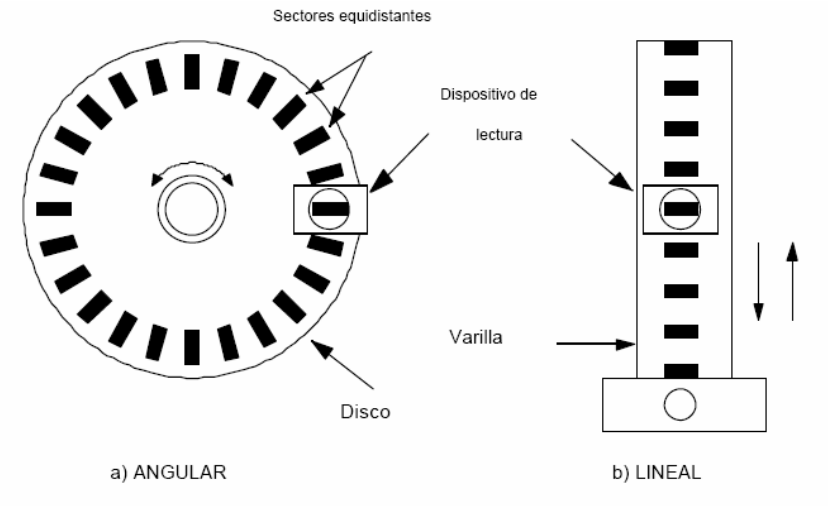


Figura 4.18. Codificador Incremental Angular y Lineal

Para garantizar el correcto posicionamiento del tubo del cañón, es fundamental colocar un codificador en la parte inferior del eje del cañón mediante un acople mecánico en forma de arco para medir el desplazamiento en dirección del tubo. Se utiliza el codificador incremental estándar TRD-N2500-RZWD, por cumplir características de construcción IP65, tamaño, costos y resolución de 2500 pulsos por vuelta, según cálculos teóricos con el acople el codificador entrega de 17 a 20 pulsos por cada milésima, lo que garantiza tener una resolución aceptable.

A continuación se presentan los cálculos realizados.

Longitud de arco (L_a) : 180 mm

Radio del eje del codificador (d_{EC}): 4mm

$Perímetro_{EC}$ = 2 p*r_{EC} = 2.(3.1416) . 4 = 25,133*

 # vueltas totales Codificador = L_a / $Perímetro_{EC}$ = 180 / 25,133 = 7,1619

 # pulsos Total = 2500 . 7,1619 = 17905 pulsos en 800 milésimas

 # pulsos/milésima = 17905 / 800 = 22.38 pulso /milésima

Figura. 4.19. Codificador Incremental TRN – 2500RMZ

Se presentan las características del codificador adquirido para el proyecto en la siguiente tabla:

Especificaciones	**TRD-N2500-RZWD**
Voltaje de Alimentación	5 – 30 VDC
Corriente	60mA máxima
Forma de Onda	Cuadrada ternaria
Frecuencia Máxima de Respuesta	100 KHz.
Tiempo máximo de Respuesta	10 µseg
Peso	250g
Condiciones Ambientales Temperatura de Ambiente: Temperatura de Almacenamiento: Humedad:	 10 – 70°C -25 – 85°C 35 – 85%

Tabla. 4.4. Especificaciones del Codificador Incremental TRD-N25OO-RZWD

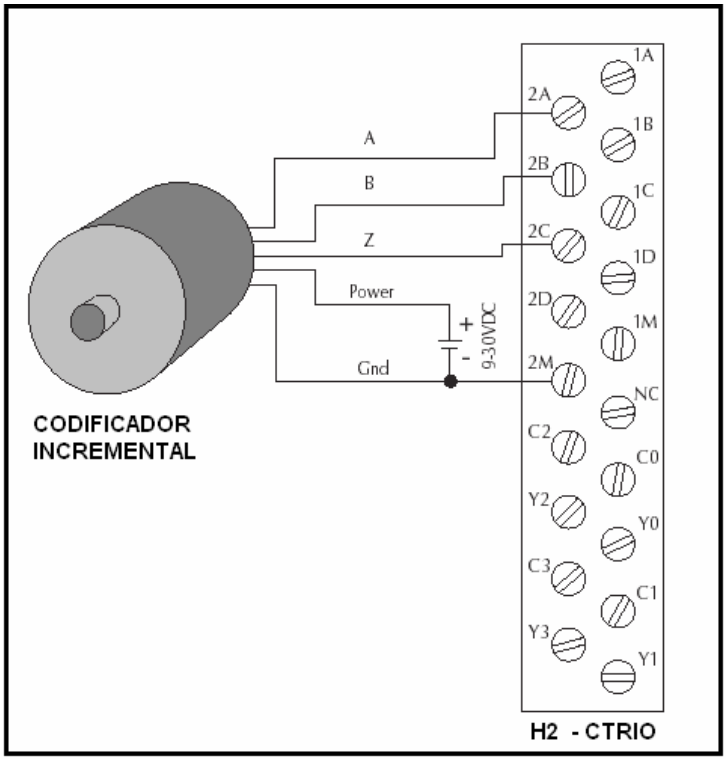


Figura. 4.20. Conexión del Codificador con H2-CTRIO

4.2.4 Radio Modems

Los radio módems están destinados a aplicaciones en las que es necesario transmitir la señal vía radio, como por ejemplo interconexión de ordenadores a través de LAN o MAN inalámbricas, envío y recepción de mensajes, telemetría, etc.

Se realizó una selección del Radio MODEM a utilizarse, tomando en cuenta que debe cumplir los siguientes requerimientos:

- Topología Multipunto.
- Distancia de transmisión en línea de vista mayor a 300m
- Interfase de comunicación Serial RS-232.
- Voltaje máximo de alimentación 48V.

Tomando en cuenta estos parámetros se selecciono el Radio Modem marca XStream – PKG **X09-019PKI-RA**, se muestra en la tabla 4.5 sus especificaciones relevantes.

Figura. 4.21. Radio Módem X09-019PKI-RA

Especificaciones	**X09-019PKI-RA**
Radio Frecuencia	902 – 928 MHz
Control de Frecuencia	FM directo
Topología de Red	Punto – punto, punto – multipunto, multi – drop
Interfase de comunicación	Serial RS 232/485/422
Velocidad de Transmisión	1200 – 57600 baudios
Voltaje de Alimentación	5 – 30 VDC
Corriente	60mA máxima
Forma de Onda	Cuadrada ternaria
Frecuencia Máxima de Respuesta	100 KHz.
Peso	250g
Condiciones Ambientales Temperatura de Ambiente: Temperatura de Almacenamiento: Humedad:	 10 – 70°C -25 – 85°C 35 – 85%

Tabla. 4.5. Especificaciones del Radio MODEM X09-019PKI-RA.

4.2.4.1 Configuración

Los radios módems están configurados para trabajar en una topología Maestro – Esclavo, donde el Maestro o Base "B" está ubicado en el CDT con una velocidad de transmisión de 19200 baudios y los esclavos o Remotos "Rn" en cada cañón con una velocidad menor, en este caso 9600 baudios.

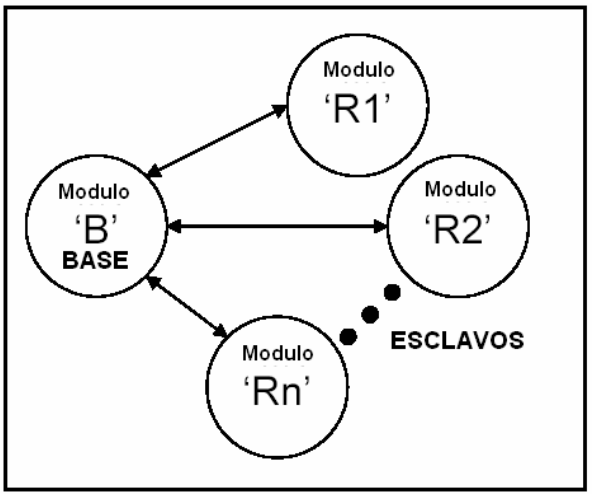


Figura. 4.22. Configuración de Radio Modems

El módulo Base puede establecer una comunicación bidireccional con todos los módulos esclavos, mientras estos solamente con la base, y no entre ellos. Se puede configurar el número de intentos de transmisión, una vez que la información sea recibida se eliminan los demás intentos. Para configurar los radios es necesario enviar los siguientes parámetros:

Module Parameters					
Command	'B'	'R1'	'R2'	. . .	'Rn'
ATDT	0x7FFF	0x8001	0x8002	. . .	0x800n
ATMK	0x8000	0x7FFF	0x7FFF	. . .	0x7FFF
ATRR	3	3	3	. . .	3

Figura. 4.22. Parámetros de Configuración

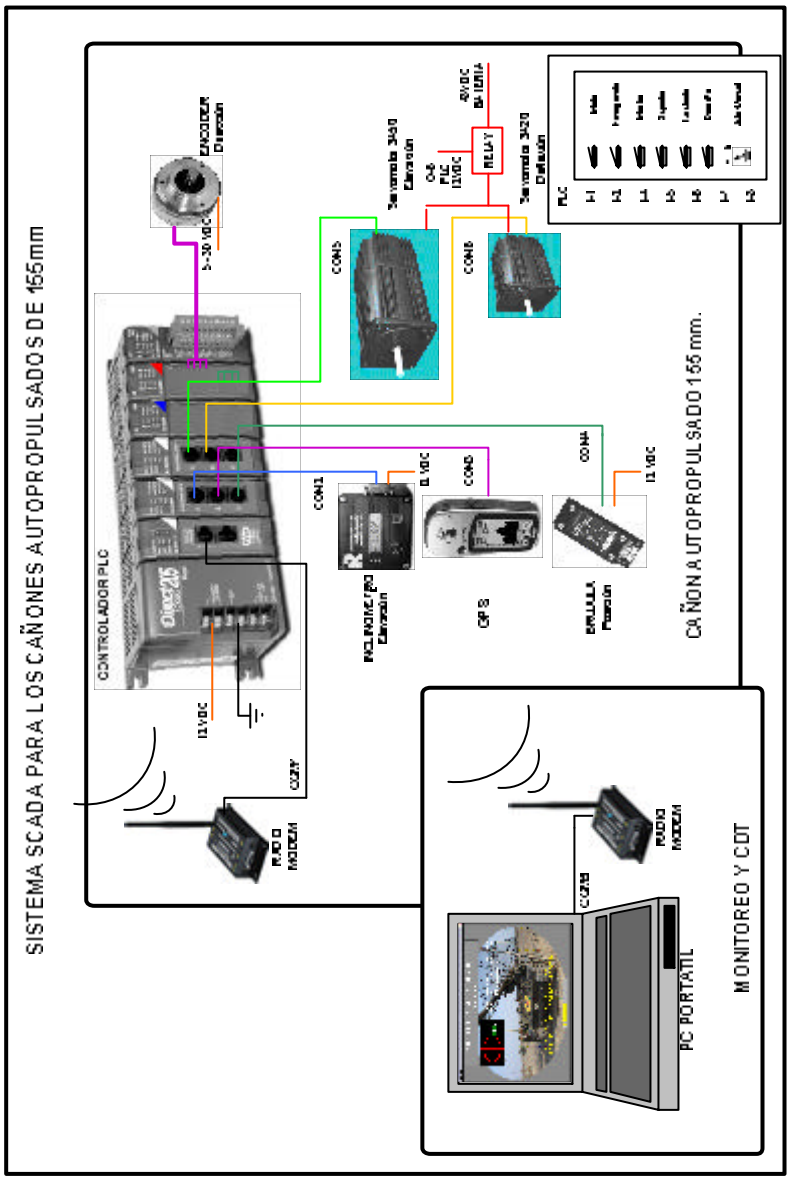
Notar que el parámetro ATDT del modulo base acepta como dirección global a todos los módulos remotos y ATRR es el número de retransmisiones del paquete, cuando un módulo está transmitiendo a la base los demás esclavos son ignorados.

Se recalca que todo el hardware adquirido es para la realización de un prototipo y su selección se basó según las necesidades técnicas del proyecto más relevantes y el bajo costo en el mercado.

4.3 MONTAJE Y CONEXIÓN

4.3.1. Conexión total del sistema

LOS SERVOMOTORES TIENEN EL MISMO COM Y FALTAN LOS CONTACTORES PARA ACCIONAMIENTO DE LOS SERVOMOTORES



4.3.2 Montaje

Para montaje de los equipos seleccionados en el cañón se tomo en cuenta el lugar de ubicación y la necesidad de plataformas de protección y acople para los mismos.

- Para el controlador y radio módem se diseño un armario de protección, para tratar de eliminar el máximo los efectos del medio ambiente como agua, polvo y otros agentes externos adicionales, que causen daños en los equipos.

- Los motores necesitaron de un acople mecánico a base de piñones y coronas para ser adaptados al sistema de engranajes propios del cañón, tanto para deflexión y elevación.

Los acoples para ambos motores están construidos a base de acero de segmentación con las siguientes relaciones de transmisión:

Elevación:

Relación	144/23
R=	18.75 cm.

Dirección:

Relación	146/23
R=	18.75 cm.

Figura. 4.24. Acoples para el motor de elevación

Figura. 4.25. Acoples para el motor de deflexión

- El codificador esta ubicado en la parte inferior posterior del cañón, acoplado al eje del tubo mediante un resorte, y se desliza sobre un semiarco de fibra.

Figura. 4.26. Acoples para el codificador

CAPITULO V

PRUEBAS DE CAMPO Y RESULTADOS

5.1 DESCRIPCIÓN DE FUNCIONALIDAD Y OPERATIVIDAD

Una vez realizado el diseño e implementación del Sistema SCADA para cañones Autopropulsados de 155mm, se puede garantizar que se cumple con las condiciones propuestas de funcionalidad y operatividad.

En el transcurso del desarrollo del proyecto se realizaron varias pruebas de campo y laboratorio para analizar las distintas fases del proyecto, por lo que se estructuro a esta capitulo de la siguiente manera:

- Pruebas de Artillería
- Pruebas de Equipos
- Pruebas del Sistema Total

5.1.1. Pruebas de Artillería

Desde los inicios del proyecto fue fundamental, comprender y analizar los distintos procedimientos de tiro y cálculo del CDT, por lo que, se realizaron viajes para adquirir conocimientos de artillería básicos relacionados con el proyecto.

5.1.1.1. Pruebas Teóricas de CDT

Los Cañones Autopropulsados de 155mm pertenecen a la Brigada Blindada Galápagos GAAP – 11, ubicada en la Provincia Chimborazo, ciudad de Riobamba, lo que obligó a realizar un visita técnica a dicho destacamento, para recabar información de los Cañones Autopropulsados, procedimientos de tiro y Cálculos del CDT.

La visita tuvo una duración de tres días, en los cuales se realizaron las siguientes prácticas:

Medición de Torque: Se tomaron medidas del torque para las manivelas en dirección y deflexión del tubo, en las que se obtuvo los siguientes resultados:

Torque máximo en elevación: 16 N.m

Torque máximo en dirección: 8 N.m

Cálculos del CDT: Se adquirieron conocimientos de CDT, gracias a un curso dictado en la brigada, con lo que se pudo realizar pruebas del Software SPD – 155 y realizar actualizaciones necesarias, las mismas que son especificadas en el Capítulo 1 de este documento. Además se realizaron los siguientes ejercicios para análisis y comprobación de los procesos de cálculos y utilización de las tablas de tiro.

Ejercicio 5.1

COORDENADAS	**ESTE**		**NORTE**
Batería:	24390	-	14650
Observador:	27610	-	23800

Azimut: 4400 milésimas
DGT: 230 milésimas
Distancia: 1450 alargar

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
8798	210	352	6

Tabla. 5.1. Valores resultantes para el primer ejercicio

Ejercicio 5.2

COORDENADAS	**ESTE**		**NORTE**
Batería:	29500	-	18500
Observador:	21550	-	25850

Azimut: 4030 milésimas
DGT: 5390 milésimas
Distancia: 1950 alargar

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
11132	5370	378	7

Tabla. 5.2. Valores resultantes para el primer cálculo del segundo ejercicio

Corrección:

Derecha: 200
Acortar: 100

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
11303	5383	388	7

Tabla. 5.3. Valores resultantes para la primera corrección del segundo ejercicio

Corrección:

Izquierda: 100
Acortar: 50

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
11194	5386	381	7

Tabla. 5.4. Valores resultantes para la segunda corrección del segundo ejercicio

COORDENADAS BALÍSTICAS PR-1 : ESTE 20172 NORTE 24688

Transporte desde PR-1

Azimut: 5820
Alargar: 3100

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
14086	5474	411	6

Tabla. 5.5. Valores resultantes para la primera corrección con transporte del segundo ejercicio

Ejercicio 5.3

COORDENADAS	**ESTE**		**NORTE**	**ALTURA**
Batería:	57197	-	17302	30
Observador:	59120	-	26150	120

Azimut: 4339 milésimas
DGT: 6400 milésimas
Distancia: 4400 alargar
Corrección por altura: 2 milésimas

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
7216	6103	259	6

Tabla. 5.6. Valores resultantes para el primer cálculo del tercer ejercicio

Corrección:

Derecha: 200
Alargar: 200

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
7384	6070	268	6

Tabla. 5.7. Valores resultantes para la primera corrección del tercer ejercicio

Corrección:

Izquierda: 200
Acortar: 100

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
7201	6088	258	6

Tabla. 5.8. Valores resultantes para la segunda corrección del tercer ejercicio

Corrección:

Derecha: 100
Alargar: 400

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
7250	6030	261	6

Tabla. 5.9. Valores resultantes para la tercera corrección del tercer ejercicio

Corrección:

Izquierda: 50
Acortar: 50

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
7205	6038	258	6

Tabla. 5.10. Valores resultantes para la cuarta corrección del tercer ejercicio

Corrección:

Derecha: 50

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
7255	6037	261	6

Tabla. 5.11. Valores resultantes para la quinta corrección del tercer ejercicio

Corrección:

Acortar: 50

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
7260	6044	261	6

Tabla. 5.12. Valores resultantes para la sexta corrección del tercer ejercicio

COORDENADAS BALÍSTICAS PR-1 : ESTE 54760 NORTE 24141

Transporte desde PR-1

Azimut: 4916
Acortar: 300
Derecha: 3000

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
9949	6203	430	6

Tabla. 5.13. Valores resultantes para la primera corrección con transporte del tercer ejercicio

Corrección:

Izquierda: 200
Alargar: 100

Alcance	Deflexión Pieza	Angulo Elevación (- corrección altura)	CARGA
9788	6188	418	6

Tabla. 5.14. Valores resultantes para la segunda corrección con transporte del tercer ejercicio

Información: Además de recoger experiencias en tiros reales de Artilleros pertenecientes a la brigada, se consiguió las tablas completas de tiro, ya que en el proyecto anterior solo se contaba con las de carga 9 bis, lo que permitió ampliar y mejorar el cálculo del CDT en el Software.

5.1.1.2. Pruebas Prácticas de CDT y Tiro

Las Fuerzas Armadas en busca de estar en continuó movimiento y adquirir experiencias en tiro reales realizó una practica real con los cañones autopropulsados de 155mm en la localidad de Playas los días 21 al 26 de Noviembre del 2004.

El Centro de Investigaciones Tecnológicas del Ejercito (CICTE) aprovechó esta oportunidad para comprobar el funcionamiento del Software del Centro Director de Tiro (CDT) realizado en Labview del “SPD-155” con las modificaciones desarrolladas en el proyecto SCADA para los cañones autopropulsados de 155mm.

El grupo de artillería GAAP-11 para realizar un tiro real necesita realizar los cálculos de tiro por lo menos con dos sistemas diferentes para tener mayor confiabilidad y

evitar accidentes a poblaciones aledañas que se encuentran en el campo donde se realizan las pruebas. Los sistemas usados para el calculo de los valores de deflexión y elevación, consta de una calculadora HP, la cuál contien un programa especializado para el cañón autopropulsado de 155mm, y una computadora portátil 386 que tiene un programa desarrollado en el lenguaje de programación C++ .

En esta practica ocurrió un problema con el software usado en la computadora portátil perteneciente al GAAP-11, el cual no poseía la opción de poder ingresar la carga que iba a ser usada en el disparo, ya por su programación interna realizaba el calculo de la carga dependiendo de su distancia. Por tal motivo se eligió que el segundo sistema de cálculo, que iba a servir como comparación del cálculo manual realizado con la calculadora, era software desarrollado en el CICTE SPD-155, el cual tiene la ventaja de ingresar el tipo de carga a ser utilizado en el disparo.

En la figura 5.1 se observa el posicionamiento de la batería de ataque y el blanco. También se puede observar al OA (observador avanzado) el cual es un miembro especializado del ejercito el cual esta entrenado para dar una apreciación de la distancia del él hacia el blanco. El OA tiene gran influencia para poder llegar a un blanco efectivo ya que se depende directamente de su apreciación y buen criterio.

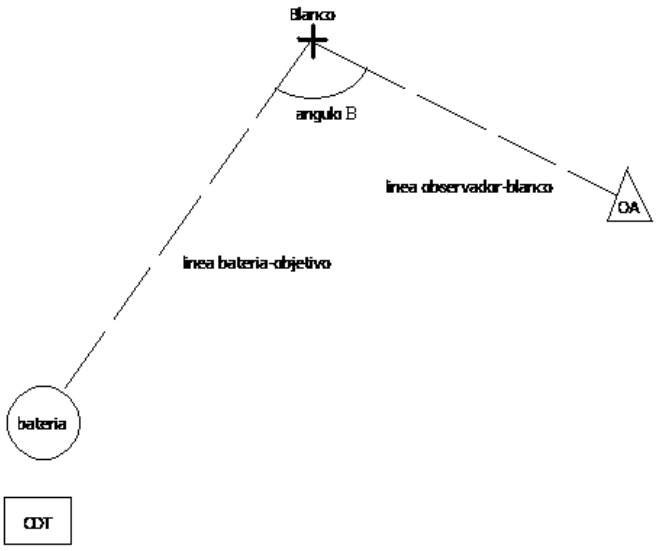


Figura. 5.1. Ubicación de la batería, blanco y del Observador Avanzado

En la semana de entrenamiento se realizó 9 disparos, los datos iniciales de tiro fueron:

Ubicación del Observador:

- Este: 55700
- Norte: 20181

Ubicación de la batería

- Este: 61384
- Norte: 12254

Dirección General de Tiro

- DGT: 5780

Tipo de Carga

- Carga: 6

Figura. 5.2. Pantalla de Software del CDT del primer disparo

Después de ingresar todos los valores iniciales en el programa del CDT desarrollado en el CICTE se obtuvo los siguientes valores para el primer disparo:

Deflexión	Elevación	Alcance
5755	416	9754

Tabla. 5.15. Valores resultantes para el primer disparo

Al realizar el primer disparo el Observador Avanzado informo que el tiro fue muy largo y que debían corregir 700 milésimas en el alcance y 170 milésimas hacia la izquierda en la dirección

Figura. 5.3. Pantalla de Software del CDT del segundo disparo

Después de ingresar los datos de corrección en el programa del CDT, se obtuvo los siguientes valores resultantes para el segundo disparo:

Deflexión	Elevación	Alcance
5693	387	9349

Tabla 5.16. Valores resultantes para el segundo disparo

En el segundo disparo el Observador Avanzado notifico que el tiro fue muy corto y que debían alargar en 150 milésimas

Figura. 5.4. Pantalla de Software del CDT del tercer disparo

Al cambiar el dato en el software por el que informo el OA, se consiguió los siguientes valores resultantes para el tercer disparo:

Deflexión	Elevación	Alcance
5739	411	9692

Tabla. 5.17. Valores resultantes para el tercer disparo

En el tercer disparo ya se logro un tiro efectivo por que los datos del observador avanzado fueron menores o igual a 100 milésimas, pero el Comandante a cargo y por motivo de practica mando la orden de hacer otra corrección por lo que el observador avanzado dispuso acortar 100 milésimas y corregir 50 milésimas a la izquierda

Figura. 5.5. Pantalla de Software del CDT del cuarto disparo

Después de ingresar los datos del OA en el programa del CDT, se obtuvo los siguientes valores resultantes para el primer disparo:

Deflexión	Elevación	Alcance
5755	416	9754

Tabla. 5.18. Valores resultantes para el cuarto disparo

Después del cuarto disparo se realizó el primer tiro de efecto para destruir completamente al primer objetivo. En el programa se procedió a realizar el transporte de

las coordenadas balísticas para tener registrado en una base de datos para informar a oficiales superiores las coordenadas exactas donde se encontraba el enemigo.

Con las coordenadas balísticas el observador avanzado realizó las siguientes correcciones: acortar 150 milésimas y corregir 50 milésimas a la derecha

Figura. 5.6. Pantalla de Software del CDT del quinto disparo

Trasladando las coordenadas balísticas y las correcciones del observador avanzado en el software del CDT, se obtuvo los siguientes valores resultantes para el primer disparo:

Deflexión	Elevación	Alcance
5706	414	9734

Tabla. 5.19. Valores resultantes para el quinto disparo

Con estas correcciones se realizó un tiro efectivo en el segundo blanco, que igualmente que el primer blanco el mayor mando a realizar otra corrección para observar la

habilidad del equipo que esta a cargo de los cañones autopropulsados de 155mm. El Observador avanzado envía su apreciación y manda a corregir 50 milésimas hacia la izquierda.

Figura. 5.7. Pantalla de Software del CDT del sexto disparo

Después de ingresar los datos del OA en el programa del CDT, se obtuvo los siguientes valores resultantes para el sexto disparo:

Deflexión	Elevación	Alcance
5707	420	9817

Tabla. 5.20. Valores resultantes para el sexto disparo

Luego del sexto disparo se realizaron dos disparos adicionales pero los cuales tuvieron una mínima modificación los cuales solo se realizaron por motivo de practica. En la Figura 5.7 se acorto 50 milésimas a la izquierda y en la Figura 5.8 se acorto 50 milésimas, luego de estos disparos se realizó un segundo tiro de efecto y operador el programa de CDT procedió a guardar los datos del segundo blanco.

Figura. 5.8. Pantalla de Software del CDT del septimo disparo

Figura. 5.9. Pantalla de Software del CDT del octavo disparo

Luego de haber realizado esta practica se comprobó que el programa realizo el cálculo de los datos de tiro en forma exacta y con mayores prestaciones de velocidad en comparación con el sistema de las calculadoras.

5.1.2. Pruebas del Equipo

Una vez realizadas las pruebas de cálculo del CDT, se realizaron pruebas con el hardware, para garantizar la efectividad y confiabilidad del equipo.

5.1.2.1. Motores

Al comienzo del proyecto se planteó la necesidad de buscar que tipo de motores se adapten mejor a los requerimientos del sistema brindando confiabilidad y eficiencia. De acuerdo con los parámetros de torque y alimentación necesarios en el posicionamiento del cañón, se adquirió motores de Pasos Sanyo Denki, por contar con una torque constante mayor que otros motores en el mercado, y cumplir características de alimentación y control, aunque no se tomó en cuenta la velocidad de los mismos. Se presentan a continuación sus características más relevantes:

El Sistema de Pasos Sanyo Denki es controlable mediante el módulo del WinPLC DL205 H2-CTRIO y esta compuesto por:

Motores de pasos Sanyo Denki NEMA 34 (103H8222-0441): motores Unipolares de Imán Permanente.

- **Drive compacto (PMM-MD-23220-10):** Trabaja con los motores y el controlador, del cual recibe un comando de posición en formato “pulso y dirección” y lo convierte en señales para el movimiento de los motores de pasos. Cuenta con una tecnología de micropasos para incrementar la resolución que puede configurarse mediante 5 dip switchs para 200, 400, 800 y 1600 pasos por revolución.

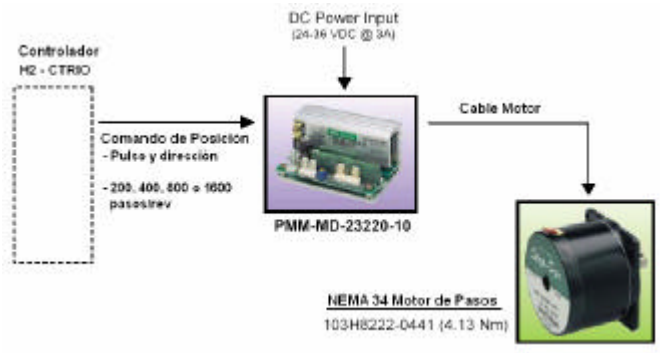


Figura. 5.10. Diagrama de bloques del Sistema de Pasos

Las especificaciones generales del sistema se presentan en la siguiente tabla

ESPECIFICACIONES DEL SISTEMA	
Velocidad Máxima	Ver Fig. 4.14 Curvas torque/velocidad
Torque Continuo	4.13 N. m
MOTOR	
Corriente (A/ fase)	2.0
Resistencia (ohms/fase)	4.0
Inductancia (mH/fase)	15.0
Inercia del Rotor	0.00029 kg – m^2
Peso	2.5 kg
Tamaño	92.2mm (largo) x 85.85mm(ancho) x 85.85mm (altura)
Diámetro del eje	12.0 mm.
Condiciones Ambientales Temperatura de Operación: Humedad:	 0 – 40°C > 90%
DRIVE	
Fuente de alimentación	24 – 36 VDC ±10%, 3ª.
Resistencia a la Vibración	4.9 m/s^2
Peso	0.15 kg
Tamaño	42mm (profundidad)x 53 mm (ancho) x 86mm (altura)

Condiciones Ambientales	
Temp. de Operación	0 – 50°C
Temp. de Almacenamiento	- 20 – 70°C
Humedad:	35 – 85%

Tabla. 5.21. Especificaciones del Sistema de Pasos Sanyo Denki

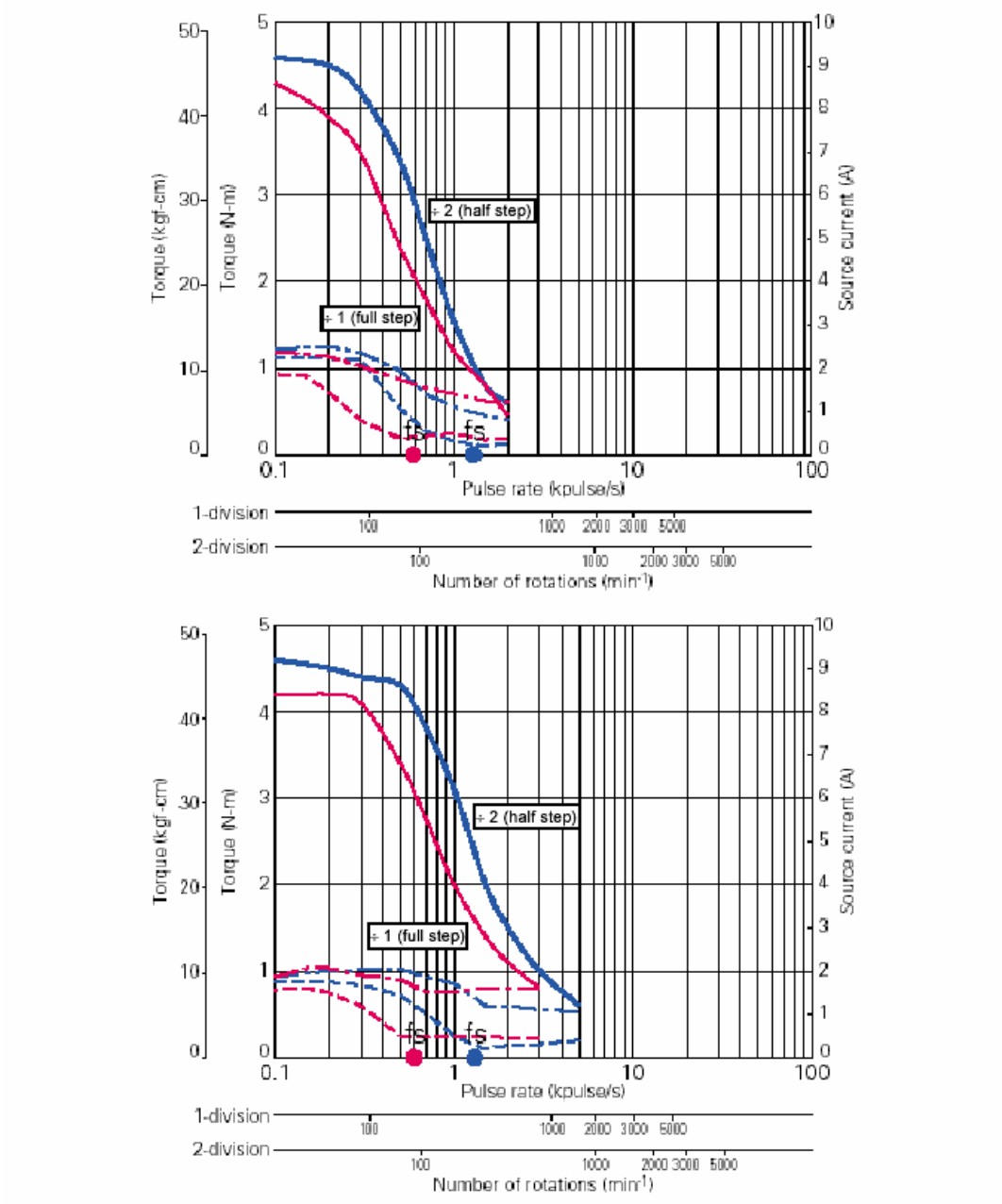


Figura. 5.11. Curvas torque/velocidad del Sistema de Pasos

Se realizaron pruebas con los motores de pasos y los acoples respectivos obteniéndose los siguientes datos:

Velocidad motor (RPM)	Tiempo total (seg)
60	42.0
80	31.5
100	25.2
120	21.0
140	18.0
160	15.8
180	14.0
200	12.6
220	11.5
240	10.5
260	9.7
280	9.0
300	**ERROR**

Tabla. 5.22. Tiempo de demora del motor de Pasos en deflexión

Velocidad motor (RPM)	Tiempo total (seg)
60	42.7
70	36.6
80	32.0
90	28.4
100	25.6
110	23.3
120	21.3
130	19.7
140	18.3
150	17.1
160	**ERROR**

Tabla. 5.23. Tiempo de demora del motor de Pasos en elevación

Como se puede observar en las tablas, los motores no pueden mover el tubo a una velocidad máxima de 280 R.P.M en deflexión y 160 R.P.M en elevación, lo que hace la respuesta del sistema muy lenta. Por lo tanto con este tipo de motores no se logra optimizar el proceso de disparo.

Otra opción que se analizó para el movimiento del tubo cañón, consiste en un sistema accionado por servomotores. Este tipo de motores, aunque poseen un menor torque continuo, tienen mejor rendimiento ya que pueden alcanzar mayor velocidad sin perder su torque. Se realizaron las mismas pruebas, que con los motores a pasos obteniéndose los siguientes resultados.

Velocidad motor (RPM)	Tiempo total (seg)
60	42.0
120	21.0
180	14.0
240	10.5
300	8.4
360	7.0
480	5.3
540	4.7
600	**4.2**

Tabla. 5.24. Tiempo de demora del Servomotor en deflexión

Velocidad motor (RPM)	Tiempo total (seg)
60	42.7
120	21.3
180	14.2
240	10.7
300	8.5
360	7.1
480	5.3
540	4.7
600	4.3

Tabla. 5.25. Tiempo de demora del Servomotor en elevación

Según tablas se demuestra que los servomotores son los más aptos para utilizarlos en este tipo de sistemas de posicionamiento, aunque en sus curvas de respuesta, muestra una linealidad en el torque con una velocidad de hasta 1000 rpm, se trabajó con una velocidad máxima de posicionamiento de 600 rpm en deflexión y 480 rpm en elevación, para evitar forzar a los motores al mismo tiempo que se ahorra energía.

5.1.2.2. Inclinómetro

Al realizar las pruebas de posicionamiento del tubo cañón en elevación se llegó a la conclusión que el sistema óptico tiene un error aproximado de dos milésimas comparado con el sistema de medición digital por causa del desgaste mecánico y la apreciación del observador. Una vez que el sistema digital fue compensado para lograr lecturas iguales al sistema óptico, el error en posicionamiento alcanzado es menor a 0.2 milésimas.

5.1.2.3. Codificador

Una vez instalado el codificador en la base del cañón fue indispensable tomar medidas para conocer el número de pulsos/milésima una vez ya acoplado el codificador a la base de giro del tubo cañón. El procedimiento realizado se basó en medir cuantos pulsos cuenta el codificador en el recorrido total de 800 milésimas.

	Número de pulsos en 800 milésimas	Pulsos / milésimas
	14401	18.00
	14389	17.99
	14387	17.98
	14387	17.98
	14387	17.98
	14387	17.98
	14389	17.99
	14387	17.98
	14387	17.98
	14387	17.98
	14393	17.99
	14390	17.99
	14393	17.99
	14390	17.99
	14399	18.00
	14400	18.00
PROMEDIO	14390.8	17.99

Tabla. 5.26. Medidas tomadas del codificador

5.1.3. Pruebas del Sistema

Una vez finalizada la implementación y pruebas de las distintas fases del proyecto, se realizó pruebas finales de todo el sistema, aplicando el ejercicio de tiro realizado en las prácticas en Playas.

Por esta razón, se lo ubicó al tanque con una DGT de 5780 milésimas y se realizó la corrección inicial de posicionamiento que comprende 300 mm en elevación y 150 milésimas en deflexión. Una vez posicionado se procedió a medir con los aparatos ópticos, goniómetro y anteojo panorámico la medida de corrección para tener una DGT exacta. Con esto el tubo cañón se encuentra posicionado para recibir datos del observador y realizar la correcciones necesarias luego del tiro. Los datos obtenidos de las pruebas de posicionamiento realizadas, se resumen en la siguiente tabla:

Valores Enviados desde Computador	Datos tomados mediante Ópticos	Datos tomados mediante sensores	ERROR SENSORES
300	302	302	2
40	41	40	0
300	300	300.4	0.4
5755	5756	5755.5	0.5
5693	5695	5693.4	0.4
5745	5747	5745.1	0.1
5625	5625	5625.5	0.5
5604	5603	5603.7	0.7
5790	5791	5790.4	0.4
5820	5820	5820.4	0.4
5875	5876	5875.5	0.5
5890	5892	5890.02	0.02
0	0	0.1	0.1
150	150	150.3	0.3
PROMEDIO			0.45

Tabla. 5.27. Valores tomados de pruebas del Sistema Total

Dados los resultados, se demuestra que el error en posicionamiento de tubo del cañón es menor a 1 milésima, y es totalmente independiente del personal de artillería, el cual está encargado de conectar el sistema y enviar datos desde el computador.

Se presenta en el Anexo 3, el manual de usuario para la utilización de el sistema completo.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

Una vez finalizado el proyecto se puede concluir lo siguiente:

- La automatización del Cañón Autopropulsado de 155mm utilizando un sistema de control tipo SCADA, es factible, puesto que los resultados obtenidos de las pruebas experimentales han ratificado la precisión de posicionamiento del tubo del cañón, llegando a obtener un error porcentual de una milésima con respecto a los equipos ópticos de puntería.
- El error mínimo del sistema que se pudo obtener fue de 0,2 milésimas, por resolución del sensor de elevación y dirección.
- Para lograr un error menor a 1 milésima es fundamental considerar la resolución de los sensores, así como el error producido en sus acoples mecánicos, para mediante cálculos compensar cualquier tipo de pérdida.

En caso del inclinómetro:

Resolución: 0.177 milésimas
Error del acople: 1 milésima

Codificador Incremental:

Resolución: 17.9 pulsos/milésima
Error del acople: ± 4 pulsos

Acoples de servomotores:

Resolución: 0.5 milésimas

- El tiempo de posicionamiento de la pieza, hasta la ejecución del primer tiro de prueba, se reduce tres veces en comparación al tiempo utilizando los métodos tradicionales.
- El tiempo de respuesta mínimo del sistema depende del tiempo de Procesamiento del PLC y el tiempo de respuesta de los sensores, por lo que,

PLC: 50ms
Codificador: 10µseg
Inclinómetro: < 0.5 seg

- Para solucionar conflictos de comunicación inalámbrica entre 2 equipos, en este caso el controlador (PLC) y el computador es necesario crear un protocolo de comunicación que sincronice la recepción y transmisión en ambos dispositivos.
- Es indispensable configurar los radio módems maestro – esclavo a velocidades 19200 baudios y 9600 baudios respectivamente.
- La programación del controlador y del software HMI, es fácil, debido a la amplia gama de herramientas brindadas por Think & Do y LabView, para este tipo de aplicaciones de control y comunicación.
- Para obtener mayor rendimiento de los servomotores se debe buscar la mejor relación de torque y velocidad, ya que de esta relación depende el ahorro de corriente.
- Los servomotores no pueden ser usados a mas potencia de la recomendada, 80% del valor nominal, ya que a un sobre esfuerzo provoca el aumento de amperaje lo que ocasiona daños permanentes en el motor y controlador.
- El PLC nos permite mayor robustez, confiabilidad y eficiencia en el sistema, ya que esta diseñado para cumplir condiciones extremas en la industria.
- El sistema de transmisión mecánica permite reducir el esfuerzo al motor, pero provoca reducciones en velocidad.
- La única forma de aumentar torque y velocidad en el sistema es mejorando las características técnicas del motor.

6.2 RECOMENDACIONES

- Implementar el Sistema SCADA en los cañones autopropulsados de 155mm por la confiabilidad y beneficios técnicos/tácticos que presenta.
- Se debe recalcar que el sistema desarrollado es un prototipo, en caso de implementación en todas las baterías se debe utilizar, cables, conectores y equipos que cumplan estándares militares.
- Para la construcción de acoples mecánicos es recomendable, realizar cálculos previos y utilizar un material robusto, para minimizar el error producido por estos elementos.
- Que el personal que sea designado para trabajar con este sistema, tenga una preparación y entrenamiento previo por expertos en el tema, para que puedan actuar según requerimientos e imprevistos que se pudieren presentar.
- El sistema diseñado e implementado brinda la posibilidad de actualizaciones y mejoras, por lo que se recomienda instalar en el cañón una interfaz propia de comunicación con el PLC.
- Adquirir un Goniómetro digital, para reemplazar la brújula y garantizar la exactitud y precisión de las medidas.
- Cuantificar la radiación de los radio modems de acuerdo a los estándares de la FCC (Federal Communications Commission) y la ICNIRP (Comisión Internacional de Protección contra la radiación no ionizante)

BIBLIOGRAFÍA

- **http://www.smartmotor.com/web/motors.html,** Servomotores 3450 y 3420.
- **http://entivity.custhelp.com/cgi-bin/entivity.cfg/php/enduser/std_alp.php,** Software de Programación para el controlador WinPLC.
- **http://www.hosteng.com/Main.htm**, Software CTRIO Workbench .
- **http://www.automationdirect.com,** Codificador Incremental, PLC, cables
- **http://www.motiononline.com/home.htm?welcome.htm&2,** Motores de Pasos.
- **http://www.ni.com,** Programación y herramientas de LabView.
- ORTIZ, Hugo, **Control de Procesos,** segunda edición, ESPE.
- **Conocimiento y Servicio del Obus de 155mm. Modelo F-3, autopropulsado,** curso de artillería.
- **Sistema de Puntería Digital para los cañones Autopropulsados de 155 mm,** Proyecto de Tesis.
- **Think & Do Live User's Guide,** Manual de usuario.
- **Wireless Radio Modems B &B Electronics,** Configuración

ANEXO 1

Cálculos de rangos de desplazamiento para el algoritmo de control.

MOTOR EN DEFLEXIÓN

	VELOCIDAD								
mil	1	2	3	4	5	6	7	8	RPS
	60	120	180	240	300	360	420	480	RPM
1	1.34	1.18	0.00	0.00	0.00	0.00	0.00	0.00	
2	2.34	1.68	0.00	0.00	0.00	0.00	0.00	0.00	
3	3.34	2.18	2.69	0.00	0.00	0.00	0.00	0.00	
4	4.34	2.68	3.03	0.00	0.00	0.00	0.00	0.00	
5	5.34	3.18	3.36	3.53	0.00	0.00	0.00	0.00	
6	6.34	3.68	3.69	3.78	0.00	0.00	0.00	0.00	
7	7.34	4.18	4.03	4.03	4.24	0.00	0.00	0.00	
8	8.34	4.68	4.36	4.28	4.44	0.00	0.00	0.00	
9	9.34	5.18	4.69	4.53	4.64	0.00	0.00	0.00	
10	10.34	5.68	5.03	4.78	4.84	5.05	0.00	0.00	
11	11.34	6.18	5.36	5.03	5.04	5.22	0.00	0.00	
12	12.34	6.68	5.69	5.28	5.24	5.39	0.00	0.00	
13	13.34	7.18	6.03	5.53	5.44	5.55	0.00	0.00	
14	14.34	7.68	6.36	5.78	5.64	5.72	5.99	0.00	
15	15.34	8.18	6.69	6.03	5.84	5.89	6.13	0.00	
16	16.34	8.68	7.03	6.28	6.04	6.05	6.28	0.00	
17	17.34	9.18	7.36	6.53	6.24	6.22	6.42	0.00	
18	18.34	9.68	7.69	6.78	6.44	6.39	6.56	0.00	
19	19.34	10.18	8.03	7.03	6.64	6.55	6.70	6.94	
20	20.34	10.68	8.36	7.28	6.84	6.72	6.85	7.07	
21	21.34	11.18	8.69	7.53	7.04	6.89	6.99	7.19	
22	22.34	11.68	9.03	7.78	7.24	7.05	7.13	7.32	
23	23.34	12.18	9.36	8.03	7.44	7.22	7.28	7.44	
24	24.34	12.68	9.69	8.28	7.64	7.39	7.42	7.57	
25	25.34	13.18	10.03	8.53	7.84	7.55	7.56	7.69	
26	26.34	13.68	10.36	8.78	8.04	7.72	7.70	7.82	
27	27.34	14.18	10.69	9.03	8.24	7.89	7.85	7.94	
28	28.34	14.68	11.03	9.28	8.44	8.05	7.99	8.07	
29	29.34	15.18	11.36	9.53	8.64	8.22	8.13	8.19	
30	30.34	15.68	11.69	9.78	8.84	8.39	8.28	8.32	
31	31.34	16.18	12.03	10.03	9.04	8.55	8.42	8.44	
32	32.34	16.68	12.36	10.28	9.24	8.72	8.56	8.57	
33	33.34	17.18	12.69	10.53	9.44	8.89	8.70	8.69	
34	34.34	17.68	13.03	10.78	9.64	9.05	8.85	8.82	
35	35.34	18.18	13.36	11.03	9.84	9.22	8.99	8.94	
36	36.34	18.68	13.69	11.28	10.04	9.39	9.13	9.07	
37	37.34	19.18	14.03	11.53	10.24	9.55	9.28	9.19	
38	38.34	19.68	14.36	11.78	10.44	9.72	9.42	9.32	
39	39.34	20.18	14.69	12.03	10.64	9.89	9.56	9.44	
40	40.34	20.68	15.03	12.28	10.84	10.05	9.70	9.57	
41	41.34	21.18	15.36	12.53	11.04	10.22	9.85	9.69	
42	42.34	21.68	15.69	12.78	11.24	10.39	9.99	9.82	
43	43.34	22.18	16.03	13.03	11.44	10.55	10.13	9.94	
44	44.34	22.68	16.36	13.28	11.64	10.72	10.28	10.07	

45	45.34	23.18	16.69	13.53	11.84	10.89	10.42	10.19
46	46.34	23.68	17.03	13.78	12.04	11.05	10.56	10.32
47	47.34	24.18	17.36	14.03	12.24	11.22	10.70	10.44
48	48.34	24.68	17.69	14.28	12.44	11.39	10.85	10.57
49	49.34	25.18	18.03	14.53	12.64	11.55	10.99	10.69
50	50.34	25.68	18.36	14.78	12.84	11.72	11.13	10.82
51	51.34	26.18	18.69	15.03	13.04	11.89	11.28	10.94
52	52.34	26.68	19.03	15.28	13.24	12.05	11.42	11.07
53	53.34	27.18	19.36	15.53	13.44	12.22	11.56	11.19
54	54.34	27.68	19.69	15.78	13.64	12.39	11.70	11.32
55	55.34	28.18	20.03	16.03	13.84	12.55	11.85	11.44
56	56.34	28.68	20.36	16.28	14.04	12.72	11.99	11.57
57	57.34	29.18	20.69	16.53	14.24	12.89	12.13	11.69
58	58.34	29.68	21.03	16.78	14.44	13.05	12.28	11.82
59	59.34	30.18	21.36	17.03	14.64	13.22	12.42	11.94
60	60.34	30.68	21.69	17.28	14.84	13.39	12.56	12.07
61	61.34	31.18	22.03	17.53	15.04	13.55	12.70	12.19
62	62.34	31.68	22.36	17.78	15.24	13.72	12.85	12.32
63	63.34	32.18	22.69	18.03	15.44	13.89	12.99	12.44
64	64.34	32.68	23.03	18.28	15.64	14.05	13.13	12.57
65	65.34	33.18	23.36	18.53	15.84	14.22	13.28	12.69
66	66.34	33.68	23.69	18.78	16.04	14.39	13.42	12.82
67	67.34	34.18	24.03	19.03	16.24	14.55	13.56	12.94
68	68.34	34.68	24.36	19.28	16.44	14.72	13.70	13.07
69	69.34	35.18	24.69	19.53	16.64	14.89	13.85	13.19
70	70.34	35.68	25.03	19.78	16.84	15.05	13.99	13.32
71	71.34	36.18	25.36	20.03	17.04	15.22	14.13	13.44
72	72.34	36.68	25.69	20.28	17.24	15.39	14.28	13.57
73	73.34	37.18	26.03	20.53	17.44	15.55	14.42	13.69
74	74.34	37.68	26.36	20.78	17.64	15.72	14.56	13.82
75	75.34	38.18	26.69	21.03	17.84	15.89	14.70	13.94
76	76.34	38.68	27.03	21.28	18.04	16.05	14.85	14.07
77	77.34	39.18	27.36	21.53	18.24	16.22	14.99	14.19
78	78.34	39.68	27.69	21.78	18.44	16.39	15.13	14.32
79	79.34	40.18	28.03	22.03	18.64	16.55	15.28	14.44
80	80.34	40.68	28.36	22.28	18.84	16.72	15.42	14.57
81	81.34	41.18	28.69	22.53	19.04	16.89	15.56	14.69
82	82.34	41.68	29.03	22.78	19.24	17.05	15.70	14.82
83	83.34	42.18	29.36	23.03	19.44	17.22	15.85	14.94
84	84.34	42.68	29.69	23.28	19.64	17.39	15.99	15.07
85	85.34	43.18	30.03	23.53	19.84	17.55	16.13	15.19
86	86.34	43.68	30.36	23.78	20.04	17.72	16.28	15.32
87	87.34	44.18	30.69	24.03	20.24	17.89	16.42	15.44
88	88.34	44.68	31.03	24.28	20.44	18.05	16.56	15.57
89	89.34	45.18	31.36	24.53	20.64	18.22	16.70	15.69
90	90.34	45.68	31.69	24.78	20.84	18.39	16.85	15.82
91	91.34	46.18	32.03	25.03	21.04	18.55	16.99	15.94
92	92.34	46.68	32.36	25.28	21.24	18.72	17.13	16.07
93	93.34	47.18	32.69	25.53	21.44	18.89	17.28	16.19
94	94.34	47.68	33.03	25.78	21.64	19.05	17.42	16.32

95	95.34	48.18	33.36	26.03	21.84	19.22	17.56	16.44
96	96.34	48.68	33.69	26.28	22.04	19.39	17.70	16.57
97	97.34	49.18	34.03	26.53	22.24	19.55	17.85	16.69
98	98.34	49.68	34.36	26.78	22.44	19.72	17.99	16.82
99	99.34	50.18	34.69	27.03	22.64	19.89	18.13	16.94
100	100.34	50.68	35.03	27.28	22.84	20.05	18.28	17.07
101	101.34	51.18	35.36	27.53	23.04	20.22	18.42	17.19
102	102.34	51.68	35.69	27.78	23.24	20.39	18.56	17.32
103	103.34	52.18	36.03	28.03	23.44	20.55	18.70	17.44
104	104.34	52.68	36.36	28.28	23.64	20.72	18.85	17.57
105	105.34	53.18	36.69	28.53	23.84	20.89	18.99	17.69
106	106.34	53.68	37.03	28.78	24.04	21.05	19.13	17.82
107	107.34	54.18	37.36	29.03	24.24	21.22	19.28	17.94
108	108.34	54.68	37.69	29.28	24.44	21.39	19.42	18.07
109	109.34	55.18	38.03	29.53	24.64	21.55	19.56	18.19
110	110.34	55.68	38.36	29.78	24.84	21.72	19.70	18.32
111	111.34	56.18	38.69	30.03	25.04	21.89	19.85	18.44
112	112.34	56.68	39.03	30.28	25.24	22.05	19.99	18.57
113	113.34	57.18	39.36	30.53	25.44	22.22	20.13	18.69
114	114.34	57.68	39.69	30.78	25.64	22.39	20.28	18.82
115	115.34	58.18	40.03	31.03	25.84	22.55	20.42	18.94
116	116.34	58.68	40.36	31.28	26.04	22.72	20.56	19.07
117	117.34	59.18	40.69	31.53	26.24	22.89	20.70	19.19
118	118.34	59.68	41.03	31.78	26.44	23.05	20.85	19.32
119	119.34	60.18	41.36	32.03	26.64	23.22	20.99	19.44
120	120.34	60.68	41.69	32.28	26.84	23.39	21.13	19.57
121	121.34	61.18	42.03	32.53	27.04	23.55	21.28	19.69
122	122.34	61.68	42.36	32.78	27.24	23.72	21.42	19.82
123	123.34	62.18	42.69	33.03	27.44	23.89	21.56	19.94
124	124.34	62.68	43.03	33.28	27.64	24.05	21.70	20.07
125	125.34	63.18	43.36	33.53	27.84	24.22	21.85	20.19
126	126.34	63.68	43.69	33.78	28.04	24.39	21.99	20.32
127	127.34	64.18	44.03	34.03	28.24	24.55	22.13	20.44
128	128.34	64.68	44.36	34.28	28.44	24.72	22.28	20.57
129	129.34	65.18	44.69	34.53	28.64	24.89	22.42	20.69
130	130.34	65.68	45.03	34.78	28.84	25.05	22.56	20.82
131	131.34	66.18	45.36	35.03	29.04	25.22	22.70	20.94
132	132.34	66.68	45.69	35.28	29.24	25.39	22.85	21.07
133	133.34	67.18	46.03	35.53	29.44	25.55	22.99	21.19
134	134.34	67.68	46.36	35.78	29.64	25.72	23.13	21.32
135	135.34	68.18	46.69	36.03	29.84	25.89	23.28	21.44
136	136.34	68.68	47.03	36.28	30.04	26.05	23.42	21.57
137	137.34	69.18	47.36	36.53	30.24	26.22	23.56	21.69
138	138.34	69.68	47.69	36.78	30.44	26.39	23.70	21.82
139	139.34	70.18	48.03	37.03	30.64	26.55	23.85	21.94
140	140.34	70.68	48.36	37.28	30.84	26.72	23.99	22.07
141	141.34	71.18	48.69	37.53	31.04	26.89	24.13	22.19
142	142.34	71.68	49.03	37.78	31.24	27.05	24.28	22.32
143	143.34	72.18	49.36	38.03	31.44	27.22	24.42	22.44
144	144.34	72.68	49.69	38.28	31.64	27.39	24.56	22.57

145	145.34	73.18	50.03	38.53	31.84	27.55	24.70	22.69
146	146.34	73.68	50.36	38.78	32.04	27.72	24.85	22.82
147	147.34	74.18	50.69	39.03	32.24	27.89	24.99	22.94
148	148.34	74.68	51.03	39.28	32.44	28.05	25.13	23.07
149	149.34	75.18	51.36	39.53	32.64	28.22	25.28	23.19
150	150.34	75.68	51.69	39.78	32.84	28.39	25.42	23.32
151	151.34	76.18	52.03	40.03	33.04	28.55	25.56	23.44
152	152.34	76.68	52.36	40.28	33.24	28.72	25.70	23.57
153	153.34	77.18	52.69	40.53	33.44	28.89	25.85	23.69
154	154.34	77.68	53.03	40.78	33.64	29.05	25.99	23.82
155	155.34	78.18	53.36	41.03	33.84	29.22	26.13	23.94
156	156.34	78.68	53.69	41.28	34.04	29.39	26.28	24.07
157	157.34	79.18	54.03	41.53	34.24	29.55	26.42	24.19
158	158.34	79.68	54.36	41.78	34.44	29.72	26.56	24.32
159	159.34	80.18	54.69	42.03	34.64	29.89	26.70	24.44
160	160.34	80.68	55.03	42.28	34.84	30.05	26.85	24.57
161	161.34	81.18	55.36	42.53	35.04	30.22	26.99	24.69
162	162.34	81.68	55.69	42.78	35.24	30.39	27.13	24.82
163	163.34	82.18	56.03	43.03	35.44	30.55	27.28	24.94
164	164.34	82.68	56.36	43.28	35.64	30.72	27.42	25.07
165	165.34	83.18	56.69	43.53	35.84	30.89	27.56	25.19
166	166.34	83.68	57.03	43.78	36.04	31.05	27.70	25.32
167	167.34	84.18	57.36	44.03	36.24	31.22	27.85	25.44
168	168.34	84.68	57.69	44.28	36.44	31.39	27.99	25.57
169	169.34	85.18	58.03	44.53	36.64	31.55	28.13	25.69
170	170.34	85.68	58.36	44.78	36.84	31.72	28.28	25.82
171	171.34	86.18	58.69	45.03	37.04	31.89	28.42	25.94
172	172.34	86.68	59.03	45.28	37.24	32.05	28.56	26.07
173	173.34	87.18	59.36	45.53	37.44	32.22	28.70	26.19
174	174.34	87.68	59.69	45.78	37.64	32.39	28.85	26.32
175	175.34	88.18	60.03	46.03	37.84	32.55	28.99	26.44
176	176.34	88.68	60.36	46.28	38.04	32.72	29.13	26.57
177	177.34	89.18	60.69	46.53	38.24	32.89	29.28	26.69
178	178.34	89.68	61.03	46.78	38.44	33.05	29.42	26.82
179	179.34	90.18	61.36	47.03	38.64	33.22	29.56	26.94
180	180.34	90.68	61.69	47.28	38.84	33.39	29.70	27.07
181	181.34	91.18	62.03	47.53	39.04	33.55	29.85	27.19
182	182.34	91.68	62.36	47.78	39.24	33.72	29.99	27.32
.								.
.								.
.								.
798	798.34	399.68	267.69	201.78	162.44	136.39	117.99	104.32
799	799.34	400.18	268.03	202.03	162.64	136.55	118.13	104.44
800	800.34	400.68	268.36	202.28	162.84	136.72	118.28	104.57

MOTOR EN ELEVACIÓN

	VELOCIDAD								
mil	1	2	3	4	5	6	7	8	RPS
	60	120	180	240	300	360	420	480	RPM
1	1.26	1.02	0.00	0.00	0.00	0.00	0.00	0.00	
2	2.26	1.52	0.00	0.00	0.00	0.00	0.00	0.00	
3	3.26	2.02	0.00	0.00	0.00	0.00	0.00	0.00	
4	4.26	2.52	2.78	0.00	0.00	0.00	0.00	0.00	
5	5.26	3.02	3.11	0.00	0.00	0.00	0.00	0.00	
6	6.26	3.52	3.45	3.45	0.00	0.00	0.00	0.00	
7	7.26	4.02	3.78	3.70	0.00	0.00	0.00	0.00	
8	8.26	4.52	4.11	3.95	0.00	0.00	0.00	0.00	
9	9.26	5.02	4.45	4.20	0.00	0.00	0.00	0.00	
10	10.26	5.52	4.78	4.45	4.43	0.00	0.00	0.00	
11	11.26	6.02	5.11	4.70	4.63	0.00	0.00	0.00	
12	12.26	6.52	5.45	4.95	4.83	0.00	0.00	0.00	
13	13.26	7.02	5.78	5.20	5.03	5.06	0.00	0.00	
14	14.26	7.52	6.11	5.45	5.23	5.22	0.00	0.00	
15	15.26	8.02	6.45	5.70	5.43	5.39	0.00	0.00	
16	16.26	8.52	6.78	5.95	5.63	5.56	0.00	0.00	
17	17.26	9.02	7.11	6.20	5.83	5.72	0.00	0.00	
18	18.26	9.52	7.45	6.45	6.03	5.89	5.98	0.00	
19	19.26	10.02	7.78	6.70	6.23	6.06	6.12	0.00	
20	20.26	10.52	8.11	6.95	6.43	6.22	6.27	0.00	
21	21.26	11.02	8.45	7.20	6.63	6.39	6.41	0.00	
22	22.26	11.52	8.78	7.45	6.83	6.56	6.55	0.00	
23	23.26	12.02	9.11	7.70	7.03	6.72	6.70	0.00	
24	24.26	12.52	9.45	7.95	7.23	6.89	6.84	6.91	
25	25.26	13.02	9.78	8.20	7.43	7.06	6.98	7.03	
26	26.26	13.52	10.11	8.45	7.63	7.22	7.12	7.16	
27	27.26	14.02	10.45	8.70	7.83	7.39	7.27	7.28	
28	28.26	14.52	10.78	8.95	8.03	7.56	7.41	7.41	
29	29.26	15.02	11.11	9.20	8.23	7.72	7.55	7.53	
30	30.26	15.52	11.45	9.45	8.43	7.89	7.70	7.66	
31	31.26	16.02	11.78	9.70	8.63	8.06	7.84	7.78	
32	32.26	16.52	12.11	9.95	8.83	8.22	7.98	7.91	
33	33.26	17.02	12.45	10.20	9.03	8.39	8.12	8.03	
34	34.26	17.52	12.78	10.45	9.23	8.56	8.27	8.16	
35	35.26	18.02	13.11	10.70	9.43	8.72	8.41	8.28	
36	36.26	18.52	13.45	10.95	9.63	8.89	8.55	8.41	
37	37.26	19.02	13.78	11.20	9.83	9.06	8.70	8.53	
38	38.26	19.52	14.11	11.45	10.03	9.22	8.84	8.66	
39	39.26	20.02	14.45	11.70	10.23	9.39	8.98	8.78	
40	40.26	20.52	14.78	11.95	10.43	9.56	9.12	8.91	
41	41.26	21.02	15.11	12.20	10.63	9.72	9.27	9.03	
42	42.26	21.52	15.45	12.45	10.83	9.89	9.41	9.16	
43	43.26	22.02	15.78	12.70	11.03	10.06	9.55	9.28	
44	44.26	22.52	16.11	12.95	11.23	10.22	9.70	9.41	

45	45.26	23.02	16.45	13.20	11.43	10.39	9.84	9.53
46	46.26	23.52	16.78	13.45	11.63	10.56	9.98	9.66
47	47.26	24.02	17.11	13.70	11.83	10.72	10.12	9.78
48	48.26	24.52	17.45	13.95	12.03	10.89	10.27	9.91
49	49.26	25.02	17.78	14.20	12.23	11.06	10.41	10.03
50	50.26	25.52	18.11	14.45	12.43	11.22	10.55	10.16
51	51.26	26.02	18.45	14.70	12.63	11.39	10.70	10.28
52	52.26	26.52	18.78	14.95	12.83	11.56	10.84	10.41
53	53.26	27.02	19.11	15.20	13.03	11.72	10.98	10.53
54	54.26	27.52	19.45	15.45	13.23	11.89	11.12	10.66
55	55.26	28.02	19.78	15.70	13.43	12.06	11.27	10.78
56	56.26	28.52	20.11	15.95	13.63	12.22	11.41	10.91
57	57.26	29.02	20.45	16.20	13.83	12.39	11.55	11.03
58	58.26	29.52	20.78	16.45	14.03	12.56	11.70	11.16
59	59.26	30.02	21.11	16.70	14.23	12.72	11.84	11.28
60	60.26	30.52	21.45	16.95	14.43	12.89	11.98	11.41
61	61.26	31.02	21.78	17.20	14.63	13.06	12.12	11.53
62	62.26	31.52	22.11	17.45	14.83	13.22	12.27	11.66
63	63.26	32.02	22.45	17.70	15.03	13.39	12.41	11.78
64	64.26	32.52	22.78	17.95	15.23	13.56	12.55	11.91
65	65.26	33.02	23.11	18.20	15.43	13.72	12.70	12.03
66	66.26	33.52	23.45	18.45	15.63	13.89	12.84	12.16
67	67.26	34.02	23.78	18.70	15.83	14.06	12.98	12.28
68	68.26	34.52	24.11	18.95	16.03	14.22	13.12	12.41
69	69.26	35.02	24.45	19.20	16.23	14.39	13.27	12.53
70	70.26	35.52	24.78	19.45	16.43	14.56	13.41	12.66
71	71.26	36.02	25.11	19.70	16.63	14.72	13.55	12.78
72	72.26	36.52	25.45	19.95	16.83	14.89	13.70	12.91
73	73.26	37.02	25.78	20.20	17.03	15.06	13.84	13.03
74	74.26	37.52	26.11	20.45	17.23	15.22	13.98	13.16
75	75.26	38.02	26.45	20.70	17.43	15.39	14.12	13.28
76	76.26	38.52	26.78	20.95	17.63	15.56	14.27	13.41
77	77.26	39.02	27.11	21.20	17.83	15.72	14.41	13.53
78	78.26	39.52	27.45	21.45	18.03	15.89	14.55	13.66
79	79.26	40.02	27.78	21.70	18.23	16.06	14.70	13.78
80	80.26	40.52	28.11	21.95	18.43	16.22	14.84	13.91
81	81.26	41.02	28.45	22.20	18.63	16.39	14.98	14.03
82	82.26	41.52	28.78	22.45	18.83	16.56	15.12	14.16
83	83.26	42.02	29.11	22.70	19.03	16.72	15.27	14.28
84	84.26	42.52	29.45	22.95	19.23	16.89	15.41	14.41
85	85.26	43.02	29.78	23.20	19.43	17.06	15.55	14.53
86	86.26	43.52	30.11	23.45	19.63	17.22	15.70	14.66
87	87.26	44.02	30.45	23.70	19.83	17.39	15.84	14.78
88	88.26	44.52	30.78	23.95	20.03	17.56	15.98	14.91
89	89.26	45.02	31.11	24.20	20.23	17.72	16.12	15.03
90	90.26	45.52	31.45	24.45	20.43	17.89	16.27	15.16
91	91.26	46.02	31.78	24.70	20.63	18.06	16.41	15.28
92	92.26	46.52	32.11	24.95	20.83	18.22	16.55	15.41
93	93.26	47.02	32.45	25.20	21.03	18.39	16.70	15.53
94	94.26	47.52	32.78	25.45	21.23	18.56	16.84	15.66

95	95.26	48.02	33.11	25.70	21.43	18.72	16.98	15.78
96	96.26	48.52	33.45	25.95	21.63	18.89	17.12	15.91
97	97.26	49.02	33.78	26.20	21.83	19.06	17.27	16.03
98	98.26	49.52	34.11	26.45	22.03	19.22	17.41	16.16
99	99.26	50.02	34.45	26.70	22.23	19.39	17.55	16.28
100	100.26	50.52	34.78	26.95	22.43	19.56	17.70	16.41
101	101.26	51.02	35.11	27.20	22.63	19.72	17.84	16.53
102	102.26	51.52	35.45	27.45	22.83	19.89	17.98	16.66
103	103.26	52.02	35.78	27.70	23.03	20.06	18.12	16.78
104	104.26	52.52	36.11	27.95	23.23	20.22	18.27	16.91
105	105.26	53.02	36.45	28.20	23.43	20.39	18.41	17.03
106	106.26	53.52	36.78	28.45	23.63	20.56	18.55	17.16
107	107.26	54.02	37.11	28.70	23.83	20.72	18.70	17.28
108	108.26	54.52	37.45	28.95	24.03	20.89	18.84	17.41
109	109.26	55.02	37.78	29.20	24.23	21.06	18.98	17.53
110	110.26	55.52	38.11	29.45	24.43	21.22	19.12	17.66
111	111.26	56.02	38.45	29.70	24.63	21.39	19.27	17.78
112	112.26	56.52	38.78	29.95	24.83	21.56	19.41	17.91
113	113.26	57.02	39.11	30.20	25.03	21.72	19.55	18.03
114	114.26	57.52	39.45	30.45	25.23	21.89	19.70	18.16
115	115.26	58.02	39.78	30.70	25.43	22.06	19.84	18.28
116	116.26	58.52	40.11	30.95	25.63	22.22	19.98	18.41
117	117.26	59.02	40.45	31.20	25.83	22.39	20.12	18.53
118	118.26	59.52	40.78	31.45	26.03	22.56	20.27	18.66
119	119.26	60.02	41.11	31.70	26.23	22.72	20.41	18.78
120	120.26	60.52	41.45	31.95	26.43	22.89	20.55	18.91
121	121.26	61.02	41.78	32.20	26.63	23.06	20.70	19.03
122	122.26	61.52	42.11	32.45	26.83	23.22	20.84	19.16
123	123.26	62.02	42.45	32.70	27.03	23.39	20.98	19.28
124	124.26	62.52	42.78	32.95	27.23	23.56	21.12	19.41
125	125.26	63.02	43.11	33.20	27.43	23.72	21.27	19.53
126	126.26	63.52	43.45	33.45	27.63	23.89	21.41	19.66
127	127.26	64.02	43.78	33.70	27.83	24.06	21.55	19.78
128	128.26	64.52	44.11	33.95	28.03	24.22	21.70	19.91
129	129.26	65.02	44.45	34.20	28.23	24.39	21.84	20.03
130	130.26	65.52	44.78	34.45	28.43	24.56	21.98	20.16
131	131.26	66.02	45.11	34.70	28.63	24.72	22.12	20.28
132	132.26	66.52	45.45	34.95	28.83	24.89	22.27	20.41
133	133.26	67.02	45.78	35.20	29.03	25.06	22.41	20.53
134	134.26	67.52	46.11	35.45	29.23	25.22	22.55	20.66
135	135.26	68.02	46.45	35.70	29.43	25.39	22.70	20.78
136	136.26	68.52	46.78	35.95	29.63	25.56	22.84	20.91
137	137.26	69.02	47.11	36.20	29.83	25.72	22.98	21.03
138	138.26	69.52	47.45	36.45	30.03	25.89	23.12	21.16
139	139.26	70.02	47.78	36.70	30.23	26.06	23.27	21.28
140	140.26	70.52	48.11	36.95	30.43	26.22	23.41	21.41
141	141.26	71.02	48.45	37.20	30.63	26.39	23.55	21.53
142	142.26	71.52	48.78	37.45	30.83	26.56	23.70	21.66
143	143.26	72.02	49.11	37.70	31.03	26.72	23.84	21.78
144	144.26	72.52	49.45	37.95	31.23	26.89	23.98	21.91

145	145.26	73.02	49.78	38.20	31.43	27.06	24.12	22.03
146	146.26	73.52	50.11	38.45	31.63	27.22	24.27	22.16
147	147.26	74.02	50.45	38.70	31.83	27.39	24.41	22.28
148	148.26	74.52	50.78	38.95	32.03	27.56	24.55	22.41
149	149.26	75.02	51.11	39.20	32.23	27.72	24.70	22.53
150	150.26	75.52	51.45	39.45	32.43	27.89	24.84	22.66
151	151.26	76.02	51.78	39.70	32.63	28.06	24.98	22.78
152	152.26	76.52	52.11	39.95	32.83	28.22	25.12	22.91
153	153.26	77.02	52.45	40.20	33.03	28.39	25.27	23.03
154	154.26	77.52	52.78	40.45	33.23	28.56	25.41	23.16
155	155.26	78.02	53.11	40.70	33.43	28.72	25.55	23.28
156	156.26	78.52	53.45	40.95	33.63	28.89	25.70	23.41
157	157.26	79.02	53.78	41.20	33.83	29.06	25.84	23.53
158	158.26	79.52	54.11	41.45	34.03	29.22	25.98	23.66
159	159.26	80.02	54.45	41.70	34.23	29.39	26.12	23.78
160	160.26	80.52	54.78	41.95	34.43	29.56	26.27	23.91
161	161.26	81.02	55.11	42.20	34.63	29.72	26.41	24.03
162	162.26	81.52	55.45	42.45	34.83	29.89	26.55	24.16
163	163.26	82.02	55.78	42.70	35.03	30.06	26.70	24.28
164	164.26	82.52	56.11	42.95	35.23	30.22	26.84	24.41
165	165.26	83.02	56.45	43.20	35.43	30.39	26.98	24.53
166	166.26	83.52	56.78	43.45	35.63	30.56	27.12	24.66
167	167.26	84.02	57.11	43.70	35.83	30.72	27.27	24.78
168	168.26	84.52	57.45	43.95	36.03	30.89	27.41	24.91
169	169.26	85.02	57.78	44.20	36.23	31.06	27.55	25.03
170	170.26	85.52	58.11	44.45	36.43	31.22	27.70	25.16
171	171.26	86.02	58.45	44.70	36.63	31.39	27.84	25.28
172	172.26	86.52	58.78	44.95	36.83	31.56	27.98	25.41
173	173.26	87.02	59.11	45.20	37.03	31.72	28.12	25.53
174	174.26	87.52	59.45	45.45	37.23	31.89	28.27	25.66
175	175.26	88.02	59.78	45.70	37.43	32.06	28.41	25.78
176	176.26	88.52	60.11	45.95	37.63	32.22	28.55	25.91
177	177.26	89.02	60.45	46.20	37.83	32.39	28.70	26.03
178	178.26	89.52	60.78	46.45	38.03	32.56	28.84	26.16
179	179.26	90.02	61.11	46.70	38.23	32.72	28.98	26.28
180	180.26	90.52	61.45	46.95	38.43	32.89	29.12	26.41
181	181.26	91.02	61.78	47.20	38.63	33.06	29.27	26.53
182	182.26	91.52	62.11	47.45	38.83	33.22	29.41	26.66
.								.
.								.
.								.
998	998.26	499.52	334.11	251.45	202.03	169.22	145.98	128.66
999	999.26	500.02	334.45	251.70	202.23	169.39	146.12	128.78
1000	1000.26	500.52	334.78	251.95	202.43	169.56	146.27	128.91

ANEXO 2

Tablas de Tiro para Carga 6, 7, 8 y 9 bis

ANGULO DE ELEVACIÓN	DURACIÓN DEL TRAYECTO	DERIVACIÓN	ALCANCE	CARGA
2.3	0.2	0	100	**6 VIVA**
4.6	0.4	0	200	
6.9	0.6	0	300	
9.3	0.9	0	400	
11.6	1.1	0	500	
14.1	1.3	0	600	
16.5	1.5	0	700	
19	1.8	0	800	
21.5	2	0	900	
24	2.2	0	1000	
26.6	2.5	-1	1100	
29.2	2.7	-1	1200	
31.8	2.9	-1	1300	
34.5	3.2	-1	1400	
37.1	3.4	-1	1500	
39.9	3.7	-1	1600	
42.6	3.9	-1	1700	
45.4	4.2	-1	1800	
48.3	4.4	-1	1900	
51.1	4.7	-1	2000	
54	4.9	-1	2100	
57	5.2	-1	2200	
59.9	5.4	-1	2300	
62.9	5.7	-1	2400	
66	6	-1	2500	
69.1	6.2	-1	2600	
72.2	6.5	-2	2700	
75.4	6.8	-2	2800	
78.6	7	-2	2900	
81.8	7.3	-2	3000	
85.1	7.6	-2	3100	

88.5	7.9	-2	3200
91.8	8.2	-2	3300
95.2	8.5	-2	3400
98.7	9.2	-2	3500
102.2	9	-2	3600
105.8	9.3	-2	3700
109.4	9.6	-2	3800
113	9.9	-3	3900
116.7	10.2	-3	4000
120.4	10.5	-3	4100
124.2	10.9	-3	4200
128.1	11.2	-3	4300
132	11.5	-3	4400
135.9	11.8	-3	4500
139.9	12.1	-3	4600
143.9	12.4	-3	4700
148	12.8	-3	4800
152.1	13.1	-4	4900
156.3	13.4	-4	5000
160.5	13.7	-4	5100
164.8	14.1	-4	5200
169.1	14.4	-4	5300
173.4	14.7	-4	5400
177.9	15.1	-4	5500
187.3	15.4	-4	5600
186.8	15.7	-5	5700
191.4	16.1	-5	5800
196	16.4	-5	5900
200.6	16.8	-5	6000
205.3	17.1	-5	6100
210	17.5	-5	6200
214.8	17.8	-5	6300
219.7	18.2	-5	6400
224.6	18.5	-6	6500

229.5	18.9	-6	6600
234.5	19.3	-6	6700
239.5	19.6	-6	6800
244.6	20	-6	6900
249.8	20.3	-6	7000
254.9	20.7	-6	7100
260.2	21.1	-7	7200
265.5	21.5	-7	7300
270.8	21.8	-7	7400
276.2	22.2	-7	7500
281.7	22.6	-7	7600
287.2	23	-7	7700
292.7	23.4	-8	7800
298.4	23.8	-8	7900
304.1	24.2	-8	8000
309.8	24.6	-8	8100
315.6	25	-8	8200
321.5	25.4	-8	8300
327.4	25.8	-9	8400
333.4	26.2	-9	8500
339.5	26.6	-9	8600
345.7	27	-9	8700
351.9	27.4	-9	8800
358.2	27.9	-10	8900
364.6	28.3	-10	9000
371	28.7	-10	9100
377.5	29.2	-10	9200
384.2	29.6	-10	9300
390.9	30	-11	9400
397.7	30.5	-11	9500
404.6	30.9	-11	9600
411.6	31.4	-11	9700
418.8	31.9	-12	9800
426	32.3	-12	9900

433.4	32.8	-12	10000
440.9	33.3	-12	10100
448.5	33.8	-13	10200
456.3	34.3	-13	10300
464.2	34.8	-13	10400
472.3	35.3	-14	10500
480.6	35.9	-14	10600
489	36.4	-14	10700
497.7	37	-14	10800
506.7	37.5	-15	10900
515.9	38.1	-15	11000
525.4	38.7	-16	11100
535.2	39.3	-16	11200
545.4	39.9	-16	11300
555.9	40.6	-17	11400
566.9	41.2	-17	11500
578.4	41.9	-18	11600
590.4	42.7	-18	11700
603.1	43.4	-19	11800
616.7	44.2	-19	11900
631.3	45.1	-20	12000
646.3	46	-21	12100
661.2	47.1	-22	12200
685.9	48.2	-23	12300
711.3	49.7	-24	12400
747.4	51.7	-26	12500

ANGULO DE ELEVACIÓN	DURACIÓN DEL TRAYECTO	DERIVACIÓN	ALCANCE	CARGA
1.6	0.2	0	100	**7**
3.3	0.4	0	200	
4.9	0.5	0	300	

6.6	0.7	0	400
8.3	0.9	0	500
10.1	1.1	0	600
11.8	1.3	0	700
13.6	1.5	0	800
15.4	1.7	0	900
17.2	1.9	0	1000
19	2.1	0	1100
20.8	2.3	0	1200
22.7	2.5	0	1300
24.6	2.7	0	1400
26.5	2.9	-1	1500
28.4	3.1	-1	1600
30.4	3.3	-1	1700
32.4	3.5	-1	1800
34.4	3.7	-1	1900
36.4	3.9	-1	2000
38.5	4.1	-1	2100
40.5	4.4	-1	2200
42.6	4.6	-1	2300
44.8	4.8	-1	2400
46.9	5	-1	2500
49.1	5.2	-1	2600
51.3	5.5	-1	2700
53.6	5.7	-1	2800
55.8	5.9	-1	2900
58.1	6.2	-1	3000
60.4	6.4	-1	3100
62.8	6.6	-1	3200
65.2	6.9	-1	3300
67.6	7.1	-1	3400
70	7.4	-2	3500
72.5	7.6	-2	3600
75	7.9	-2	3700

77.5	8.1	-2	3800
80.1	8.4	-2	3900
82.7	8.6	-2	4000
85.3	8.9	-2	4100
88	9.1	-2	4200
90.7	9.4	-2	4300
93.4	9.7	-2	4400
96.2	9.9	-2	4500
99	10.2	-2	4600
101.8	10.5	-2	4700
104.7	10.7	-2	4800
107.6	11	-3	4900
110.6	11.3	-3	5000
113.5	11.6	-3	5100
116.6	11.9	-3	5200
119.6	12.1	-3	5300
122.8	12.4	-3	5400
125.9	12.7	-3	5500
129.1	13	-3	5600
132.3	13.3	-3	5700
135.6	13.6	-3	5800
138.9	13.9	-3	5900
142.3	14.2	-3	6000
145.7	14.5	-4	6100
149.1	14.8	-4	6200
152.6	15.2	-4	6300
156.2	15.5	-4	6400
159.8	15.8	-4	6500
163.4	16.1	-4	6600
167.1	16.4	-4	6700
170.8	16.7	-4	6800
174.5	17.1	-4	6900
178.4	17.4	-5	7000
182.2	17.7	-5	7100

186.1	18.1	-5	7200
190.1	18.4	-5	7300
194	18.7	-5	7400
198.1	19.1	-5	7500
202.2	19.4	-5	7600
206.3	19.7	-5	7700
210.5	20.1	-5	7800
214.7	20.4	-6	7900
219	20.8	-6	8000
223.3	21.1	-6	8100
227.6	21.5	-6	8200
232	21.8	-6	8300
236.5	22.2	-6	8400
241	22.6	-6	8500
245.5	22.9	-7	8600
250.1	23.3	-7	8700
254.8	23.6	-7	8800
259.4	24	-7	8900
264.2	24.4	-7	9000
268.9	24.7	-7	9100
273.8	25.1	-7	9200
278.6	25.5	-8	9300
283.6	25.9	-8	9400
288.5	26.3	-8	9500
293.6	26.6	-8	9600
298.7	27	-8	9700
303.8	27.4	-8	9800
309	27.8	-9	9900
314.3	28.2	-9	10000
319.5	28.6	-9	10100
324.9	29	-9	10200
330.3	29.4	-9	10300
335.8	29.8	-9	10400
341.3	30.2	-10	10500

346.9	30.6	-10	10600
352.5	31	-10	10700
358.3	31.4	-10	10800
364	31.9	-10	10900
369.9	32.3	-11	11000
375.8	32.7	-11	11100
381.8	33.1	-11	11200
387.8	33.6	-11	11300
393.9	34	-11	11400
400.1	34.5	-12	11500
406.4	34.9	-12	11600
412.8	35.4	-12	11700
419.3	35.8	-12	11800
425.9	36.3	-13	11900
432.5	36.8	-13	12000
439.3	37.2	-13	12100
446.1	37.7	-13	12200
453.1	38.2	-14	12300
460.1	38.7	-14	12400
467.3	39.2	-14	12500
474.6	39.7	-14	12600
482	40.2	-15	12700
489.6	40.7	-15	12800
497.4	41.2	-15	12900
505.4	41.8	-16	13000
513.5	42.3	-16	13100
521.9	42.9	-16	13200
530.4	43.5	-17	13300
539.3	44.1	-17	13400
548.3	44.7	-17	13500
557.6	45.3	-18	13600
567.3	45.9	-18	13700
577.2	46.6	-19	13800
587.6	47.3	-19	13900

598.4	48	-20	14000
609.7	48.7	-20	14100
621.7	49.5	-21	14200
634.5	50.3	-21	14300
48.2	51.2	-22	14400
663.2	52.1	-23	14500
680	53.1	-24	14600
699.3	54.3	-25	14700
722.7	55.8	-26	14800
755.5	57.7	-28	14900

ANGULO DE ELEVACIÓN	DURACIÓN DEL TRAYECTO	DERIVACIÓN	ALCANCE	CARGA
1.2	0.2	0	100	**8**
2.4	0.3	0	200	
3.6	0.5	0	300	
4.8	0.6	0	400	
6	0.8	0	500	
7.2	0.9	0	600	
8.5	1.1	0	700	
9.7	1.3	0	800	
11	1.4	0	900	
12.3	1.6	0	1000	
13.6	1.8	0	1100	
14.9	1.9	0	1200	
16.2	2.1	0	1300	
17.6	2.3	0	1400	
19	2.4	0	1500	
20.3	2.6	0	1600	
21.7	2.8	0	1700	
23.1	3	0	1800	
24.5	3.1	-1	1900	
26	3.3	-1	2000	
27.4	3.5	-1	2100	
28.9	3.7	-1	2200	

30.4	3.9	-1	2300
31.9	4	-1	2400
33.4	4.2	-1	2500
34.9	4.4	-1	2600
36.5	4.6	-1	2700
38	4.8	-1	2800
39.6	5	-1	2900
41.2	5.2	-1	3000
42.3	5.4	-1	3100
44.5	5.6	-1	3200
46.2	5.8	-1	3300
47.6	6	-1	3400
49.5	6.2	-1	3500
51.3	6.4	-1	3600
53	6.6	-1	3700
54.6	6.8	-1	3800
56.5	7	-1	3900
58.3	7.2	-1	4000
60.2	7.4	-1	4100
62	7.6	-1	4200
63.9	7.9	-1	4300
65.9	8.1	-1	4400
67.7	8.3	-2	4500
69.6	8.5	-2	4600
71.6	8.7	-2	4700
73.6	9	-2	4800
75.6	9.2	-2	4900
77.5	9.4	-2	5000
79.7	9.7	-2	5100
81.8	9.9	-2	5200
83.9	10.1	-2	5300
88	10.4	-2	5400
88.2	10.6	-2	5500
90.4	10.9	-2	5600
92.6	11.1	-2	5700
94.9	11.3	-2	5800
97.1	11.6	-2	5900
99.5	11.9	-2	6000
101.8	12.1	-2	6100

104.2	12.4	-3	6200
106.6	12.6	-3	6300
109	12.9	-3	6400
111.4	13.1	-3	6500
113.9	13.4	-3	6600
116.4	13.7	-3	6700
119	13.9	-3	6800
121.6	14.2	-3	6900
124.2	14.5	-3	7000
126.8	14.8	-3	7100
129.5	15.1	-3	7200
132.2	15.3	-3	7300
135	15.6	-3	7400
137.8	15.9	-4	7500
140.6	16.2	-4	7600
143.5	16.5	-4	7700
146.4	16.8	-4	7800
149.3	17.1	-4	7900
152.3	17.4	-4	8000
155.3	17.7	-4	8100
158.3	18	-4	8200
161.4	18.3	-4	8300
164.5	18.6	-4	8400
167.7	18.9	-5	8500
170.9	19.2	-5	8600
174.1	19.5	-5	8700
177.4	19.9	-5	8800
180.7	20.2	-5	8900
184.1	20.5	-5	9000
187.4	20.8	-5	9100
190.9	21.2	-5	9200
194.4	21.5	-5	9300
197.9	21.8	-5	9400
201.4	22.2	-6	9500
205.1	22.5	-6	9600
208.7	22.8	-6	9700
212.4	23.2	-6	9800
216.1	23.5	-6	9900
219.9	23.8	-6	10000

223.7	24.2	-6	10100
227.6	24.5	-6	10200
231.5	24.9	-7	10300
235.4	25.2	-7	10400
239.4	25.6	-7	10500
243.4	25.9	-7	10600
247.4	26.3	-7	10700
251.5	26.7	-7	10800
255.7	27	-7	10900
259.8	27.4	-7	11000
264.1	27.7	-8	11100
268.3	28.1	-8	11200
272.6	28.5	-8	11300
277	28.8	-8	11400
281.3	29.2	-8	11500
285.8	29.6	-8	11600
290.3	30	-9	11700
294.8	30.4	-9	11800
299.4	30.7	-9	11900
304	31.1	-9	12000
308.7	31.5	-9	12100
313.4	31.9	-9	12200
318.1	32.5	-10	12300
322.9	32.7	-10	12400
327.8	33.1	-10	12500
332.7	33.5	-10	12600
337.6	33.9	-10	12700
342.6	34.3	-10	12800
347.7	34.7	-11	12900
352.8	35.1	-11	13000
358	35.5	-11	13100
363.2	35.9	-11	13200
368.4	36.3	-11	13300
373.7	36.8	-12	13400
379.1	37.2	-12	13500
384.5	37.6	-12	13600
390	38	-12	13700
395.6	38.5	-12	13800
401.2	38.9	-13	13900

406.9	39.3	-13	14000
412.7	39.8	-13	14100
418.5	40.2	-13	14200
424.4	40.7	-13	14300
430.4	41.2	-13	14400
436.4	41.6	-14	14500
442.6	42.6	-14	14600
448.8	42.6	-14	14700
455	43	-15	14800
461.4	43.5	-15	14900
467.8	44.8	-15	15000
474.4	44.5	-16	15100
481.1	45	-16	15200
487.9	45.5	-16	15300
494.8	46	-16	15400
501.8	46.5	-17	15500
509	47.1	-17	15600
516.3	47.6	-17	15700
523.8	48.2	-18	15800
531.5	48.7	-18	15900
539.3	49.3	-18	16000
547.3	49.9	-19	16100
555.5	50.5	-19	16200
563.9	51.1	-19	16300
572.5	51.7	-20	16400
581.3	52.3	-20	16500
590.4	53	-21	16600
599.9	53.6	-21	16700
609.9	54.3	-22	16800
619.8	55.1	-22	16900
630.5	55.8	-23	17000
641.7	56.6	-23	17100
653.6	57.4	-24	17200
666.3	58.3	-25	17300
680.2	59.3	-25	17400
695.5	60.3	-26	17500

ANGULO DE ELEVACIÓN	DURACIÓN DEL TRAYECTO	DERIVACIÓN	ALCANCE	CARGA
1	0.1	0	100	**9 bis**
1.9	0.3	0	200	
2.9	0.4	0	300	
3.9	0.6	0	400	
4.9	0.7	0	500	
5.9	0.8	0	600	
6.9	1	0	700	
7.9	1.1	0	800	
9	1.3	0	900	
10	1.4	0	1000	
11.1	1.6	0	1100	
12.1	1.7	0	1200	
13.2	1.9	0	1300	
14.3	2	0	1400	
15.4	2.2	0	1500	
16.5	2.3	0	1600	
17.6	2.5	0	1700	
18.8	2.7	0	1800	
19.9	2.8	0	1900	
21.1	3	0	2000	
22.2	3.1	0	2100	
23.4	3.3	0	2200	
24.6	3.5	-1	2300	
25.8	3.6	-1	2400	
27.1	3.8	-1	2500	
28.3	4	-1	2600	
29.5	4.1	-1	2700	
30.8	4.3	-1	2800	
32.1	4.5	-1	2900	
33.4	4.6	-1	3000	
34.7	4.8	-1	3100	
36	5	-1	3200	
37.3	5.2	-1	3300	
38.7	5.4	-1	3400	
40	5.5	-1	3500	
41.4	5.7	-1	3600	
42.8	5.9	-1	3700	

44.2	6.1	-1	3800
45.6	6.3	-1	3900
47.1	6.5	-1	4000
48.5	6.6	-1	4100
50	6.8	-1	4200
51.5	7	-1	4300
53	7.2	-1	4400
54.5	7.4	-1	4500
56.1	7.6	-1	4600
57.6	7.8	-1	4700
59.2	8	-1	4800
60.8	8.2	-1	4900
62.4	8.4	-1	5000
64	8.6	-1	5100
65.7	8.8	-2	5200
67.4	9.1	-2	5300
69.1	9.3	-2	5400
70.8	9.5	-2	5500
72.5	9.7	-2	5600
74.3	9.9	-2	5700
76	10.1	-2	5800
77.8	10.3	-2	5900
79.6	10.6	-2	6000
81.5	10.8	-2	6100
83.3	11	-2	6200
85.2	11.3	-2	6300
87.1	11.5	-2	6400
89.1	11.7	-2	6500
91	11.9	-2	6600
93	12.2	-2	6700
95	12.4	-2	6800
97	12.7	-2	6900
99	12.9	-2	7000
101.1	13.1	-2	7100
103.2	13.4	-3	7200
105.3	13.6	-3	7300
107.5	13.9	-3	7400
109.7	14.1	-3	7500
111.9	14.4	-3	7600

114.1	14.7	-3	7700
116.4	14.9	-3	7800
118.7	15.2	-3	7900
121	15.4	-3	8000
123.3	15.7	-3	8100
125.7	16	-3	8200
128.1	16.3	-3	8300
130.6	16.5	-3	8400
133	16.8	-3	8500
135.5	17.1	-4	8600
138.1	17.4	-4	8700
140.6	17.6	-4	8800
143.2	17.9	-4	8900
145.8	18.2	-4	9000
148.5	18.5	-4	9100
151.2	18.8	-4	9200
153.9	19.1	-4	9300
156.7	19.4	-4	9400
159.4	19.7	-4	9500
162.3	20	-4	9600
165.1	20.3	-5	9700
168	20.6	-5	9800
170.9	20.9	-5	9900
173.9	21.2	-5	10000
176.9	21.5	-5	10100
179.9	21.8	-5	10200
183	22.2	-5	10300
186.1	22.5	-5	10400
189.2	28.2	-5	10500
192.4	23.1	-5	10600
195.6	23.5	-6	10700
198.9	23.8	-6	10800
202.2	24.1	-6	10900
205.5	24.4	-6	11000
208.8	24.8	-6	11100
212.2	25.1	-6	11200
215.7	25.4	-6	11300
219.2	25.8	-6	11400
222.7	26.1	-6	11500

226.2	26.5	-7	11600
229.8	26.8	-7	11700
233.5	27.2	-7	11800
237.1	27.5	-7	11900
240.8	27.9	-7	12000
244.5	28.2	-7	12100
248.3	28.6	-7	12200
252.1	28.9	-8	12300
255.9	29.3	-8	12400
259.8	29.6	-8	12500
263.7	30	-8	12600
267.7	30.3	-8	12700
271.6	30.7	-8	12800
275.7	31.1	-8	12900
279.7	31.4	-8	13000
283.8	31.8	-9	13100
288	32.2	-9	13200
292.2	32.6	-9	13300
296.4	32.9	-9	13400
300.7	33.3	-9	13500
305	33.7	-9	13600
309.4	34.1	-10	13700
313.7	34.5	-10	13800
318.2	34.9	-10	13900
322.6	35.2	-10	14000
327.1	35.6	-10	14100
331.7	36	-10	14200
336.3	36.4	-11	14300
340.9	36.8	-11	14400
345.6	37.2	-11	14500
350.3	37.6	-11	14600
355.1	38	-11	14700
360	38.4	-12	14800
364.8	38.9	-12	14900
369.7	39.3	-12	15000
374.7	39.7	-12	15100
379.7	40.1	-12	15200
384.7	40.5	-12	15300
389.8	41	-13	15400

395	41.4	-13	15500
402	41.8	-13	15600
405.5	42.2	-13	15700
410.9	42.7	-14	15800
416.3	43.1	-14	15900
421.7	43.6	-14	16000
427.3	44	-14	16100
432.9	44.5	-14	16200
438.5	44.9	-15	16300
444.2	45.4	-15	16400
450	45.9	-15	16500
455.8	46.3	-15	16600
461.7	46.8	-16	16700
467.7	47.3	-16	16800
473.8	47.8	-16	16900
479.9	48.3	-16	17000
486.2	48.8	-17	17100
492.6	49.3	-17	17200
499	49.8	-17	17300
505.6	50.3	-18	17400
512.3	50.8	-18	17500
519.2	51.4	-18	17600
526.2	51.9	-19	17700
533.3	52.5	-19	17800
540.5	53	-19	17900
547.9	53.6	-20	18000
555.4	54.2	-20	18100
563.1	54.8	-20	18200
570.9	55.4	-21	18300
579	56	-21	18400
587.2	56.6	-22	18500
595.7	57.3	-22	18600
604.4	57.9	-22	18700
613.4	58.6	-23	18800
622.7	59.3	-23	18900
632.4	60.1	-24	19000
642.6	60.8	-24	19100
653.2	61.6	-25	19200
664.4	62.5	-26	19300

676.4	63.4	-26	19400	
689.2	64.3	-27	19500	
703.2	65.3	-28	19600	
718.5	66.5	-29	19700	
735.7	67.7	-30	19800	
756.8	69.2	-31	19900	
787.7	71.4	-34	20000	
826	74.1	-37	20100	

MANUAL DE USUARIO

Este software fue desarrollado por el Centro de Investigaciones del Ejercito (CICTE), para contribuir con sus conocimientos tecnológicos con las Fuerzas Armadas del Ecuador.

HACERCA DE ESTE MANUAL

Este manual especifica el uso de cada pantalla, y además provee de posibles errores que el usuario pueda cometer con sus respectivas soluciones.

El software consta de cinco ventanas las cuales están concatenadas y enlazadas entre ellas, por lo cuál, se aconseja al operador que si existen problemas con el programa cierre todas las ventanas abiertas y vuelva a comenzar desde la primera ventana.

Las ventanas en el Sistema SCADA son las siguientes:

- Presentación del software y elección del puerto de comunicación

Figura 1: Pantalla de la presentación del Software

- Detección de los Sistemas Esclavos (PLC)

Figura 2: Pantalla de la detección de los Sistemas Esclavos

- Pantalla de la DGT

Figura 3: Pantalla de la DGT

- Ventana del HMI (Interfase hombre máquina)

Figura 4: Pantalla del monitoreo de los sensores

- Sistema de cálculo del Centro Directo de Tiro (CDT)

Figura 5: Pantalla del CDT

PRESENTACIÓN DEL SOFTWARE Y ELECCIÓN DEL PUERTO DE COMUNICACIÓN

Esta pantalla esta diseñada principalmente para la presentación del programa del sistema SCADA. Adicionalmente permite configurar el puerto de comunicación que el computador principal utilizará para el intercambio de información con el

controlador. Para comenzar la comunicación entre estos dos dispositivos se debe seleccionar cuidadosamente el puerto de comunicación que tendrá el computador, caso contrario no existirá comunicación entre estos dispositivos.

En la figura 6 se muestra como se selecciona el puerto serial para el ordenador, una vez realizada esta acción, se debe proceder a dar un clic izquierdo del ratón en el botón ACEPTAR.

Figura 6: Selección del Puerto de Comunicación

Nota*: *En caso de que el computador no posea puertos seriales, el usuario debe usar un convertidor USB a serial y observar el COMM del puerto USB del ordenador. No es necesario configurar el radio MODEM ni el software en este caso.

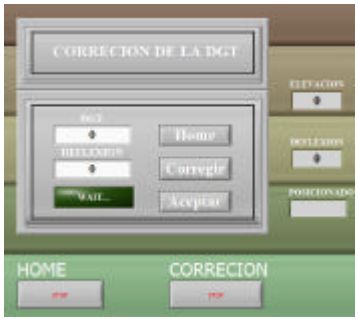
DETECCION DE LOS SISTEMAS ESCLAVOS (PLC)

Esta pantalla nos permite observar que sistemas esclavos están transmitiendo información al computador principal, en caso de que el operador observe que algún dispositivo no está transmitiendo se aconseja que verifique las conexiones y alimentación de los radio modems y el puerto de comunicación del ordenador. Si el problema persiste se debe verificar la velocidad de transmisión del puerto del computador en la plataforma de Labview.

Nota: *En caso de modificar la tasa de transmisión, se recomienda reiniciar al computador y los Sistemas Esclavos para que cambios al puerto de comunicación no provoque errores de transmisión.*

El operador del software del Sistema SCADA al verificar que, por lo menos un indicador este activado de los Sistemas Esclavos puede continuar a la siguiente ventana de programa al dar un clic en el botón Aceptar.

PANTALLA DE LA DGT (DIRECCIÒN GENERAL DE TIRO)



Esta ventana permite realizar el posicionamiento inicial del tubo del cañón que son, 300 milésimas en elevación y 150 milésimas en dirección. Además permite realizar correcciones en la DGT para que el obus este listo para realizar correcciones para el disparo.

Para colocar el tubo del cañón en su posición inicial se debe presionar el botón de HOME, luego de esperar aproximadamente un minuto, los sensores marcaran 300 en elevación y -150 en dirección, (valor en milésimas de la distancia de la posición de "guardado" y la posición donde el tubo del cañón se pone en paralelo con las orugas.), luego de lo cual en posicionado aparecerá la palabra LISTO y el indicador se colocará en rojo, como lo muestra la figura 16

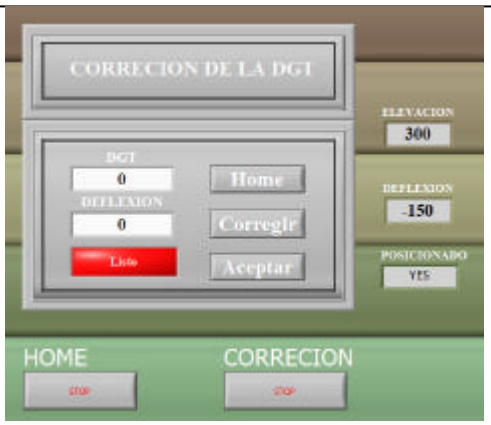
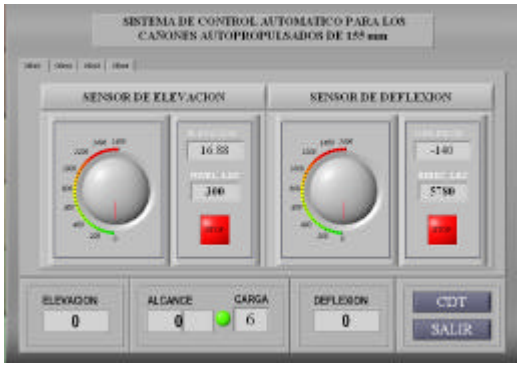


Figura 7: El tubo de cañón ya esta en su posición inicial

El operador para ubicar el tubo en la DGT debe ingresar su valo r teórico (DGT) y su valor real (DEFLEXIÓN) y dar un clic en el botón de Corregir y automáticamente corregirá la posición del tubo para comenzar a realizar el primer tiro con los valores calculados en el CDT. En la figura 17 se muestra como esta corrigiendo al valor de la DGT.

Figura 8: Corrigiendo errores de posicionamiento

VENTANA DEL HMI



Esta ventana sirve para que el operador pueda monitorear en que posición está ubicado el tubo del cañón. Además posee indicadores en los cuales muestra los valores calculados en el CDT al cual debe llegar a posicionarse el tubo cañón.

En la figura 18 se explica cada indicador y su función en el monitoreo de cada sensor.

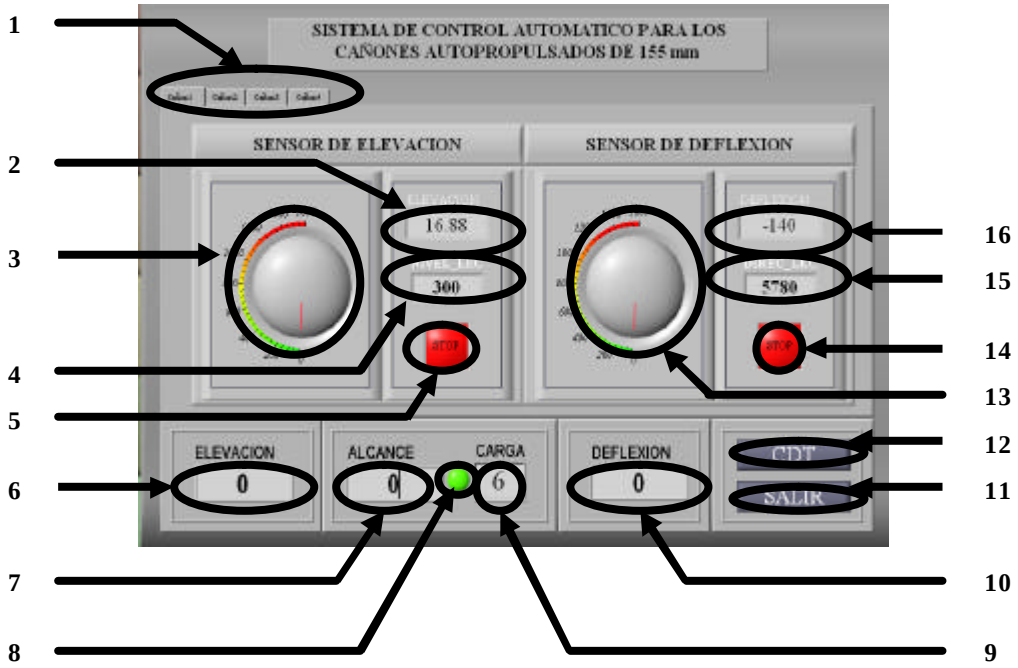


Figura 9: HMI

1. Selecciona el cañón que quiere monitorear de la batería
2. Indica el valor del sensor en elevación en datos reales (grados)

3. Muestra en forma gráfica en donde está ubicado el tubo del cañón en milésimas en elevación.
4. Muestra en forma de digital el valor en milésimas del tubo del cañón.
5. Indica cuando el tubo del cañón esta en posiciones de riesgo en elevación
6. Muestra el valor calculado en el CDT en elevación
7. Indica cual va hacer el alcance de la carga
8. El indicador se acciona cuando se posiciona el tubo del cañón
9. Muestra que carga debe utilizar para el disparo
10. Muestra el valor calculado en el CDT en dirección
11. Salir del software
12. Cambia a la pantalla del CDT
13. Muestra en forma gráfica en donde está ubicado el tubo del cañón en milésimas en deflexión
14. Indica cuando el tubo del cañón esta en posiciones de riesgo en dirección
15. Muestra en milésimas la posición del tubo en deflexión
16. Muestra en milésimas los datos del encoder.

VENTANA DE CÁLCULOS PARA EL CDT

Esta ventana está basada en la tesis SPD-155, la cual permite realizar los cálculos de tiro, para posicionar el tubo del cañón y observar que alcance puede tener dependiendo de la carga que los operadores tienen a disposición. Adicionalmente en esta ventana se tiene la opción de transporte de las coordenadas balísticas, el ingreso de datos de la ubicación de la batería la cual puede ser manual o en forma automática con la ayuda del GPS, la orden de enviar a posicionar el tubo del cañón, un botón para regresar al HMI para monitorear los sensores y la opción de salir la cual posiciona el tubo del cañon en su posición para guardado..

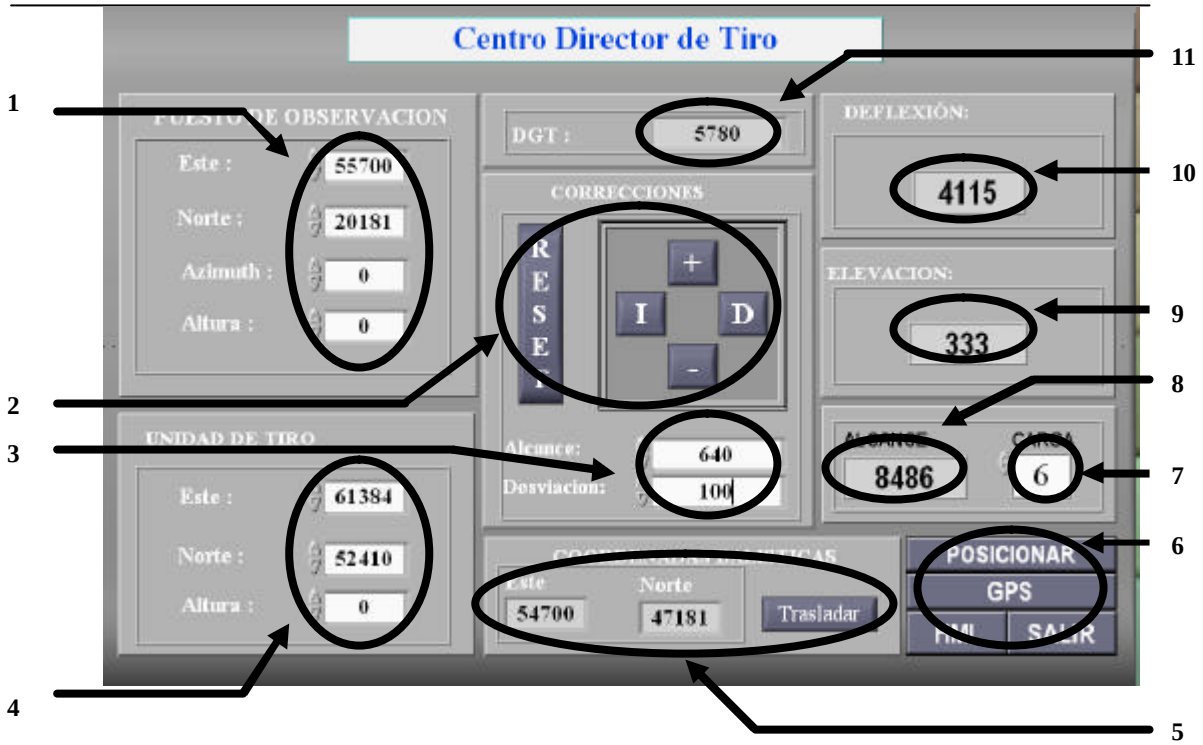


Figura 10: CDT

1. Datos de ubicación del observador avanzado de la posición del blanco
2. Botones para realizar correcciones de tiro
3. Datos de correcciones enviados por el observador avanzado
4. Datos de ubicación de la batería
5. Traslado de las coordenadas balísticas
6. Botones para posicionar, ingreso de los datos del GPS, cambiar a la interfase HMI y para salir y guardar el tubo del cañón
7. Selecciona la carga
8. Indica el alcance del disparo
9. Muestra la elevación del tubo del cañón
10. Indica la dirección del tubo del cañón

INCLINÓMETRO

RDI Series
Installation & Mounting Instructions

- 2 -

RIEKER ELECTRONICS INC

Figure 1: Standard RDI Display (inches / mm)

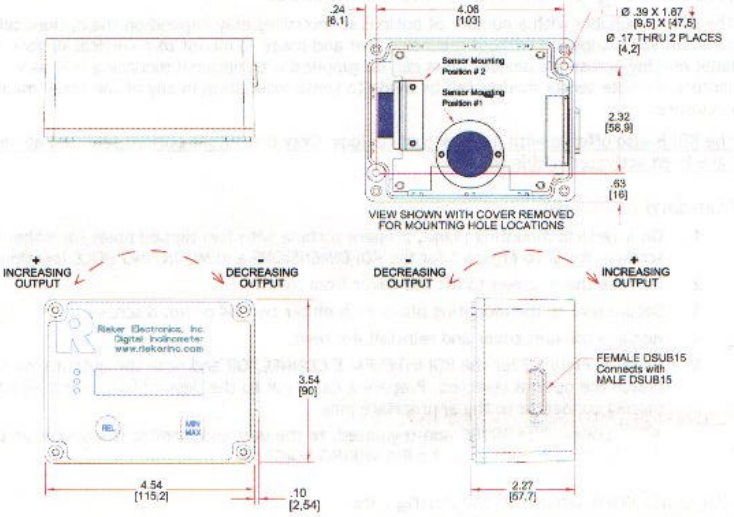
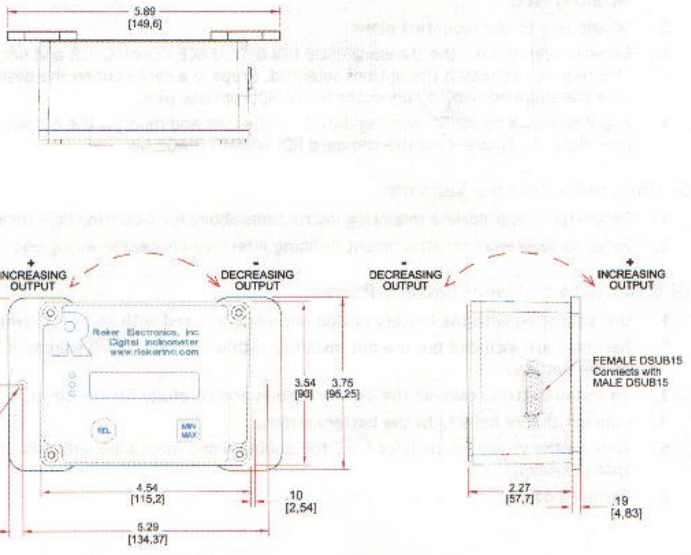


Figure 2: Standard RDI Display with Mounting Feet (inches / mm)



PO Box 127 • 777 Henderson Blvd • Park Square North Bldg • Bay #7 • Folcroft • PA • 19032 • USA
voice: 610-534-9000 fax: 610-534-4670 email: support@riekerinc.com web: www.riekerinc.com

FORM Number: RD001E 08/03 REVISED: 12/03

RIEKER ELECTRONICS INC

Figure 3: RDI DSUB Connector Wiring

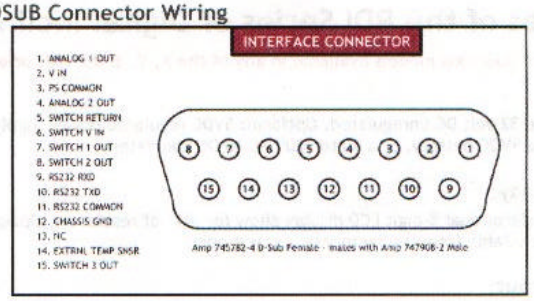
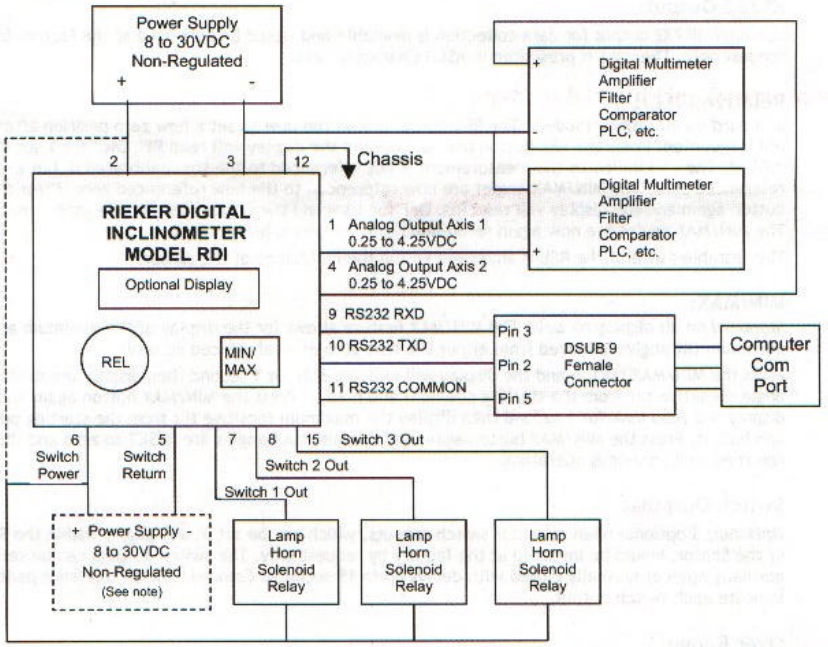


Figure 4: RDI Wiring Diagram



Notes:
1. Switch Power can be connected to the same power source that powers the unit or it can be connected to an isolated power source. If connected to the unit power source, a separate switch power return must still be connected.
2. The RS232 Output is continuously generated in approximately 250msec intervals. The output is transmitted one set of readings per line. Single axis units will display one angle per line; dual axis units will display two angles per line with axis 1 first then axis 2. It is transmitted at a 9600-baud rate, 8 data bits, No parity, 1 stop bit.

PO Box 127 • 777 Henderson Blvd • Park Square North Bldg • Bay #7 • Folcroft • PA • 19032 • USA
voice: 610-534-9000 fax: 610-534-4670 email: support@riekerinc.com web: www.riekerinc.com

RDI Series

Installation & Mounting Instructions

RIEKER ELECTRONICS INC

Features of the RDI Series of Digital Inclinometers

- Single and dual axis models available in any of the X, Y, and Z-axis orientations

Power:

Standard: 8 to 32 volt DC unregulated. *Optional:* 5VDC regulated power supply input, a cigarette lighter plug, a 9VDC battery, or a 90 to 220V AC to DC converter.

Digital Display:

Standard: Single or dual 8-digit LCD display allow for .01° of resolution. *Optional:* Alternate display settings are available from the factory by request only.

Analog Output:

Optional: 0 to 5VDC output is available and would be activated at the factory by request only.

RS232 Output:

Optional: RS232 output for data collection is available and would be activated at the factory by request only. The data is presented in ASCII Character Text.

Relative (REL):

Standard on all display models: The REL button allows the user to set a new zero position after the RDI is mounted. Press the REL button and release and the display will read REL ON * for 1 sec then 0.00° *. The (*) indicates the measurement is not referenced to the true calibrated 0, but a referenced zero. The MIN/MAX angles are now referenced to the new referenced zero. Press the REL button again and the display will read REL OFF for 1 second then return to the true calibrated zero. The MIN/MAX angles are now again referenced to the true calibrated zero.

The settable range of the REL is anywhere within the Full Range of the sensor.

MIN/MAX:

Standard on all display models: The MIN/MAX feature allows for the display of the minimum and maximum tilt angles achieved from either the true zero or a referenced zero tilt.

Press the MIN/MAX button and the display will indicate MIN for 1 second then display the minimum angle (negative tilt from the starting position) and hold it. Press the MIN/MAX button again and the display will read MAX for 1 second then display the maximum (positive tilt from the starting position) and hold it. Press the MIN/MAX button again and the MIN/MAX angles are RESET to zero and the RDI is returned to its previous operation.

Switch Outputs:

Optional: 3 optional open collector switch outputs, which can be set at any angles within the Range of the Sensor, would be installed at the factory by request only. The switch outputs can be set as normally open or normally closed with delays up to 15 seconds. Colored LEDs on the front panel indicate each switch output.

Over Range:

Standard on all display models: When the RDI is tilted beyond the Full Range of the sensor the display will indicate an OVER RANGE condition. Once the RDI is tilted back to its operating range the display will indicate normal operation again.

PO Box 127 • 777 Henderson Blvd • Park Square North Bldg • Bay #7 • Folcroft • PA • 19032 • USA
voice: 610-534-9000 fax: 610-534-4670 email: support@riekerinc.com web: www.riekerinc.com

RDI Series: RS232
Supplemental Information

RIEKER ELECTRONICS INC

Connecting an RDI to a PC (RS232 Serial Output)

If using a laptop computer in conjunction with an RDI Digital Inclinometer a modified power cord is needed, which is split to provide both a serial port connector and a power cable with pigtail leads.

1. Connect the serial port connector to the COM1 (or COM2) port on the computer for RS232 communication. Make sure to use the serial port and not the parallel port.
2. Connect power cable RED LEAD to 8-30 VDC supply voltage and the BLACK LEAD to power supply ground. Apply power.
3. Click on the Windows START menu on the computer. Then click on PROGRAMS > ACCESSORIES > HYPERTERMINAL > HYPER TERMINAL.
4. A "Connection Description" window should pop up. Enter a filename under "Name". Keep the default for Icon. Click OK.
5. A "Connect To" window will pop up that will ask you to enter details for the phone number that you want to dial. In the "Connect Using" space select the COM port you are using - either COM1 or COM2. Click OK.
6. A "COM1 OR COM2 Properties" window will pop up. Select the following Port Settings:

RS232 Port Settings Table:	
BITS PER SECOND:	9600
DATA BITS:	8
PARITY:	NONE
STOP BITS:	1
FLOW CONTROL:	NONE

   Click OK.
7. At this point HyperTerminal should be connected. The RDI indicator should be recording a continuous stream of angles to the laptop. The angles are output approximately every 250msec - one angle per line. (If data is not being processed click on CALL > CONNECT.) To stop the flow of data, click on CALL > DISCONNECT. Data flow can also be controlled through the RDI unit by the MIN/MAX button. When this button is pressed, data from the unit is temporarily halted until it is returned back into normal mode. In practice, we recommend you press the MIN/MAX button two times to stop data collection - this will display the last MAX reading recorded. When ready to start recording data press the MIN/MAX button one more time to "Reset" the unit. It will resume sending data to the computer.
8. To save data in HyperTerminal, click on FILE > SAVE. Click SAVE AS to change the data file name.
9. To transfer data to an EXCEL sheet click on the following in Hyperterminal: EDIT > SELECT ALL > EDIT > COPY. Open up EXCEL, create a new sheet and click on EDIT > PASTE. From there the data can be graphed.

PO Box 127 • 777 Henderson Blvd • Park Square North Bldg • Bay #7 • Folcroft • PA • 19032 • USA
voice: 610-534-9000 fax: 610-534-4670 email: support@riekerinc.com web: www.riekerinc.com

BRÚJULA

Honeywell

SENSOR PRODUCTS

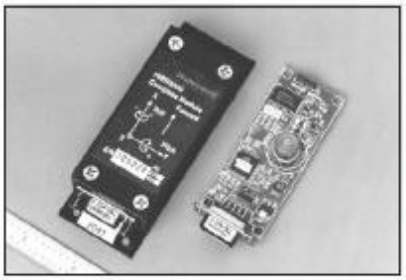
APPLICATIONS

- Oceanographic
  - Marine Compassing
  - Positioning of Buoys, Underwater Structures
- Drilling
  - Down Hole and Directional
- Attitude Reference
- Heading
  - Navigation of Unmanned Vehicles
  - Avionic Compassing
- Integration with GPS
  - Dead Reckoning
- Satellite Antenna Positioning
- Laser Range Finders
  - Surveying Applications

Digital Compass Module

HMR3000

Electronic compass module that provides heading, pitch and roll output for navigation and guidance systems. Honeywell's solid state magnetoresistive sensors make this strapdown compass both rugged and reliable. This compass provides fast response time up to 20 Hertz and high heading accuracy of 0.5° with 0.1° resolution.



FEATURES AND BENEFITS

Fast Response Time	Built with solid state magnetic sensors and no moving parts improves response time, allowing faster updates compared to gimballed fluxgates.
Small Size	Available as a circuit board 1.2 x 2.95 inches, weighing less than one ounce, or in an aluminum enclosure.
Low Power	Operates with less than 35 mA, allowing for long operation with a battery.
High Accuracy	Accuracy better than 0.5° with 0.1° resolution for critical positioning applications.
Wide Tilt Range	Tilt range of ±40° for both the roll and pitch allows operation for most applications.
Hard Iron Compensation	Calibration routines to compensate for distortion due to nearby ferrous objects and stray fields, such as vehicles.
User Configurable Features	User settings of baud rate, update rate, output format, units, filter settings, deviation angles, alarms and warnings are stored internally in non-volitile memory.

HMR3000

SENSOR PRODUCTS

INTERFACE SIGNAL DESCRIPTIONS

Communication
HMR3000 communicates with an external host via RS-232 or RS-485 electrical standard through simple ASCII character strings. ASCII characters are transmitted and received using 1 Start bit, 8 Data bits, (LSB first, MSB always 0), no parity, and 1 Stop bit. Baud rate is user configurable to 1200, 2400, 4800, 9600, 19,200 or 38,400. HMR3000 responds to all valid inputs received with correct checksum value.

Compass Output
HMR3000 can output three NMEA standard sentences, (HDG, HDT and XDR), three proprietary sentences (HPR, RCD and CCD), and an ASCII heading output for a digital display. HDG, HDT and HPR are the most commonly used sentences; the formats are given below.

$HCHDG, Heading, Deviation, Variation
$HCHDG,85.5,0.0,E,0.0,E*77
$HCHDT, Heading, True
$HCHDT,271.1,T*2C
$PTNTHPR, Heading, Pitch and Roll
$PTNTHPR, Heading,Heading Status,Pitch,Pitch Status,Roll,Roll Status*hh<cr><lf>
$PTNTHPR,85.9,N,-0.9,N,0.8,N*2C

The table shows pin assignments for the 9-pin D-shell connector. Power input can be either regulated 5V dc or unregulated 6V to 15V. Only one of the two power pins (9 or 8) should be connected in a given installation.

Name	In/Out	Pin	Description	Typ	Min (1)	Max (1)	Units
TxD / B	Out	2	RS-232 transmit out / RS-485	—	-18	18	V dc
RxD / A	In	3	RS-232 receive in / RS-485	—	-18	18	V dc
GND	In	5	Power and signal common	—			
6-15V	In	9	Unregulated power input	6 – 15	0	30	V dc
5V	In	8	Regulated power input	5 ± 5%	0	7.5	V dc
Oper / Calib (2)	In	1	Operate / Calibrate (3) input (open = Operate)	0 – 5	-20	20	V dc
Run / Stop (2)	In	6	Run / Stop (3) input (open = Run)	0 – 5	-20	20	V dc
Ready / Sleep (2)	In	4	Ready / Sleep (3) input (open = Ready)	0 – 5	-20	20	V dc
Cont / Reset (2)	In	7	Continue / Reset (3) input (open = Continue)	0 – 5	-20	14	V dc

(1) Absolute maximum ratings.
(2) Sink current requirement; 200 (Typ) 400 (Max) µA.
(3) Open input = high logic state.

HMR3000

SENSOR PRODUCTS

SPECIFICATIONS

	Parameter	Value	Comments
Heading			
	Accuracy (1)	< 0.5° RMS (2) < 1.5° RMS	Dip < 50°, Tilt <20° * Dip < 75°, Tilt <20° *
	Repeatability (3) (4)	± 0.3°	
	Resolution	0.1°	
	Units	degrees / mils	User selectable
Pitch and Roll			
	Range	± 40°	
	Accuracy	± 0.4° ± 0.6°	Tilt < 20° Tilt ≥ 20° *
	Repeatability (3) (4)	± 0.2°	
	Resolution	0.1°	
	Units	degree/ mils	User selectable
Magnetic Field (3)			
	Dynamic Range	± 1.0 Gauss max	± 0.5 Gauss range
	Resolution	1 mGauss	
Electrical (4)			
	Supply Voltage	5.0 Vdc regulated 6 - 15 Vdc unregulated	
	Power	35 mA @ 6 Vdc 13 mA 2.0 mA	Normal operation STOP Mode SLEEP Mode
Interface			
	Serial	RS-232 RS-485	Half Duplex
	Baud Rate	1200 to 38400 bps	
	Standard	NMEA 0183	
	Update Modes	Continuous Strobed	1/min to 20 Hz per sentence selectable averaging
Physical (4)			
	Weight	0.75 oz (22 g) 3.25 oz (92 g)	Circuit card only Housed
	Dimensions	1.2 x 2.95 x 0.760 1.5 x 4.2 x 0.88	Circuit card Housed compass
Environment (5)			
	Operating Temp	-20 to 70° C	
	Storage Temperature	-35 to 100° C	
	Shock	30 inch drop	MIL-STD-810E; TM 516.4
	Vibration	20 - 2000 Hz Random 2 hrs/axis	MIL-STD-810E; TM 514.4
Manufacturing			
	PCB	IPC 6012	
	Assembly	IPC 610	Class II or better

1. Heading accuracy assumes the Earth's magnetic field is only disturbed by hard iron fields, and has been compensated through calibration.
2. Calculated values.
3. Guaranteed by characterization or design.
4. Typical
5. Meet or exceed.

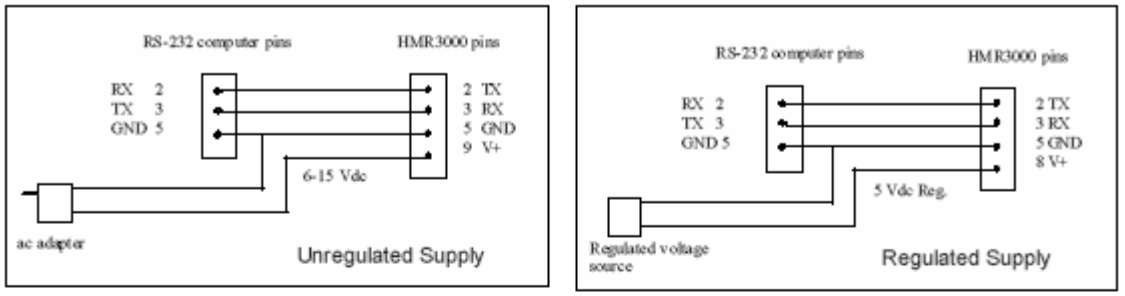
* Device orientation not to exceed 75° during operation or storage—may cause temporary loss of accuracy.

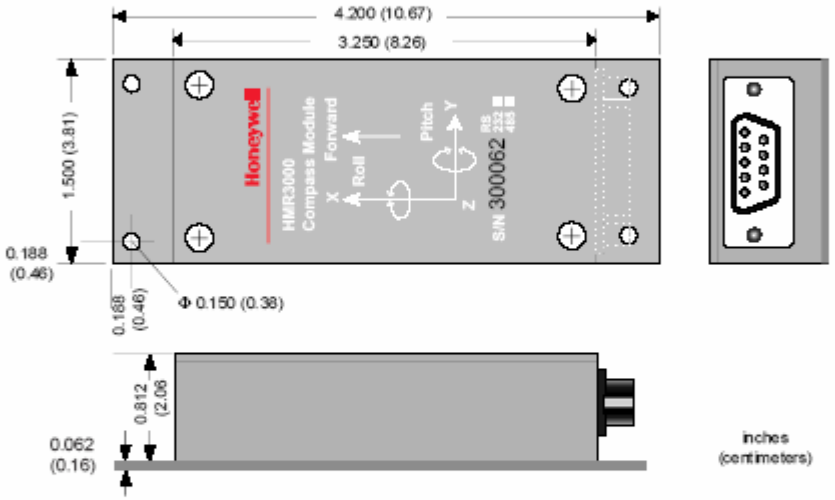
HMR3000

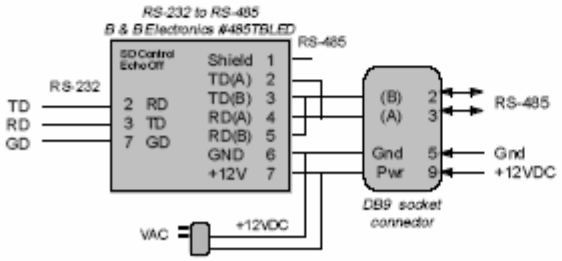
SENSOR PRODUCTS

SPECIFICATIONS

HMR3000 CONNECTION DIAGRAM—COMPUTER RS232 TO HMR3000







ORDERING INFORMATION

Type	Output	Enclosure
HMR3000-Demo-232*	RS232	
HMR3000-D00-232	RS232	None
HMR3000-D21-232	RS232	Extended Base
HMR3000-D00-485	RS485	None
HMR3000-D21-485	RS485	Extended Base

*Development Kit includes one module in aluminum enclosure, cabling with power supply, demonstration software for PC running Windows™ and User's Manual.

GPS Y ALTÍMETRO

Physical

Case:	Fully-gasketed, high-impact plastic alloy, waterproof to IPX7 standards (waterproof to one meter for 30 minutes)
Size:	4.4"H x 2"W x 1.2"D
Weight:	Approx. 5.3 ounces (150g) w/batteries
Temperature Range:	5° to 158°F (-15° to 70°C) (operating)

Performance

Receiver:	Differential-ready, 12 parallel channel
Acquisition time:	Approx. 15 seconds (warm start) Approx. 45 seconds (EZinit/cold start) Approx. 5 minutes (First Time/AutoLocate™)
Update Rate:	1/second, continuous
GPS Accuracy:	<15 meters (49 ft) RMS, 95% typical[2]
DGPS (USGC) Accuracy:	3-5 meters (10-16 ft), 95% typical with DGPS corrections[1]
DGPS (WAAS) Accuracy:	<3 meters (10 ft), 95% typical with DGPS corrections[1]
Velocity Accuracy:	0.05 meter/sec steady state
Dynamics:	Performs to specifications to 6 g's
Interfaces:	NMEA 0183, RTCM 104 (for DGPS corrections) and RS-232 for PC interface
Antenna:	Built-In patch

Power

Input:	Two 1.5-volt AA batteries[3]
Power Consumption:	0.5 watts max.
Battery Life:	Up to 12 hours of typical use in 'Battery Saver' mode[4]

Specifications are subject to change without notice.

[1]With optional GARMIN Differential Beacon Receiver Input (such as GARMIN GBR 21 or 23).

[2]Subject to accuracy degradation to 100m 2DRMS under the U.S. DoD-imposed Selective Availability program.

[3]The temperature rating for the eTrex Vista may exceed the usable range of some batteries. Alkaline batteries can rupture at high temperatures. External power can only be applied using the GARMIN Auto Power Adapter or PC Interface Cable with Auto Power Adapter. These cables contain a 12 VDC to 3 VDC voltage regulator. Modifications or other applications will void the product warranty

[4]Alkaline batteries lose a significant amount of their capacity as temperature decreases. Use lithium batteries when operating the eTrex Vista in below-freezing conditions. Extensive use of screen backlighting and the electronic compass will significantly reduce battery life. Different brands of batteries will vary in performance.

Interface formats are selected from the Setup 'Interface Page' on page 56 of this manual. The input/output lines on your eTrex Vista unit are RS-232 compatible, allowing easy interface to a wide range of external devices, including PC's, differential beacon receivers, marine autopilots and /or a second GPS receiver.

The NMEA 0183 version 2.3 interface format is supported by the eTrex Vista and enables the unit to drive up to three NMEA devices.

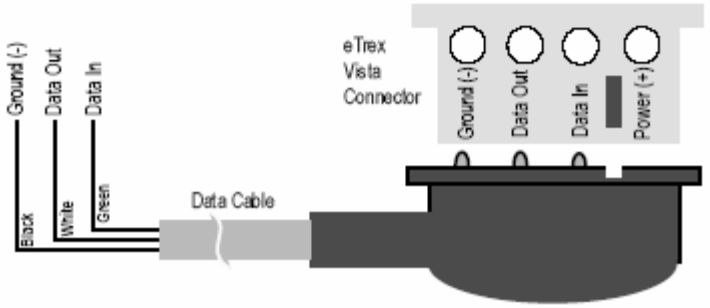
NMEA 0183 Version 2.3 Approved Sentences:
GPGGA, GPGLL, GPGSA, GPGSV, GPRMB, GPRMC, GPRTE, GPWPL, GPBOD

GARMIN Proprietary Sentences:
PGRME (estimated error), PGRMZ (altitude), PSLIB (beacon receiver control)

DGPS (Differential GPS) corrections are accepted in RTCM SC-104 version 2.0 format through the 'Data In' line. The GARMIN GBR 21 is the recommended beacon receiver for use with the eTrex Vista. Other beacon receivers with the correct RTCM format may be used, but may not correctly display status or allow tuning control from the GPS unit.

The eTrex Vista may be hard-wired to a serial connector using the Data Cable (see Appendix C for ordering information). Refer to the wiring diagram and the eTrex Vista unit data connection illustrated below.

The PC Interface Cable provided with this unit allows you to connect your eTrex Vista to a PC-compatible computer's serial port.



DL205 WinPLC: Windows® CE-based CPU

DL205 WinPLC
H2-WPLC1-EN
H2-WPLC2-EN
H2-WPLC3-EN
PC-WPLC-START*

Specifications	H2-WPLC1-EN	H2-WPLC2-EN	H2-WPLC3-EN
Processor	Hitachi SH3 Series 7708 Processor		
Processor Speed	40 Mhz	100 Mhz	100 Mhz
Pre-loaded Software	Runtime engine compatible with Entivity Studio or Think & Do Live		
Memory	4MB FLASH EE ROM, 2MB RAM, 64kB battery-backed RAM10Mbps		8MB FLASH EE ROM, 8MB RAM, 64kB battery-backed RAM10Mbps
Indicators	Power, Link/Act, Run, Error		
Local I/O Points	256 (224 if using H2-ERM in module slot for Ethernet remote I/O)		
Ethernet Remote I/O pts.	256 (using H2-ERM master in local WinPLC base and H*-EBC or T1H-EBC remote slave		
Port 0	RJ12, 6-pin modular, serial port, supports K-sequence, or any protocol from Windows CE		
Port 1	RJ45, 8-pin modular, Ethernet 10MBPS		
I/O Interface	Backplane to DL205 (Up to 9-Slot base), expandable with H2-ERM		
Power Consumption	680 mA at 5VDC		
Weight	6 oz.		
Operating Temperature	0–60°C		
Storage Temperature	-20-70°C		
Agency Listings	UL Listing		

PC control with a WinPLC

The WinPLC provides a Windows® CE operating system environment in our DL205 CPU hardware. The small size and low cost of DL205 products is desirable, but the operating systems of the D2-230, 240, 250-1 and 260 CPUs are proprietary (like most PLCs). The WinPLC provides a hybrid PC PLC solution that brings the best of the PLC and PC control worlds together. A WinPLC system is the best solution if your applications requires:

- Complex math
- Heavy serial communications (can use the H2-SERIO module)
- Advanced data manipulation
- Advanced handling of string or array data
- Up to 64 PID loops

Here's how it works

The WinPLC module is plugged into the CPU slot of the DL205 base. It uses Windows® CE, a real-time operating system combined with the advantages of open standard software such as OPC, ActiveX and other Microsoft communications tools. The WinPLC offers both deterministic control and open communications. It uses advanced software development tools for control, data management, communication and integration with business systems. The WinPLC supports the following DL205 modules only:

- All discrete and analog modules
- Temperature input modules
- H2-SERIO serial communications module
- H2-ERM module for Ethernet remote I/O (limited to one ERM and one EBC slave per system)
- H2-CTRIO Counter I/O module

DL205 specialty modules not listed above are not supported by the WinPLC.

Built-in Ethernet port

The WinPLC is programmed via a built-in 10MB Ethernet port. WinPLCs can use OPC or DDE to link to an HMI or other application using this high-speed port. Or, share tags with any controller running Entivity software for coordinated control with a PC system. The built-in Ethernet port can also be used for peer-to-peer communications between multiple WinPLCs.

Built-in serial port

A built-in RS-232C serial port lets you connect an EZTouch, EZText or other operator interfaces to the WinPLC. You can also connect to devices such as barcode readers, weight scales or serial modems to the serial port. Unlike most RLL programming, the Entivity programming method is designed for easy communication programming and string manipulation. Up to nine additional serial ports can be added to a WinPLC system by using the H2-SERIO serial communication module. See "Additional Serial Ports for the WinPLC" later in this section for more information on the H2-SERIO module.

Programming the WinPLC

To create flowcharts (projects) for the WinPLC, you'll need one of the following development packages running on a desktop PC equipped with an Ethernet card: Think & Do Live (PC-ENT-LIVE) or Entivity Studio (PC-ENT-SDD). Since each WinPLC includes its own run-time license, you can program as many WinPLCs as you need, at no additional cost. When you build (compile) your project, the PC will automatically download the flowcharts into the WinPLC. Then at runtime (or at powerup), the WinPLC will run the flowchart program.

CE-only version WinPLC

This version of the WinPLC is not preconfigured with any control software. It's for qualified OEMs or software developers who want to develop their control code in VB or C++. AUTOMATIONDIRECT does not sell this version of the WinPLC. If you are interested in the CE-only version, visit www.hosteng.com for details.

**See the Entivity PC Control software section in this desk reference for information on the PC-WPLC-START Starter Kit.*

WinPLC and Serial I/O Module Installation and Operation

Manual Number: H2-WPLC-M

WARNING

Thank you for purchasing automation equipment from ***Automationdirect.com***™. We want your new ***Direct***LOGIC™ automation equipment to operate safely. Anyone who installs or uses this equipment should read this publication (and any other relevant publications) before installing or operating the equipment.

To minimize the risk of potential safety problems, you should follow all applicable local and national codes that regulate the installation and operation of your equipment. These codes vary from area to area and usually change with time. It is your responsibility to determine which codes should be followed, and to verify that the equipment, installation, and operation is in compliance with the latest revision of these codes.

At a minimum, you should follow all applicable sections of the National Fire Code, National Electrical Code, and the codes of the National Electrical Manufacturer's Association (NEMA). There may be local regulatory or government offices that can also help determine which codes and standards are necessary for safe installation and operation.

Equipment damage or serious injury to personnel can result from the failure to follow all applicable codes and standards. We do not guarantee the products described in this publication are suitable for your particular application, nor do we assume any responsibility for your product design, installation, or operation.

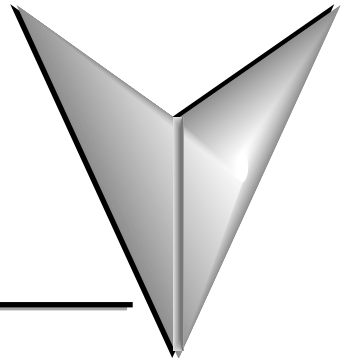
If you have any questions concerning the installation or operation of this equipment, or if you need additional information, please call us at 770-844-4200.

This publication is based on information that was available at the time it was printed. At Automationdirect.com E we constantly strive to improve our products and services, so we reserve the right to make changes to the products and/or publications at any time without notice and without any obligation. This publication may also discuss features that may not be available in certain revisions of the product.

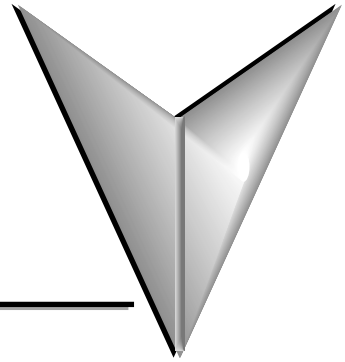
Trademarks

This publication may contain references to products produced and/or offered by other companies. The product and company names may be trademarked and are the sole property of their respective owners. Automationdirect.comE disclaims any proprietary interest in the marks and names of others.

TABLE OF CONTENTS



MANUAL REVISIONS



If you contact us in reference to this manual, remember to include this revision number.

Title: WinPLC and Serial I/O Module Installation and Operation
Manual Number: H2-WPLC-M

Issue	Date	Description of Changes
Original	***4/99***	Original Issue
2nd Edition	***12/99***	Describe T&D ESP usage Added H2-SERIO Chapter 2
3rd Edition	***3/01***	QuarkXPress conversion Added Workbench Chapter 3

GETTING STARTED

In This Chapter...

Manual Overview

Overview of this Publication

The WinPLC and Serial I/O manual describes the installation of the modules, port configuration, power budgeting, and basic operation of the WinPLC and Serial I/O modules. There is also a brief discussion of Ethernet cabling issues.

Other Reference Materials

You may find other technical publications useful for your application. For technical information related to your PC-based control software or Windows® CE, please refer to the appropriate publication for those products. For more information about the ***Direct***LOGIC™ products, you may want to read the following:

- DL205 Installation and I/O Manual

Who Should Read This Manual

You will find the WinPLC manual helpful if you have chosen to use the following:

- WinPLC running PC-based Control software
- Our DL205 I/O

You will find that a familiarity with Ethernet communications and with the setup and installation of PLCs is helpful. An understanding of electrical codes and industrial control is essential.

Technical Support

We strive to make our manuals the best in the industry. We rely on your feedback to let us know if we are reaching our goal. If you cannot find the solution to your particular application, or, if for any reason you need additional technical assistance, please call us at **770-844-4200**.

Our technical support group is glad to work with you in answering your questions. They are available weekdays from 9:00 a.m. to 6:00 p.m. Eastern Time. We also encourage you to visit our website where you can find technical and non-technical information about our products and our company. Visit us at **www.automationdirect.com**.

Special Symbols

When you see the "notepad" icon in the left-hand margin, the paragraph to its immediate right will be a **special note**.

When you see the "exclamation mark" icon in the left-hand margin, the paragraph to its immediate right will be a **warning**. This information could prevent injury, loss of property, or even death (in extreme cases).

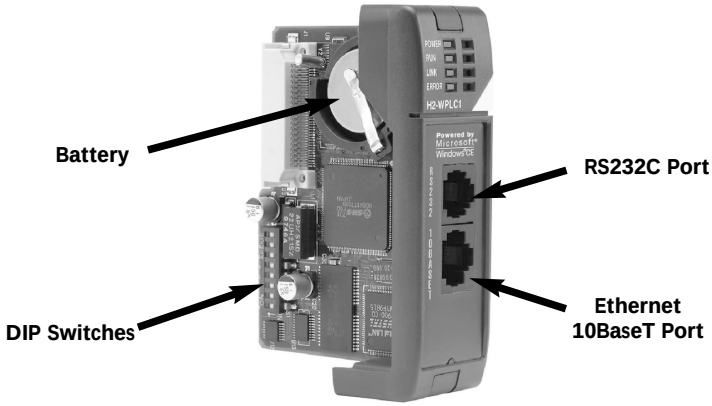
WinPLC Overview

The WinPLC (part number H2-WPLC1-xx) is an open-platform CPU running the Windows→ CE operating system. It plugs into the CPU-slot of a ***Direct***LOGIC DL205 base and "talks" across the backplane to standard digital and analog input and output modules.

The Windows→ CE operating system is a familiar favorite for embedded systems in a wide variety of applications. Using Windows→ CE in the WinPLC makes it a flexible control platform with the ability to run PC-based Control software from a number of sources, as well as Visual Basic and Visual C programs.

The operating system is resident in the module and does not require battery back-up. The user program is backed by a five-year lithium battery.

The WinPLC's operating characteristics will largely be determined by the PC-based Control software running in it. The PC-based Control software provider chooses how to use the available features in their implementation of the product.



Features Depend on Software Implementation

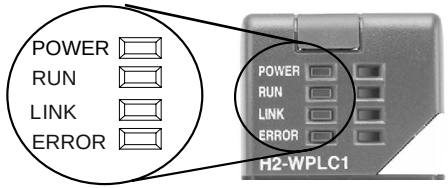
Support of the following features depends on your PC-based Control software implementation:

- the frequency of I/O updates
- the available support for RS232C serial communications
- the uses of the DIP switches
- the uses of the LEDs

If you are using the Visual Basic for CE or Visual C for CE version of the WinPLC, you will receive an SDK (software development kit), a utility called WinPLC Workbench, and a Viewer. The SDK will provide functions to access the features above, WinPLC Workbench will give you a means to set up the WinPLC, and the Viewer will make it possible to load your Visual Basic for CE or Visual C for CE programming the WinPLC.

The WinPLC LEDs

The WinPLC module has four LED indicator lights. The green POWER and RUN LEDs are individually addressable. Their exact meaning will depend on the PC-based Control software you are using. The green LINK LED has a double function. It indicates that the unit is connected successfully to an Ethernet network, and it indicates that there is activity on the network. The LINK LED will come on intermittently to indicate that it sees Ethernet traffic. The LINK LED will blink faster to indicate an increase in network activity. The red ERROR LED comes on steady to indicate that a hardware error has occurred internal to the WinPLC.



Inserting the H2-WPLCx into the DL205 Base

The H2-WPLC1 plugs into the "CPU" slot of any DL205 base.

- Locate the grooves on the inside top and bottom of the DL205 base.
- Align the module with the grooves and slide the module into the slot until the face of the module is flush with the power supply.
- Push in the retaining clips to secure the module.

Align the H2-WPLC1 with the grooves in the base and slide in.

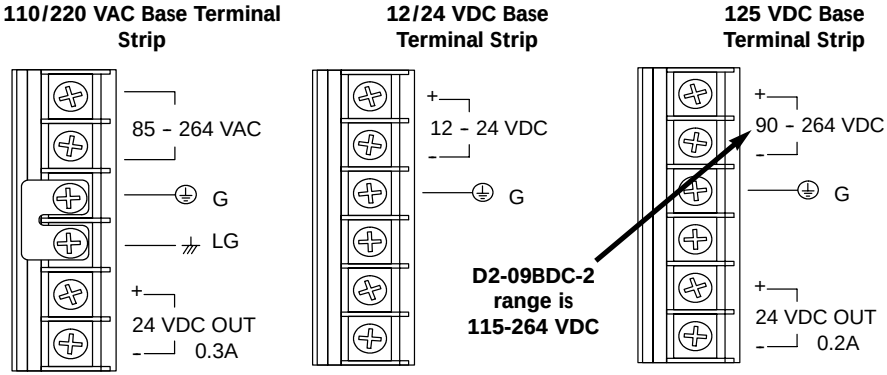
Push the retaining clips in to secure the module to the DL205 base

WARNING: To minimize the risk of electrical shock, personal injury, or equipment damage, always disconnect the system power before installing or removing any system component.

DL205 Power Wiring and Grounding

The DL205 power supply is an integral part of the base. The DL205 also has three power options: 12/24VDC, 125VDC, and 120/240VAC.

The diagram shows the terminal connections located on the power supply of the DL205 bases. The base terminals can accept up to 16 AWG. You may be able to use larger wiring depending on the type of wire used, but 16 AWG is the recommended size.



*NOTE: You can connect either a 120 VAC or 240 VAC supply to the AC terminals. Special wiring or jumpers are not required as with some of the other **Direct**LOGIC™ products.*

WARNING: Once the power wiring is connected, install the plastic protective cover. When the cover is removed there is a risk of electrical shock if you accidentally touch the wiring or wiring terminals.

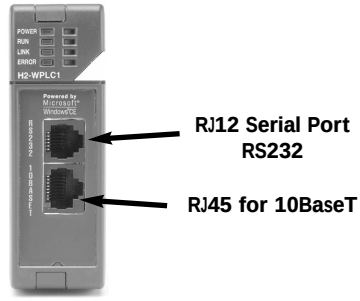
10BaseT Network Cabling

The H2–WPLC1–xx supports the Ethernet 10BaseT standard. The 10BaseT standard uses twisted pairs of copper wire conductors.

10BaseT Connections

The H2-WPLC1-xx has an eight-pin modular jack that accepts RJ45 connector plugs. UTP (Unshielded Twisted-Pair) cable is rated according to its data-carrying ability (bandwidth) and is given a "category" number. We strongly recommend using a category 5 cable for all Ethernet 10BaseT connections. For convenient and reliable networking, we recommend that you purchase commercially manufactured cables (cables with connectors already attached).

H2-WPLC-xx

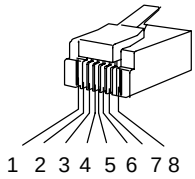


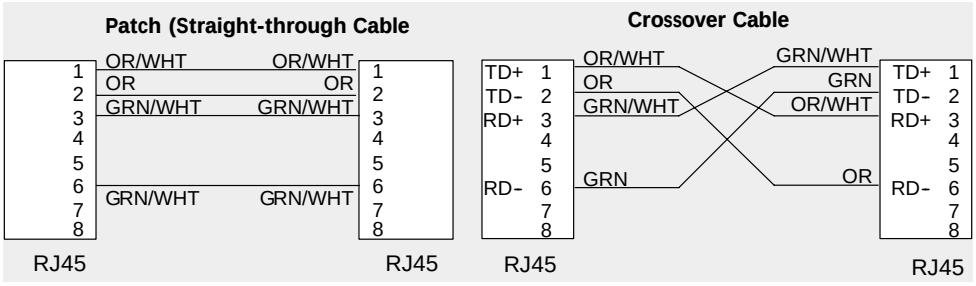
10BaseT

To connect an H2-WPLC1-xx (or PC) to a hub or repeater, use a patch cable (sometimes called a straight-through cable). The cable used to connect a PC directly to a WinPLC or to connect two hubs is called a crossover cable.

The diagram below illustrates the standard wire positions in the RJ45 connector. We recommend all WinPLC 10BaseT cables to be Category 5, UTP cable.

8-pin RJ45 Connector (8P8C)



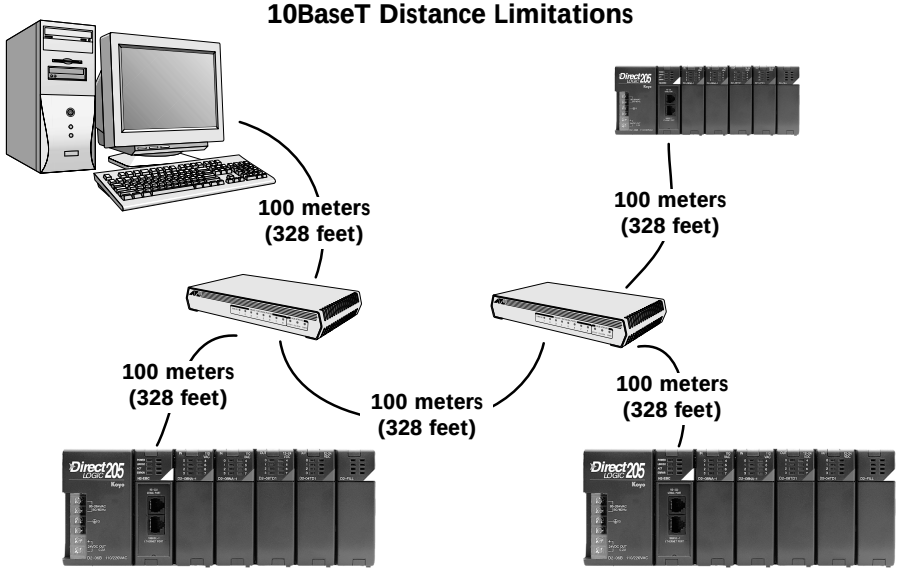


NOTE: See page 1-7 for 10BaseT distance limitations.

Maximum Ethernet Cable Length

The maximum distance per 10BaseT cable segment is 100 meters or 328 feet. Repeaters extend the distance. Each cable segment attached to a repeater can be up to 100 meters. Two repeaters connected together extend the total range to 300 meters.

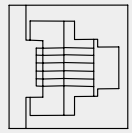
10BaseT Distance Limitations



H2-WPLC-xx Serial Port Pinouts

Pin		Pin Assignments for: H2-WPLC-xx serial port
1	0V	Power (–) Connection (GND)
2	5V	Power (+) Connection
3	RXD	Receive Data (RS232C)
4	TXD	Transmit Data (RS232C)
5	RTS	Request to Send
6	0V	Signal Ground (GND)

RJ12 (6P6C) Female Modular Connector



Power Budget for the DL205 with H2-WPLC1-xx

Managing your Power Resource

When determining which I/O modules you will be using in the DL205 WinPLC system, it is important to remember that there is a limited amount of power available from the power supply. We have provided a table showing the power available from the various DL205 base power supplies and a table showing the maximum power consumed by the WinPLC and each of the I/O modules supported by the WinPLC. Following these two tables is an example of a completed power budgeting worksheet and then a blank worksheet you can use for your own calculations.

If the I/O modules you choose exceed the maximum power available from the smaller DL205 base power supplies, you will need to use a D2-09B9-slot base. This base supplies more power than the other bases, as you can see in the table below.

WARNING: It is extremely important to calculate the power budget. If you exceed the power budget, the system may operate in an unpredictable manner which may result in a risk of personal injury or equipment damage.

WinPLC Power Specifications

The following table shows the amount of electrical current available at the two voltages supplied from the DL205 base. Use these values when calculating the power budget for your system.

The Auxiliary 24V power source mentioned in the table is available at the base terminal strip. You can connect to external devices or DL205 I/O modules that require 24VDC, but be sure not to exceed the maximum current supplied.

Available Electrical Current		
Bases	***5V Current Supplied***	***Auxiliary 24VDC Current Supplied***
D2-03B	1550 mA	200 mA
D2-04B	1550 mA	200 mA
D2-06B	1550 mA	200 mA
D2-09B	2600 mA	300 mA
D2-03BDC-1	1550 mA	None
D2-04BDC-1	1550 mA	None
D2-06BDC-1	1550 mA	None
D2-09BDC-1	2600 mA	None
D2-03BDC-2	1550 mA	200 mA
D2-04BDC-2	1550 mA	200 mA
D2-06BDC-2	1550 mA	200 mA
D2-09BDC-2	2600 mA	300 mA

DL205 Module Power Requirements

The chart below shows the maximum amount of electrical current required to power each of the DL205 WinPLC or I/O modules. Use these values when calculating the power budget for your system. The Power Budget for the DL205 with H2--WPLC1--xx

Device	5V Current Req. (mA)	24V Aux. Current Req. (mA)
CPUs		
H2-WPLC-xx	680	0
H2-EBC	530	0
H2-EBC-F	670	0
DC Input Modules		
D2-08ND3	50	0
D2-16ND3-2	100	0
D2-32ND3	25	0
AC Input Modules		
D2-08NA-1	50	0
D2-08NA-2	100	0
D2-16NA	100	0
Input Simulator Module		
F2-08SIM	50	0
DC Output Modules		
D2-04TD1	60	20
D2-08TD1	100	0
D2-16TD1-2	200	80
D2-16TD2-2	200	0
D2-32TD1	350	0
AC Output Modules		
D2-08TA	250	0
D2-12TA	350	0

Device	5V Current Req. (mA)	24VDC Aux. Current Req. (mA)
Relay Output Modules		
D2-04TRS	250	0
D2-08TR	250	0
F2-08TR F2-08RRS	670	0
D2-12TR	450	0
Combination In/Out Module		
D2-08CDR	200	80
Analog Modules		
F2-04AD-1	50	80
F2-04AD-1L	50	90mA @12V
F2-04AD-2	60	80
F2-04AD-2L	60	90mA @12V
F2-08AD-1	50	80
F2-08AD-2	50	80
F2-02DA-1	40	60
F2-02DA-1L	40	70mA @12V
F2-02DA-2	40	60
F2-02DA-2L	40	70mA @12V
F2-02DAS-1	100	50
F2-02DAS-2	100	60
F2-08DA-2	60	90
F2-4AD2DA	60	80
F2-04RTD	90	0
F2-04THM	110	60

Power Budget Calculation Example

The following example shows how to calculate the power budget for the DL205 system.

Base # 1	***Device Type***	***5 VDC (mA)***	***External Power 24 VDC (mA)***
Available Base Power			
Base	**D2-09B**	**2,600**	**300**
Power Required			
CPU SLOT	H2-WPLC-xx	480	0
SLOT 0	D2-16ND3-2	100	0
SLOT 1	D2-16ND3-2	100	0
SLOT 2	D2-16NA	100	0
SLOT 3	F2-04AD-1	50	100
SLOT 4	F2-02DA-1	40	80
SLOT 5	D2-08TA	250	0
SLOT 6	D2-08TD1	100	0
SLOT 7	D2-08TR	250	
Other			
Operator interface	DV-1000	150	
Handheld programmer	D2-HPP	200	
Maximum Power Required		**1820**	**180**
Remaining Power Available		2600-1820=**780**	300-180=**120**

1. Using the table on the previous page, fill in the information for the base power supply, the WinPLC1-xx, I/O modules, and any other devices that will use system power including devices that use the 24 VDC output. Pay special attention to the current supplied by the base power supply. The 9-slot base has a larger current capacity than the smaller bases.
2. Add the current columns starting with the row for the **CPU slot** and work your way down to the **"Other"** category. Put the total in the row labeled **"Maximum power required"**.
3. Subtract the row labeled **"Maximum power required"** from the row labeled **"Available Base Power"**. Place the difference in the row labeled **"Remaining Power Available"**.
4. If **"Maximum Power Required"** is greater than **"Available Base Power"** in either of the two columns, the power budget will be exceeded. It will be unsafe to use this configuration, and you will need to restructure your I/O.

Power Budget Calculation Worksheet

This blank chart is provided for you to copy and use in your power budget calculations.

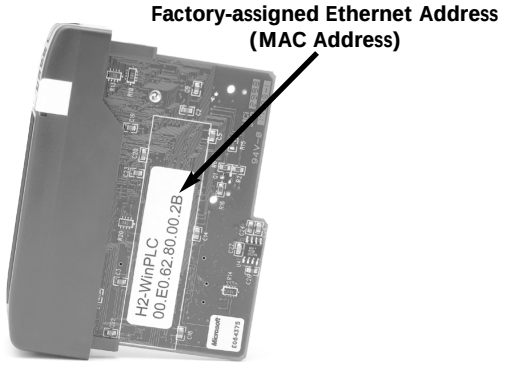
Base # ____	***Device Type***	***5 VDC (mA)***	***External Power 24 VDC (mA)***
Power Supplied			
Base			
Power Required			
CPU SLOT			
SLOT 0			
SLOT 1			
SLOT 2			
SLOT 3			
SLOT 4			
SLOT 5			
SLOT 6			
SLOT 7			
Other			
Maximum Power Required			
Remaining Power Available			

1. Using the table on the previous page, fill in the information for the base power supply, the WinPLC-xx, I/O modules, and any other devices that will use system power including devices that use the 24 VDC output. Pay special attention to the current supplied by the base power supply. The 9-slot base has a larger current capacity than the smaller bases.
2. Add the current columns starting with the row for the **CPU slot** and work your way down to the **"Other"** category. Put the total in the row labeled **"Maximum power required"**.
3. Subtract the row labeled **"Maximum power required"** from the row labeled **"Available Base Power"**. Place the difference in the row labeled **"Remaining Power Available"**.
4. If **"Maximum Power Required"** is greater than **"Available Base Power"** in either of the two columns, the power budget will be exceeded. It will be unsafe to use this configuration, and you will need to restructure your I/O.

Locating the Ethernet Address Label

Ethernet Address

A unique Ethernet Address is assigned to each module at the factory and cannot be changed. It is a twelve digit number (six pairs of hexadecimal numbers) and is printed on a label permanently attached to the WinPLC module.



Setting Up the WinPLC

If you are using Think & Do Studio, Version 6.1 or later, use Chapter 2, Workbench Utility Operation, to help you set up the WinPLC.

If you are using Think & Do, Version 5.2 or 5.3, use Appendix A, Using The ESP Utility To Set Up The WinPLC When Using Think & Do vers. 5.2 or 5.3, to help you set up the WinPLC.

Diagnosing Network Cable Problems

If you are experiencing communication problems, swapping cables is one of the simplest diagnostic procedures you can perform. If the network operates correctly with a different cable, you have isolated and cured the problem. If possible, use a short run of cable to test the network because problems with longer cable runs can be more difficult to diagnose and are more often intermittent.

If you are unable to swap cables, verify the proper operation of all other network components. You probably have a cable problem if you have verified that your:

- WinPLC module is working correctly.
- WinPLC module configuration is correct.
- PC-based Control program is correct.
- any hubs are working correctly.
- Windows configuration is correct.
- network adapter card is the correct type, and it is working correctly.

It is a good maintenance practice to test network cables periodically and maintain a permanent record of cable characteristics. A number of cable test instruments are available to test 10BaseT networks. These instruments will check the electrical characteristics of your cabling, including:

- Continuity — This is a check to make sure the communication pairs are wired correctly, and that the wires are continuous from end to end.
- Attenuation — This refers to the amount of signal loss over the cable segment at the signal frequency of interest. The 10BaseT specification allows for a maximum signal loss of 11.5 decibels (dB) for the entire link at the signal frequency used by 10Mbps Ethernet.
- Crosstalk — Crosstalk occurs when a signal in one pair of wires is electromagnetically coupled to an adjacent pair.

NOTE: Any significant difference between the cable characteristics of the transmitter and receiver can cause communication errors.

Ethernet devices continually monitor the “receive data” path for activity as a means of verifying their link is working correctly. When the network is idle, each network device (including the WinPLC module) sends a periodic link test signal to verify that the network is working. If the link test signal or other network activity is not received periodically, the LINK LED on the WinPLC module is turned off.

WORKBENCH UTILITY OPERATION

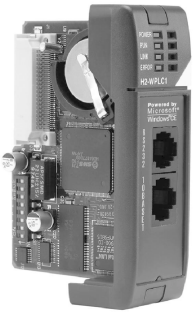
In This Chapter...

Note: This Chapter only applies if you are using the WinPLC with Think & Do Studio version 6.0 or later. Use Appendix A if using the WinPLC with Think & Do versions 5.2 or 5.3.

WinPLC Workbench Overview

WinPLC Workbench is a utility to configure and check out a WinPLC I/O system. It is also used to load new ROM images on the WinPLC. Use Workbench with a new WinPLC to set its IP address, thereby allowing other devices or software products to connect with the WinPLC.

Since the WinPLC may be used with various software packages and user developed applications, Workbench can be helpful in troubleshooting to verify that the WinPLC and its I/O are functioning properly.



Workbench is intended for use with the the following WinPLC products.

H2-WPLC1

H2-WPLC2

H2-WPLC1-KW

H2-WPLC2-KW

It is **not** recommended for use with these WinPLC products:

H2-WPLC1-TD

H2-WPLC2-TD

Configuring Your WinPLC

PC Setup

Copy Workbench files to a directory on your PC.

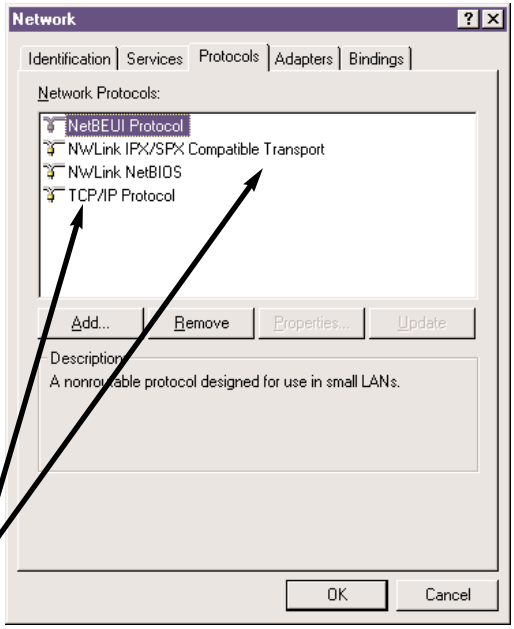
We recommend that you set up a Desktop or Start Button program menu Shortcut.

Make sure your PC has TCP/IP and IPX/SPC Protocols installed. To check, click on your computer's Start button, select Settings and then select Control Panel. Select Network and click on the Protocols tab (see screen).

Direct-connect WinPLC using a cross over cable (see Section 1).

You can connect to the WinPLC across a network; however, setting up the proper IP Address, SubNet Mask, and Gateway are beyond the scope of this manual. See your LAN Administrator for assistance with these settings.

Also, set up the WinPLC module so that it's easy to cycle the power.



IPX/SPX and TCP/IP protocols must be installed on your PC

Catching the WinPLC: Using Workbench To Find Your WinPLC Using its Ethernet (MAC) Address

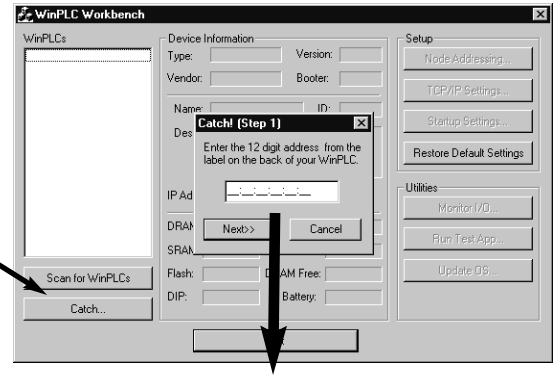
The "Catch" feature can find a WinPLC by its Ethernet Address (MAC Address). This address is found on the WinPLC label and is set at the factory and cannot be changed. Catch is a robust way to locate the WinPLC in order to setup the TCP/IP communications. Most Workbench features are not enabled until TCP/IP communication has been established.

Factory-assigned Ethernet address (MAC address)

Note: *Since the Catch feature uses the IPX/SPX protocol, the PC and WinPLC must be on the same LAN to work properly. Workbench must see the WinPLC while the WinPLC is in its boot-up state, which is indicated by the flashing green RUN LED.*

Follow these steps.

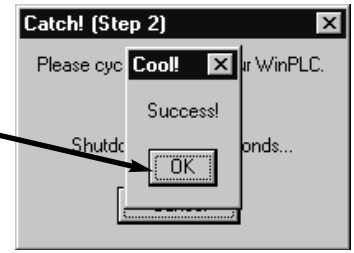
Click on "Catch"



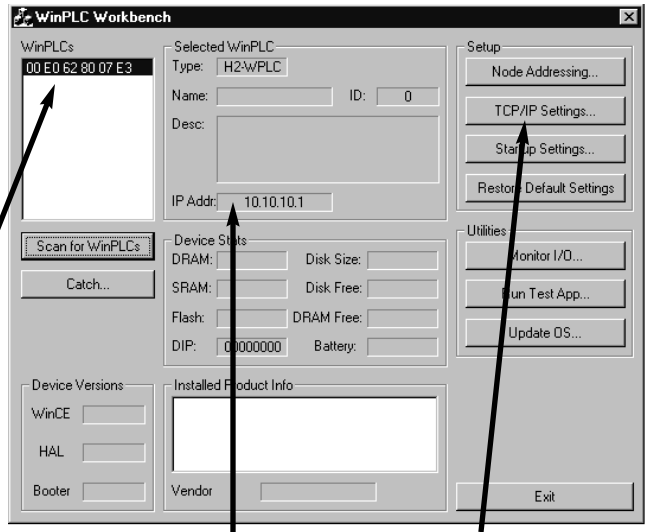
Enter 12-character Ethernet address here, click on "Next"

Cycle power

Click on "OK"



Workbench has found your WinPLC and it's Ethernet address appears in the "WinPLC" window. The green RUN LED should also be flashing. If you have problems, check to make sure you have the correct Ethernet address entered and that the IPX/SPX protocol is loaded on your PC.



Ethernet Address

TCP/IP Settings

NOTE: If the WinPLC has no IP address, Workbench displays the PC's IP address in this field.

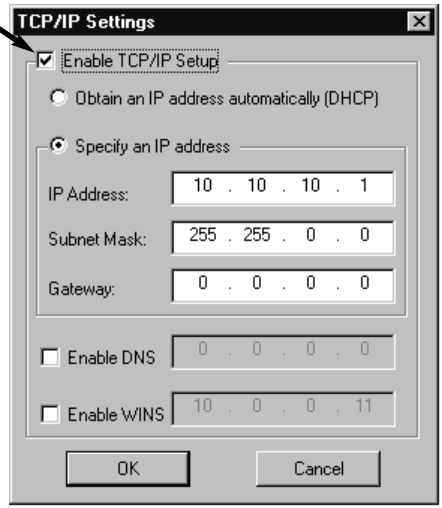
Setting TCP/IP Communications

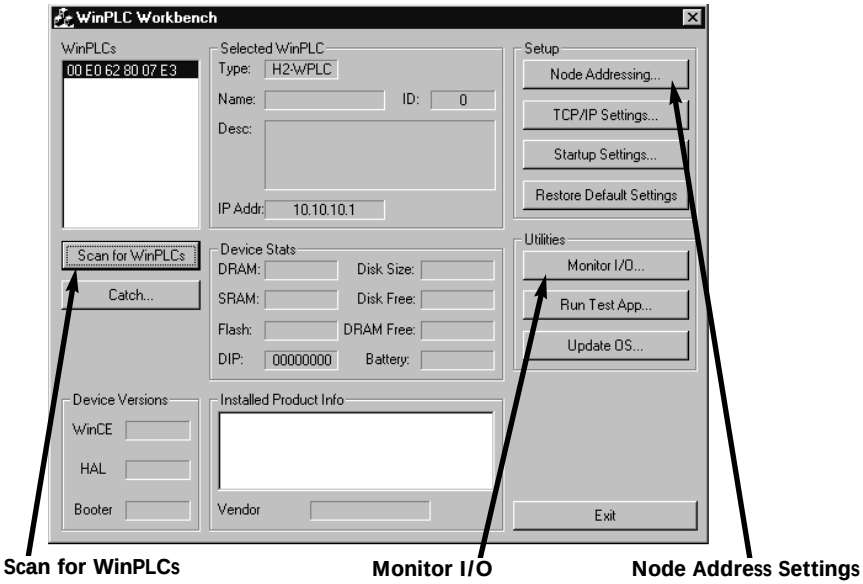
Next, click on "TCP/IP Settings" to bring up this screen. Make sure "Enable TCP/IP Setup" is selected.

Enable TCP/IP Setup

With "Specify an IP address" selected, you can manually enter the IP address. Enter the IP address, click "OK" and cycle power to the WinPLC to activate the address.

Note: *Unless you have detailed knowledge of IP protocol, we recommend that your PC and WinPLC have the same subnet mask. DNS and WINS settings are optional (see your LAN administrator).*





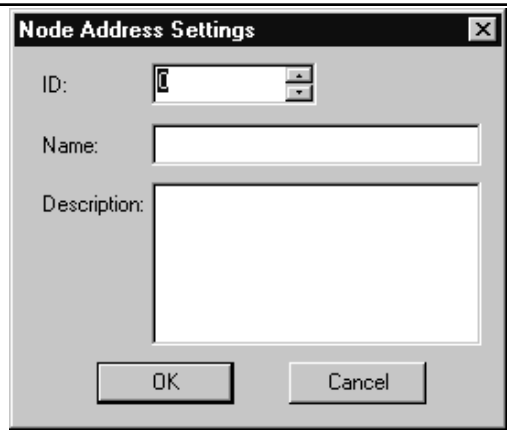
Now that that the IP address is set, Workbench should be able find the WinPLC automatically if it is run after the WinPLC is powered up and connected. If the WinPLC is connected after Workbench is running, just click on “Scan for WinPLCs”.

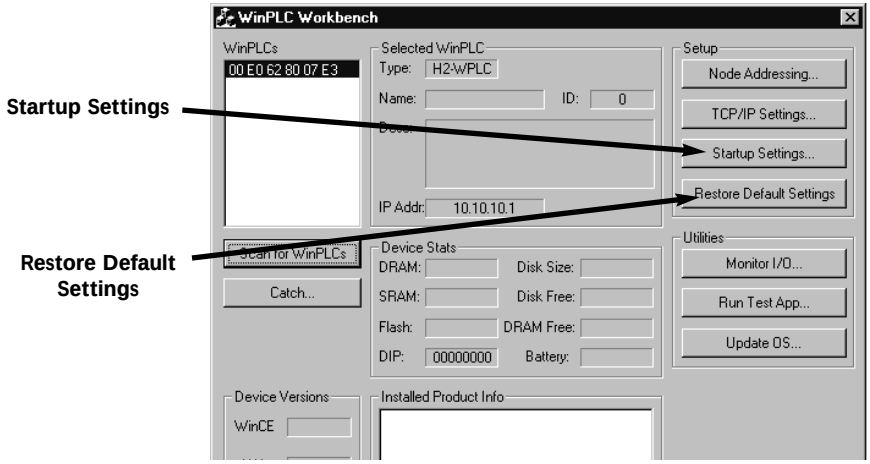
The Workbench window now appears like above, showing information about the WinPLC module. In addition, the “Monitor I/O” utility is now activated.

From this point Workbench is using TCP/IP protocol, and with the proper IP address setting you can remotely attach to the WinPLC.

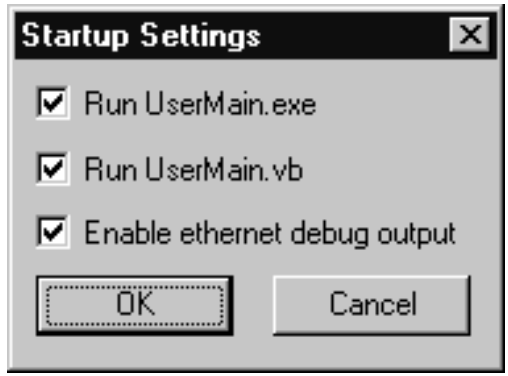
NOTE: While the Catch feature does allow you to capture the WinPLC, even without an IP address, the feature cannot be routed between LANS.

The “Node Address Setttings” selection allows you to enter descriptive information for each Win PLC module. For example, you can assign the WinPLC a module ID, name or description.

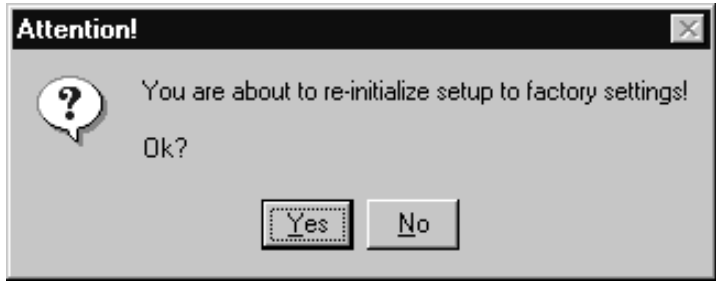




"Startup Settings" are only used by the H2-WPLC1 and H2-WPLC2 models for C and VB programming. See H24-SDK-M for more information on these settings.



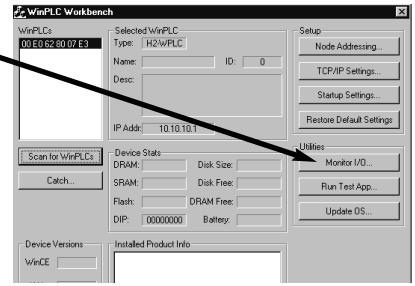
Selecting "Restore Default Settings" returns the WinPLC to its factory default settings. You must cycle power before this occurs, so if you accidentally select "Yes", you can recover by resetting the setup parameters before cycling power.



Following is a description of the utilities provided by Workbench. These programs allow you to monitor and test your I/O modules and your programming connection to the WinPLC without having to write a specific program.

Monitoring the I/O

Monitor I/O



"Monitor I/O" gives you a way to read from and write to the I/O modules in the base using your WinPLC. It allows you to see the current state of the discrete and analog inputs, toggle your discrete outputs and write values to your analog outputs.

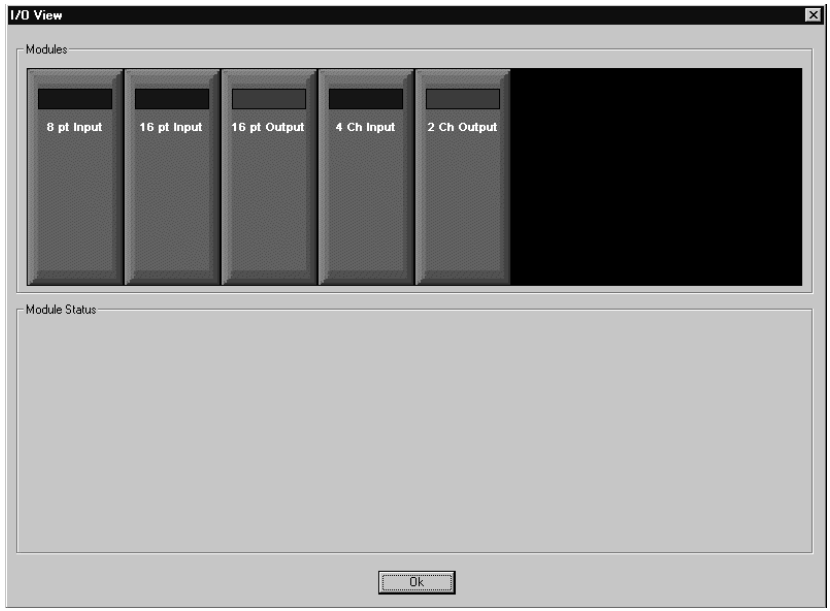
NOTE: *The "Monitor I/O" utility uses TCP/IP protocol, so it will not be active until the WinPLC is assigned a valid TCP/IP address.*

When you click here, Workbench scans the backplane and displays a graphical representation of the modules it finds.

Slots containing specialty I/O modules which are not currently supported by Workbench will be displayed but will be nonfunctional.

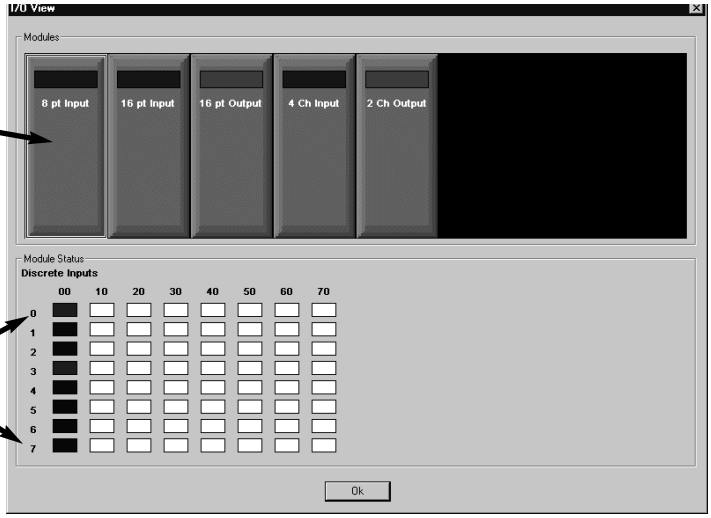
Click on a module to open a window with details about that module.

WARNING: Because this utility allows you to manipulate the actual I/O, be very careful not to cause personal injury or equipment damage.



Discrete Input Modules

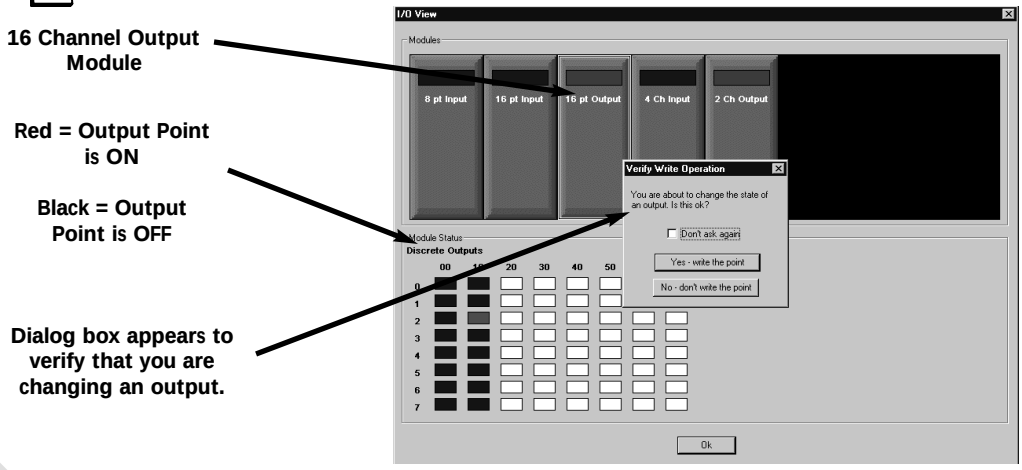
Below is an example where an 8 channel discrete input module has been selected. Notice that 8 blocks are displayed, each representing one channel. For discrete input modules, points that are ON will be blue, while points that are OFF will be black.



Discrete Output Modules

Below is an example where a 16 channel discrete output module has been selected. Notice that 16 blocks are displayed, each representing one channel. For discrete output modules, points that are ON will be red, while points that are OFF will be black. To turn ON an ouput, double-click on the black box, which brings up a window asking you to verify that you want to change the output. Make sure it is safe for you to turn the output on or off.

NOTE: Some older WinPLCs will not allow the state of ouput points to change.

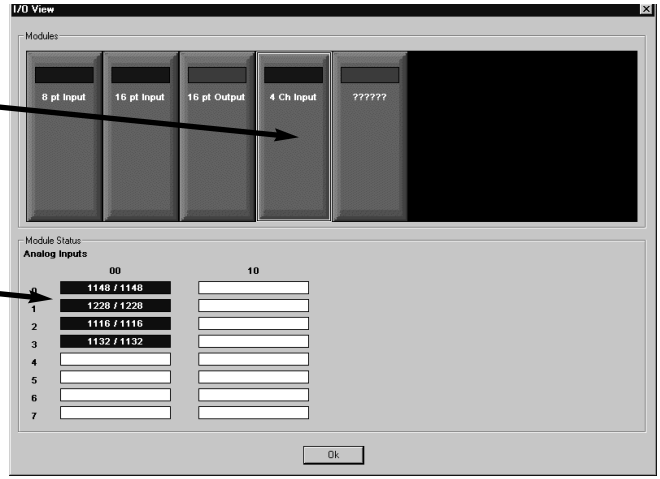


Analog Input Modules

Below is an example where a 4 channel analog input module has been selected. Notice that 4 blocks are illuminated, each representing one channel and displaying some non-zero digital value representing the sensed value. The exact digital value depends on the module resolution and range. For example, a 12 bit input module displays 4095 for a full-scale input.

4 Channel Analog Input Module

Each block contains the digital value for that channel



Analog Output Modules

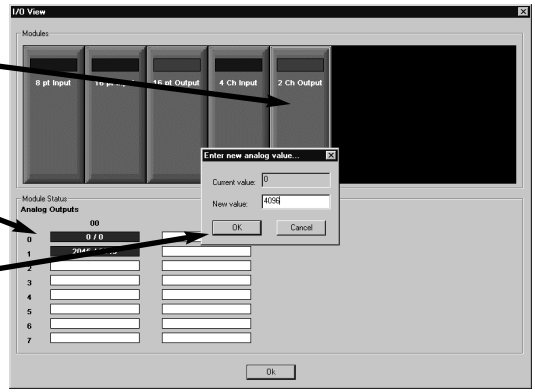
Below is an example where a 2 channel analog output module has been selected. Notice that 2 blocks are illuminated red, each representing one channel and displaying a zero when the output is OFF. To turn ON an output, double-click its block, which brings up a screen allowing you to enter a digital value representing the portion of the full-scale output you desire. The full-scale digital value depends on the bit resolution of the module. For example, set a 10V, 12 bit voltage module to 4095 for a 10V output signal. Enter a value and click OK. A window pops up asking you to verify that you want to turn on/off an output. Make sure it is safe to do so.

NOTE: *Some older WinPLCs will not allow the state of ouput points to change.*

4 Channel Analog Output Module

Each red block has 0 (zero) displayed if the output point is OFF, or a digital value if it is on.

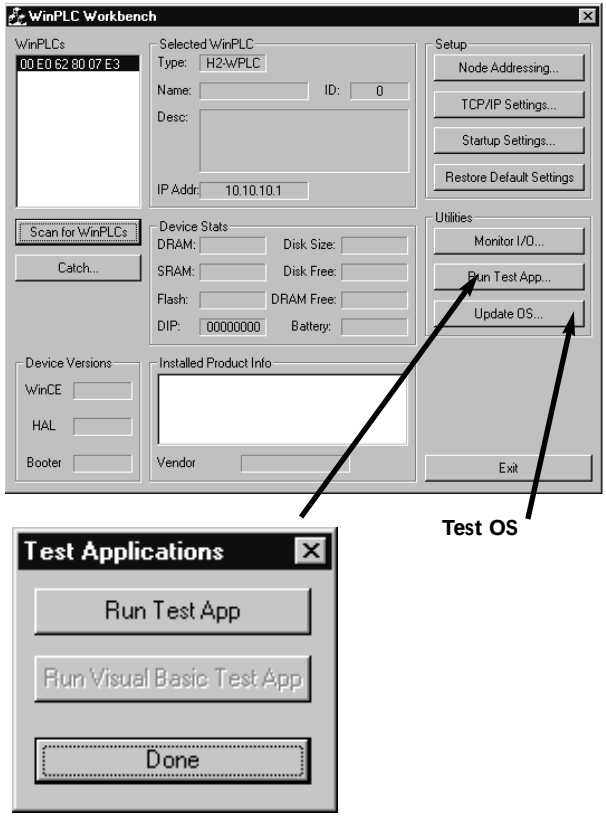
Dialog box appears to verify that you are changing an output.



Test Applications Utility

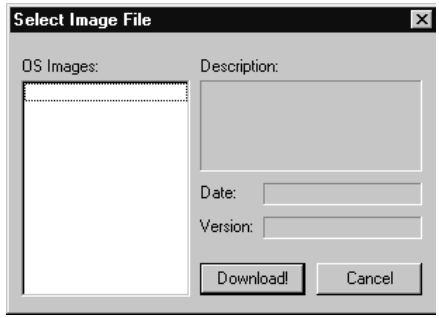
The "Run Test App" feature is only used by the H2-WPLC1 and H2-WPLC2 models for C and VB programming. See H24-SDK-M for more information.

This utility lets you test your development PC's ability to download a program to the WinLPLC and have the WinPLC run that program. The utility decides which test application to run by looking at the operating system image in the WinPLC.



Update OS Utility

This utility should only be used if directed to by a Technical Support person to update the EEPROM image stored on your WinPLC. If the manufacturer issues new operating system images for your WinPLC, this utility can be used to do the update. Clicking on any entry in the OS Images displays the image's description, date of release and other version information. The size of the FLASH drive is determined by the amount of ROM left over after the operating system is loaded, so updating the operating system image deletes the entire FLASH drive and rebuilds it to accomodate the new operating system image. If there is anything in the FLASH drive you want to save, do so before updating the operating system.



Serial I/O Module Installation & Operation (Using T & D Studio ver. 6.0 or later)

In This Chapter...

Note: This Chapter only applies if you are using the WinPLC with Think & Do Studio version 6.0 or later. Use Appendix B if using the WinPLC with Think & Do versions 5.2 or 5.3.
Only Think & Do WinPLCs (H2-WPLC1-TD and H2-WPLC2-TD) support the H2-SERIO module.

H2–SERIO Overview

The Scope of This Manual

This chapter introduces the use of the H2-SERIO module using the WinPLC with Think & Do Studio, version 6.0 or later). See Appendix B if you are using Think & Do versions 5.2 or 5.3.

This chapter will not describe in detail how to build a project or connect to a WinPLC. Depending on which version of Think & Do you are using, further information can be found in:

Chapter 2 of this manual, Workbench Utility Operation

Appendix A of this manual, Using The ESP Utility To Set Up The WinPLC

The Think & Do Studio Learning Guide, Chapter 2.

The basic steps in using this module are:

1. Install the Serial I/O module in the base.
2. Connect power to the base.
3. Bring up Think & Do Studio.
4. Select the WinPLC as the target.
5. Connect to the WinPLC.

Add Serial Ports to Your WinPLC

The Serial I/O module plugs into the DL205 I/O base and is used exclusively with the WinPLC to provide additional RS232 serial ports. The WinPLC communicates with the H2–SERIO module across the DL205 backplane.

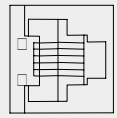
As Many as Ten Serial Ports

The WinPLC has one built-in serial port. Now, you can add as many as nine additional serial ports for Think & Do applications requiring multiple serial devices, such as barcode scanners.

Setting Communication Parameters Using Think & Do

Use I/O View to set baud rate, parity, data bits, and stop bits for each port. Choose from 300 to 57,600 baud communication speeds. Think & Do Studio allows each port to be designated as a MODBUS slave or a generic serial device. Each port on the H2–SERIO module is capable of hardware handshaking.

RS-232 Wiring

Pin Assignments for: H2-SERIO ports			RJ12 (6P6C) Female Modular Connector
1	0V	Power (–) connection (GND)	
2	CTS	Clear to Send	
3	RXD	Receive Data (RS232C)	
4	TXD	Transmit Data (RS232C)	
5	RTS	Request to Send	
6	0V	Signal Ground (GND)	

NOTE: The serial port on-board the WinPLC has a different pinout from the H2–SERIO module. Refer to page 1–7 for the WinPLC serial port pin assignments.

H2-SERIO Specifications	
Module type	Intelligent module for use with H2–WPLC1–TD
Maximum number of modules supported by one WinPLC	3
Recommended cable	Belden 9729 or equivalent
Connector	RJ12 jack
Power consumption	230mA @ 5VDC
Operating environment	0° to 60°C (32°F to 140°F), 5% to 95% RH (non-condensing)
Manufacturer	Host Engineering

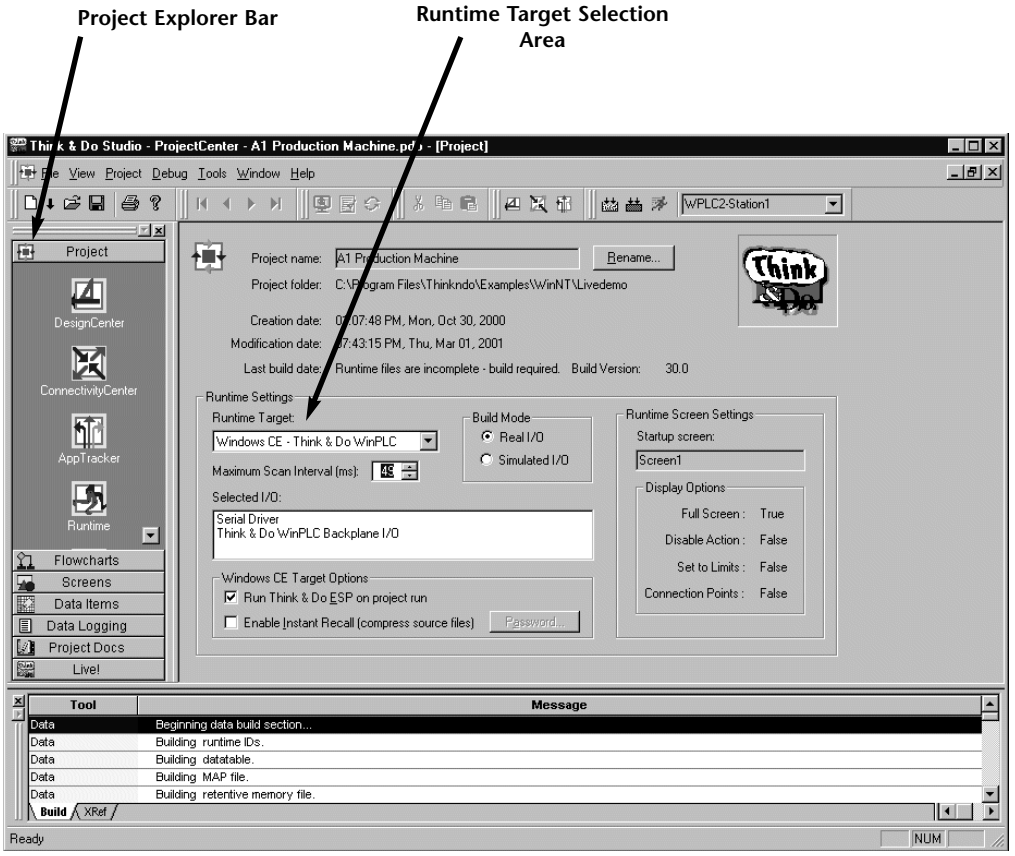
Using Think & Do to Set Serial Port Parameters

Installing the H2-SERIO

Install the H2-WPLC1-TD or H2-WPLC2-TD, and the H2–SERIO module in your DL205 base. Please refer to the guidelines elsewhere in this publication for information about installation, power wiring, and Ethernet connections. The WinPLC must be recognized on the network to proceed, so use Think & Do to establish your link to the WinPLC.

Setting the WinPLC as the Runtime Target

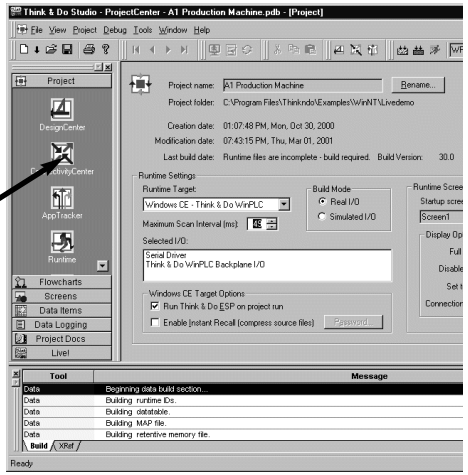
With Think & Do Studio ProjectCenter open, click the "Project Explorer Bar", and project information will display in the main ProjectCenter window. In the "Runtime Target" area, select "Windows CE - Think & Do WinPLC" from the drop-down list.



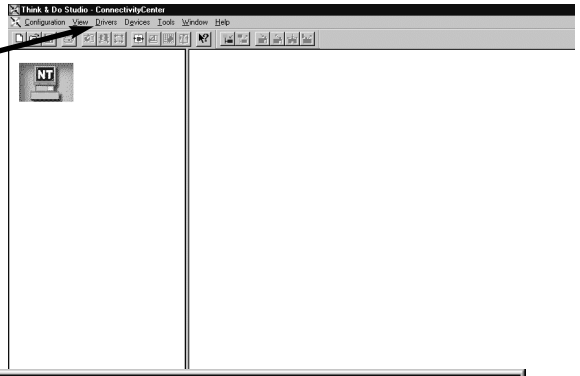
Using Think & Do ConnectivityCenter to Set Up The Serial I/O Module

ConnectivityCenter is the Think & Do Studio tool for configuring I/O devices. See the Think & Do Studio Learning Guide (Chapter 2) for more detailed information on using ConnectivityCenter.

Open up the ConnectivityCenter.



Click here to open ConnectivityCenter From ProjectCenter



Drivers menu

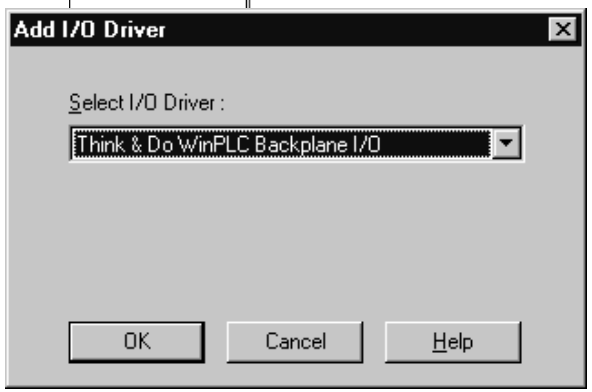
This frame shows an initial ConnectivityCenter screen with no WinPLC connected.

Adding the Serial I/O Module Driver

Again, see the Think & Do Studio Learning Guide for more information on adding I/O drivers.

Either click on the "Drivers" menu and select "Add", or click on the Add Driver toolbar button.

Select "Think & Do WinPLC Backplane I/O" as the target.

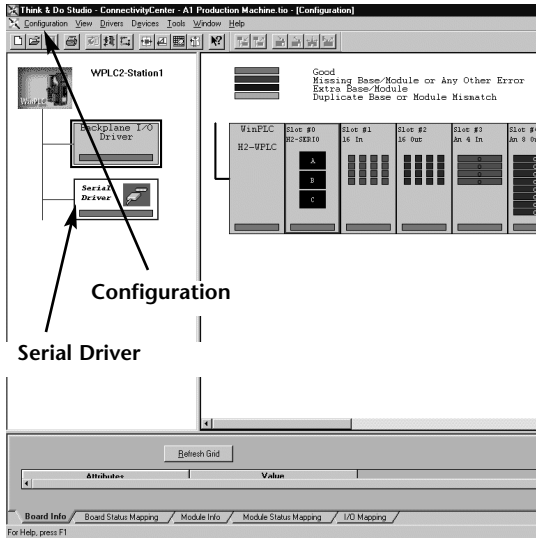


Connecting To The WinPLC

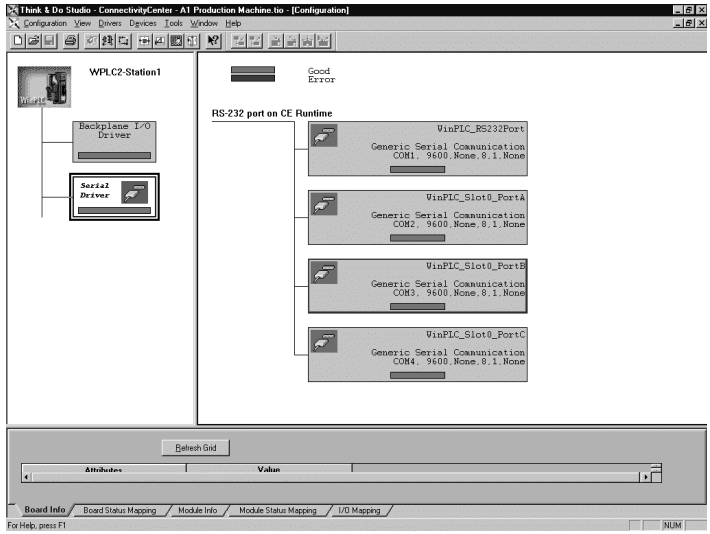
To connect to the WinPLC, click "Configuration", and select "Connect".

Think & Do recognizes the DL205 base as you have configured it. The WinPLC is displayed in the CPU slot, and the Serial I/O module is displayed where you have installed it.

Click on "Serial Driver".



You will see a port configuration box for each serial port Think & Do Studio recognizes. In our example to the right, Think & Do sees four serial ports. One is on the WinPLC and the other three are on the Serial I/O module.

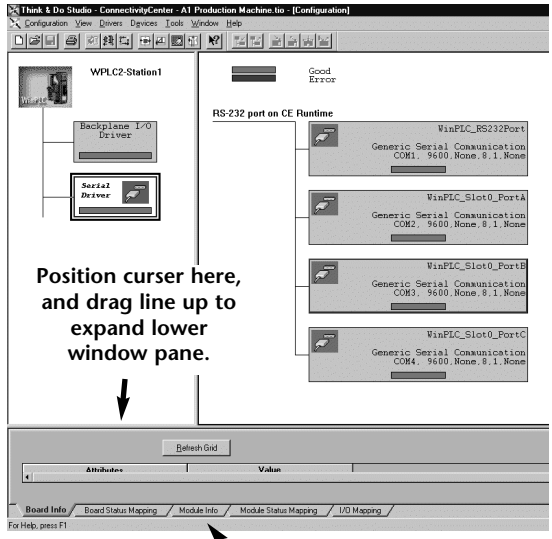


Notice that the ports are numbered COM 1 through COM 4 in Think & Do. COM 1 is on the WinPLC. COM 2 through COM 4 are on the first Serial I/O module in the base. Think & Do counts the serial ports from top to bottom (on the Serial I/O module) and from left to right in terms of slot position.

If you install additional Serial I/O modules at a later time, be aware that the order of the modules in the base determines their COM numbers. If you install a Serial I/O module between an existing Serial I/O module and the CPU, your port settings will remain the same, but the COM number will change.

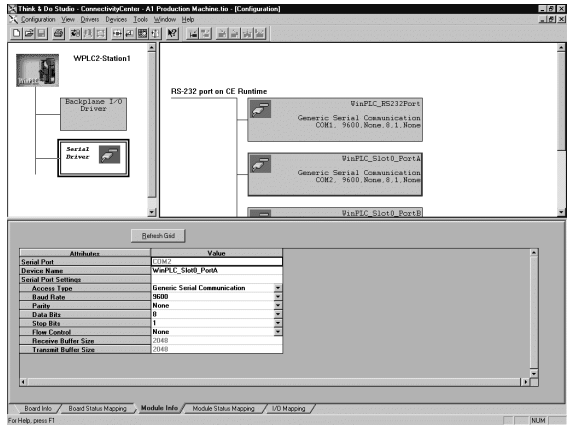
Setting Serial Port Parameters

To set the serial port parameters, click on the "Serial Driver" in the left pane of the ConnectivityCenter window. You will see a port configuration box for each serial port Think & Do recognizes. In our example to the right, Think & Do sees four serial ports. One is on the WinPLC and the other three are on the Serial I/O module.



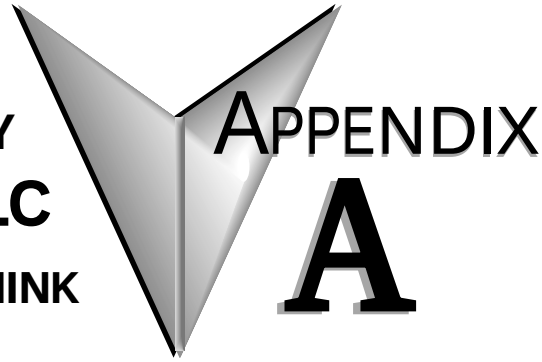
Expand the Window Pane

Position your cursor on the line that separates the upper window panes from the lower window pane. Move this line up by dragging your mouse. Click on the tab at the bottom of the lower window pane marked "Module Info." You will see a screen that looks similar to the one shown here. Pull-down menus allow you to change the serial port parameters.



Select the port whose parameters you want to change by clicking on that port in the upper right pane. Make the changes in the lower pane, and save the changes using the Ctrl + S keys.

USING THE ESP UTILITY TO SET UP THE WINPLC WHEN WHEN USING THINK & DO VER. 5.2 OR 5.3



In This Appendix...

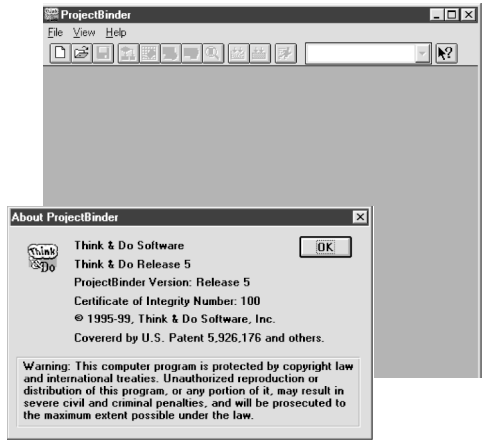
Note: This Appendix only applies if you are using the WinPLC with Think & Do versions 5.2 or 5.3. Use Chapter 2 if using the WinPLC with Think & Do Studio version 6.0 or later.

Using the Think & Do ESP Utility to Set Up the WinPLC

Check Think & Do Version First

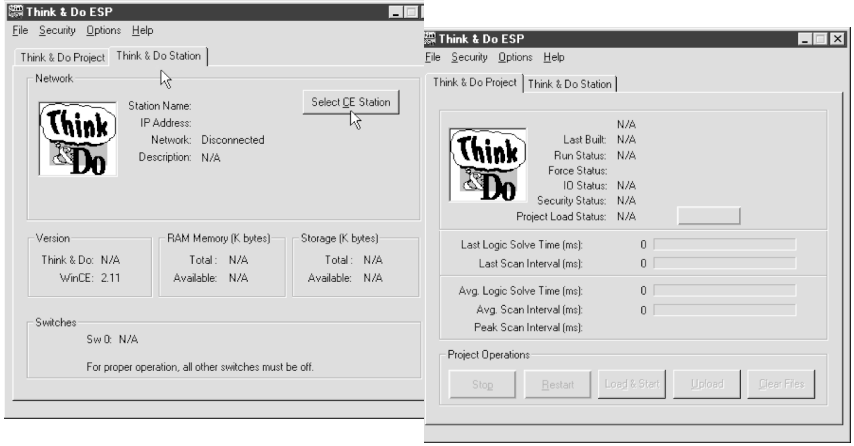
You will need Version 5.2 (or later) of Think & Do, to recognize the H2–SERIO module. To determine whether you have the right version, open the Project Binder. As the Project Binder opens, you may notice a screen that tells you which Version of Think & Do you are opening. That screen disappears as the Project Binder opens.

After the Project Binder is open, you can click on "Help," and the bottom menu option, "About Project Binder," will tell you which Version you are using.



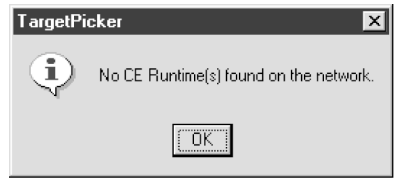
After you load Think &Do, Version 5.2 or 5.3, you will notice a Think & Do ESP icon on your desktop. If you double click this icon, you will start a utility that helps you establish network parameters for the WinPLC.

Click on the "Think & Do Station" tab, then click on the "Select CE Station" button to open the "TargetPicker".



TargetPicker

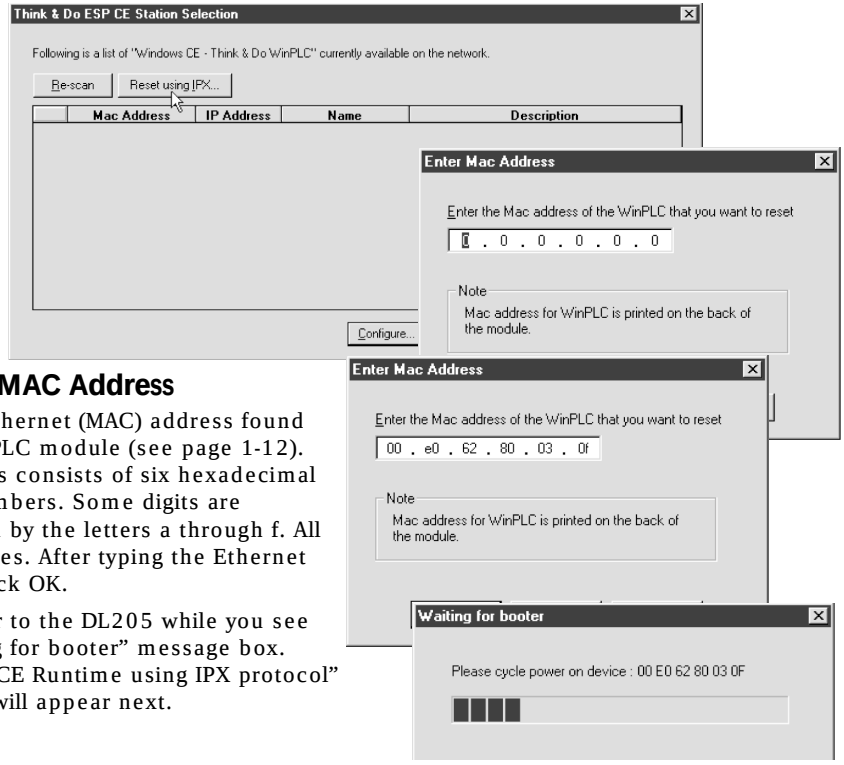
A "TargetPicker" message box opens to notify you that no CE targets are currently visible on the network.



Acknowledge this message by clicking OK, and the "Think &Do ESP CE Station Selection" window will open. If you have not already done so, install the WinPLC and connect power to the DL205 base. See pages 1-4 through- 1-7 for important wiring and installation information.

NOTE: The following link procedure assumes that you are directly connected from your Think & Do Development System computer to your WinPLC. For more information about making this connection, consult the Think &Do Software Learning Guide. If your WinPLC is connected via your office or plant network, please consult your Network Administrator for appropriate network settings.

Click the "Reset using IPX" button on the Station Selection window. This will allow Think & Do to link to the WinPLC target using its Ethernet address (MAC address). The IPX protocol must be loaded on your Think & Do Development System Computer. For more information, consult the Think & Do Software Learning Guide.



Ethernet or MAC Address

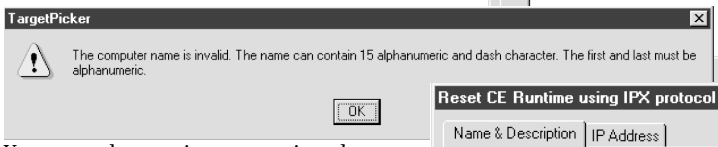
Enter the Ethernet (MAC) address found on the WinPLC module (see page 1-12). This address consists of six hexadecimal pairs of numbers. Some digits are represented by the letters a through f. All 0's are zeroes. After typing the Ethernet address, click OK.

Cycle power to the DL205 while you see the "Waiting for booter" message box. The "Reset CE Runtime using IPX protocol" dialog box will appear next.

WinPLC Name is Required

The "Reset CE Runtime using IPX protocol" window requires you to name the WinPLC module. You can use up to 15 alphanumeric characters.

If the name you select does not conform to the length or character usage requirements, you will see the TargetPicker error message shown below

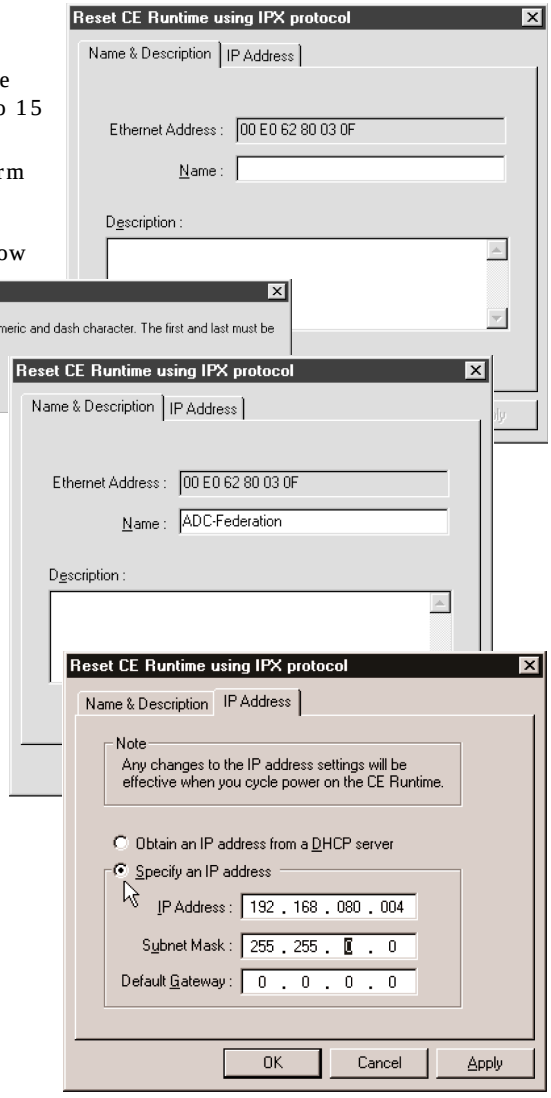


You can also assign an optional description to the WinPLC in the field provided.

IP Address is Required

Now, click on the "IP Address"tab. Assign an IP Address and Subnet Mask that are compatible with the IP Address and Subnet Mask of your Think & Do Development System computer.

When you have completed the IP Address and Subnet Mask (and default Gateway, if necessary), click OK.

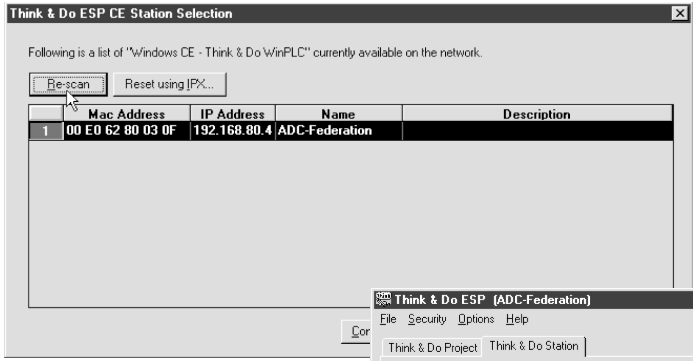


Cycle Power After Assigning IP Address

A TargetPicker message box will pop up to let you know you must cycle power to make the new IP address effective.

After you click OK and cycle power to the DL205, the "Think & Do ESP CE Station Selection" window will reappear (as shown below). Click on the button labeled Re-scan, and you should see the WinPLC module listed by its MAC address, IP address, Name, and Description.

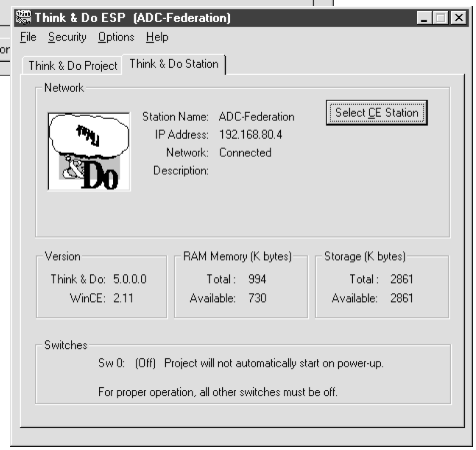
If you do not see your WinPLC module listed, check to be sure the power is on to the DL205. If the power is on, recheck your IP addresses on the WinPLC and your Think & Do Development System computer for compatibility. Also, make sure you have loaded the IP protocol on your Development System computer and that you are using the appropriate connecting cable (straight-through or crossover).



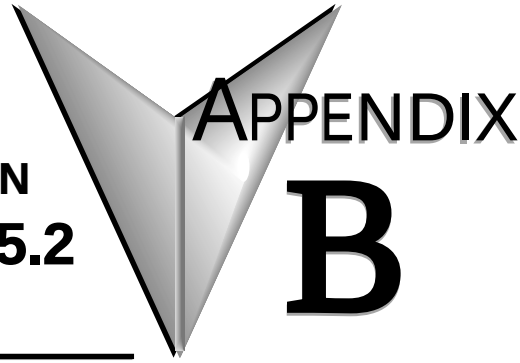
Select, Exit, You're Connected

If your module is listed on the window above but not highlighted, click on the module information and click the button labeled Select. Then, click Exit.

You will return to the initial Think & Do ESP window which will now show your Think & Do and Windows CE Version numbers as well as memory usage information. At this point, you are successfully linked to the WinPLC. You can continue to develop your project or download it to the WinPLC using the methods described in the Think & Do Software Learning Guide.



SERIAL I/O MODULE INSTALLATION / OPERATION WHEN USING T&D VER. 5.2 OR 5.3



In This Appendix...

Note: This Appendix only applies if you are using the Serial I/O Module with Think & Do versions 5.2 or 5.3. Use Chapter 3 if using the Serial I/O Module with Think & Do Studio version 6.0 or later.

Only Think & Do WinPLCs (H2-WPLC1-TD and H2-WPLC2-TD) support the H2-SERIO module.

H2-SERIO Overview

Add Serial Ports to Your WinPLC

The Serial I/O module plugs into the DL205 I/O base and is used exclusively with the WinPLC to provide additional RS232 serial ports. The WinPLC communicates with the H2–SERIO module across the DL205 backplane.

As Many as Ten Serial Ports

The WinPLC has one built-in serial port. Now, you can add as many as nine additional serial ports for Think & Do applications requiring multiple serial devices, such as barcode scanners.

Setting Communication Parameters Using Think & Do

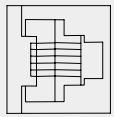
Use I/O View to set baud rate, parity, data bits, and stop bits for each port. Choose from 300 to 57,600 baud communication speeds. Think & Do allows each port to be designated as a MODBUS slave or a generic serial device. Each port on the H2–SERIO module is capable of hardware handshaking.

RS-232 Wiring

Pin Assignments for: H2-SERIO ports

1	0V	Power (–) connection (GND)
2	CTS	Clear to Send
3	RXD	Receive Data (RS232C)
4	TXD	Transmit Data (RS232C)
5	RTS	Request to Send
6	0V	Signal Ground (GND)

RJ12 (6P6C) Female Modular Connector



NOTE: The serial port on-board the WinPLC has a different pinout from the H2–SERIO module. Refer to page 1–7 for the WinPLC serial port pin assignments.

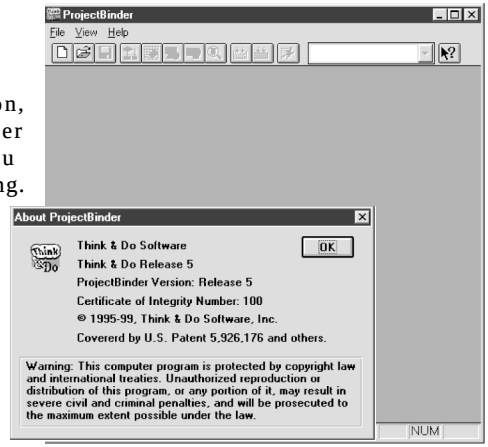
H2-SERIO Specifications	
Module type	Intelligent module for use with H2–WPLC1–TD
Maximum number of modules supported by one WinPLC	3
Recommended cable	Belden 9729 or equivalent
Connector	RJ12 jack
Power consumption	230mA @ 5VDC
Operating environment	0° to 60°C (32°F to 140°F), 5% to 95% RH (non-condensing)
Manufacturer	Host Engineering

Using Think & Do to Set Serial Port Parameters

Check Think & Do Version First

You will need Version 5.2 of Think & Do, to recognize the H2–SERIO module. To determine whether you have the right version, open the Project Binder. As the Project Binder opens, you may notice a screen that tells you which Version of Think & Do you are opening. That screen disappears as the Project Binder opens.

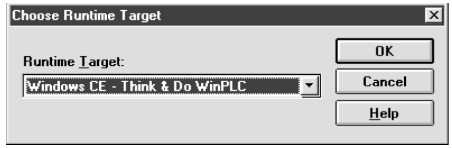
After the Project Binder is open, you can click on "Help," and the bottom menu option, "About Project Binder," will tell you which Version you are using.

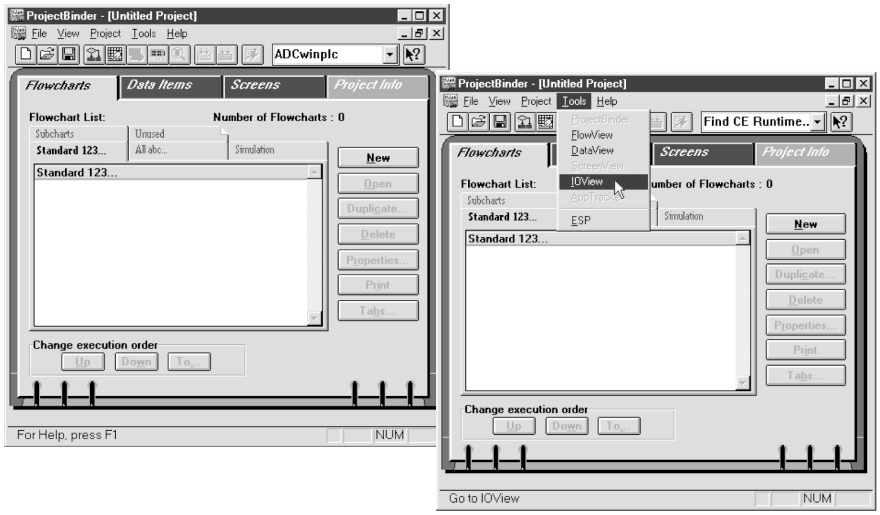


New Project Using H2-SERIO Module

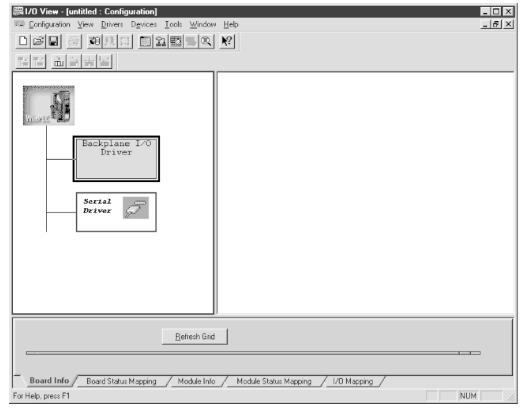
Start a new project by clicking on the blank document button. A dialog box will pop up asking you to "Choose Runtime Target." Select "Windows CE – Think & Do WinPLC."

You will see a new Untitled Project open. Next, click on the "Tools" menu and select I/O View.



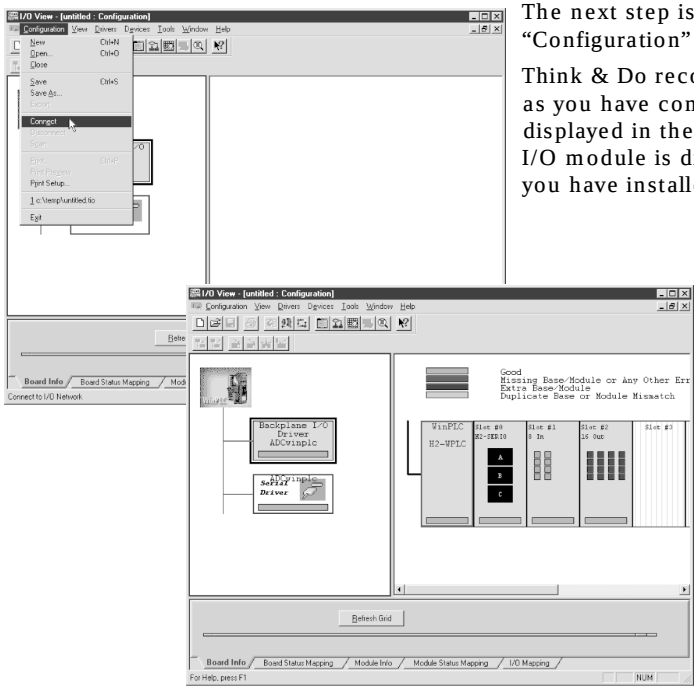


Notice in I/O View that the drivers for the DL205 backplane and the WinPLC serial port are already loaded. You will see them graphically represented in the left pane of the I/O View window.



Connecting to the WinPLC

Prior to the next step, you will need to install the H2–WPLC1–TD and the H2–SERIO module in your DL205 base. Please refer to the guidelines elsewhere in this publication for information about installation, power wiring, and Ethernet connections. The WinPLC must be recognized on the network to proceed. Use “Think & Do ESP” to establish your link to the WinPLC, as described on pages 1–12 through 1–15.



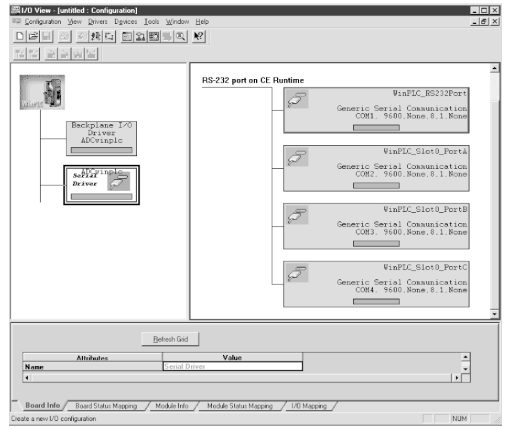
The next step is to click on “Configuration” and select “Connect.”

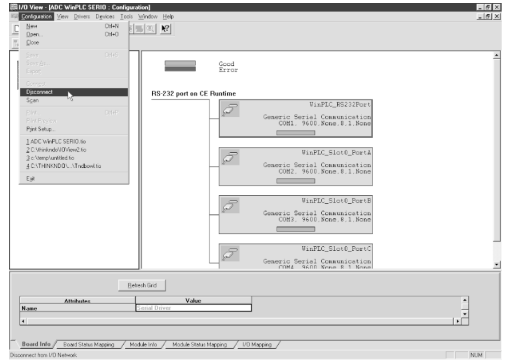
Think & Do recognizes the DL205 base as you have configured it. The WinPLC is displayed in the CPU-slot, and the Serial I/O module is displayed in the slot where you have installed it.

Click on the Serial Driver in the left pane of the I/O View window. You will see a port configuration box for each serial port Think & Do recognizes. In our example to the right, Think & Do sees four serial ports. One is on the WinPLC and the other three are on the Serial I/O module.

Notice that the ports are numbered COM 1 through COM 4 in Think & Do. COM 1 is on the WinPLC. COM 2 through COM 4 are on the first Serial I/O module in the base. Think & Do counts the serial ports from top to bottom (on the Serial I/O module) and from left to right in terms of slot position.

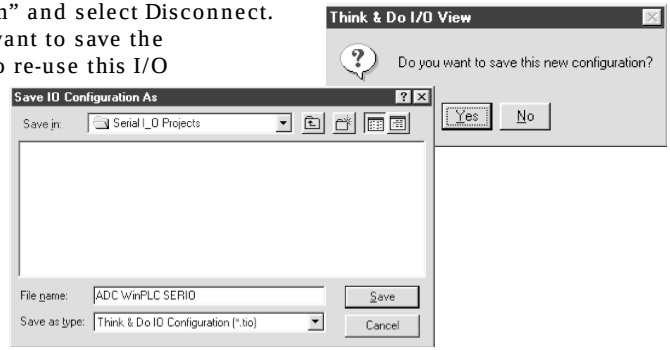
If you install additional Serial I/O modules at a later time, be aware that the module's slot position determines its COM number. If you install a Serial I/O module between an existing Serial I/O module and the CPU, your port settings will remain the same, but the COM number will change.





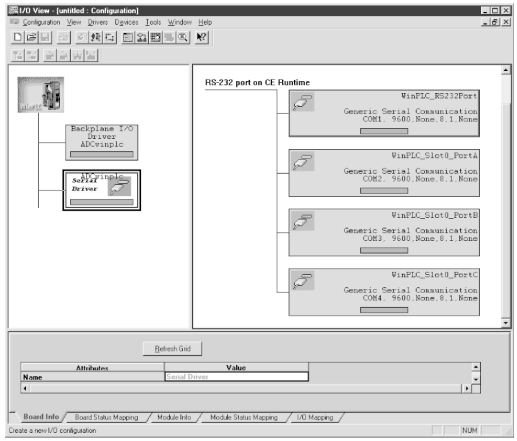
NOTE: You must be disconnected from the WinPLC and the I/O base in order to change the serial port parameters on the H2–SERIO module. To disconnect, you have two choices. You can click on Configuration/Disconnect as shown above, or you can physically disconnect the WinPLC by removing power or by removing the Ethernet cable.

Now, click on "Configuration" and select Disconnect. Think & Do will ask if you want to save the configuration. If you want to re-use this I/O Configuration later, click yes. If you click yes, you will see the "Save I/O Configuration As" screen. Name the configuration and click save.



Setting Serial Port Parameters

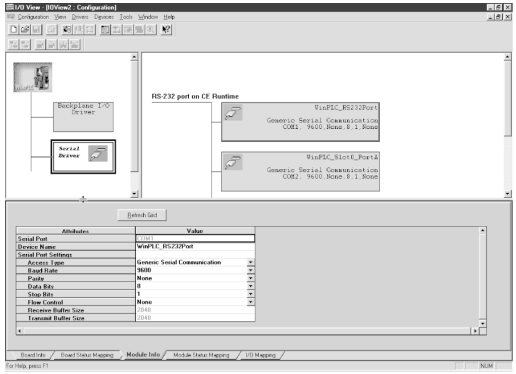
To set the serial port parameters, click on the Serial Driver in the left pane of the I/O View window. You will see a port configuration box for each serial port Think & Do recognizes. In our example to the right, Think & Do sees four serial ports. One is on the WinPLC and the other three are on the Serial I/O module.



Expand the Window Pane

Position your cursor on the line that separates the upper window panes from the lower window pane. Move this line up by dragging your mouse. Click on the tab at the bottom of the lower window pane marked Module Info. You will see a screen that looks similar to the one shown here. Pull-down menus allow you to change the serial port parameters.

Select the port whose parameters you want to change by clicking on that port in the upper right pane. Make the changes in the lower pane, and save the changes using the Ctrl + S keys.



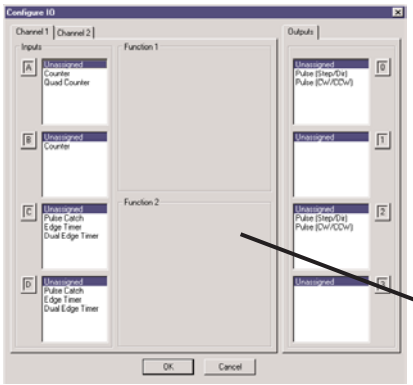
High-Speed Counter I/O Module

Overview

The High-Speed Counter I/O (CTRIO) module is designed to accept high-speed pulse-type input signals for counting or timing applications and designed to provide high-speed pulse-type output signals for stepper motor control, monitoring, alarm or other discrete control functions. The CTRIO module offers great flexibility for applications that call for precise counting or timing, based on an input event or for high speed control output applications.

The CTRIO module has its own microprocessor and operates asynchronously with respect to the PLC/Controller. This means that the on-board outputs respond in real time to incoming signals so there is no delay waiting for the PLC/Controller to scan I/O.

The H2-CTRIO module is designed to work with incremental encoders or other field devices that send pulse outputs.



Use Configure I/O dialog to assign the CTRIO input and output functions

CTRIO features

The CTRIO modules offer the following I/O features:

- 8 DC sink/source inputs, 9-30VDC
- 4 isolated sink/source DC outputs, 5-36 VDC, 1A per point

Inputs supported:

- 2 quadrature encoders counters up to 100KHz, or 4 single channel counters up to 100KHz using module terminals Ch1A, Ch1B, Ch2A and Ch2B
- High speed edge timers, dual edge timers, pulse catch, count reset, count inhibit or count capture or home search limits using module terminals Ch1C, Ch1D, Ch2C or Ch2D

Outputs supported:

- 4 independently configurable high-speed discrete outputs or 2 channels pulse output control (20Hz-25KHz per channel or 50KHz if only using one channel)
- Pulse and direction or cw/ccw pulses supported for pulse output control
- Raw control of discete output directly from user control program

Software Configuration

All scaling and configuration is done via CTRIO Workbench, a Windows software utility program. This eliminates the need for PLC ladder programming or other interface device programming to configure the module. CTRIO Workbench runs under Windows 98/2000/XP and NT 4.0 SP5 or later.

CTRIO Workbench main configuration screen

Typical applications

- High-speed cut-to-length operations using encoder input
- Pick-and-place or indexing functions controlling a stepper drive
- Dynamic registration for web material control
- Accurate frequency counting for speed control with onboard scaling
- PLS (Programmable Limit Switch) functions for high-speed packaging, gluing, or labeling
- Sub 10 usec pulse-catch capability for high-speed product detection
- Functions for level or flow

Supported systems

Multiple CTRIO modules can reside in the same base provided that the backplane power budget is adequate. Depending upon which CPU/interface module is used, there may be I/O base slot restrictions for the CTRIO module. Refer to the CTRIO High Speed Counter Manual (HX-CTRIO-M) for I/O slot restrictions.

*Direct*LOGIC DL205 PLC

You can use the H2-CTRIO module with the D2-240, D2-250(-1) and D2-260 CPUs. (It is not supported in local expansion bases or in D2-RSSS serial remote I/O bases.)

DL205 Win PLC

The H2-CTRIO module can be used in DL205 WinPLC systems (H2-WPLC*-**).

PC-based Ethernet I/O control systems

The H2-CTRIO module can be used in PC-based control systems using the H2-EBC interface module

ERM to EBC systems

The H2-CTRIO module is supported in H2-EBC slaves in H*-ERM systems. This includes the supported DL205 CPUs and WinPLC systems.

Profibus systems

The H2-CTRIO module can be used in Profibus systems using the H2-PBC slave interface module.

High-Speed Counter

I/O Specifications

General	
Module Type	Intelligent
Modules Per Base	Limited only by power consumption
I/O Points Used	None, I/O map directly in PLC V-memory or PC control access
Field Wiring Connector	Standard removable terminal block
Internal Power Consumption	400mA Max at +5V from Base Power Supply, Maximum of 6 Watts (All I/O in ON State at Max Voltage/Current)
Operating Environment	32°F to 140°F (0°C to 60°C), Humidity (non-condensing) 5% to 95%
Manufacturer	Host Engineering
Isolation	2500V I/O to Logic, 1000V among Input Channels and All Outputs

H2-CTRIO Input Specifications	
Inputs	8 pts sink/source 100 kHz max.
Minimum Pulse Width	5 µsec
Input Voltage Range	9-30VDC
Maximum Voltage	30VDC
Input Voltage Protection	Zener Clamped at 33VDC
Rated Input Current	8mA typical 12mA maximum
Minimum ON Voltage	9.0VDC
Maximum OFF Voltage	2.0VDC
Minimum ON Current	5.0mA (9VDC required to guarantee ON state)
Maximum OFF Current	2.0mA
OFF to ON Response	Less than 3 µsec
ON to OFF Response	Less than 3 µsec

H2-CTRIO Output Specifications	
Outputs	4 pts, independently isolated, current sourcing or sinking FET Outputs: open drain and source with floating gate drive
Voltage range	5VDC - 36VDC
Maximum voltage	36VDC
Output clamp voltage	60VDC
Maximum load current	1.0A
Maximum load voltage	36VDC
Maximum leakage current	100µA
Inrush current	5A for 20ms
OFF to ON response	less than 3µsec
ON to OFF response	less than 3µsec
ON state V drop	0.3V max.
External power supply	for loop power only, not required for internal module function*
Overcurrent protection	15A max
Thermal shutdown	Tjunction = 150°C
Overtemperature reset	Tjunction = 130°C
Duty cycle range	1% to 99% in 1% increments (default = 50%)
Configurable Presets a) single b) multiple	a) each output can be assigned one preset, or b) each output can be assigned one table of presets, one table can contain max. 128 presets, max. predefined tables = 255

*** User supplied power source required for stepper drive configuration.**

H2-CTRIO Input Resources	
Counter/Timer	4, (2 per 4 input channel group)
Resource Options	1X, 2X, or 4X Quadrature, Up or Down Counter, Edge Timer, Dual Edge Timer, Input Pulse Catch, Reset, Inhibit, Capture
Timer Range / Resolution	4.2 billion (32 bits); 1 µsec
Counter Range	+ / - 2.1 billion (32 bits or 31 bits + sign bit)

H2-CTRIO Output Resources	
Pulse output / Discrete outputs	Pulse outputs: 2 channels (2 outputs each channel) Discrete outputs: 4 pts.
Resource Options	Pulse outputs: pulse/direction or cw/ccw; Profiles:Trapezoid, S-Curve, Symmetrical S-Curve, Dynamic Position, Dynamic Velocity, Home Search, Velocity Mode, Run to Limit Mode and Run to Position Mode Discrete outputs: 4 configurable for set, reset, pulse on, pulse off, toggle, reset count functions (assigned to respond to Timer/Counter input functions). Raw mode: Direct access to discrete output from user application program
Target Position Range	+ / - 2.1 billion (32 bits or 31 bits + sign bit)

HIGH-SPEED COUNTER

Status indicators

H2-CTRIO LED Descriptions	
OK	Module OK
ER	User Program Error
1A	Channel 1 Status
2A	Channel 2 Status
0 - 3	Output Status

H2-CTRIO LED Diagnostic Definitions		
LED OK	***LED ER***	***Description***
ON	OFF	All is well - RUN Mode
ON	ON	Hardware Failure
Blinking	Blinking	Boot Mode - Used for Field OS Upgrades
Blinking	OFF	Program Mode
OFF	Blinking	Module Self-diagnostic Failure
OFF	ON	Module Error Due to Watchdog Timeout
OFF	OFF	No Power to Module

H2-CTRIO LED Diagnostic Definition	
1A/2A	
Blinking 7 times per second	Input is configured as Counter and is changing
Following state of input	Input is not configured as counter
0-3	
Follow actual output state: ON = output is passsing current	

Installation and wiring

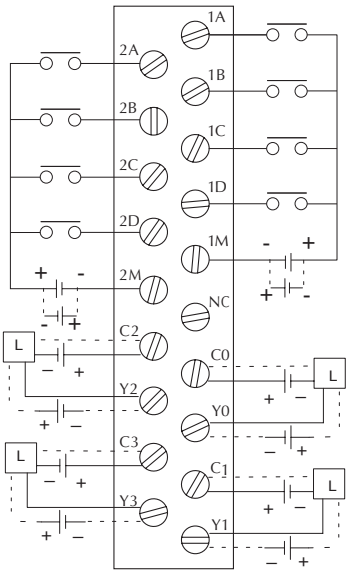
The H2-CTRIO module has two independent input channels, each consisting of 4 optically isolated input points (pts. 1A-1D on common 1M and pts. 2A-2D on common 2M). The inputs can be wired to either sink or source current.

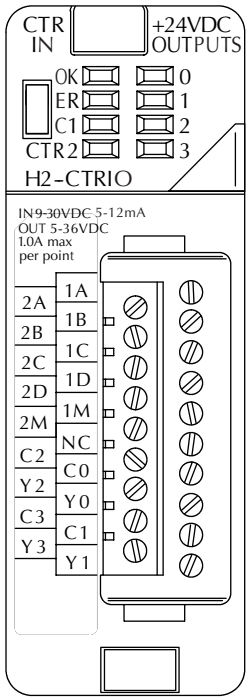
The module has 4 optically isolated output points (pts. Y0-Y3 with isolated commons C0-C3, respectively). The outputs must be wired so positive current flows into Cn terminal and then out of the Yn terminal (see the diagram on the following page).

Remember that the internal jumpers can be used to connect the input commons or outputs/output commons together.

The module is configured, using CTRIO Workbench, to accommodate the user's application. The function of each input (counting, timing, reset, etc.) and output (pulse output, discrete output, etc.) is defined in the configuration of the module.

See the notes below for further details about power source considerations, circuit polarities, and field devices.





Notes:

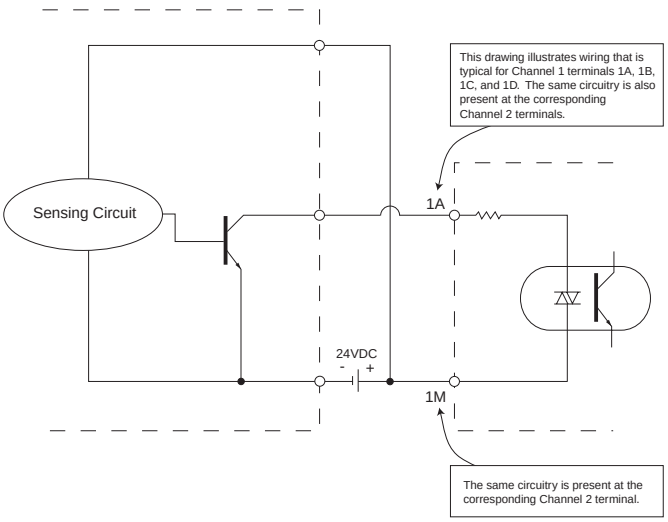
1. ***Inputs (1A, 1B, 1C, 1D and 2A, 2B, 2C, 2D) require user-provided 9-30VDC power sources. Terminals 1M and 2M are the commons for Channel 1 and Channel 2 inputs. Maximum current consumption is 12mA per input point.***
2. ***Polarity of the input power sources can be reversed. Consideration must be given, however, to the polarity of the field device. Many field devices are designed for only one polarity and can be damaged if power wiring is reversed.***
3. ***Outputs have one polarity only and are powered by user-provided 5-36VDC power sources. The maximum allowable current per output circuit is 1A.***

High-Speed Counter

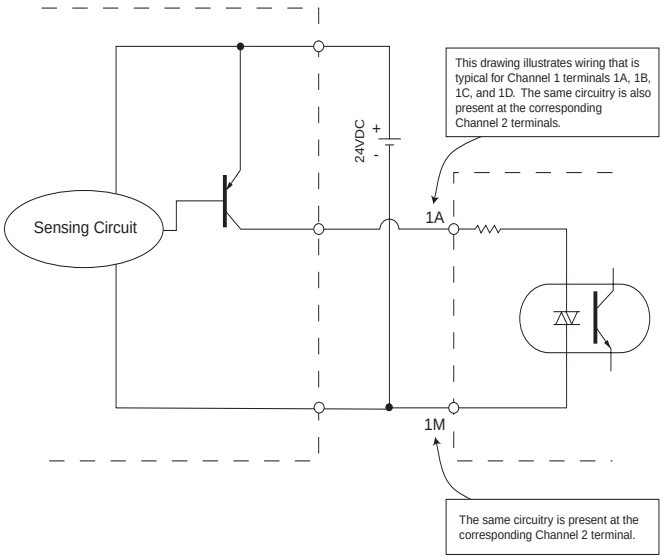
Solid state input wiring device

DC types of field devices are configured to either sink or source current. This affects the wiring of the device to the CTRIO module. Refer to the sinking/sourcing appendix in this desk reference for a complete explanation of sinking and sourcing concepts.

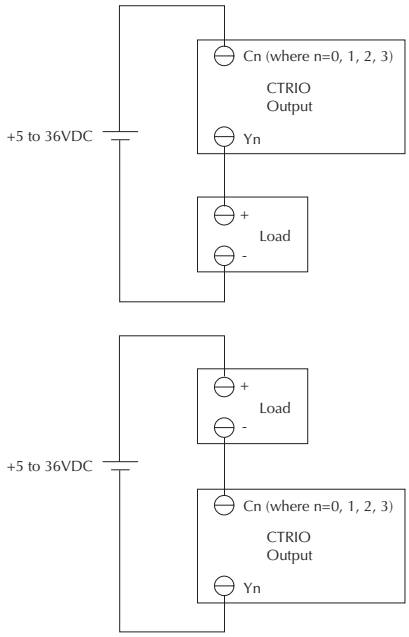
NPN Field Device (sink)



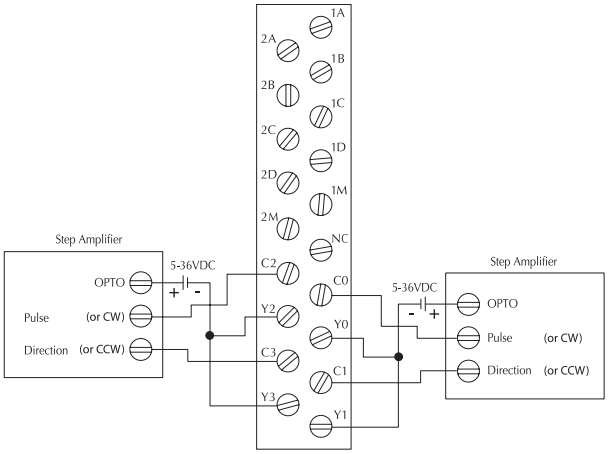
PNP Field Device (source)



Pulse output schematic



Stepper/Servo drive wiring example

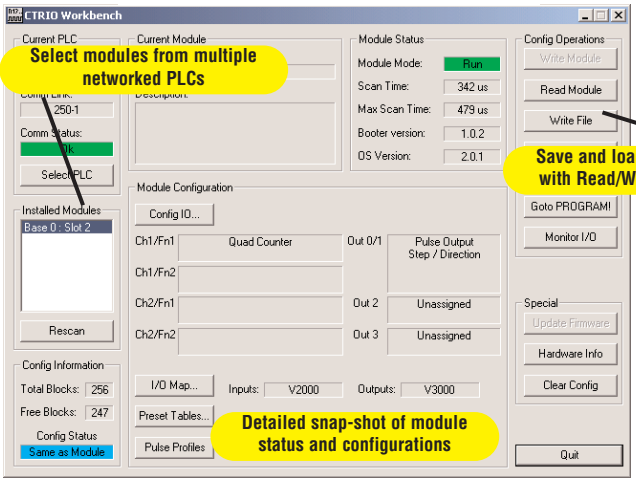


HIGH-SPEED COUNTER

Fill-in-the-blank configuration software

The CTRIO Workbench is the software utility used to configure the CTRIO module and to scale signals to desired engineering units. Workbench also allows you to perform various other functions, such as switching between the CTRIO's Program mode and Run mode, monitoring I/O status and functions, and diagnostic control of module functions. The CTRIO Workbench utility ships with the CTRIO User Manual. You can also download the latest version free at the Host Engineering Web site: www.hosteng.com.

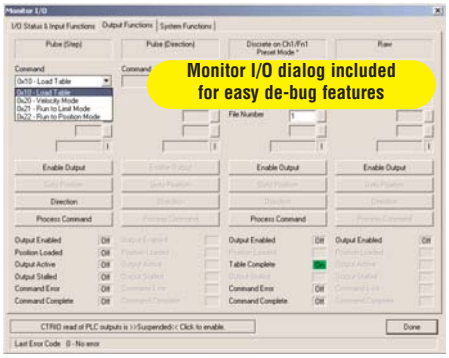
CTRIO Workbench main configuration screen



CTRIO Workbench diagnostics and monitoring

The Monitor I/O dialog is accessible from the main Workbench dialog when the module is in Run Mode. This allows for a convenient way to test and debug your configuration prior to installation. The Monitor I/O dialog is divided into three functional areas: Input Functions, Output Functions and System Functions. The data displayed under the Input Functions tab includes all input Dword parameters, status bits and the current status of each configured input and output function. The fields displayed under the Output Functions tab includes all output (D)word parameters and configuration information that can be altered during runtime and the bits that indicate successful transfers or errors. The System Functions can be used to read from or write to the CTRIO's internal registers.

Monitor I/O screen

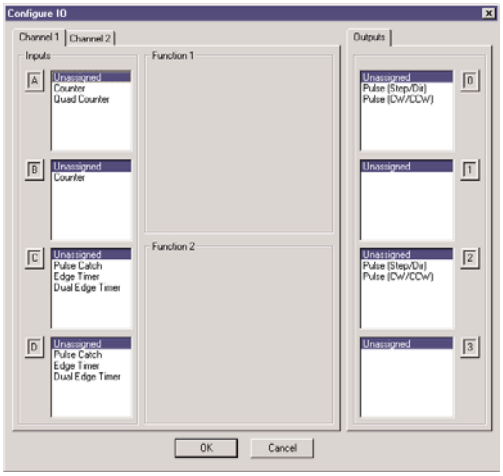


CTRIO Workbench configure I/O setup

The Configure IO dialog is the location where input and output functions are assigned to the module. The choice of input and output functions determines which options are available. The input function boxes prompt you with selections for supported functions. The Workbench software automatically disallows any unsupported configurations.

H2-CTRIO

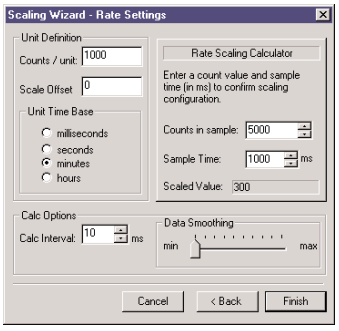
Configure I/O screen



CTRIO Workbench on-board scaling

Scaling raw signals to engineering units is accomplished using the Scaling Wizard. The Scaling Wizard options are different for the Counter functions as compared with the Timer functions. "Position" and "Rate" scaling are available when you select a Counter function. "Interval" scaling is available when you select a Timing function.

Scaling Wizard screen



High-Speed Counter

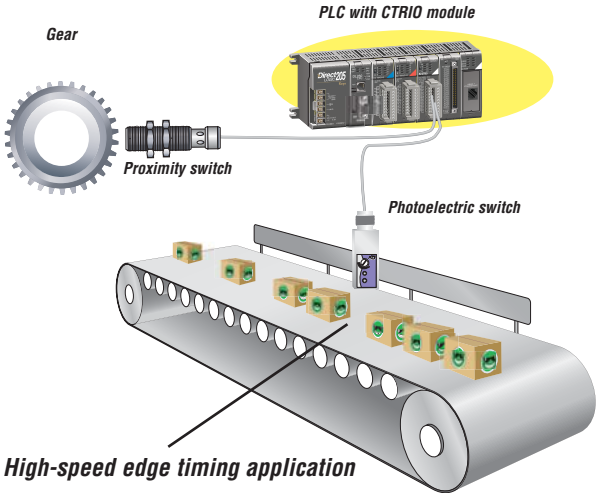
High-speed input operations

The CTRIO module is capable of a wide variety of high speed input and output operations all within one module. With its flexible 2-channel input and separate 2-channel output design, the CTRIO can satisfy both high-speed counting, timing, pulse catch operations, along with high speed discrete output or several profile choices of pulse output operations. Not all combinations of input functions and output functions are possible within the resources of the module, but the following examples are some of the most common applications for the CTRIO. Check out these examples and see how they relate to your high speed application needs.

High-speed timing

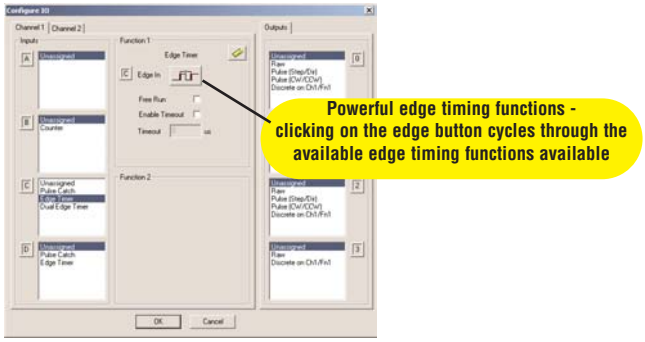
The CTRIO can be configured for timing functions based on both count or rate. Using a common configuration of a proximity switch sensing the teeth on a gear, the module is able to calculate the velocity of the gear based on the rate it receives its counts. This value can be scaled within the module to the engineering units required for the application.

High-speed timing application



High-speed edge timing application

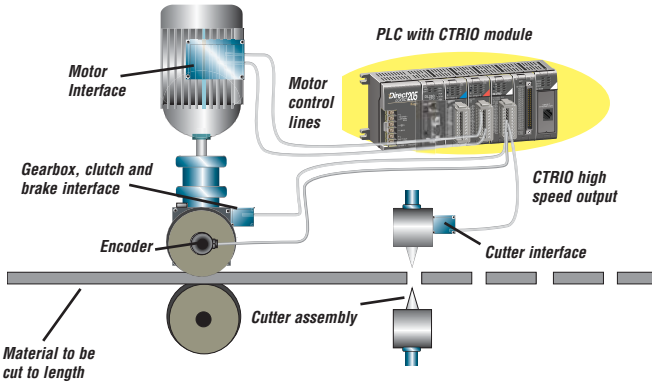
Using Configure I/O screen to configure CTRIO for high-speed timing



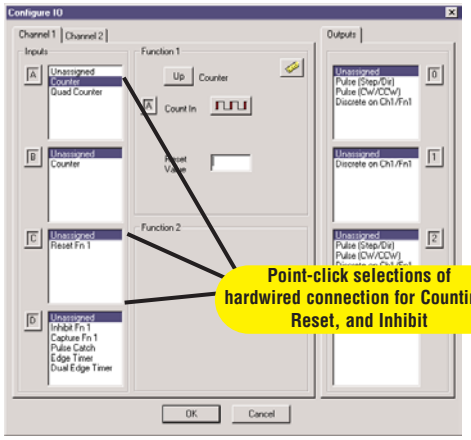
High-speed counting

The CTRIO can be configured for counting functions for the use of an encoder input, (up to two quadrature encoders per module) with available connections for external reset, capture and inhibit signals. In a simple cut to length application as shown, the encoder provides an input position reference for the material to the module. The module's high speed outputs are wired to the cutting device and to the clutch and/or braking device. When the count from the encoder is equal to a pre-programmed setpoint within the module, the high speed outputs are activated to stop and cut the material to a repeatable fixed length. Additionally, the clutch/brake signal can be used for an inhibit signal to not accumulate counts while the material is being cut.

High-speed cut-to-length application



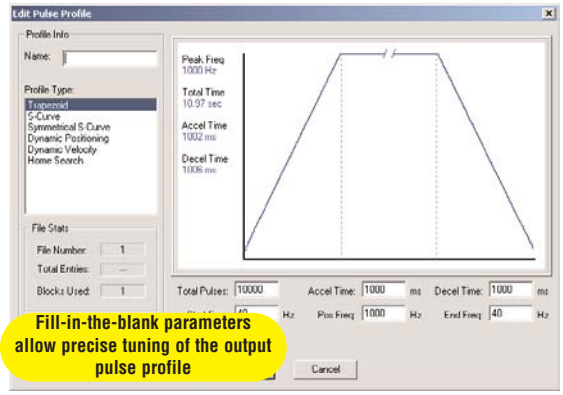
Using Configure I/O screen to configure CTRIO for high-speed counting

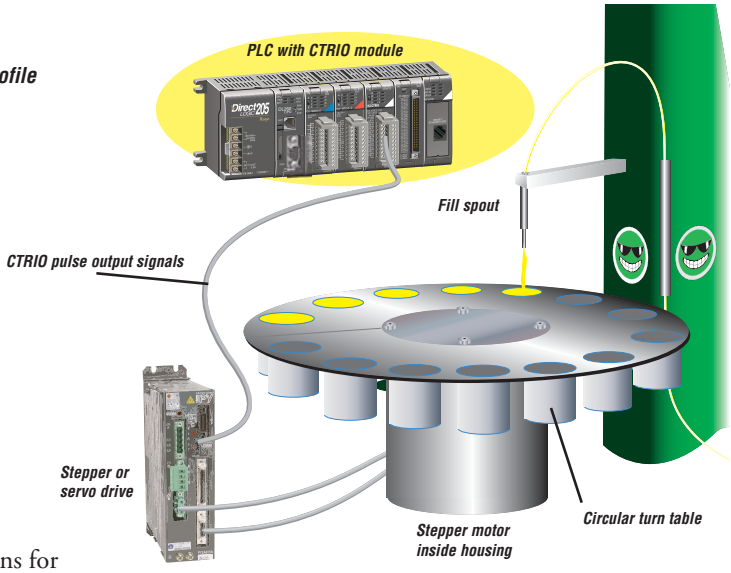


HIGH-SPEED COUNTER

Pulse output operations

Using Edit Pulse Profile screen to select Trapezoid pulse output profile





Rotary indexing liquid fill application

Pulse output for stepper/servo control

The CTRIO module is capable of multiple configurations for pulse output control, most often when connected to a stepper or servo drive system. The module can deliver a pulse output signal up to a maximum of 25Khz on two channels with support for pulse-and-direction or CW/CCW pulses. The available profile choices include Trapezoid, S-Curve, Symmetrical S-Curve, Dynamic Positioning, Dynamic Velocity and Home Search. All profiles can be easily configured using the CTRIO Workbench software with fill-in-the-blank parameter fields and a graphic representation of the selected profile. Three additional profiles are available that are completely controlled by the user program (no CTRIO profile is configured). They are Velocity Mode, Run to Limit Mode and Run to Position Mode.

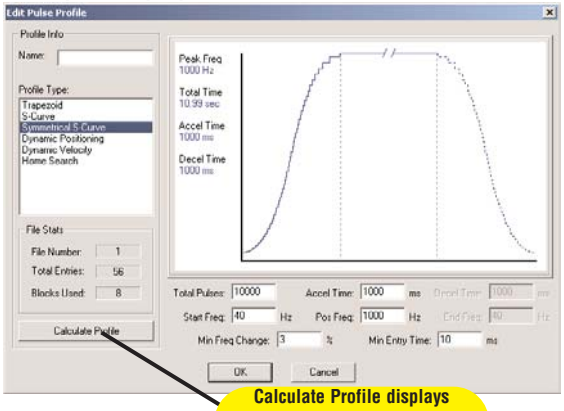
Example application

In a simple rotary indexing application, as shown above, a fixed Trapezoid profile is chosen. The CTRIO for this application is wired to a stepper drive for pulse-and-direction. The requirement for this application is to provide a smooth movement of the rotary table to allow product to be filled into individual containers equal distance apart. The predetermined number of pulses required for each movement is entered into the CTRIO Workbench as "Total Pulses" along with the Starting Frequency, Ending Frequency, and Positioning Frequency (speed after acceleration). The Acceleration and Deceleration parameters are entered in units of time, so no ramp-distance calculations are required. After all parameters are entered, a graphical representation of the configured profile is shown automatically. Once the configuration has been downloaded to the module, all that is needed is from the PLC CPU is the Enable Output signal to begin a movement.

Other common pulse output applications:

- S-Curve accel/decel profile for signaling a stepper or servo drive that needs a curved acceleration and deceleration profile, i.e. for diminishing any initial "jerk" upon movement of static products, boxes on conveyors, liquids in containers on an indexer, printing registrations, etc.
- Dynamic Positioning for any run-to-a-specific-position requirement, either by a pre-programmed count of an external high speed discrete input wired to the module. This is popular in winding or webcontrol with any dynamic registration mark or variable speed requirement.
- Home search routines to seek a home position based on CTRIO discrete input limit(s).

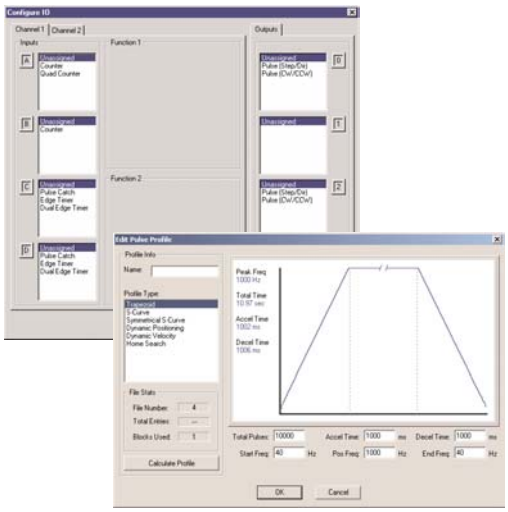
Example of S-Curve acceleration and deceleration pulse output profile



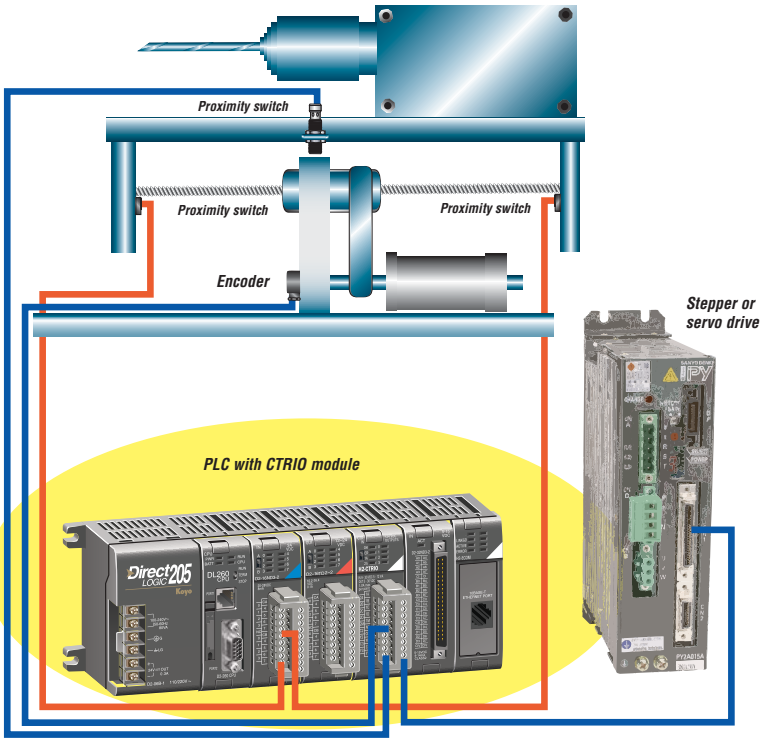
High-Speed Counter

Combining high-speed input and pulse output operations

Using CTRIO Workbench to configure the module for simultaneous high-speed input and high-speed pulse output operation



Multihead drill machine application

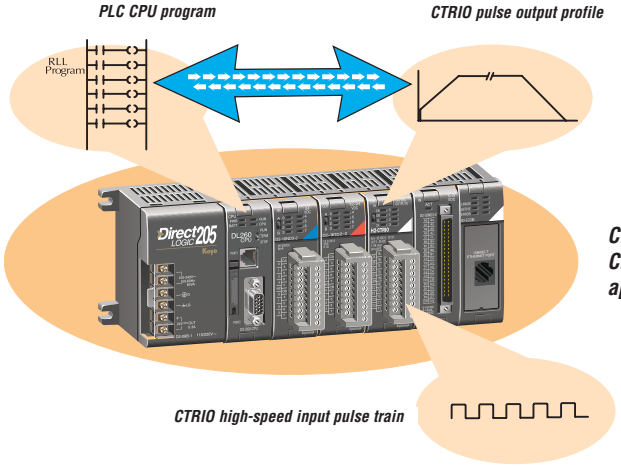


High-Speed inputs and pulse output combinations

The flexible design of the CTRIO module allows for combining high speed inputs and delivering high speed pulse outputs signals simultaneously. There are limitations to this type of configuration in that the module does not internally support closed loop control. Providing closed loop control with the CTRIO involves additional PLC code to coordinate this control, making the application subject to the PLC CPU program scan. Simple position/speed monitoring via a high speed counting input for non-critical response while providing pulse outputs to a drive, is easily achievable for the CTRIO.

Example application

In the simple drill-head application shown above, the CTRIO pulse outputs are wired to a stepper and/or servo drive. The inputs are wired to an encoder attached to the lead screw on the movable portion of the drill-head assembly. The CTRIO module output pulse train to the drive allows the motor to spin the lead screw making the drill move forward into the passing material. The encoder monitors the speed and position of the drill-head. Prox switches at each end act as limit switches ensuring the drill-head will not over-travel. A home sensor is positioned in the middle of the assembly which allows the PLC to reset the count.



Closed loop control for the CTRIO module requires PLC CPU program interaction to close the loop. This makes the application subject to the PLC CPU scan.

ANIMATICS

Page Number:	Page 1 of 34
Revision:	H
Effective:	May 5, 2001
Replaces:	November 16, 1999

SmartMotor™ and RTC3000 Controller
Expanded Capabilities List

Motion Control Chip Firmware Version :
4.00-Q1 to 4.00-Q3 ,
4.10 to 4.15,4.75
4.40 to 4.41

Animatics is upgrading SmartMotors™ with its new firmware Version 4.15,4.75,4.40 & 4.41 Motion Control Firmware. This new version is backward compatible with your older programs, but has substantially increased functionality.

This document is intended to serve as an addendum to the existing manual until a new manual can be published.

A CD containing the Windows based terminal software, SmartMotor Interface "SMI", should come with the manual. The software is also downloadable on our website at www.animatics.com or www.smartmotor.com. For first time users, refer to the Help section in SmartMotor Interface (SMI) software.

To find out your SmartMotor firmware version, type the following command in the "SmartMotor Terminal" window "**RSP**" and the motor responds by returning the sample rate and firmware version. For example a response of "**24576/412**" the numbers after the slash "/" is the firmware version number "412" means you have a 4.12 firmware version motor. A "xxxxxx/Qx" means you have a version 4.00-Qx motor (where x are numbers). A "xxxxx-Mx" means you have a version 3.4 motor.

TABLE OF CONTENTS

INTRODUCTION

This section covers features and motion commands that are not covered in the current SmartMotor Manual.

The SmartMotor is a brushless DC servo-motor that uses a built in controller and amplifier to perform programmed motion. The servo controller uses closed loop PID control. The SmartMotor and RTC3000 can either be run from a program downloaded to its memory or from commands received from RS232 or RS485 communication cable.

We recommend first time users to program the SmartMotor and RTC3000 in the Windows™ based SmartMotor Interface (SMI) software. SmartMotors can also run on the DOS based interface software TermV400. Both software comes on the CD with the SmartMotor and is also available on the website at:
http://www.animatics.com/support.shtml

Click on SMIsetup.exe (single file) or SMIdisk1.exe to SMIdisk4.exe to save onto floppy disks.

The SmartMotor and RTC3000 can be to receive commands from other software that sends ASCII characters such as C++,C,Visual Basic , data acquisition software and industrial devices.

For first time users refer to help-topics in the Windows based SmartMotor Interface Software (SMI). To get to these files open the SMI software and select Help>Helptopics from the toolbar. The help topics include material covered in the printed manual, additional program examples, hardware diagrams and the full command sets of the different firmware versions.

When starting the SMI and TermV400 software for the first time it defaults to com port 1, 9600 baud, RS232 protocol. Please change these settings for your system com port. The SmartMotor controller and RT3000 defaults to 9600 baud, RS232, 8data bits, 1stop bit, non-parity through the 7 pin connector.

FIRWARE VERSIONS:

Firmware version 4.15:

Differences from 4.12:

- Added Directional Limit and Limit Trigger High/Low software selectable function. (LIMD,LIMN,LIMH and LIML)
  Default is non-directional limit (LIMN) and limit trigger low (LIML).
- Added Coordinated Motion (Multi-axis contouring)
- The motor will not auto-execute the stored program on the EEPROM if the characters "EE" is transmitted to the motor within ½ second after power up.
- ADDR=<expression> used for setting motor address (SADDR<value> is also accepted) on dasiy chain/multidrop . (SADDR<value> is also accepted)
- F=8 command added. Command causes the PID integral term to be cleared at the end of the trajectory.
- MD50 command added. This puts the motor into torque mode and sets the "T" by reading analog voltage at port A. The "T" value uses a linear scale where:
  5.0v input sets T=1023
  2.5v sets T=0,
  0v sets T=-1023

Firmware version 4.75:

Differences from 4.15:

- Default is Directional limit (LIMD) and limit trigger high (LIMH).

Firmware version 4.40/4.41:

Differences from 4.12:

- Added Direction Limit and Limit Trigger High/Low software selectable function. (LIMD,LIMN,LIMH and LIML)
  Default is non-directional limit (LIMN) and limit trigger low (LIML).
- Added Coordinated Motion (Multi-axis contouring)
- No External Encoder Inputs (ENCA input & ENCB input)
- No Mode Follow, Mode Follow with ratio and Mode CAM.
- The motor will not auto-execute the stored program on the EEPROM if the characters "EE" is transmitted to the motor within ½ second after power up.
- ADDR=<expression> used for setting motor address on RS232 daisy chain/ RS485 multi-drop . (SADDR<value> is also accepted)

SECTION 1.0: CREATING MOTION

The main modes of operation of the SmartMotor and RTC3000 controller:
1) Position Mode-User wants to control position of motor.
Motor moves to set position in at user defined acceleration and velocity.
SmartMotors default to this mode on power-up, unless otherwise defined in the program downloaded to program memory.

2) Velocity Mode-User wants to have a constant controlled velocity.
Motor rotates at a user set acceleration and constant velocity.

3) Torque Mode-User wants to control torque and speed of motor.

To decelerate to a stop issue "X". To stop as fast as possible issue "S" command.

POSITION MODE:
Position Mode controls the position of the shaft based on encoder signal. The SmartMotor defaults to reads its differential internal encoder (section 1.3 covers external encoder implementation).
SmartMotors come standard with :
500 line encoder for SM1720,SM23xx
1000 line encoder in the SM34xx,SM42xx,SM56xx
The encoder lines are read in quadrature (the 4 edges of encoder lines are read) that allows the motors to be controlled to a resolution of 2000 encoder counts (steps) per revolution for a 500 line encoder (0.18 degree per encoder count. And 4000 encoder counts (steps) per revolution for a 1000 line encoder (0.09 degree per encoder count).

In this mode you want to get to a position **"P"** at a desired acceleration **"A"** and desired velocity **"V"**. The controller accelerates to the set velocity and decelerates (at same rate of acceleration) to a stop at set position. Note commands are in uppercase and variable names are in lower case letters.

Position Mode can be used as absolute position **"P"** (move relative zero position) or in relative postion move **"D"** (move relative to current shaft position). Refer to Section 1.1 to calculate "A" and "V" values.
Note a positive velocity value is a clockwise rotation if you are looking at the mounting plate and shaft. Negative is counterclockwise. Position uses same sign convention. Note SmartMotor commands are shown in bold and a different font.

Sample:

```
MP          ' set to position mode
O=0         ' set current shaft position as  zero position called Origin or Home
            '  (command is letter O equal zero)
A=100       'set acceleration
V=32212     'set velocity
            Absolute Position Move
P=2000      ' move to position 2000 in the positive direction relative to the zero position.
G           'go and start move

            Relative Position Move
D=-8000     ' move 8000 encoder counts in negative direction relative to current shaft position.
G           'go and start move
```

VELOCITY MODE:

Velocity mode is used when a controlled constant velocity is desired. In this mode you have to set the velocity and acceleration. Note a positive velocity value is a clockwise rotation if you are looking at the mounting plate and shaft. Negative is counterclockwise.

Note that in this mode you can make on-the-fly (real-time) velocity changes, the new velocity is initiated when the go command **"G"** is issued.

Example:

```
MV          'set to mode velocity, clockwise rotation.
A=80        ' set acceleration
V=32212*5   ' set velocity
G           'go and start motion

V=32212*20  'change desired velocity
A=200       'change acceleration
G           ‘go and start this new velocity and new acceleration.

V=-50000    'change desired velocity, counter clockwise. This accelerates at the last acceleration value that
             ' was set.
G           ‘go and start this new velocity and current acceleration.
```

TORQUE MODE

In this mode the SmartMotor controls the amount of power to the motor. In this mode only issue values of "T", do not issue a "G" this will cause the motor to exit torque mode . The "T" is a torque factor the range is from 0 to 1023. Where 1023 is maximum power (speed and torque) available from the power supply.

Note that Animatics torque speed curves are rated at 48V DC. To get rated speed and torque use Animatics p/n PS42V6A, 42V unregulated power supply or equivalent. Maximum speed will decrease if using a lower voltage power supply. Note the “AMPS” command will not set the current limit. In torque mode the power to the motor is controlled by “T” value to limit the torque. Position error limit has no effect in torque mode , it will not cause the motor to stop.

```
MT          'set to mode torque
T=50        ' set to about 5% max power available (positive direction)
T=-500      'change to about 50% max power available (negative direction)
T=1023      'set to maximum power available (speed and torque)
T=0         'set to zero speed and torque
```

SECTION 1.1: COMPUTING VELOCITY (V), ACCELERATION (A) and WAIT values

To achieve a better level of control and improved performance over the speed range the SmartMotor internally scales the servo-samples. The controller uses velocity in units of scaled counts per second and acceleration in units of scaled counts per second squared. We have provided equations and sample calculations that convert standard units (Rpm, rps and rev/s^2) to the SmartMotor units in section 1.1. Section 1.1a covers firmware versions 4.10 to 4.15, 4.75,4.40 to4.41. Firmware version 4.00-Q1 to 4.00-Q3 are covered in section 1.1b covers 4.00-Q1 to4.00-Q3. Please refer to label on motor for firmware version or send the command "RSP" to the motor.

The SmartMotor's servo-controller takes samples the encoder position at a sample rate of 4069 times per second (version 4.00 and above) .

SECTION 1.1a: COMPUTING V, A, and WAIT values for FirmwareVersion 4.10 to 4.15,4.75, 4.40 to 4.41

Standard Units: Angular Velocity: Rpm = revolution/minute or Rps = revolution/ second
Angular Acceleration: revolution/ second^2

SmartMotor Units: velocity [**V**] = scaled encoder counts/sec
acceleration [**A**] = scaled encoder counts/sec^2

The Following are the equations to convert from standard units to SmartMotor **V**elocity & **A**cceleration values

`V=<expression>` 'where the expression can be an integer, variable
`A=<expression> '` or one mathematical operation.

Refer to Animatics product selection chart to determine standard encoder resolution.

For a 2000 encoder count per revolution:

1 revolution=131,072,000 scaled counts
V = (# rpm)*(536.87633 scaled encoder counts per sec/rpm)
or V = (# rps)*(32212.578 scaled encoder counts per sec/ rps)
A = (# rev/s^2)*(7.9166433 scaled encoder counts per sec^2/ rev per sec^2)

For a 4000 encoder count per revolution:

1 revolution=262,141,000 scaled counts
V = (# rpm)*(1073.7526 scaled encoder counts per sec/rpm)
or V = (# rps)*(64425.156 scaled encoder counts per sec/ rps)
A = (1 rev / s^2)*(15.833286 scaled encoder counts per sec^2/ rev per sec^2)

Sample Rate = 24576.25 seconds / 100,000,000 servo samples = 1 second / 4068.9695 servo samples .

`WAIT=` #servo samples For example: **`WAIT=4069`** is 1 second

SAMPLE CALCULATION:

If you want: velocity of 1 revolution/second (equal to 60 rpm) and an acceleration of 1 rev/s^2 set "V" and "A".

2000 count encoder: (SM17xx, SM23xx)
Your SmartMotor (**`V= #`**) and (**`A=#`**) are:

V=(60 rpm)*536.87633=32212.58 **`V=60*537`**
Or V=(1 rev/sec)*32212.578=32212.58 **`V=32213`**
A=(1 rev/s^2)*7.9166433=7.92 **`A=8`**

SAMPLE CALCULATION:
If you want: velocity of 1 revolution/second (equal to 60 rpm) and an acceleration of 1 rev/s^2 set "V" and "A".

4000 count encoder (SM34xx, SM42xx, SM56xx)
Your SmartMotor V= <expression> and A=< expression > are:

	V=(60 rpm)*1073.7526=644425.16	**V=64425**
Or	V=(1 rev/sec)* 64425.156 =64425.16	**V=64425**
	A=(1 rev/s^2)* 15.833286=15.83	**A=16**

SECTION 1.1b: COMPUTING V, A, and WAIT values for FirmwareVersion 4.00-Q1 to 4.00-Q3

Standard Units: Angular Velocity: Rpm = revolution/minute
or Rps = revolution/ second
Angular Acceleration: revolution/ second^2
SmartMotor Units: Velocity [**V**] = scaled encoder counts/sec
Acceleration [**A**] = scaled encoder counts/sec^2

For SM17xx series SmartMotor™

Sample Rate = 24576.25 seconds / 100,000,000 servo samples = 1 second / 4068.9695 servo samples .

WAIT= #servo samples
For example: **WAIT=4069** is 1 second

The following are equations to convert from standard units to SmartMotor **V**elocity & **A**cceleration values.
Velocity and Acceleration values are defined as: **V=expression**
A=expression

V = (# rpm)*(536.87633 scaled encoder counts per sec/rpm)
or V = (# rps)*(32212.578 scaled encoder counts per sec/ rps)
A = (# rev/s^2)*(7.9166433 scaled encoder counts per sec^2/ rev per sec^2)

SAMPLE CALCULATION:
If you want: velocity of 1 revolution/second (equal to 60 rpm) and an acceleration of 1 rev/s^2 .

2000 count encoder:
1 rev=131,072,000 scaled encoder counts
Your SmartMotor (**V=** and (**A=#**) are:

	V=(60 rpm)*536.87633=32212.58	**V=32213**
or	V=(1 rev/sec)*32212.578=32212.58	**V=32213**
	A=(1 rev/s^2)*7.9166433=7.92	**A=8**

For SM23xx and SM34xx series SmartMotor™

Sample Rate = 24824.24 seconds / 100,000,000 servo samples . 1 second / 4028.3207 servo samples .

WAIT= #servo samples

For example: **WAIT=4069** is 1 second

2000 count encoder:

1 rev = 131,072,000 scaled encoder counts

V = (# rpm)*(542.29383 scaled encoder counts per sec/rpm)

or V = (# rps)*(32537.628 scaled encoder counts per sec/ rps)

A = (# rev/s^2)*(8.07722 scaled encoder counts per sec^2/ rev per sec^2)

4000 count encoder:

1 rev = 262,144,000 scaled encoder counts

V = (# rpm)*(1084.587596 scaled encoder counts per sec/rpm)

or V = (# rps)*(65075.226 scaled encoder counts per sec/ rps)

A = (# rev/s^2)*(16.15444 scaled encoder counts per sec^2/ rev per sec^2)

SAMPLE CALCULATION : SM23xx and SM34xx series version 4.00-Qx:

If you want: velocity of 1 revolution/second (equal to 60 rpm) and an acceleration of 1 rev/s^2 .

2000 count encoder:

Your SmartMotor **V** and **A** values are:

V=(60 rpm)* 542.29383 =32537.63 **V=32538**

or V=(1 rev/sec)* 32537.628 =32537.63 **V=32538**

A=(1 rev/s^2)* 8.07722 =8.08 **A=8**

4000 count encoder:

Your SmartMotor V= <expression> and A=< expression > are:

V=(60 rpm)* 1084.587596=65075.23 **V=65075**

or V=(1 rev/sec)* 65075.226 =65075.23 **V=65075**

A=(1 rev/s^2)* 15.833286=15.83 **A=16**

SECTION 1.2: INTEGRAL BRAKE COMMANDS

For motors equipped with Animatics integral brakes, a series of new commands and dedicated internal I/O facilitate the use of the integral brake on version 4.12 and newer.

BRKENG	Engage the brake.
BRKRLS	Release the brake.
BRKSRV	Engage the brake whenever the motor is not servo-ing.
BRKTRJ	Brake on and servo off, if no trajectory

SECTION 1.3: EXTERNAL ENCODER COMMANDS

Please refer to the Animatics Product Selection Chart or Pinout drawing to see if your motor has the ability to accept external encoder inputs. To use a differential external encoder input you have to setup the hardware by wiring encoder A signal to the ENCA pin (port A) and encoder B signal to ENCB pin (port B). To allow the controller to use the external encoder as the primary encoder, in place of the internal encoder , issue the "ENC1" command from the interface software or in the motor's program memory.

Encoder commands:

ENC0 Encoder zero: (Default) Use internal encoder as primary encoder

ENC1 Encoder one: Use external encoder as primary encoder and the shaft position is recorded in the counter (CTR)

SECTION 1.4: Vertical Axis Applications , Gravity Constant (KG, KGON and KGOFF commands)

COMMAND DESCRIPTION

KGON This command reduces torque ripple and smoothes motion against an externally applied constant force on the axis. An example of such a force is gravity acting on a lead screw driven vertical axis slide. At low speeds, the peak torque increases, but the continuous torque decreases. The stability also changes so the PID filter coefficients should be modified.

KG=<value> A positive value of KG is in the positive direction (clockwise looking at the shaft) and applies a constant amount of power to the motor to compensate for the constant applied force.

KGOFF Turns off the KGON function.

To use the KGON issue following commands in this sequence:

```
KG=<value>   'sets amount of power to motor to compensate for constant
             'applied force.
F            'Update the PID (F)ilter.
KGON         'Turn on the KG function.
```

SECTION 1.5: Directional and Non-directional Limit inputs (Left/Right Limit Switches)

Port D and C is set to Left Limit and Right Limit Input functions by default. The limits switches can be set as directional or non-directional on firmware version 4.15, 4.75, 4.40 and 4.41.

On version 4.15,4.40 and 4.41 the default is non-directional limit "LIMN" and limit active low "LIML" and can be changed by issuing LIMD and LIMH.

On version 4.75 the default is directional limit "LIMD" and limit active high "LIMH".

Directional Limits Version:

Firmware version 4.15, 4.75, 4.40 and 4.41, have the directional limits capability. To use directional limits obtain SmartMotor Interface (SMI) Software version 1.221 or newer (available from Animatic's tech support).

Non-Directional Limits Version:

Firmware version (4.13 and lower, 4.00-Qx, 3.00) have non-directional limits and can not use the LIMH,LIML,LIMD and LIMN commands. For example, if the left limit input (port D) is triggered the motor will stop. However if you issue a move and "G" going left (Clockwise, looking at the shaft) the motor will go left.

Directional commands:(firmware version 4.15, 4.75, 4.40 and 4.41)

LIMN LIMits are Non-directional, as they have been before. This is the default.

LIMD LIMits are Directional.

A new occurrence of either limit halts the motor. A move begun on a limit is only allowed to move in the opposite direction of the limit.

If the left limit (counter clockwise) is triggered, the motor is not allowed to go past that left limit point.

If the right limit (clockwise) is triggered, the motor is not allowed to go past that right limit point.

Limit Trigger Low/High Commands: (versions 4.15, 4.40 and newer)

LIML LIMit active Low triggers when signal goes from low input. This is the default.

LIMH LIMit active high, sets limit input to trigger on high input

SECTION 1.6: MOTOR and LOAD PROTECTION FEATURES

The SmartMotorTM servo motor is equipped with several protection features and diagnostic tools that allow the user to protect and perform diagnostic functions on the load. These features are: power limit, temperature , error and power monitoring functions.

SECTION 1.6a: Error Limit

The allowable position error limit is software settable by issuing **E=<value>**. The range of the values is 0 to 32000 (default is **E=1000**). When the position error reaches the error limit "**E**" the motor stops execution of the program, stops the shaft and turns off the amplifier and the error limit status bit "**Be**" goes to 1. To reset the error limit bit issue a go **"G"**.

This affects the following commands: **E=**, **RE**, **=E**, **RPE**, and **=@PE**.
This error limit range is useful in compensating for large position errors while using the VRE (Variable Resolution Encoder).

If error limit is achieved in applications where motor is run at a constant high speed, increase/decrease the PID filter velocity constant "**KV**". The constant compensates for the non-changing error value associated in high speed operations. Issue this sequence of commands.

KV=<value> 'set constant velocity error PID filter constant.

F 'update PID Filter.

SECTION 1.6b: Peak Power Limit

The internal peak power limit of the SmartMotor™ servomotor is set by the AMPS command. This function controls amount of power that is delivered to the motor. Note, the AMPS command limits output torque and maximum speed, as well.
The valid range of the "AMPS" command is a scale factor with range 0 to 1023. The default setting is 1000.

For example, a setting of **`AMPS=512`** will limit the output stall torque to ½ of the peak torque rating and limit the maximum velocity to ½ of the specification. To get full torque and speed set the value, AMPS=1023.

`AMPS` can be assigned to a variable. For example, **`i=AMPS`** will store the value of AMPS in the variable i.

SECTION 1.6c: Temperature and RMS (Continuous) Power Limits

The RMS, continuous, power consumption is constantly monitored by the SmartMotorTM servo motor. If the RMS power exceeds continuous output rating of the SmartMotorTM servo motor for a programmable amount of time, the amplifier will shut down and indicate an overheat error (see status bit Bh). This programmable time is set by the THD (Temperature Hot, time Delay) function.

The valid range for THD is 0 through 65535 (version 4.40 THD range: 0-19967), with units in servo samples. For example, `THD=4069` will set the thermal shut down delay to one second. This means that the RMS input power must exceed the specification for 1 second before the amplifier will shut down.
The default value: THD=12000 (approximately three seconds). THD can not be assigned to a variable.

In addition, the SmartMotorTM servo motor also monitors its internal temperature. If the internal temperature exceeds a programmable set point, the amplifier shuts down and indicates an overheat error (see status bit Bh). The SmartMotorTM servo motor will remain in an overheat condition until the internal temperature drops 5° C below the programmable set point. This set point is determined by the function TH. The valid range for **`TH`** is 0 to 70, with units in degrees Celsius.

For example, if you enter, TH=50 the amplifier will indicate an overheat if the internal temperature reaches 50°C and will come out of the overheat condition when the temperature drops below 45°C.
The default value for **`TH=70`**.
THD can be stored into a variable. For example, t=TH will store the value of TH to the variable t.

SECTION 1.6d: Temperature , Voltage and Current Monitoring

TEMPERATURE MONITORING: The Temperature and RMS power of the SmartMotor™ servo motor can be monitored for diagnostic, preventative maintenance reasons. The real time temperature is read by the TEMP function and is given in units of degrees Celsius. TEMP is read by storing it to any variable. For example,

`t=TEMP`	this command ill assign the internal temperature to the variable "t".
`Rt`	this command will report variable "t". (Note: Any variable name can used .)

VOLTAGE MONITORING: The bus voltage is monitored by the user "J" analog input using the UJA function. User J pin is not physically accessible by the user. UJA provides the input bus voltage in tenths of volts. The accuracy of the reading is +0.1VDC.
For example, **`v=UJA`** will assign the input voltage to the variable "v".

If the reading is 336, the input voltage is 33.6+0.1 VDC.

CURRENT (I_{RMS}) MONITORING: The RMS current is monitored by the user "I" analog input port using the **`UIA`** command. User defined port I pin is not physically accessible by the user.
UIA will provide the measured RMS current in hundredths of ampere.
The accuracy of the reading is 0.1A.

For example, **`i=UIA`** will assign the RMS current to the variable i.
If the reading is 234, the measure current is 2.34+0.1 Amps.

SECTION 1.7: Servo-Amplifier OFF Command
On version 4.00 and newer motors you end the motor from servo-ing place by shutting off the servo-amplifer by issuing "OFF". This will turn off the servo-amplifer and the shaft will be able to rotate freely .

`OFF` Turn motor servo off.

SECTION 1.8: External Program Memory Chip
SmartMotors are available with internal memory on the "D" type I/O connector. On the SQUARE motors (Molex type connectors) the memory chip is external.

External program memory modules (EEPROM) are now available in 32k byte and 8k byte for square type motors.
D type motors (with 15 pin I/O connector) come standard with internal program memory.
Please refer to Figures 1.81 and 1.82 for Square and D-type motors.

Typical "SQUARE" Type Motor:

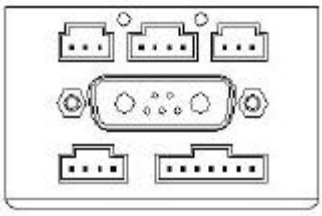


Figure 1.81: SQUARE Type Motor (Molex™ connectors and 7pin-combination connector)

Typical D-Type Motor:

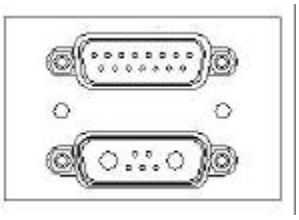


Figure 1.82: D-Sub Type Motor (15pin D-sub connector and 7pin-combination connector)

SECTION 2.0: ADDITIONAL MOTION MODES

SECTION 2.1a: Mode Follow with ratio, "electronic gearing"

SmartMotor™ users wanted variable mode follow ratios. To address that need with a fixed point system, we implemented a numerator/denominator function. This uses an 'integer fraction' to provide the equivalent of a floating point relationship between the motor and an incoming encoder, or step and direction, signal. For example, a mode follow relationship of 4.125 can be produced by setting MFDIV=8 and MFMUL=41.

Mode Follow with Ratio Commands

MFMUL=<expression>	Set numerator (signed 16 bit integer)
MFDIV=<expression>	Set denominator (signed 16 bit integer)
MFR	Engage Mode Follow Ratio

SECTION 2.1b: Mode Follow Phase Offset

A phase offset may be introduced when using the mode following commands. A phase offset allows the motor shaft to be aligned relative to the signal being followed. The offset is initiated by the G command, if the D command variable holds a (non-zero) offset. The V command variable controls the rate at which the offset is applied. The offset value in D decreases in magnitude until it reaches 0. V remains unchanged. The A command variable is used for internal calculations, so any prior value will be lost. If no offset is desired, be sure D=0 when issuing the G command while in MFR or MSR mode. While the offset is being applied, changes to D,V, or A will have immediate effect.

Mode Follow with Phase Offset Sequence:

MF4	Set counter mode (resets external counter to zero)
MFMUL=1	Set ratio multiplier
MFDIV=1	Set ratio divisor
MFR	Set Follow-Ratio Mode, initiating ratio calculation
D=0	Ensure there is no unintended phase offset
G	Start (applies new ratio to all subsequent external encoder changes)

Let's say we observe the motor shaft off ¼ revolution relative to the external encoder, both turning about 1 rps.

```
D=500     'Set positive ¼ revolution relative offset for motors with a 500 line encoder
V=410     'Want the relative offset to complete in 20 seconds.
          '(500 counts/20 seconds * 65536 scaled counts/count * 1 second/4000 servo samples =
          '410 scaled counts/servo sample).
G         'Begin phase offset
V=0       'Pause phase offset
RD        'Shaft move 299 counts relative to gear, 201 counts to go
V=410     'Resume phase offset
RD        'Reports Phase offset complete
```

SECTION 2.2: Mode Step and Direction and Mode Step with Ratio

MSR	Engage Mode Step Ratio
MFMUL=<expression>	Set numerator (signed 16 bit integer)
MFDIV=<expression>	Set denominator (signed 16 bit integer)
O=0	Set current shaft position as origin. (Uppercase letter O equal to zero)

SECTION 2.3: Mode Cam & Mode Cam with Ratio

For reciprocating motion such as the output from a variable profile CAM, there is a new mode initiated by the command "MC" (Mode Cam). This function allows a complex relationship between an incoming encoder signal and the motor's own encoder. It is described by as many as 100 sixteen bit words loaded in the array (aw[value]), and will linearly interpolate velocity & acceleration values between those points.
The BASE represents the number of external encoder counts to define one cam cycle or cam revolution.

BASE = <expression> *where: 2 <= BASE <= 32768*

The number of points in the cam profile is determined by the table entries given by:

SIZE =<expression> *where: 2<= SIZE <= 100*

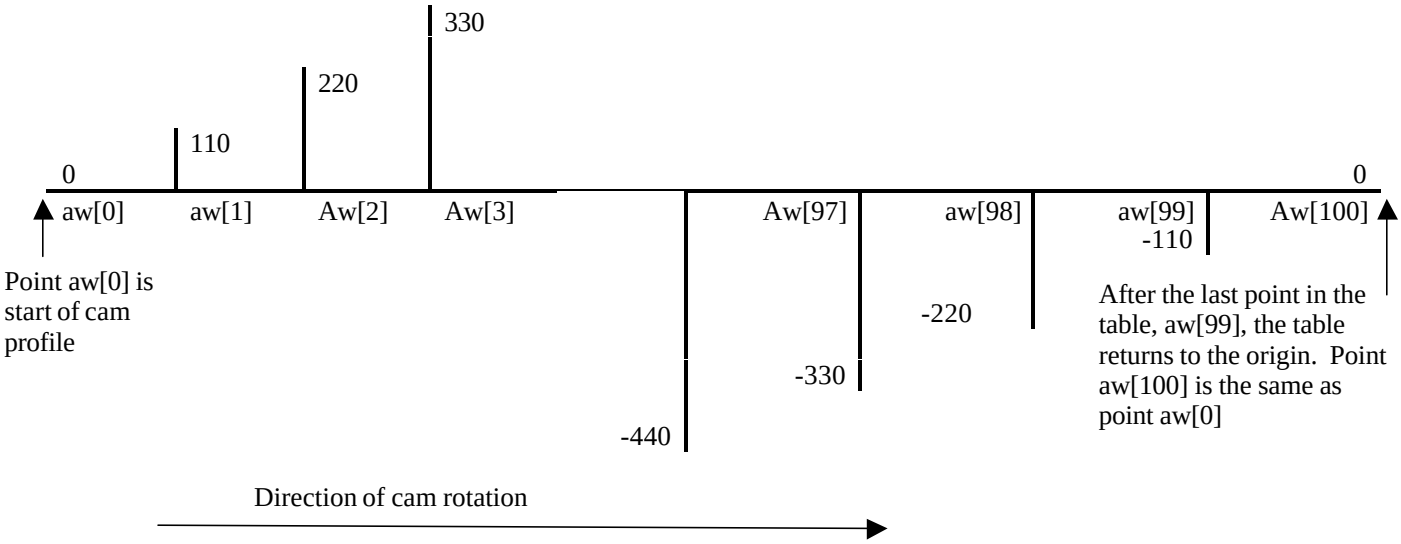
SIZE <= BASE

16 bit data values can be stored (values between – x to +x) to Cam Table.
They are to be stored in aw[0]…to aw[SIZE-1]. The data entries must represent a sample point at evenly spaced intervals of the required offset

Example Code:

```
MF4             'Reset external encoder mode and zero counter
BASE=4000       'Set the number of external encoder counts per cycle ( 4000 counts)
SIZE=100        'Tell how many array points, or table entries, to use
E=10000         'It may be necessary to increase error band to ensure the motor, when rotating at a high
                'speed, does not assumme it is seeing a position error as it tries to move to the next point
                'in the cam table.

                'CAUTION: ensure that the error band is not set so high that it allows the motor to
                'perform a violent move that damages machinery or personnel.

aw[0] 0 110 220 330 440 … -440 -330 -220 -110.   Load the array

                'The maximum line length is 128 characters. If more characters are required, use a
                'carriage return. At the last point in the table, aw[99] the 'cam' has almost completed
                'one revolution. At position aw[100] the cam has completed one revolution returns to the
                'first point in the table, the origin aw[0].. Note the array must end with a period.

MC              'Set Mode Cam, performs buffered calculation
G               'Start, resets motor position to zero – points at beginning of table and
                'engages CAM follow mode
```

The cam profile of this example is shown in the following sketch:



In this example, 10000 counts from the external encoder drives one 'rotation' of the cam. To continue cycling through the cam table, external encoder readings are translated to an offset within the range 0-10000. An external encoder measurement of 10350 translates to 10350 – 10000 = 350

Since the BASE = 10000 and there are 100 table entries, each table entry represents a segment of magnitude 10000/100 or 100 counts. The table entries, aw[ ], correspond to external encoder readings at 100 count intervals:

aw[0] 0, aw[1] 100, aw[2] 200, aw[3] 300, aw[4] 400,……..
……aw[96] 9600, aw[97] 9700, aw[98] 9800, aw[99] 9900

In our example, if the external encoder reading translates to 350, the sample point lies between aw[3] and aw[4]. The actual requested shaft position is calculated using linear interpolation.

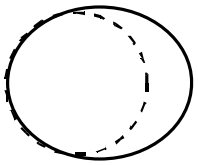
From the cam table:
aw[3] = 330
aw[4] = 440
If external encoder = 350,
Required position = sample position aw[3] + a fractional part of (aw[4] –aw[3])
Required position = 330 + ((350-300)/100 * (440 – 330) = 385
= 330 + 50/100 * 110
= 330 + 55 = 110

Note: The SmartMotor sets its position to ZERO (P=0) where the mode is engaged. The table offsets are relative to ZERO.
The external counter may reverse at any time.
Both positive and negative external encoder readings map to the data table.
The normal PID update rate (4 kHz) is maintained.
The required position calculation is rounded to the nearest encoder count.

EXTREME CAM PROFILES

At the end of each cam revolution, the table returns to the origin. Remember that abrupt changes in cam profile, or increases/decreases in cam diameter, will require high acceleration from the motor to ensure the shaft moves to the required angular position during one interval. Extreme changes could cause error problems, especially if the system is rotating at high speed.

Cam profile with smooth transitions



Cam profile with abrupt transitions

CAM MODE COMMANDS

Mode cam can multiply encoer inputs by issuing the follwowing commands.

MC2	Mode cam with final position request multiplied by 2
MC4	Mode cam with final position request multiplied by 4
MC8	Mode cam with final position request mutilplied by 8

SECTION 2.4: Coordinated Motion (Multi-axis Contouring Applications)

Coordinated Motion is a new feature of SmartMotors that have firmware versions 4.15 ,4.75, 4.40,4.41 and later. It allows SmartMotors on a daisy chain to be synchronized and precisely controlled by feeding the SmartMotors coordinate and time data. The Coordinated Motion command prompts for an ASCII formatted coordinate file (save as ***.txt*** file). The file must only have integer values separated by tabs:

	motor 1:	*motor 2:*	*motor 3:*
line 1:	**position clock**	**position clock**	**position clock**
line 1:	**position clock**	**position clock**	**position clock**

Sample file:

```
0     0     0     0
100   128   -100  128
200   256   -300  256
400   384   -100  384
800   512   0     512
```

The difference between position values must not exceed 8,388,607, otherwise an invalid position delta error will occur. The difference between clock values must not exceed 32,768 otherwise an invalid time delta error will occur. The clock values must also differ by values that are powers of 2 (for example 1, 2, 4, 8, 16, 32, 64, 128, .. , 32768). The Coordinated Motion command automatically addresses the SmartMotors on a daisy chain according to their position on the daisy chain.

If the first clock values are zeros, SMI assumes the position coordinates are relative to the current trajectory position (not the immediate position as reported by RP). Otherwise, the coordinates are absolute, and movement is done from the current trajectory position (not RP) and the first clock value is the "homing" time.
To return the trajectory position, issue the commands: aa=P, RP.

Note that the immediate position (returned by RP) and the trajectory position (variable P) must be the same, otherwise a position error may occur if the position differs by more than the error band E.

When the file is being sent and Coordinated Motion is in progress a dialog box pops up.
You can press the escape key or click anywhere in the dialog box to cancel Coordinated Motion.
You should not perform any CPU-intensive operations while running Coordinated Motion.
If SMI is not able to send data fast enough, possibly because another program interrupts it, a buffer underflow error will occur. If you cancel the motion or the file runs out of coordinates, the motors are sent an S command to halt motion and servo in place. When the trajectory clock values end for a motor, an S is sent to have the SmartMotor servo in place.

For more advanced functionality of Coordinated Motion, like dynamically generated coordinates, etc., the SMIEngine toolkit has been developed. SMIEngine is a component object that implements various primitives for communicating with SmartMotors and gives full control of Coordinated Motion.

SMI is supplied with an executable CMotion.exe that is used to control Coordinated Motion. The command-line format for CMotion.exe is: "CMotion portname filename" (ex. "CMotion COM1 c:\test.txt")
(Do not include quotation marks in the arguments.) This is the same executable that is run from the SMI Tools menu.

SECTION 3: INPUT/OUTPUT FUNCTIONS

In addition to SmartMotors special functions (right limit, left limit, step & direction ...etc) ports A to G are equipped to process Inputs/outputs.

SECTION 3.1: Software Selectable Functions

Each of the I/O pins have many devices attached internally that perform different I/O functions at each pin. I/O pins A to G can receive analog input (0-5V, 10 bit analog-to-digital converter), digital input (5V TTL) and send digital output (5V TTL). These functions are software selectable. Each I/O pin can change the selected function at anytime. To use a particular I/O function issue I/O commands below. Please refer to section 3.2 for sample code for I/O initialization.

INPUT /OUTPUT COMMANDS

New flexibility and functionality has been added to all I/O:
List of I/O commands

UA= expression	(set pin OUT LATCH to lsb, 0 or 1)
UAI	Assign pin A to input state
UAO	Assign pin A to output state
variable**=UAA**	Read pin a as 10 bit analog input
UB=expression	(set pin B output latch state)
UBI	Assign pin B to input state
UBO	Assign pin B to output state
variable**=UBA**	Read pin B as 10 bit analog input
UC= expression	(set pin C output latch state)
UCI	Re-assign right limit to input state
UCO	Re-assign right limit to output state
variable**=UCA**	Read pin C as 10 bit analog input
UCP	Restore pin to right (plus) limit function
UD=expression	(set pin D output latch state)
UDI	Re-assign left limit to input state
UD0	Re-assign left limit to output state
variable**=UDA**	Read pin D as 10 bit analog input
UDM	Restore pin D to left limit (minus limit) function
UE= expression	(set pin E output latch state)
UEI	Assign Anilink Data pin to input state
UEO	Assign Anilink Data pin to output state
variable**=UEA**	Read pin E as 10 bit analog input
UF= expression	(set pin F output latch state)
UFI	Assign Anilink Clock pin to input state
UFO	Assign Anilink Clock pin to output state
variable**=UFA**	Read pin F as 10 bit analog input

UG	Restore G sync line functionality
UG=expression	(set pin G output latch state)
UGI	Assign pin G to input state
UGO	Assign pin G to output state
variable**=UGA**	Read pin G as 10 bit analog input

SECTION 3.2: SAMPLE I/O PIN INITIALIZATION AND READ/WRITE SEQUENCE

Examples show: initialize Port B as an Analog Input, initialize Port C as an Analog Input and initialize Port D as an Digital Output

EXAMPLE 1) Digital Input: initializing Port A as an Digital Input .

Note: The SmartMotor Input ports have an internal 5kΩ pullup resistor, default state of input port is high and triggered when input line goes low. The exception is when **LIMH** limit High command issued triggers Left/Right Limit on high signal (version 4.15 & 4.40) .

```
UAI          'Initializes (U)ser defined port (A) as an (I)nput
b=UAI        ' Store digital input (at this instant in time) into variable "b"
Rb           ' Reports ( prints on screen) the value stored in the variable "b"
dd=UAI+5     ' Math operation is allowable on .
```

EXAMPLE 2) Digital Output: initializing Port B as an Digital Output .

```
UBO          'Initializes (U)ser defined port (B) as digital (O)utput
UB=1         'Sends logical high output to I/O pin B
             'Latches that logical state until new state is outputed.
UB=0         'Sends logical low output to I/O pin B.
```

EXAMPLE 3) Analog Input: initializing Port B as an Analog Input .

```
             'Initializes (U)ser defined port (E) as an (I)nput
c=UEA        ' stores analog input value at this instant in time
             ' into variable "c" (value of A/D Converter is 0 to 1023 ).
             'Sets (U)ser defined port (E) as (A)nalog input (10 bit A/D Converter).
             ' you can store the analog input into any of the variable names.
Rc           ' Reports (prints on screen) the value stored in the variable "c"
ee=UEA*100   'Math operation on the analog input is allowable
```

SECTION 3.3: Motor I/O Connector types:

SmartMotors have two types of I/O connectors:

D-SUB type (15pin D-sub connector and 7pin-combination connector)
SQUARE type (Molex™ connectors and 7pin-combination connector)

Refer to Figure 3.31 for typical D-sub Motor and Figure 3.31 for typical D-sub Motor.

Typical D-Type Motor:

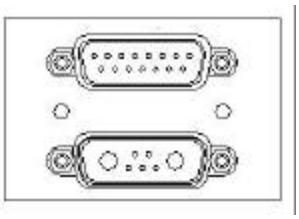


Figure 3.31: D-Sub Type Motor (15pin D-sub connector and 7pin-combination connector)

Typical "SQ" Type Motor:

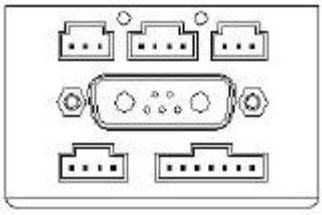
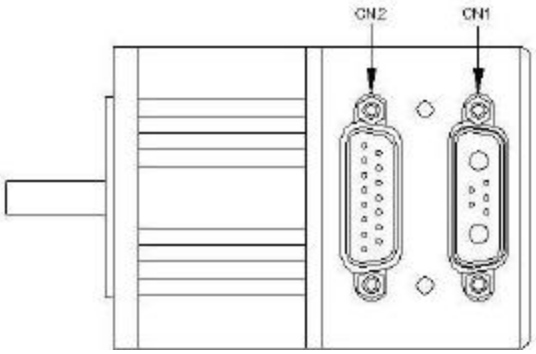


Figure 3.32: SQ Type Motor (Molex™ connectors and 7pin-combination connector)

SECTION 3.4: I/O Connector Pinouts

This section provides pinouts for the D-sub connector and Molex I/O connector. Connector 1 and 2 pinout is listed below.

D-sub Connector Pinouts



CN1: 7 PIN COMBO D-SUB MALE CONNECTOR		
PIN#	NAME	FUNCTIONS
1	Pin G	Go-Synchronization Pin
2	+5V OUT	+5V Out
3	RS232 TX	RS232-Transmit
4	RS232 RX	RS232-Receive
5	GND	Signal Ground
A1	POWER	Power,+24v to 48v
A2	PWR GND	Power Ground

CN2: 15 PIN D-SUB FEMALE CONNECTOR					
		SOFTWARE SELECTABLE I/O PIN FUNCTIONS			
PIN#	NAME			DIGITAL I/O (TTL)	ANALOG INPUT (10 bit)
1	Port A	Step Input	External Encoder A Input*	5V	0 to 5V
2	Port B	Direction Input	External Encoder B Input*	5V	0 to 5V
3	Port C	Right Limit Input		5V	0 to 5V
4	Port D	Left Limit Input		5V	0 to 5V
5	Port E	Anilink Data	RS-485 Signal A	5V	0 to 5V
6	Port F	Anilink Clock	RS-485 Signal B	5V	0 to 5V
7	Port G	Go-Synchronization Pin	RS-485 Adapter Control Line	5V	0 to 5V
8	ENCA OUT	Encoder A Output			
9	ENCB OUT	Encoder B Output			
10	RS232 TX	RS232-Transmit			
11	RS232 RX	RS232-Receive			
12	+5V OUT	+5V Out			
13	GND	Signal Ground			
14	PWR GND	Power Ground			
15	POWER	Power,+24v to 48v			

Figure 3.41: D-Sub Type Motor Pinout

(* SM2315D does not have external encoder capabilities)

Molex connectors SM17xx and 23xx (SmartMotor SQ)

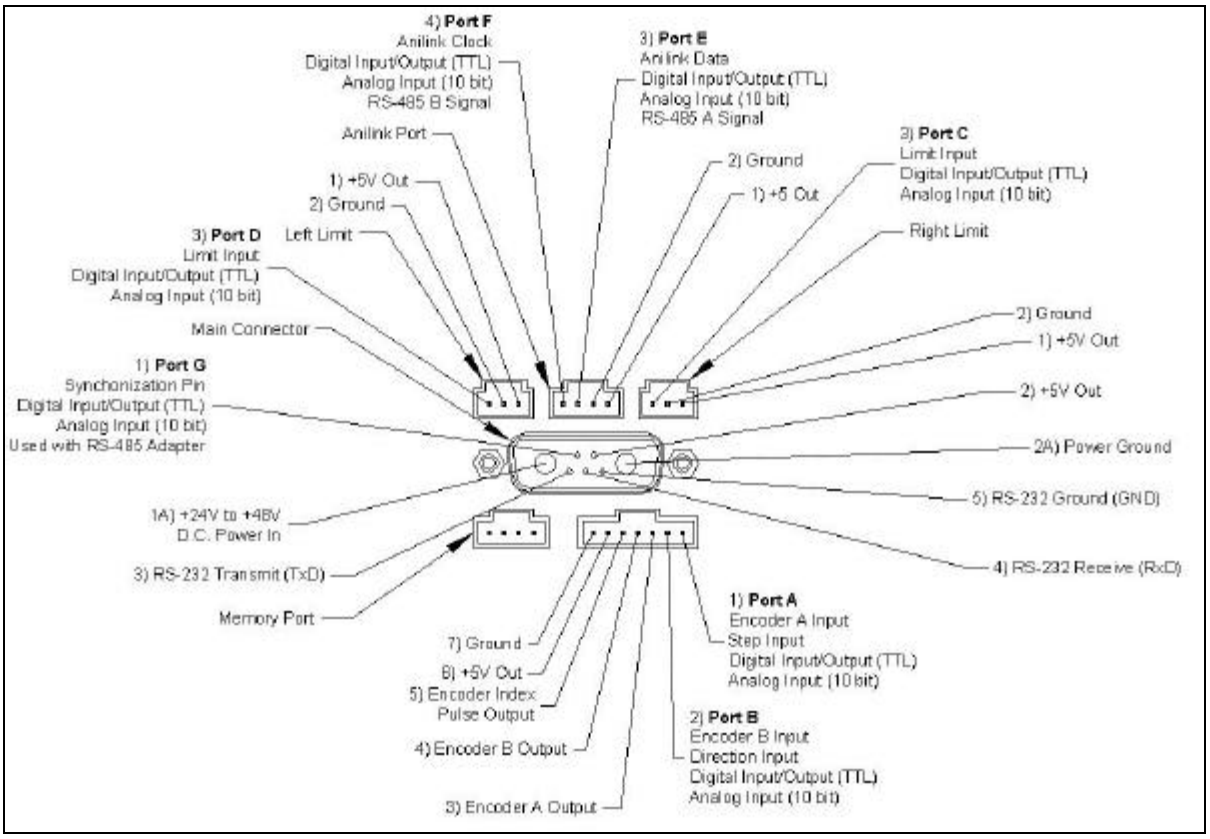


Figure 3.42: SQ Type Motor Pinout (SM17xx and SM23xx)

Molex connectors SM34xx (SmartMotor SQ)

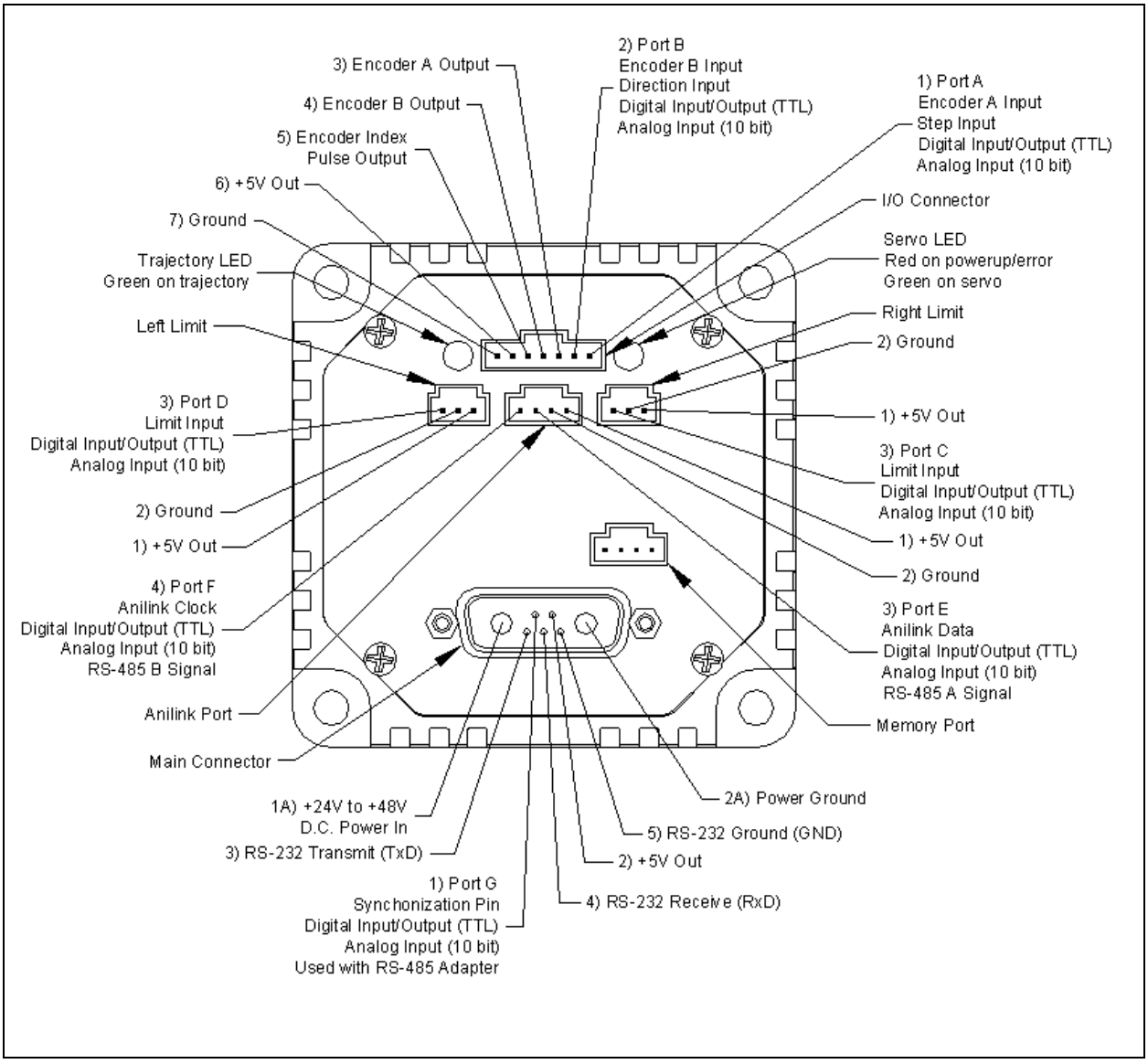


Figure 3.43: SQ Type Motor Pinout (SM34xx)

SECTION 3.5: D-sub motor Wiring diagram for Anilink peripherals

The D-sub type use port E and F as the Anilink port (Anilink clock line and Anilink data line). Figure 3.41 shows how to make a cable connecting D-sub motor to Animatics Anilink peripheral.

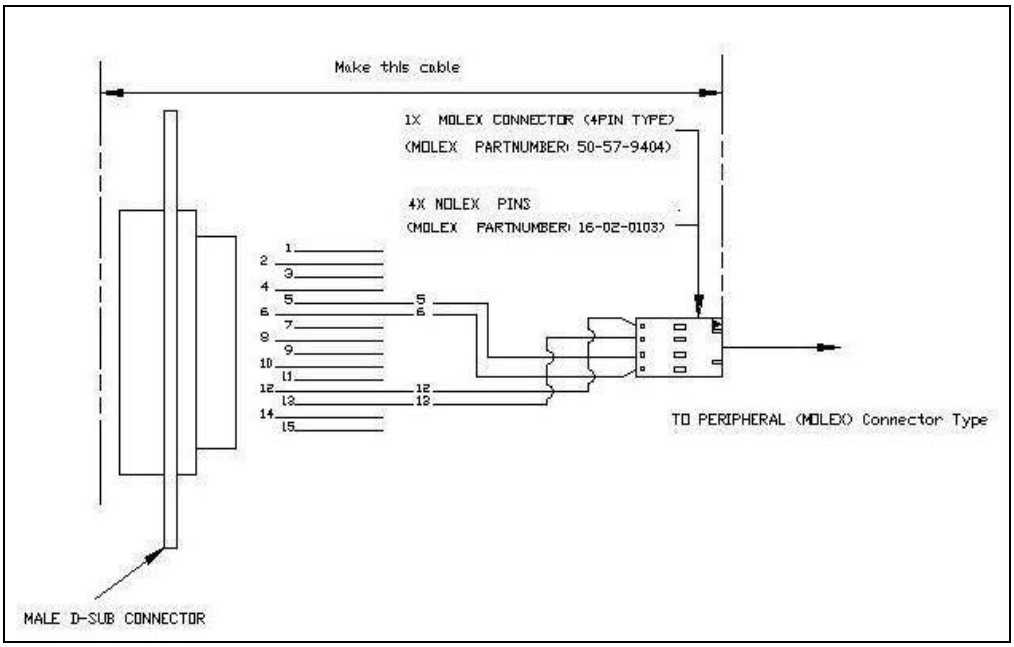


Figure 3.51: Dsub Cable To ANILINK 4 Pin Molex Connector

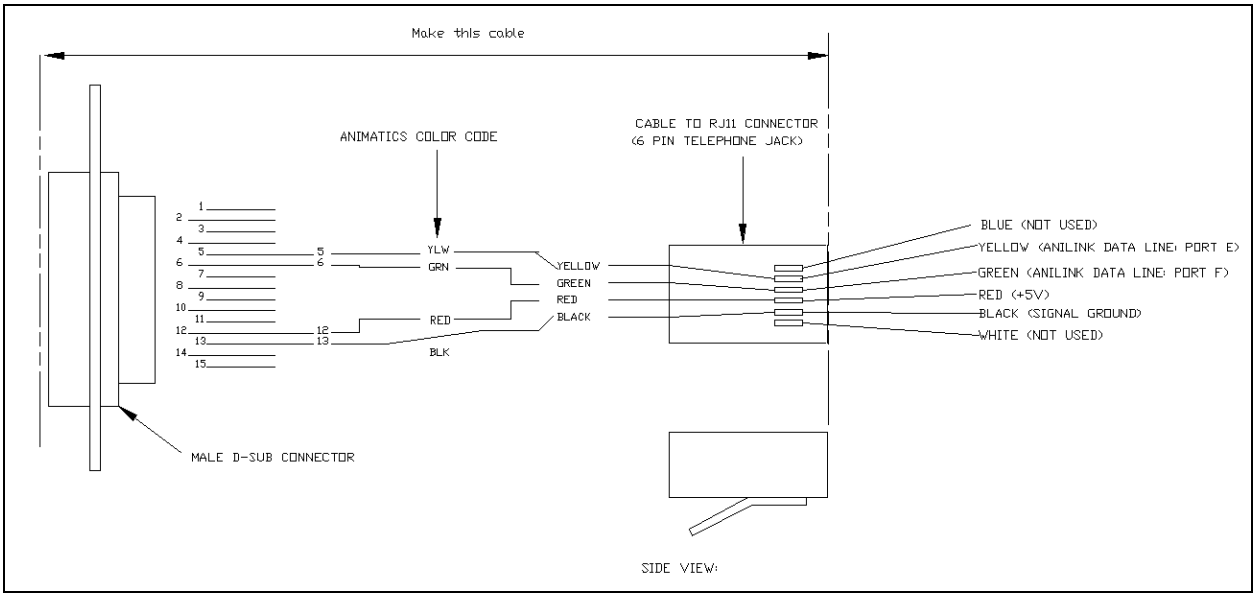


Figure 3.52: Dsub Cable To ANILINK RJ11 Connector

SECTION 4.0: PROGRAMMING LANGUAGE

SECTION 4.1a: Variable and Arrays Names

Variable Names

The number of variables have been expanded dramatically. The user now has the following 77 variable names at their disposal:

`a to z`	32 bit (first set of 26 variables)	(32 signed bits)
`aa,bb, cc ,dd…..to …zz`	32 bit (second set of 26 variables)	(32 signed bits)
`aaa, bbb, ccc to …yyy`	32 bit (third set of 25 variables)	(32 signed bits)

Note : Variable names with non-duplicated letters such as : ab, bc,cd,….abc, cde,efg,…etc… are not available.
These variable are 32 bit signed that means the can store values from - 2,147,483,647 to 2,147,483,647.

Arrays Names

As an alternative to the above two and three letter variables, sharing the same data space, the following arrays could be used. The same space can be used for three different variable types: 8 bit; 16 bit; & 32 bit. Naturally, the number of variables yielded from the space varies in accordance with the variable data lengths.

`ab[i]` Can store value of -128 to 127 (8 signed bits)
where i = 0...199
overlays aa-zz & aaa-zzz

`aw[i]` Can store values -32,768 to 32,767 (16 signed bits)
where i = 0...99
overlays aa-zz & aaa-zzz

`al[i]` Can store values from - 2,147,483,648 to 2,147,483,647 (32 signed bits)
where i = 0...49
overlays aa-zz & aaa-zzz

Where "i" can be a variable name a to z or a constant.

Reporting Variables

The array variables can be read from a host just as the traditional variables are, by prefixing the variable name with a capital "R". This is also known as the Report command which prints the value onto the communication line.

`R`*_variable_name*

For example: **`Rab[4]`** Reports Array Byte (**`ab[...]`**) 4th array location
`Rd` Report value stored in variable "d"

For flexibility, a long (signed 32 bit number) can be written and four bytes read from the same space, or visa-versa.

SECTION 4.1b: Initializing Variables and Arrays

Because of the vastly greater number of variables and additional variable types, we have implemented a convenient way of initializing a series of variables on a single line. It is shown in the following example. Note: the period after the last line entry the end of variable initialization on that line:

Example

```
a 1 -2 13 24 35.
```

Performs:

```
a=1
b=-2
c=13
d=24
e=35
```

```
aw[12] 15 20 30.
```

Performs:

```
aw[12]=15
aw[13]=20
aw[14]=30
```

SECTION 4.2: Long Term Variable Data Storage (VST,VLD Commands)

In addition to the program memory there is an additional internal 8k EEPROM chip for the long term storage of data (not standard on some models, please refer to product selection chart). It is accessed by first locating a pointer into that memory space and then doing single or bulk stores and loads. The associated commands are:

EPTR	Locates the EEPROM pointer, range (0……..7999)
VST	Stores data at that location and forward
VLD	Loads (retrieves) data from that location and forward

Example: Storing values to data memory (VST)

EPTR=1000	Set the pointer to 1000 to a memory location
VST(aw[15],5)	This command stores five words from the array starting with aw[15]. Each word is two bytes, the five words will take up10 bytes of space. Memory required for the variable type is automatically allocated.
p=15	In this example we initialize variable p and q
q=5	
EPTR=2000	Set the pointer arbitrarily to 2000
VST(ab[p],q)	You can use variables in place of constants

Example: Reading values from the data memory (VLD)

EPTR=1300	Set the pointer to read the required memory address in the EEPROM
VLD(t,6)	This command reads six variables from the EEPROM and writes to t and then u,v,w,x, and y.
VLD(aw[10],3)	Read three words from EEPROM and store to aw[10], aw[11], aw[12] in RAM.

SECTION 4.3: Control Flow

The following commands can be used with Version 4.00–Q3 and below series SmartMotors™
The number of subroutine labels have been expanded. Following are the new limitations:

GOSUBn	Where n = 0...999	(This results in 1000 possible subroutine labels)
GOTOn	Where n = 0...999	
Cn	Where n = 0...999	(Subroutine labels)
STACK	Reset user program stack pointer (Repairs stack damage from GOTOing out of a nested subroutine)	

The "IF" statement now has optional "ELSE" and "ELSEIF.

Example 1): Gosub Statements

```
IF a==1
      GOSUB1
ELSEIF a==2
      GOSUB2
ELSE
      GOSUB3
ENDIF
```

Example 2): Switch Statement

A "SWITCH" structure has been added. This switch statement does the same to the example 1. Do not use variable "zzz" if using switch statement.

```
SWITCH a                  'Depending on value of variable "a"
                          'the following cases
                          'select.
CASE 1                    ' If a=1.
        GOSUB1            'go to subroutine 1
        BREAK             'break out of switch statement
CASE 2                    'If a= 2.
        GOSUB2            'go to subroutine 2
        BREAK             'break out of switch statement
DEFAULT                   'Where "a" is anything else
        GOSUB3            'go to subroutine 3
        BREAK             'break out of switch statement
ENDS                      'END of Switch statement
```

There is no limit to the number of WHILE…LOOPs statement.

SECTION 4.4: System Status Bits

State Variables:
State variables are binary variables either set (1) or reset (0). They are reported using the **RB**<bit name> command. Note, some RB<?> are not supported by VRE High Resolution encoder.
For example: **RBe** 'reports the excessive error bit

Report Status Byte/Word Command:
RS Report status byte
RW Report status word

STATUS BYTE:

Bit Name		DESCRIPTON	BIT#	VALUE	COMMENT
Bo	=1	Motor is OFF	Bit7	128	Real time
Bh	=1	Excessive temperature	Bit6	64	Real time
Be	=1	Excessive position error occurred	Bit5	32	Reset by "G"
Bw	=1	Wraparound occurred	Bit4	16	reset by G,MT,&Zw
Bi	=1	Index report available	Bit3	8	reset by RI
Bl	=1	Historical left limit	Bit2	4	reset by Zl, ZS, RS & RW
Br	=1	Historical right limit	Bit1	2	reset by Zl, ZS, RS & RW
Bt	=1	Trajectory in progress	Bit1	1	real time

STATUS WORD:

Bit Name		DESCRIPTON	BIT#	VALUE	COMMENT
Bk	=1	Over current state occurred	Bit15	32768	real time
Ba	=1	Over current state occurred	Bit14	16384	reset by Za & ZS
Bs	=1	Syntax error occurred	Bit13	8192	reset by Zs & ZS
Bu	=1	User array index range error occurred	Bit12	4096	reset by Zu & ZS
Bd	=1	User math overflow occurred	Bit11	2048	reset by Zd & ZS
Bm	=1	Left limit asserted	Bit10	1024	real time
Bp	=1	Right limit asserted	Bit9	512	real time
Bx	=1	Hardware index input level	Bit8	256	real time
Bo	=1	Motor is OFF	Bit7	128	real time
Bh	=1	Excessive temperature	Bit6	64	real time
Be	=1	Excessive position error occurred	Bit5	32	reset by G
Bw	=1	Wraparound occurred	Bit4	16	reset by G,MT,&Zw
Bi	=1	Index report available	Bit3	8	reset by RI
Bl	=1	Historical left limit	Bit2	4	reset by Zl, ZS, RS & RW
Br	=1	Historical right limit	Bit1	2	reset by Zl, ZS, RS & RW
Bt	=1	Trajectory in progress	Bit0	1	real time

COMMUNICATION STATUS BITS:

Bit Name		DESCRIPTON
Bb	=1	Parity error occurred
Bc	=1	Communication overflow occurred
Bf	=1	Communications framing error occurred

States Resets: State variables are reset to zero with the following commands.

Za	Reset hardware current limit violation indication.
Zb	Reset serial data parity error indication.
Zc	Reset communications buffer overflow indication.
Zd	Reset user math overflow indication.
Zf	Reset communications framing error indication.
Zl	Reset left limit "seen" indication.
Zr	Reset right limit "seen" indication.
Zs	Reset user command syntax error indication.
Zu	Reset user array indexing out of range indication.
Zw	Reset wraparound indication.

Global Resets:

Z	Software reset; the user program counter, user variables, and all firmware states are reset.
ZS	Reset all individual user system bits listed above without performing a complete [Z] reset.

SECTION 4.5: Report Mode Commands

The following commands report (prints onto communication cable) function and mode of operation.

RF	Report last **F=<value>**
RCS1	Report and Clear 8 bit check sum of channel 1 communication bytes.
RMODE	Report SmartMotor MODE of operation the response will be one of the following:

P	Absolute position mode
R	Relative position mode
V	Velocity mode
T	Torque mode
F	Follow mode using MF1, MF2, or MF4
S	Step & direction mode
C	Cam mode
X	Follow mode or step & direction mode with ratio
E	Position Error exceeded error limit and motor shut itself off. Issue "G" to get out of error mode. Increase position error limit if necessary.
O	Letter "O" means Motor off

SECTION 4.6: Function Commands

The following commands can only be used with Version 4.11 and below series SmartMotors™

F=2	Disables error code characters "!h", "!1", and "!e" upon command error detection.
F=4	Report command responses are re-directed to RS485 channel 1(Port E and F) instead of to RS232 channel 0 (7 pin Main Connector).

SECTION 4.7: Program Execution Speed (CHANGING PID UPDATE RATE)

If necessary reducing the controller's PID update rate can increase program execution speed. The user can slow the PID update rate with the new PID commands.

The commands are:

PID1	Invokes the default rate (4069 servo samples per second on version 4.00 and up)
PID2	Divides the rate by two (2038 servo samples per second)
PID4	Divides the rate by four (1017 servo samples per second)
PID8	Divides the rate by eight (509 servo samples per second)

NOTES: Since velocity, acceleration and WAIT are functions of PID rate, modify these values to get equivalent motion or timing.

SECTION 4.8: Transmitting and Uploading User Programs on EEPROM

Commands are:

UP	Transmit stored SmartMotor program in tokenized format to the host terminal.
UPLOAD	Transmit program from motor's non-volatile memory to host terminal as seen in programing window.
ES400	Force EEPROMS to be read from & written to at 400 bits per second.
ES1000	Force EEPROMS to be read from & written to at 1000 bits per second. (Newer EEPROMS only)
RCKS	Return check sums. If an "**F**" appears following the check sums, there is a definite read/write error.

SECTION 4.9: LOCKP Command: Prevent Upload of User Program

LOCKP	Disables UP,UPLOAD commands (Host/Terminal Command Only). LOCKP can only be issued from a host computer.

SECTION 4.10: Program Checksum

User programs and subroutine jump tables both have stored checksums. The command RCKS emits the checksums followed by 'F' or 'P' to indicate Fail/Pass. System bit Bk reflects the EEPROM Write/Fail state. A new upload command, "UP", permits a byte for byte comparison to the compiled version of the user source code, since the RCKS pass response does not absolutely verify the stored EEPROM program. The original UPLOAD command recreates the original uncompiled user source code.

The commands are: **RCKS**

RCKS returns two checksums, from the label table and the program. The returned output is:

Label checksum Program checksum 'result' The result is either 'P' or 'F' for pass or fail.

RBk returns a system bit, k. If Bk = **1** it indicates the EEPROM is not verified. A return Bk =**0** indicates no EEPROM failure detected.

SECTION 5: RS232, RS485 and ANILINK COMMUNICATION CHANNELS

Section 5.1: CommunicationCommands

Dramatic changes have been implemented in the area of serial communications. The primary RS-232 channel has been enhanced so that it can input string data. By executing in the "DAT" command, the primary channel will no longer interpret commands, but rather input characters and place them in an input buffer for the user program to access. An RS-485 port has been added to the motor. It uses the same pins as the AniLink port. The AniLink port can be used for RS485 or the Animatics AniLink network. It is not possible to use both functions at the same time. The RS485 port can be opened with a vast array of different parameters and will input commands or data. It responds to the same commands with the addition of the channel number "1".

CMD	Define primary RS-232 port as command input port
CMD1	Define secondary RS-485 port as command input port
DAT	Define primary RS-232 port as data input only
DAT1	Define secondary RS-485 port as data input only
SLEEP1, WAKE1, TALK1, SILENT1	All for additional RS-485 Channel

To open a communication channel:

OCHN(COM TYPE,CHANNEL,PARITY,BAUDRATE,STOP BITS,DATA BITS,SPEC **)**

Example: **OCHN(RS2,0,N,9600,1,8,C)**	Primary host command channel 9600 7 pin combination D-sub connector.
Example: **OCHN(RS2,0,N,9600,1,8,D)**	Primary host data channel 9600
Example: **OCHN(RS4,1,N,38400,1,8,C)**	Second channel: port E (RS485 signal A) and port F (RS485 signal B).
Example: **OCHN(RS4,1,N,19200,1,8,D)**	Second channel.

To close a comm channel:

CCHN(TYPE,CHANNEL**)**

Example: **CCHN(RS2,0)**	close main channel, 7 pin combination D sub connector.
Example: **CCHN(RS4,1)**	close secondary channel, port E and port F.

To identify the existence of characters in comm channel buffer:

LEN, LEN1

To get characters from a comm channel:

GETCHR, GETCHR1

Example

```
IF LEN>0          'Ensure there are characters in the buffer before using GETCHR.
a=GETCHR          'Fetch a character from host communications channel buffer.

IF LEN1>0         'Ensure there a characters in the secondary channel (usually RS485 buffer) before using
                  'GETCHR1.
ab[3]=GETCHR1     'Fetch a character from secondary communications channel buffer to array
```

SECTION 5.2: Print to A Comm Channel

Because multiple comm channels exist now, enhancements were made in the PRINT commands:

PRINT(...)	print to the primary host channel (7 pin combination connector)
PRINTA(...) - PRINTH(...)	print to the AniLink device address A to H (address set by jumpers pin)
PRINT1(...)	print to channel 1 [Port E & F] (usually the default RS485 channel)

Examples:

1) Print to main connector [7 pin combination D-sub connector]

```
PRINT("Hello World")
```

2) Print to AniLink channel [port E (AniLink Data) and port F (AniLink Clock)]
Example: send print message to LCD address B, starting at beginning of second line.
Refer to data sheet for program code specific to your Anilink peripheral device.

```
PRINTB(#148,"Hello World")
```

3) Print RS485 communication using Port E (RS485 signal A) and port F (RS485 Signal B) channel

```
PRINT1 (" T=40 MT ")
```

SECTION 5.3: Reporting Channel Protocol

Report channel commands exist to track the status of the communication channels:

RCHN	report comm channel status - all
RCHN0	report comm host channel status
RCHN1	report comm channel 1 status

Where bit0 = overflow error
bit1 = framing error
bit2 = command scan error
bit3 = parity error

ANIMATICS®

THE SMART Motor™ USER'S GUIDE

UG 0002 (rev.1/02)

Animatics The SmartMotor™ User's Guide.

Please let us know if you find any erorrs or omissions in this work so that we may improve it for future readers. Such notifications should be sent by e-mail with the words "User's Guide" in the subject line sent to: **techwriter@animatics.com**. Thank you in advance for your contribution.

Contact Us:

Animatics Corporation
3050 Tasman Drive
Santa Clara, CA 95054
USA
Tel: 1 (408) 748-8721
Fax: 1 (408) 748-8725
www.animatics.com

TABLE OF CONTENTS

Continued on following page

TABLE OF CONTENTS

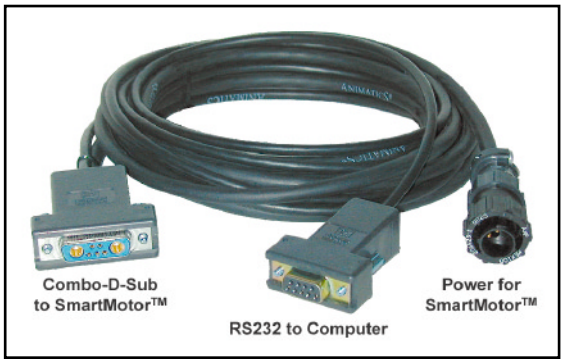
Continued from preceding page

A SmartMotor™ is delivered without any peripheral equipment due to the huge variety of systems and machines it can be used in. In order to make the SmartMotor run, the following will be needed at a minimum:

1. A SmartMotor™
2. A computer running MS Windows 95/98, 2000 or NT
3. A DC power supply for the SM1700 through SM3400 series motors or AC power cord for the SM4200 through SM5600 series motors.
4. A data cable to connect the SmartMotor to the computer's serial port
5. Host level software to communicate with the SmartMotor

The first time user of the SM1700 through SM3400 series motors should purchase the Animatics SMDEVPACK. It includes the CBLSM1-10 data and power cable, the SMI software, the manual and a connector kit.

The CBLSM1-10 cable (right) is also available separately.



Optional SmartMotor™ cable (CBLSM1-10)

Animatics also has the following DC power supplies available:

PS24V8A (24 Volt, 8 Amp) and PS42V6A (42 Volt, 6 Amp)

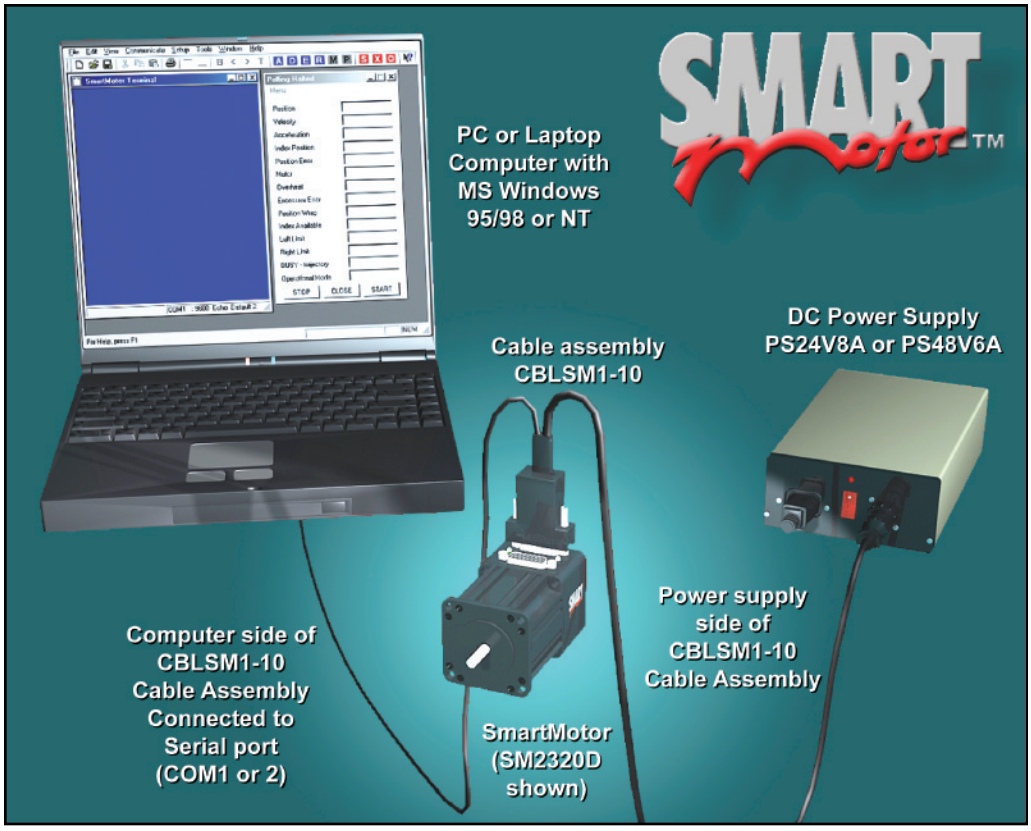
Optional PS24V8A or PS48V6A power supply

Both power supplies will work with the DC motors, but the PS42V6A supply allows the motors to operate at higher speeds. Note that the Speed-Torque curves are taken at 48VDC, the upper limit. Special care must be taken when near the upper limit or in vertical applications that can back-drive the SmartMotor. Gravity influenced applications can turn the SmartMotor into a generator and back-drive the power supply voltage above the safe limit. Many vertical applications require a SHUNT to protect the SmartMotor from damage. Larger open frame power supplies are also available and may be more suitable for cabinet mounting.

"Many vertical applications require a SHUNT to protect the SmartMotor from damage"

For the AC SmartMotors, SM4200 through SM5600 series, Animatics offers:

CBLSMA1-10	10' communication cable
CBLAC110-10	10' 110 volt AC single phase power cord



Connecting a SM2320D SmartMotor using a CBLSM1-10 cable assembly and PS24V8A power supply

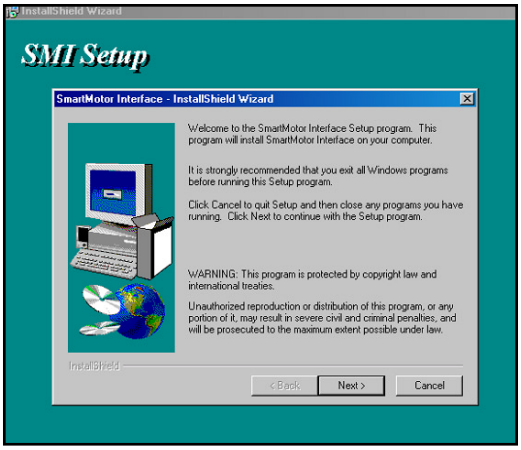
CBLAC200-10 10' 208-230 volt AC 3 phase power cord

SOFTWARE INSTALLATION

***Appendix F** is a complete screen by screen software installation guide*

Either the downloaded software (see **Appendix E**) or the SmartMotor CD-ROM software can be installed with the following procedure. Any differences are noted.

Double click the downloaded file, **SMISetup.exe**, or **Setup.exe** on the SmartMotor CD-ROM. The ***SMI Setup*** window (right) should now be visible.



*The **SMI Setup** window*

If the defaults are accepted on all of the setup screens after this screen, the software will be installed in the **C:/Programs/Animatics** directory. The **Programs** directory, that is accessed from the Windows **Start** button, will now have **Animatics** listed as an option with four subdirectories;

SmartMotor Interface

SmartMotor Tuning Utility

SMI Help

SMIEngine Help

A QUICK LOOK AT THE SMARTMOTOR INTERFACE

The **SmartMotor Interface** (**SMI**) is part of a suite of programs, that can be downloaded free from our web site (***www.smartmotor.com***) or purchased from Animatics Corporation on CD or a set of floppies. The suite runs on the **MS Windows 95/98, Windows 2000** or **NT** operating systems.

*The **SMI** suite is a free download from our web site (see **Appendix E**) or it can be purchased on a single CD or a set of floppies.*

The software connects a SmartMotor, or a series of SmartMotors, to a computer or workstation and gives a user the capability to control and monitor the status of the motors directly from the computer. Every SmartMotor has an ASCII interpreter built in. It is not necessary to use **SMI** to operate a SmartMotor. The interface does, however, allow the user the ability to write programs and download them into the SmartMotor's non-volatile EEPROM memory.

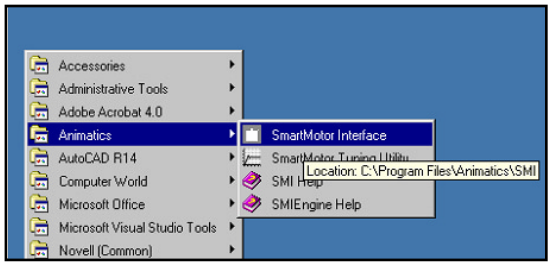
*Every SmartMotor has an ASCII interpreter built in. It is not necessary to use **SMI** to operate a SmartMotor.*

The suite includes the **SmartMotor™ Tuning Utility**, **SMI Help** and **SMIEngine Help** in addition to the **SmartMotor™ Interface** itself. With these tools the SmartMotor(s) can be tuned, addressed, monitored, tested and run from one computer. The SmartMotor can operate in a variety of different configurations. One or more motors can be completely controlled from a host computer, or by a Master SmartMotor programmed to control the others. Alternatively, each SmartMotor can have an independent program. The SmartMotor has three basic functions: **Motion Generation**, **Program Execution** and **Communications**. Each SmartMotor can do all three simultaneously, although the user should take care in implementing communications so as to avoid data collisions when programming multiple SmartMotors to send data over the same network.

PREPARING TO RUN THE SMARTMOTOR

Make sure everything is connected properly (see diagram on preceding page) then turn on the power supply and computer. A red LED should light on the motor indicating that it is powered and ready to go to work.

*Selecting **SmartMotor Interface** from the Windows' **Start** button*



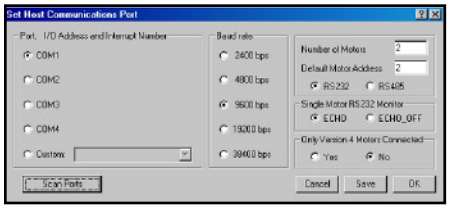
From the Windows desktop click on the **Start** button and then click on **Programs**, **Animatics** and the **SmartMotor Interface**.

__SmartMotor Terminal__ window

The dark blue **SmartMotor Terminal** window (right) should now be on screen. This is the main window that's used to communicate with a SmartMotor.

Click **Setup** on the menu bar, then click **Configure Host Port** from the drop-down menu. The **Set Host Communications Port** window should now be on screen.

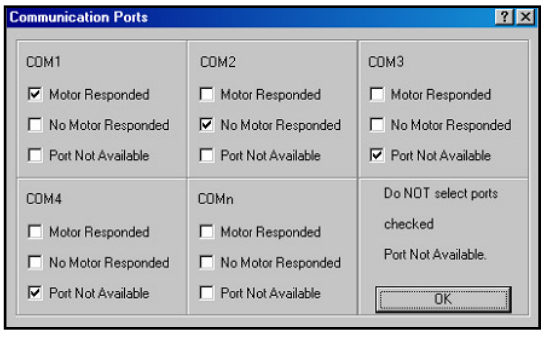
__Set Host Communications Port__ window



Select the **Scan Ports** button (lower left corner) and the **Communication Ports** window (below) should open.

This window only reports the status of the computer's serial ports and cannot be edited. If a SmartMotor is found (COM1 in the example) the **Motor Responded** box will have a check mark in it and the other two boxes under COM1 will be blank. If no motor is found (COM2) the **No Motor Responded** box will be checked. The computer used for these examples only has two COM ports (1 and 2) so the other COM ports (3 and 4) have **Port Not Available** checked. COMn isn't addressed.

__Communication Ports__ window



If the motor being tested returns similar results to those in the example, the computer and SmartMotor are properly interconnected and the SmartMotor is ready to transmit and receive data through its RS-232 serial port.

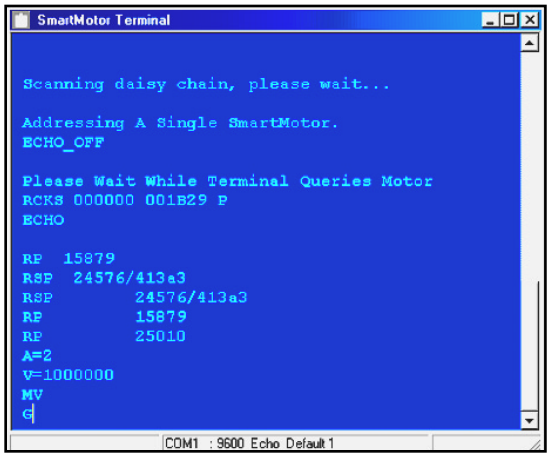
If no COM resource is available, independent measures should be taken to obtain an available port. This could be done either through system re-configuration or through the use of a USB to RS-232 or similar port converter.

MAKING THE SMARTMOTOR RUN

This is a quick tutorial designed to help the new **SmartMotor™** user get started. The commands used here are only described in enough detail to help in understanding what is happening.

The larger SmartMotors can shake and move suddenly and should be restrained for safety.

Click on the **A** button on the **SMI** tool bar. A few messages and responses will appear in the blue **SmartMotor Terminal** window (right).



***SmartMotor Terminal** window with example data*

Now the computer and the SmartMotor are communicating and ready for commands. The example is showing the data returned from the SmartMotor used to get these screen shots. The data returned from a different motor may be quite different.

Transmitting commands

In the **SmartMotor Terminal** window type **RSP** (with CAPS LOCK *on*) followed by the enter key. The SmartMotor should respond with a string of data in the terminal window containing version information.

Type **RP** and then Enter. The motor responds with its current position (15879 in the example, above). Rotate the SmartMotor's shaft a little and type **RP** again. Note the change in the position (25010, above).

The **RP** (**R**eport **P**osition) and **RSP** (**R**eport **S**ample **P**eriod and version number) are Report to Host commands and are only two out of over sixty that can be used to query almost every aspect of a SmartMotor's status and performance. When any of the commands are entered into the **SmartMotor Terminal** window they return whatever data the command solicits.

Initiating motion

Now enter the following lines into the **Terminal** window (omitting the comments to the right).

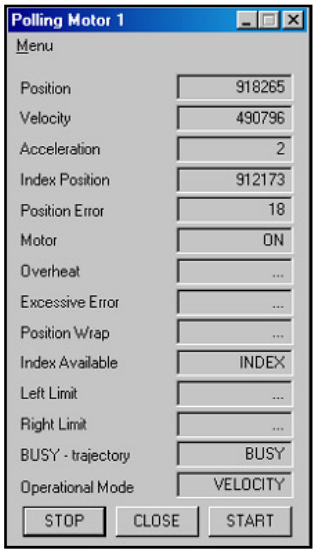
```
A=100          'sets the Acceleration
V=1000000      'sets the maximum Velocity accelerated to
P=300000       'sets the target Absolute Position
G              'Go, initiates motor movement
```

1000000 Scaled Counts/Sample= about 1860 RPM for SM2300 series motors and about 930 RPM for series SM3400, 4200 and 5600 motors

After the final **G** command has been entered, the SmartMotor will accelerate up to speed, slew and then decelerate to a stop at the absolute target position.

Monitoring motor status

Click on the M button on the toolbar and the **Polling Motor 1** status window (right) will open. This window shows the current status of the test SmartMotor (number 1).



Polling Motor 1 *dialog box. (The number refers to the motor number)*

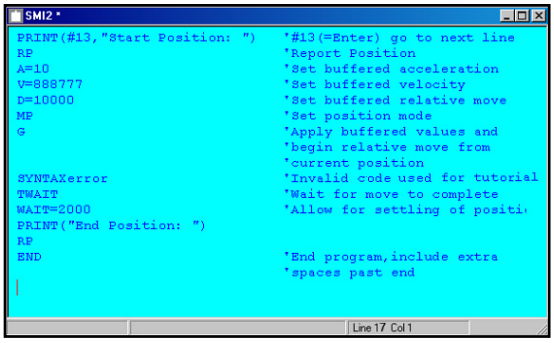
While the monitor is polling the motor, enter a new position and repeat the **G** command to initiate the move. With the monitor on, the progress of the move can be seen. The changing of the **Index Position** can also be seen as the encoder's index passes with each revolution (Motors equipped with a variable resolution encoder do not have an index marker).

To stop the motor in the middle of a move, enter the **X** command in the **SmartMotor Terminal** window. The SmartMotor will decelerate to a stop at the same rate the **A**cceleration parameter was last set (**A=100**).

Writing a user program

Press the button on the toolbar and the **SMI2*** program editing window (below) will open. This window is where SmartMotor programs are entered and edited.

Enter the following program in the editing window. It's only necessary to enter the boldface text. The text preceded by a single quote is a comment and is for information only. Comments and other text to the right of the single quotation mark do not get sent to the motor. The text is usually used to describe what's going on at that moment in the program. Pay close attention to spaces and capitalization. The code is case sensitive and a space is a programming element.



Program editing window.

*If the **SmartMotor** used in this test needs a memory module make sure one is installed.*

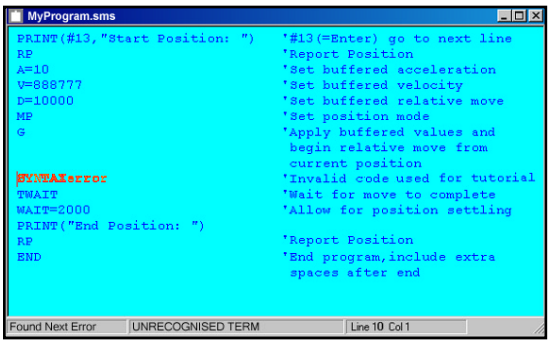
```
PRINT(#13,"Start Position: ")
                    '#13(=Enter) go to next line
RP                  'Report Position
A=10                'Set buffered acceleration
V=888777            'Set buffered velocity
D=10000             'Set buffered relative move
MP                  'Set position mode
G                   'Go, apply buffered values and begin
                      relative move from current position
SYNTAXerror         'Invalid code used for tutorial
TWAIT               'Wait for move to finish
WAIT=2000           'Allow for position settling
```

```
PRINT("End Position: ")
RP                  'Report Position
END                 'End program, include extra
                       spaces after END for test
```

After the program has been entered, select **File** from the menu bar and **Save as . . .** from the drop down menu. In the **Save File As** window give the new program a name such as "MyProgram.sms" and click on the **Save** button. The file name will replace the edit window title (**SMI2***).

Transmitting the program to a SmartMotor

To check the program and transmit it to the SmartMotor, click on the T button located on the tool bar. A small window will open with **Errors Found**. Click **OK** to close the new window. The error, "SYNTAXerror", should now be colored red. Click the > button and the cursor will jump to the line with the error.



Program editing window with "SYNTAXerror" highlighted

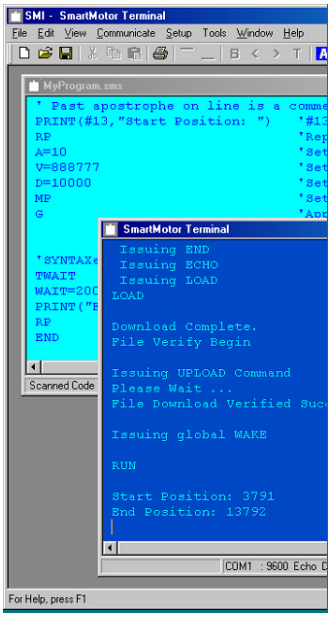
If there were additional errors the > button would send the cursor to the next error each time it was clicked. The < button steps through the errors backwards, the ⎸ sends the cursor to the beginning of the program and the ⎹ sends it to the end.

The bottom status bar of the editing window should now have three entries; **Found Next Error**, **UNRECOGNIZED TERM** and **Line 10 Col 1** (the present cursor position if it's located at the beginning of the line with the error).

To correct the error, use the cursor to highlight just the "SYNTAXerror" line, including the single quote and the comment, "Invalid code for tutorial" and press the delete key.

If the error had been a typo (**TWAIP** possibly) it could have been fixed by highlighting just the error (the P) and typing the proper letter (a capital **T**).

After the error has been corrected, click on the T button again. Some messages should start scrolling down the **SmartMotor Terminal** window and a progress bar showing **SMI** is transmitting the program to the SmartMotor should also be on screen (right).



Transmitting program to SmartMotor

Running the downloaded program

The programmed motor now returns data (the **PRINT** commands), and these characters can cause errors in the data returned to the **Polling Motor 1** window (page 10). To keep this from happening click on the **STOP** button at the bottom of the **Polling Motor 1** window and then click R on the toolbar to run the program for the first time.

The SmartMotor accelerates at the rate set (**A=10**) to the programmed speed or **V**elocity (**V=888777**) and then maintains that speed for the **D**istance (**D=10000**), determined by the number of motor shaft rotations, and then decelerates to a stop at the rate set by the **A**cceleration command (**A=10**). As the motor spins up watch the data in the **SmartMotor Terminal** window. Run the program several times by clicking on R and watch the results.

Because servomotors operate based on position error feedback, position may oscillate slightly.

Turn off the power to the SmartMotor and after a few seconds turn it back on. The motor will run the resident program every time it is turned on until a new program is downloaded. The SmartMotor will do this whether there's a computer connected to it or not. All it needs now is power.

TALKING TO SMARTMOTORS

The dark blue **SmartMotor Terminal** window opens whenever the **SMI**software is turned on. If the window has been closed, another operation requiring its use will open a new window.

To transmit a command to the SmartMotor in the **SmartMotor Terminal** window type the command on a new line and Enter on the same line.

Any line, or part of a line, can be transmitted through the serial port by moving the cursor to the line's end or anywhere along the line and pressing the Enter key. Everything to the left of the cursor position will immediately be transmitted to the SmartMotor and the cursor will jump to the end of the listing, past the last line of text and after a copy of what was just transmitted. If the highlighted line of text contains a macro, the macro will be expanded and transmitted as well. Macros are explained later in this chapter.

The Tab key sends the cursor to a new line after the last line of text without transmitting any data to the SmartMotor, no matter where the cursor was located in the **SmartMotor Terminal** window.

SMI is continuously monitoring the serial ports for data, so the asynchronous input is displayed by the **SmartMotor Terminal** window.

More than one command can be entered on a line when separated by a space. For example; **RA RV RP** reports **A**cceleration, **V**elocity, and **P**osition.

The SmartMotor Terminal status bar displays the COM port, baud rate, and Default Motor Address.

ECHO and ECHO_OFF Modes

The **SmartMotor Terminal** runs in either **ECHO** or **ECHO_OFF** mode. In **ECHO** mode, the SmartMotors echo back every character they receive and the **SmartMotor Terminal** refrains from displaying both the sent and received lines. When in **ECHO** mode, **SMI** will warn the user if it fails to receive the expected echoed response. In the **ECHO_OFF** mode, the terminal has no expectation of receiving an echo of the command sent. The **Send ECHO** and **Send ECHO_OFF** sub-items of the **Communicate** drop-down menu can serve to synchronize the **SmartMotor Terminal** and SmartMotor(s). Otherwise, if **SMI** is expecting an echo and the motor is not in echo mode, warnings will appear.

*The **SMI** software uses the ECHO feature to Auto-Address a Daisy Chain of SmartMotors.*

A single RS-232 SmartMotor can operate in either state, but a daisy chain of RS-232 SmartMotors must operate in **ECHO** mode for data to get through the network. RS-485 SmartMotors must operate in the **ECHO_OFF** mode or there will be data collisions.

Motors using RS-485 must address themselves by way of a stored program.

The third item on the **SmartMotor Terminal** status bar displays "Echo" if the Terminal is in the **ECHO** mode.

***Note**: When operating in ECHO_OFF mode the SmartMotor Terminal window has no way to synchronize with SmartMotors. If the values appear to be incorrect in the Polling Motor window, STOP the polling and START it again using the buttons at the bottom of the window.*

Addressing SmartMotors

In general, a single SmartMotor doesn't have to be addressed before using **SMI**. All of the motors in multiple SmartMotor installations (daisy chain) must be addressed first, otherwise the **SmartMotor Terminal** window may not work properly. The software stores information about each of the SmartMotors as their addresses are assigned.

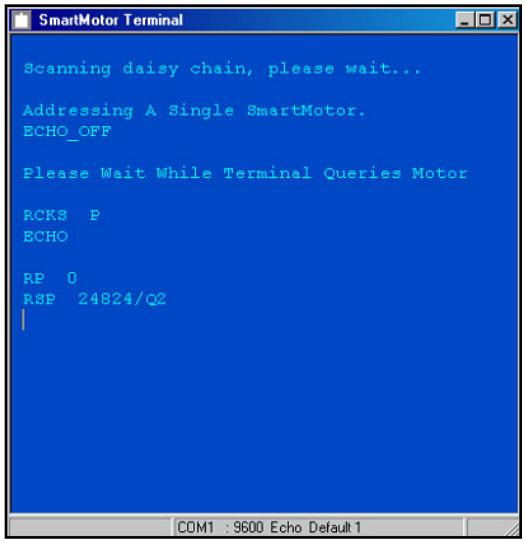
Addressing the SmartMotor(s) can be done by selecting:

Communicate from the ***Menu bar*** and **Address Motor Chain** from the drop-down menu.

OR Click the A button on the ***Toolbar***

OR Enter an **&** (ampersand) character in the **SmartMotor Terminal** window.

Data returned after addressing a single SmartMotor

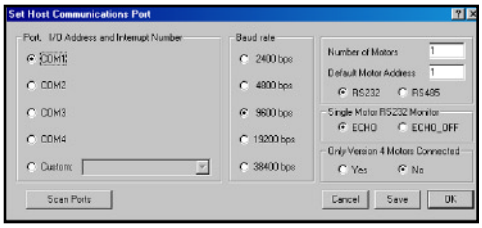


When a single motor is addressed it should return results similar to those shown in this window to the right.

Changing the baud rate

When switching to a different baud rate, change the motors baud rate first, and then change the **SMI** baud rate. The supported rates are 2400, 4800, 9600, 19200 and 38400.

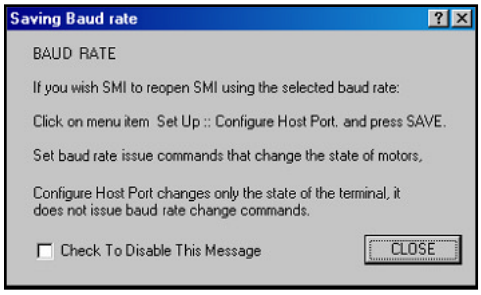
Set Host Communications Port *window*



To change the SmartMotor's and **SMI's** baud rate, click on the **Communicate** drop-down menu and click on **Send New Baud Rate**. If the changes are to be used whenever **SMI** is started, click **Setup** on the menu bar and from that drop-down menu select **Configure Host Port** and then **Save** at the bottom of the **Set Host Communications Port** window. When the **Send New Baud Rate** command is chosen, the **Saving Baud Rate** window (left) gives instructions for saving the new settings.

If a SmartMotor is powered on with no program in its memory, it defaults to 9600 baud.

Saving Baud Rate *window*



Using Macros

Macros are used to store frequently used single commands or multiple command strings.

The maximum number of macros that can be recalled is 50, the maximum number of characters in a macro's name is 20 and the command string character limit is 220. Up to 4 Macros can be nested (a macro contained within another macro). The macro list is loaded when **SMI** is opened and saved when **SMI** is closed.

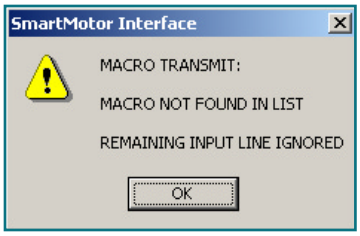
To define a macro use the following syntax followed by the enter key:

%MacroName Commands%

For example this line: %CmdS **RA RV RP**% defines a macro named **CmdS** which issues the **RA**, **RV** and **RP** commands (Notice the lower case letters in the title, **CmdS**).

To run a macro use the following syntax followed by the enter key: /MacroName

Macro's are case sensitive and the title must be entered exactly as it was typed. On the right is the error message received if a mistake is made.



Macro transmit error window. Click on **OK,** *retype the macro's name and try again*

To run the macro defined above enter the title exactly as it was typed: **/CmdS**

To list all of the defined macros select:

Communicate from the ***Menu bar*** and **List Macro Definitions** from the drop-down menu

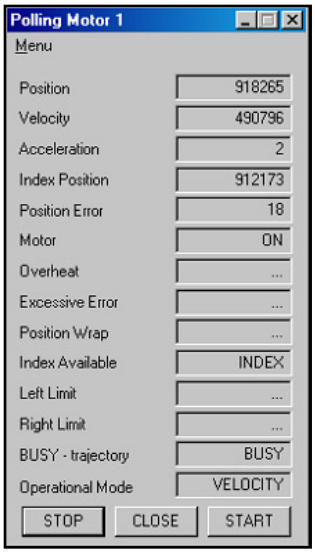
OR enter **//** in the **SmartMotor Terminal** window.

To delete an existing macro type: -MacroName

To edit an existing macro, type: +MacroName

MONITORING SMARTMOTOR PARAMETERS

The **Polling Motor** window displays the SmartMotor's current status and parameter values.



To open the **Polling Motor** window, select:

Communicate from the ***Menu bar*** and **Monitor Status** from the drop-down menu

OR click on the **M** button on the ***Toolbar***

Polling Motor *window*

The most commonly used SmartMotor parameters are shown in this window. When the window is first opened, it immediately starts polling the SmartMotor(s) and updates the parameters.

Polling can be stopped with the **STOP** button and restarted with the **START** button at the bottom of the **Polling Motor** window. The **CLOSE** button closes the window. This window is read-only and cannot be edited.

Typing data into the **SmartMotor Terminal** window will pause the data stream to the **Polling Motor** window. Polling remains paused until the command is finally transmitted to the motor with the enter key.

If a program is written that transmits data through the serial port (the **PRINT** commands), it may conflict with the data displayed in the **Polling Motor** window. If this type of data is being used, stop polling.

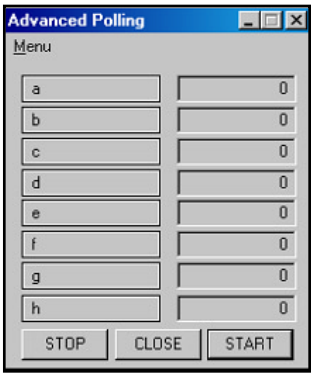
Advanced Polling

The **Advanced Polling** window displays up to eight user defined parameters.

To open the **Advanced Polling** window select:

Communicate from the ***Menu bar*** and **Advanced Status** from the drop-down menu

OR click on the **P** button on the ***Toolbar***

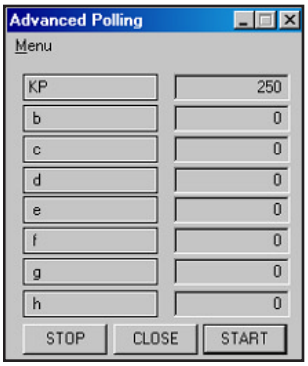


Advanced Polling *window*

The value of a variable, status of a port or other customized parameters need to be monitored. Like the **Polling Motor** window, the **Advanced Polling** window (right) sends out report commands to the motor(s) in the background and shows the received responses in the window. If any of the blocks on the left side of the window are clicked, the **Standard Polling Variables** window will open.

These polling variables can be selected to replace the block clicked. For example click on the first block (the **a** block) in the **Advanced Polling** window, and click on the **KP** button in the **Standard Polling Variable** window. Then click on **CLOSE**. A window will open with instructions on how to save the advanced polling settings. Close the window to return to

the **Advanced Polling** window. Now the **a** in the first block is changed to **KP**. **P**ress the **START** button to start polling the motors again. The value of the **KP** parameter should be in the data box on the right side (250 in the example, right).

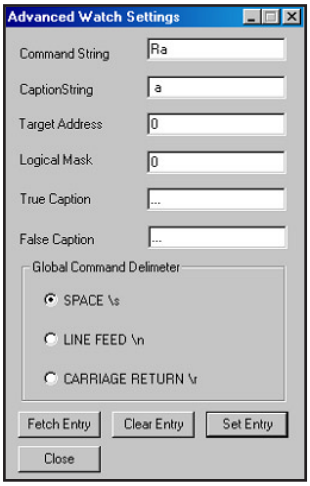


*Monitoring the **KP** parameter in the **Advanced Polling** window*

Note: Pressing any of the blocks on the left side of the Advanced Polling window will automatically stop polling. It can only be restarted with the START button.

At times it's necessary to monitor the status of a hardware port or an internal SmartMotor variable. Polling parameters can be customized using the **Advanced Watch Settings** window (below). To open the window, click on the **Advanced** button on the lower right corner of the **Standard Polling Variables** window.

The following is a description of each data entry box in the **Advanced Watch Settings** window:



***Advanced Watch Settings** window*

Command String: The actual data transmitted to the SmartMotor(s).

Caption String: Data displayed in the boxes on the left side of the **Advanced Polling** window.

Target Address: If there is more than one SmartMotor, enter the address of the motor to monitor.

Logical Mask: Enter the logical mask (in base 10) to apply to the SmartMotor response. If this value is "0" no mask is applied.

True caption: The data displayed if the final response value is not zero. The final value of the response is calculated by ANDing the SmartMotor's response with the logical mask value.

False caption: The data displayed if the final response value is zero. The final value of the response is calculated by ANDing the SmartMotor's response with the logical mask value.

Global Command Delimiter: The command delimiter in the command string. Usually commands are separated by one space.

Fetch Entry: Fetches current user defined configuration and updates the window entries.

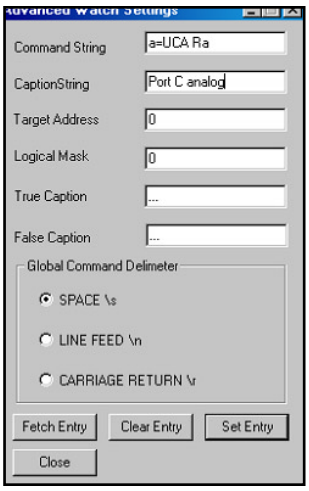
Clear Entry: Resets the current entry to null settings.

Set Entry: Places the settings in current watch configuration

Close: Saves any changes and closes the window.

Advanced settings for monitoring custom parameters.

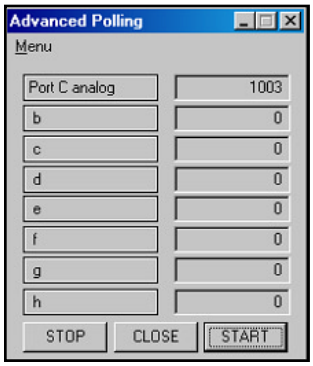
Advanced Polling Example



The following example shows how to set the advanced polling parameters to monitor motor data.

1. Click on the **a** box in the **Advanced Polling** window. The **Standard Polling Variables** window will open.
2. Click on the **Advanced** button. The **Advanced Watch Settings** window (right) will open.
3. Enter the commands to be sent to the SmartMotor in the **Command String** box. In this example the analog value of port C needs to be monitored. Note that there must be a space between **a=UCA** and **Ra**.
4. Enter a title for the string in the **Caption String** box (**Port C analog** in the example)

Monitoring the analog value of port C

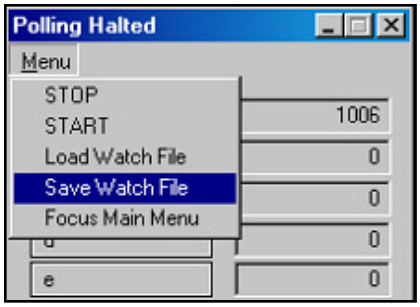


5. If the SmartMotor is part of a daisy chain, enter the motor's address in the **Target Address** box and modify the title in the **Caption String** to reflect the motor being polled. If there is only one motor connected to the computer the **Target Address** box can be ignored.
6. Click on the **Set Entry** button, and then close the window. Close the **Standard Polling Variables** window.
7. In the **Advanced Polling** window, press the **START** button and watch the selected parameter polling (above).

***Note**: Lengthy command strings can cause communication errors. Advanced polling parameters are polled in the background and programs that send characters through the serial port during polling may cause problems.*

Saving and loading advanced settings

Saving and loading advanced settings



Advanced Polling window settings can be saved for future use. Click on **Menu** and select **Save Watch File** from the drop-down menu. In the save window, enter a name for the file. To load a previously saved file, select **Load Watch File** from the drop-down menu.

PROGRAMMING WITH SMI

This section focuses on the **SMI tools** used to create and edit SmartMotor programs. Later sections will focus on actual programming.

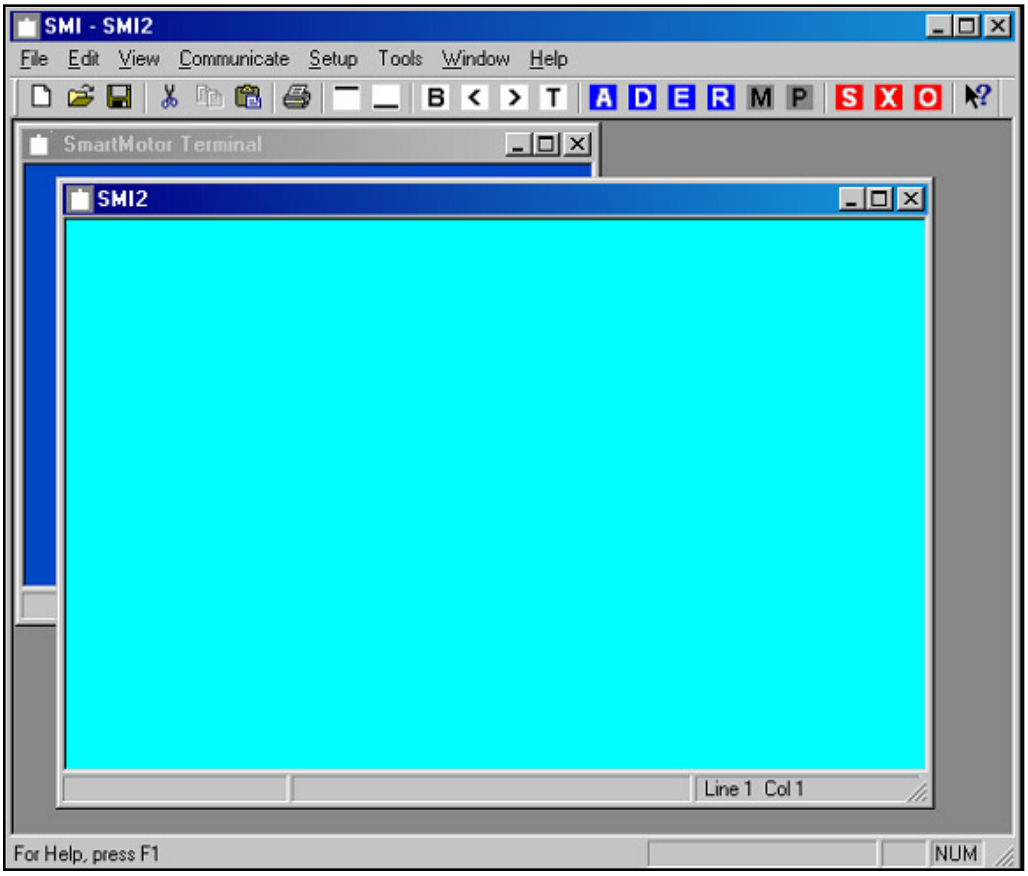
To create a new program file, select:

File from the ***Menu bar*** and **New** from the drop-down menu

OR Click on the button on the ***Toolbar***

OR Press the Ctrl+N keys.

A new window will open (below).



Program edit window

This is a simple text-editing window that's used to edit or create new programs. It works like all Windows based text editors with all of the Cut, Copy, Paste and Delete capabilities. When a new edit window is opened, it will have a default name (**SMI2** in this example).

*There are several sample programs in the **SMI Help** file. Just copy and paste them into the **SMI** text editor.*

Saving program files

The currently active program can be saved by selecting:

File from the ***Menu bar*** and **Save** or **Save As** from the drop-down menu.

OR Click on the button on the ***Toolbar***

OR Press the Ctrl+S keys for **Save** or Shift+Ctrl+S for **Save As**.

When saving a document for the first time, **SMI** displays the **Save As** window so the document can be named before it is saved. To change the name of an existing document, use the **Save As** command. There are two types of documents that can be saved by the **SMI** software:

- Program source files (.sms).
- ASCII source files (.src).

A **SmartMotor Terminal** window can be saved with file extension .mon, but it will be read only when it is reopened, which means it can't be modified and commands can't be sent to the motors through it. Only one **SmartMotor Terminal** window can be open at a time. To use a saved window, close the open **SmartMotor Terminal** window and open the saved window by clicking on **File** on the menu bar and then **Open** from the pull down menu and then double click on the file desired.

Loading program files

To open a previously saved program, select:

File from the ***Menu bar*** and **Open** from the drop-down menu.

OR Click on the button on the ***Toolbar***

OR Press the Ctrl+O keys.

Use this command to open an existing program source document or terminal document in a new window.

Use the **Open** command to open ASCII source files (with extension .src) if the **ASCII source** type was selected. The file is automatically renamed with a .sms extension. This replaces the **Import ASCII Source** command in older versions of **SMI**.

Building (Compiling) an SMI program

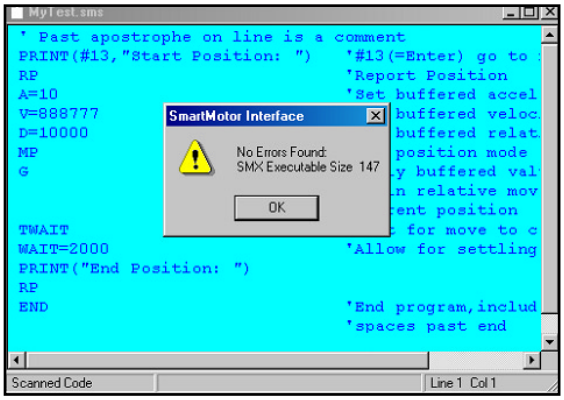
Compiling a program with no errors

Check and Compile a program by selecting:

Edit from the ***Menu bar*** and **Make SmartMotor Executable** from the drop-down menu.

OR Click on the button on the ***Toolbar***

OR Press the Ctrl+E keys.



The program is scanned for syntax and statement label errors. A window will show the end of scanning. If no errors were found, the first pane of the **Edit** window ***Status bar*** will have the **Scanned Code** message as **SMI** builds both the un-compiled .smx, and compiled, .sm, program files.

After the program has been compiled a small message box will be displayed stating: **No Errors Found: SMX Executable Size** followed by a number indicat-

ing the number of program executable bytes that will be used when this file is downloaded. Just click on **OK** to continue.

If errors are found, the message box will state **Errors Found** and the errors will be colored red.

Locating the Errors

The following are shortcuts for moving the cursor around the editing window:

To go to the beginning of the program, select:

Edit from the ***Menu bar*** and **Go To Top Of File** from the drop-down menu

OR Click on the button on the ***Toolbar***

OR Press Ctrl+Home on the keyboard

To go to the end of the program select:

Edit from the ***Menu bar*** and **Go To End Of File** sub-item from the drop-down menu

OR Click on the button on the ***Toolbar***

OR Press Ctrl+End on the keyboard

To go to the next error position select:

Edit from the ***Menu bar*** and **Find Next Error** from the drop-down menu

OR Click on the button on the ***Toolbar***

To go to the previous error position select:

Edit from the ***Menu bar*** and **Find Previous Error** from the drop-down menu

OR Click on the button on the ***Toolbar***

Downloading programs to a SmartMotor

To download a program to the SmartMotor select:

File from the ***Menu bar*** and **Transmit Program to Motor** from the drop-down menu.

OR Click on the button on the ***Toolbar***

OR Press the Ctrl+T keys.

If you press the button without Building or Compiling your program first, it will compile automatically, saving you the extra step.

Use this command to transmit the compiled source code of the presently active document window to the default-addressed SmartMotor.

This command first scans the currently active source document code for syntax and statement label errors. If any errors are found, they are treated as they were in the **Building (Compiling) an SMI program** section.

SmartMotor(s) must be addressed before a program can be downloaded to them.

If scanning the program finds no errors, the file will be compiled and transmitted to the addressed SmartMotor.

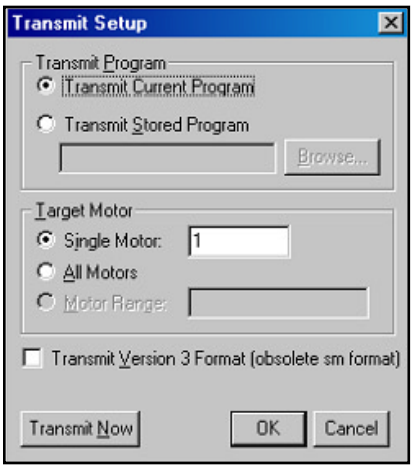
The integrity of downloaded code is validated by checksum. If the configuration defines more than one motor, then the user defined default motor will receive the file. Note that the source code comments are not sent to the SmartMotor.

If **SMI** finds an older SmartMotor not capable of handling a compiled file, it will issue a warning and transmit a .sm format file instead.

Note: *Older SmartMotors can be found with less or slower memory than has been shipping since roughly year 2000-on. Programs exceeding 8k should not be downloaded to the older 8k EEPROMS. SMI does not have the capability to determine the EEPROM capacity of a specific SmartMotor. Repeated checksum errors can be solved by issuing the ES400 command, slowing the programming and reading rate from 1,000bps to 400bps.*

Transmit Setup

*The **Transmit Setup** dialog box*



The **Transmit Setup** window (right) can be used to define the functionality of the **Transmit Program** command.

The **Transmit Setup** window is accessed by selecting:

> **File** from the ***Menu bar*** and **Transmit Setup…** from the drop-down menu.

It can be setup to always transmit the current program or a program stored in a file. When transmitting the current program, highlight the editor window that contains the program to send.

For a one-time occasion, use the **Transmit Now** button to use the currently displayed configuration and then press **Cancel** so the changes to the setup are not saved.

Check the **Transmit Version 3 Format** to transmit a current program in the **.sm** format.

Uploading a program from a SmartMotor

To upload a SmartMotor's program, select:

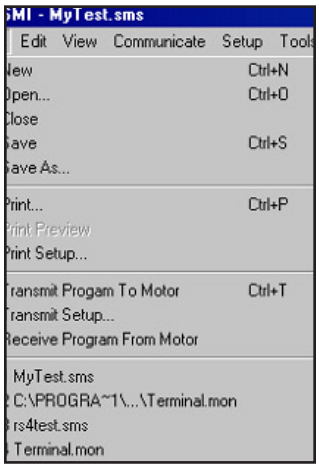
> **File** from the ***Menu bar*** and **Receive Program from Motor** from the drop-down menu

A new edit window will open and the SmartMotor's program will be uploaded to it. The program can then edited, saved and transmitted back to the motor. Note that the uploaded program has no comments. They were stripped off when the program was transmitted to the SmartMotor. Additional and unprintable compiler codes are removed during upload.

SMI MENU COMMANDS

This section describes the **SMI** ***Menu bar*** pull down menus. Many of these commands are also discussed in the **SMI** programming section.

File pull down menu:

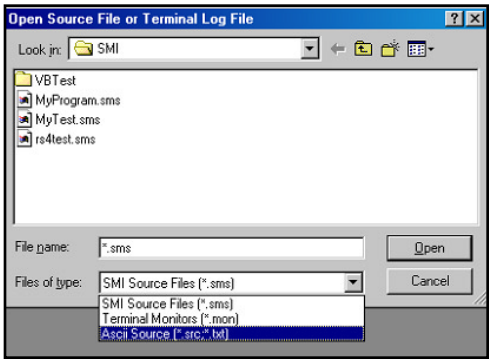


***File** menu*

New: (Ctrl+N) Create a new file in a new document window.

The file format is RTF (Rich Text Format). It cannot be read directly by an ASCII editor.

Open: (Ctrl+O) Open an existing document or terminal document in a new window. This command can be used to open ASCII source files (with extension .src) if the "ASCII source" choice was picked.

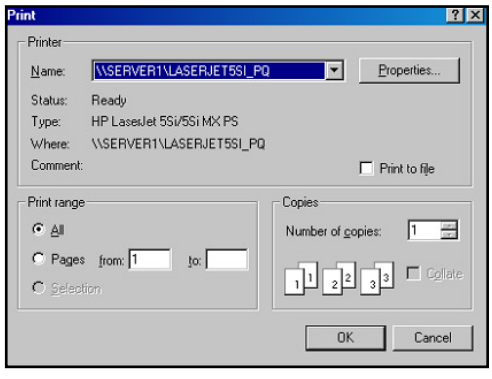


***Open** File dialog*

Close: (Ctrl+W) Use to close an open document window. If a modified document is not saved, **SMI** displays a message asking whether the changes should be saved.

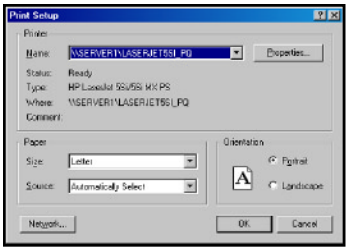
Save: (Ctrl+S) Use this command to save the active document to its current name and directory. When a document is saved for the first time, **SMI** displays the **Save As** window so it can be renamed.

Save As: (Shft+Ctrl+S) Use this command to save and name the active document. **SMI** displays the **Save File As** dialog where the document can be named and then saved.



***Print** window*

Print: (Ctrl+P) Opens the **Print** window (right) where the number of copies to be printed can be set and the destination printer, and other printer setup options can be selected or changed. When **OK** is clicked the document will print.



***Print Setup** window*

Print Preview: Disabled.

Print Setup: Opens the **Print Setup** window (left) where the printer, printer connection, paper size, and paper orientation can be selected and changed.

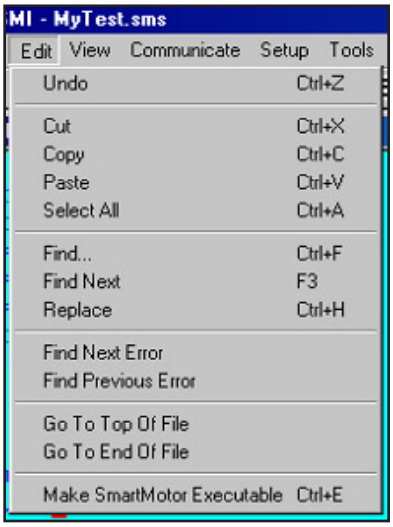
Transmit Program To Motor: (Ctrl+T) Download the program in the currently active window to the SmartMotor(s).

Transmit Setup: Adjust settings for downloading programs.

Receive Program from Motor: Receive a program from the default addressed motor to a newly opened window.

Exit: (Ctrl+Q) End the **SMI** application session.

Edit pull-down menu



Edit pull down menu

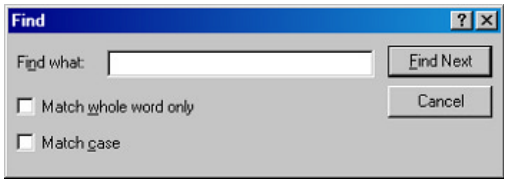
Undo: (Ctrl+Z) Reverses previous editing operation in the currently active document window, if possible. **SMI** provides only the most minimal support for the Undo operation.

Cut: (Ctrl+X) Deletes the highlighted data from the currently active document and moves it to the Windows clipboard.

Copy: (Ctrl+C) Copies the highlighted data from the currently active document to the Windows clipboard.

Paste: (Ctrl+V) Pastes data from the Windows clipboard into the currently active document.

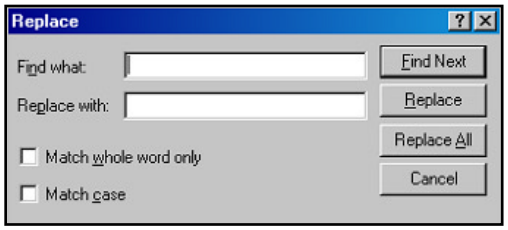
Select All: (Ctrl+A) Selects (highlights) the entire data in the presently active document, making it ready for global cut or copy operations to the windows clipboard.



Find window

Find: (Ctrl+F) Displays a "**Find**" dialog to perform a text string search in currently active document (left).

Find Next: (F3) Repeats previous Find text search operation in the currently active document.



Replace window

Replace: (Ctrl+H) Displays the **Replace** window to perform Find and Replace operations in the currently active document (left).

Find Next Error: Places the cursor at the next error in the currently active source file.

Find Previous Error: Places the cursor at the previous error in the currently active source.

Go To Top Of File: Places the cursor at the beginning of the currently active source file document.

Go To End Of File: (Ctrl+End) Places the cursor at the end of the currently active source file document.

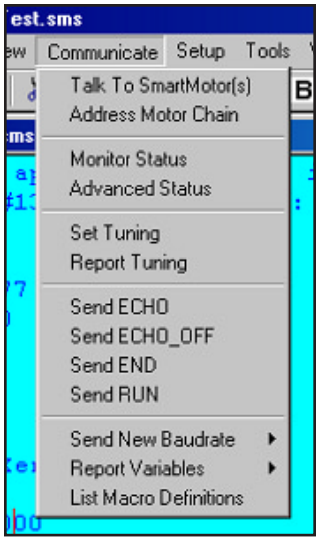
Make SmartMotor Executable: (Ctrl+E) Scans source code within the currently active document for syntax and statement label errors and creates both uncompiled and compiled program files.

View Menu Commands:

Toolbar: Shows or hides the tool bar.

Status Bar: Shows or hides the status bar.

Communicate Menu Commands:



Communicate *menu commands*

Talk to SmartMotor(s): Opens the **SmartMotor Terminal** window, if it is not already open.

Address Motor Chain: Automatically addresses the motors connected to the serial port. This only works with an RS-232 Daisy Chain.

Monitor Status: Opens the **Monitor Status** window, if not already open and starts polling the default addressed SmartMotor.

Advanced Status: Opens the **Advanced Monitor Status** window, if not already open.

Set tuning: Displays the **Set Tuning Parameter** window.

Report Tuning: Requests the tuning parameters from default addressed SmartMotor and displays them in the Terminal window.

Send ECHO: Sends a global **ECHO** command to all SmartMotors. After this command is received, a SmartMotor receiving commands or data will echo it to the next motor or back to the terminal.

Send ECHO_OFF: Sends a global **ECHO_OFF** command to all SmartMotors. After this command is received, a SmartMotor receiving commands or data will not echo it to the next motor or back to the **SmartMotor Terminal.**

Send END: Sends a global **END** command to all SmartMotors.

Send RUN: Sends a global **RUN** command to all SmartMotors.

Send New Baud rate: Issues the new bit-rate to all SmartMotors and then changes the **SMI** bit-rate to the new value. The new bit-rate is displayed in the second pane of the **SmartMotor Terminal** ***Status bar***.

Report Variables:

Report **a…z**: Request default addressed SmartMotor to report the value of variables **a** through **z** (version 3.4 and earlier SmartMotors only have variables **a** through **j)**.

Report **aa…zz**: Request default addressed SmartMotor to report the value of variables **aa** through **zz** (not available for Version 3.4 and earlier SmartMotors).

Report **aaa…zzz**: Request default addressed SmartMotor to report the value of variables **aaa** through **zzz** (not available for Version 3.4 and earlier SmartMotors).

Report **ab[0]…ab[200]**: Request default-addressed SmartMotor to report the value of 8 bit variables **ab[0]** through **ab[200]** (not available for Version 3.4 and earlier SmartMotors).

Report **aw[0]…aw[99]**: Request default-addressed SmartMotor to report the value of 16 bit variables **aw[0]** through a**w[100]** (not available for Version 3.4 and earlier SmartMotors).

Report **al[0]…al[50]**: Request default-addressed SmartMotor to report the value of 32 bit variables **al[0]** through **al[50]** (not available for Version 3.4 and earlier SmartMotors).

List Macro Definitions:

List user macro definitions to the **SmartMotor Terminal** window.

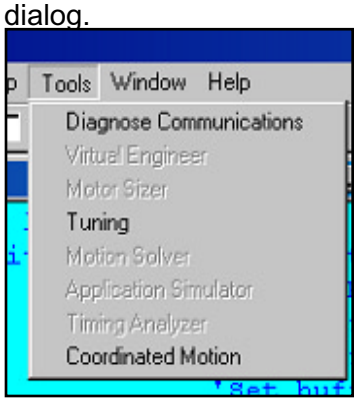
Setup Menu Commands:

Configure Host Port:

Displays the Set Default Communications Port dialog.

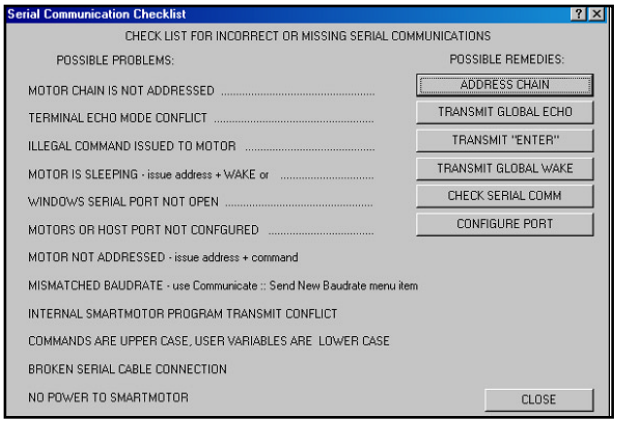
Tools Menu Commands:

Tools *drop-down menu commands*



Diagnose Communication: This command displays the **Serial Communications Checklist** window. If there is a communication problem with a SmartMotor, choose **Tools** from the menu bar and **Diagnose Communications** from the drop-down menu. This should turn on the **Serial Communication Checklist** window (below)

Serial Communication Checklist *window*



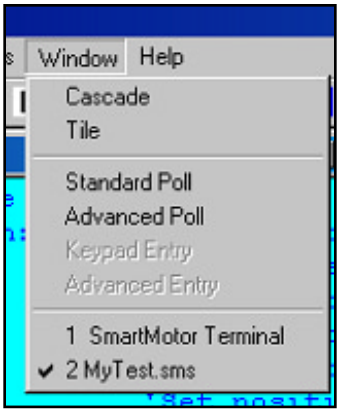
Read all of the lines on the left side and press the button to the right (if any) or do what is recommended.

Virtual Engineer: (future)

Motor Sizer: (future)

Tuning: The Tuning Utility is a stand-alone tool that can be used to plot the position of the shaft as it makes a quick

step motion. It has an integrated interface that allows the user to easily modify the **P.I.D.** Parameters and update them on-the-fly. The Tuning utility is detailed in a later section.



***Window** drop-down menu*

Motor Solver: (future)

Application Simulator: (future)

Timing Analyzer: (future)

Coordinated Motion: This is another stand-alone tool that manages the sending of coordinate data to SmartMotors.

Window Menu Commands

Cascade: Arranges the open document windows in overlapped fashion within the main program window.

Tile: Arranges the open windows in a non-overlapped fashion within the main program window.

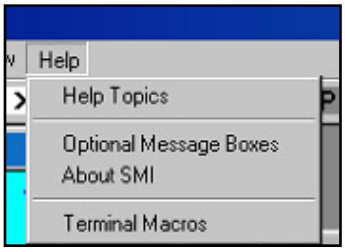
Standard Poll: Brings the Standard Poll dialog into focus (useful if no mouse is in use).

Advanced Poll: Brings the Advanced Poll dialog into focus (useful if no mouse is present).

Keypad Entry: Brings the Keypad Entry dialog into focus (useful if no mouse is present).

Advanced Entry: Brings Advanced Entry dialog into focus (useful if no mouse is present).

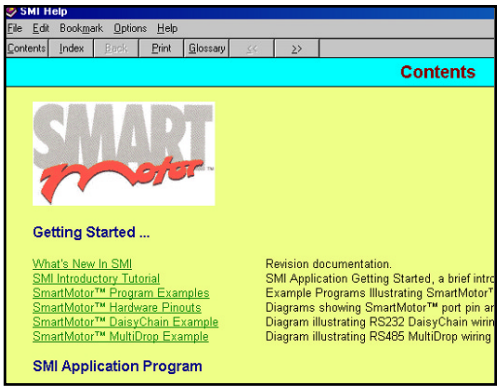
Help Menu Commands



***Help** drop-down menu*

Help Topics: Displays the **SMI** application help contents page.

Optional Message Boxes: The **SMI** application displays some message dialogs that the user may select not to show again. To allow the user to reset all option message dialogs to reappear, this command displays the **Reset All Message Dialogs** to open the window.



***Help** topics*

About SMI: Displays the About SMI dialog indicating the SMI version number.

Terminal Macros: Displays the Terminal Macros dialog window, a quick macro functions reminder.

TOOLBAR COMMANDS

The following table summarizes the functions of the ***Toolbar*** buttons:

Toolbar button description table

Toolbar button	Corresponding *Menu bar* command	Description
	File -> New	Opens a new editing window.
	File -> Open	Opens a previously saveed program source file (*.sms) or terminal file (*.mon) in an editing window.
	File -> Save	Saves the currently active edit window to a named file.
	Edit -> Cut	Deletes the highlighted data from the currently active document and moves it to the windows clipboard.
	Edit -> Copy	Copies the highlighted data from the currently active document to the Windows clipboard.
	Edit -> Paste	Pastes Windows clipboard data into the currently active document at current cursor position.
	File -> Print	Prints contents of currently active window.
	Edit -> Go To Top Of File	Places the cursor at the beginning of the currently active source file document.
	Edit -> Go To End Of File	Places the cursor at the end of the currently active source file document.
	Edit -> Make SmartMotor Executable	Scans the program for errors and creates the compiled and uncompiled files.
	Edit -> Find Previous Error	Places cursor at previous error in currently active source window.
	Edit -> Find Next Error	Places cursor at next error in currently active source window.
	File -> Transmit Program To Motor	Scans the program and if no errors were found downloads it to the motor.
	Communicate -> Address Motor Chain	Addresses all motors connected to the serial port.
	none	Set the default SmartMotor.
	Communicate -> Send END	Transmits a Global “END” command to SmartMotor(s)
	Communicate -> Send RUN	Transmits a Global “RUN” command to SmartMotor(s)
	Communicate -> Monitor Status	Starts Monitor Status dialog box in polling mode.
	Communicate -> Advanced Status	Starts the “Advanced Polling” dialog box in polling mode.
	none	Transmit global “S” (gradual stop) command to SmartMotor(s)
	none	Transmit global “X” (decelerate to stop) command to SmartMotor(s)
	none	Transmit global “OFF” (rapid stop) command to SmartMotor(s)
	none	Activates context help pointer

TUNING UTILITY OVERVIEW

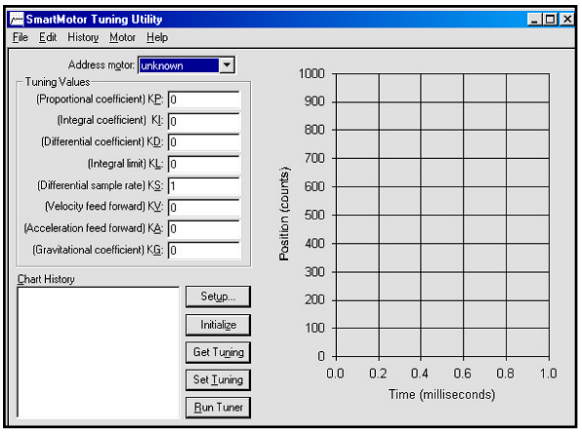
The SmartMotor tuning is not done with the traditional turning of pots on an amplifier. These physical components have been replaced with firmware and host level software. The Tuning Utility is the host level component that aids in the selection of the absolute best values for optimal performance for a given loading condition. With this utility the values of **KP**, **KI**, **KD**, **KL**, **KS**, **KV**, **KA**, and **KG** can be changed and the motor's response to a step change in target position can be seen. A later section titled "The PID Filter" describes the functions of the different terms and the optimization process in detail.

A QUICK TUTORIAL

This section is a quick guide for getting started using the **SmartMotor Tuning Utility** software.

Running the Tuner program

The **Tuning utility** was installed as part of the **SmartMotor Interface** software. To open the utility click on the Windows **Start** button and then click on **Programs**, **Animatics** and the **SmartMotor Tuning Utility**. The **Tuning Utility**'s main window, right, should now be on screen. The tuning utility can also be launched from the tools menu of the **SMI**.



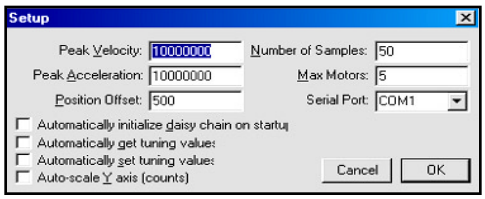
*The SmartMotor **Tuning Utility**'s main window*

Setting up the Tuner

Click on the **Setup…** button

OR select **Tuner Setup…** from the **Motor** drop-down menu.

The **Setup** dialog box should now be on screen, right.



The Setup dialog box

Make sure the following default values are set in the dialog box:

Peak Velocity: 10000000
(The Maximum Velocity)

Peak Acceleration: 10000000 (The Maximum Acceleration)

Position Offset: 500 (The relative position change)

Number of Samples: 50 (Total number of samples taken)

NOTE:

The Tuning Utility instantly rotates the motor's shaft at maximum allowed acceleration and velocity for a quarter of a turn, to an abrupt stop. This can cause the motor to shake enough to affect delicate equipment.

If the motor has an external Memory Module, remove it.

Select the number of motors in the daisy chain from the **Max Motors** edit box and the communication port that is connected to the motors and leave all other check box entries in the window unchecked. Click on "**OK**" to set these values for tuning.

Initializing a Daisy chain of motors

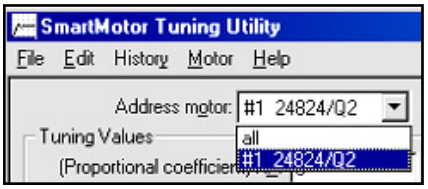
If the motor(s) have an external Memory Module, remove it and make sure all connections are OK.

Click on the **Initialize** button

OR select **Initialize Daisy Chain** from the **Motor** drop-down menu

Selecting the motor after initializing.

A small box shows that the program is initializing the motors in the daisy chain. After a few seconds the box will disappear and the number and version of the motor(s) should be visible in the **Address motor:** window. Select the motor from this combo box.

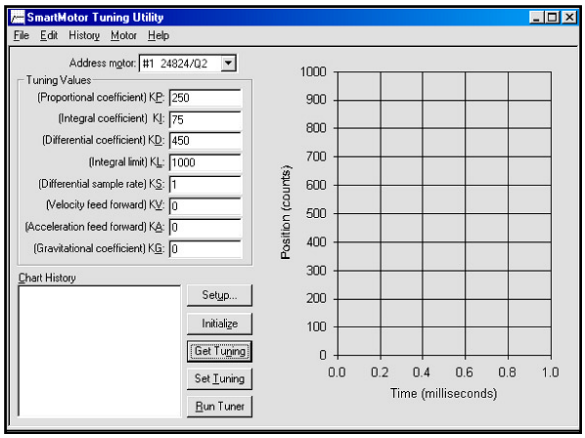


Getting the tuning values from a motor

Click on the **Get Tuning** button

OR select the **Get Tuning Values** from the **Motor** drop-down menu

Main window showing values read from motors.



This command asks the tuning values that are currently set in the motor. After a few seconds a message box appears showing the successful reading of values.

Setting the tuning values

Make sure that the values read from the motor are the motor's default values or are the correct values intended to be used in an application. The preceding figure displays the defaults for a version 4.02 motor. Modify the tuning values if necessary.

After verifying the values:

Click on the **Set Tuning** button

OR select the **Set Tuning Values** from the **Motor** drop-down menu

After a few seconds a message box shows the successful setting of motor values.

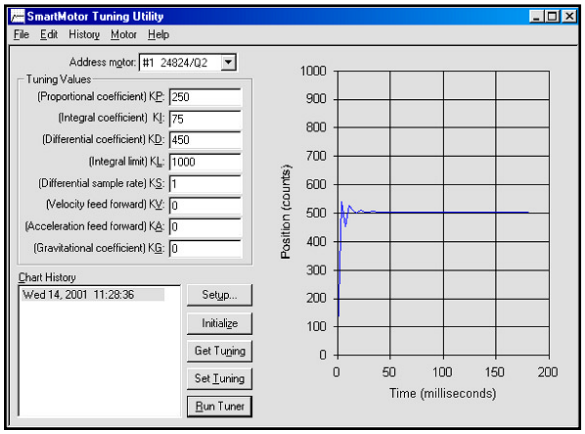
Running the Tuner

Now everything is ready to run the tuner.

Click on the **Run tuner** button

OR select the **Run tuner** from the **Motor** drop-down menu

A small box shows the program is processing the command and after a few seconds, the motor shaft will rotate rapidly and stop. After that, a record is added in the **Chart History** window, showing the date and time of operation and the results of the operation are shown in the chart on the right side of main window.



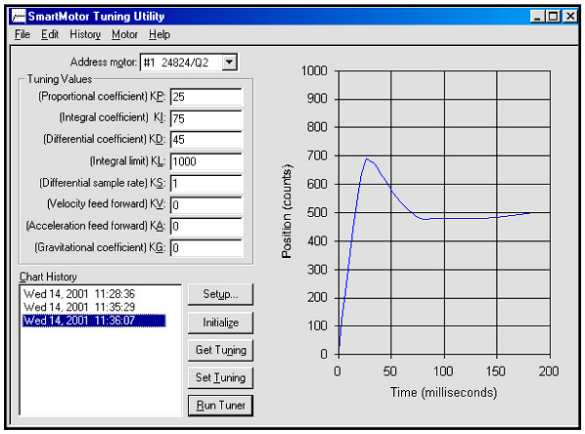
The resulting chart after running the tuner.

*See the **PID Filter** section for a greater understanding on optimizing the SmartMotor's tuning.*

Modifying the tuning values

Try other tuning values and setup parameters and compare the results.

Note: The **Set Tuning** button must be clicked each time the tuner is run. The figure, right, shows the result of running the tuner with a lowered **KP** and **KD**.



Running tuner with modified tuning values.

The date and time in the **Chart History** window is a default label that can be renamed by:

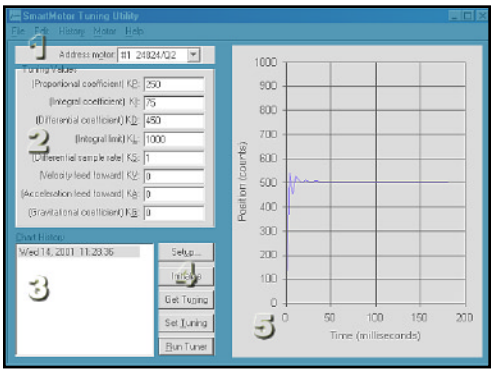
Clicking on a selected label

OR selecting **Edit** from the **History** pull-down menu and editing the label.

SMARTMOTOR TUNING UTILITY WINDOWS

There are five main elements of the **SmartMotor Tuning Utility** window.

The five elements of the "SmartMotor Tuning Utility".



1. **Address motor:** This window is updated after initialization of a single motor or a daisy chain of motors. There will be a line for each motor that can be accessed by clicking on the small down arrow at the end of the window. The data for each motor will show the sequence number, sample rate and version number. Highlight the motor needed and the tuner's commands will be sent to it.

*Note: The **PID** filter parameters are held in registers until an **F** command is received. Then they become active within the same servo cycle.*

2. **Tuning Values:** The Tuning values are 8 windows with data that defines the characteristics of a motor in response to different input data.

   (Proportional coefficient) **KP**: This is the gain of the proportional element in the **PID** filter. The higher the value of **KP**, the stiffer the motor will be.

   (Integral coefficient) **KI**: The integral compensation gain of the PID filter. The integral term of the **PID** filter creates a torque that is a function of both error and time. If the position error remains nonzero, over time, the torque becomes ever larger to enable the motor's shaft to reach its target.

   (Differential coefficient) **KD**: The derivative element of the **PID** filter. It can be thought of as the vibration-absorbing element.

   (Integral limit) **KL**: The integral limit of the **PID** filter. This value provides a limit to the amount of torque the **KI** term can produce given a non closing position error.

   (Differential sample rate) **KS**: This value represents the number of sample periods between evaluation of the **KD** parameter.

   (Velocity feed forward) **KV**: This value compensates for the predictable natural latency of the filter as it's influence grows with speed.

   (Acceleration feed forward) **KA**: This value compensates for the predictable forces due purely to acceleration and deceleration.

   (Gravitational coefficient) **KG**: This value compensates for the predictable force due to gravity in a vertical application.

***SmartMotor Tuning Utility** data can be saved with the file extension **.tnh**. Reopen the files in the **SmartMotor Tuning Utility** to view previously saved results.*

*Peak Velocity, Peak Acceleration, Position Offset, Number Of Samples, the Auto-scale option and the **PID** tuning values are all saved.*

3. **Chart History:** This window shows all of the tuning operations performed on the motor. Move through the list of motors (in step one) and see each tuning chart. Any item selected from this list can also be edited or deleted.

4. **Shortcut buttons:** Each one of these buttons has a corresponding sub-item in the "Motor" pull-down menu. The menu items operations are described in the following section.

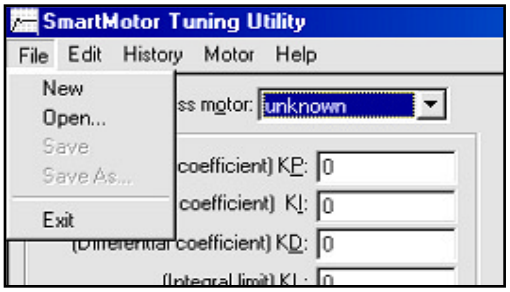
5. **Tuning Chart:** This graph displays the "SmartMotor Tuning Utility" data for the selected SmartMotor. The horizontal axis is time in milliseconds and the vertical axis is position in counts.

SMARTMOTOR TUNING UTILITY MENUS

This section describes all of the operations performed by the **SmartMotor Tuning Utility.**

File, pull-down menu

The file pull-down menu (right) has 5 sub-items that are described here.



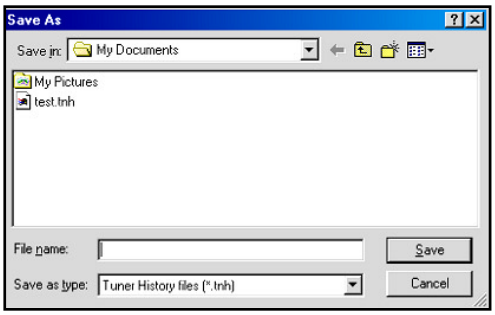
The File menu sub-items

New: Opens a new document. The document can be saved with a new file name.

Open: Opens an existing SmartMotor file. Select any file with **.tnh** extension saved in the **SmartMotor Tuning Utility**. All tuning values and tuning parameters are updated with the values stored in the file.

Save: Updates the data in the original saved file. If the data has not been saved (a new file), the **Save As** window (below) will open.

Save As: Opens the **Save As** window. Enter a new file name and save the **SmartMotor Tuning Utility** data to a new file.



*The **Save As** dialog box.*

Exit: Closes the **SmartMotor Tuning Utility**. If the file was modified since the last time it was saved, **Exit** will ask if the file should be saved.

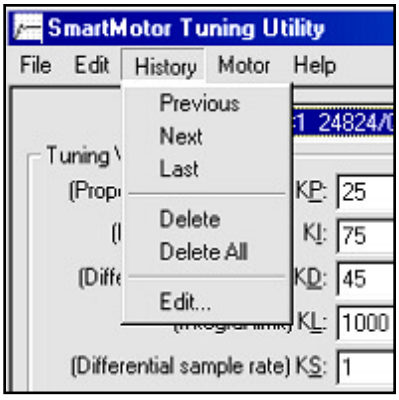
Edit pull-down menu

Copy: Will copy the data highlighted to the Windows clipboard. The data can then be pasted into another application such as a spreadsheet. It is also possible to paste a picture of the graph by using the **Paste Special** menu item in the destination application and selecting the Picture or Bit map format. The Picture format is scalable, while the Bit map format is not.

Use the SmartMotor version number (or other identification) in the filename as a reminder about which motor was used.

The data in the first column of numbers copied to the clipboard are the number of milliseconds since the start of the step motion. The second column contains the actual position values of the shaft at the corresponding times.

History pull-down menu



History menu sub-items.

The commands in this menu are related to the history of charts that are stored in the program.

Previous

This command selects the chart that is immediately before the current chart in the "Chart History" list box and shows it on the right side of main window.

Next

This command selects the chart that is immediately after the current chart in the "Chart History" list box and shows it on the right side of the main window.

Last

This command selects the last chart in the "Chart History" list box and shows it on the right side of the main window.

Delete

This command deletes the currently selected chart in the chart history list box.

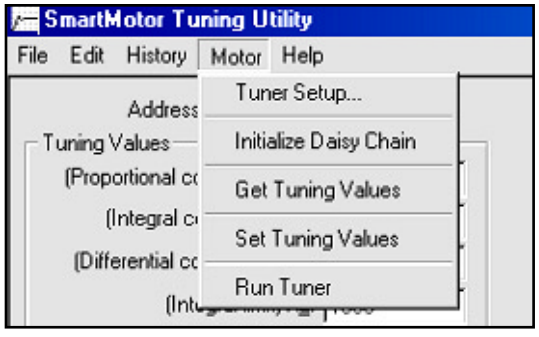
Delete All

This command deletes all of the charts in the chart history.

Edit

This command allows the user to edit the **Chart History** label. This can also be accomplished by clicking on a selected item in the list. By default, the chart label indicates the date and time the chart was created. Click on a selected chart history label and edit the label to indicate any specific notes or comments about that chart. Click on **Enter** to complete editing or **Escape** to cancel editing.

Motor drop-down menu



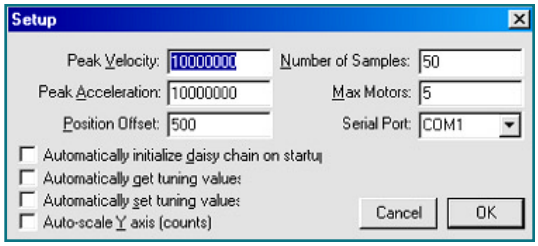
***SmartMotor Tuning Utility*/ *Motor* drop-down menu**

This menu (right) contains sub-items that perform the main actions of the **SmartMotor Tuning Utility**.

Tuner Setup...

This command invokes the **Setup** dialog box (top of facing page).

The parameters in this dialog box determine the data fed to motors for tuning. These values are as follows:

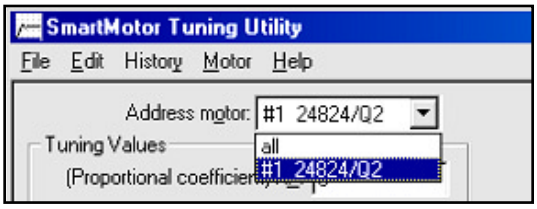


Setup dialog box.

Peak Velocity: is the target trajectory velocity that will be used in the step motion. The default value is 10 million.

Peak Acceleration: is the desired acceleration at which to rotate the shaft during the step motion. The default is 10 million. Due to limited resolution, odd numbers are rounded down to the next even integer.

Position Offset: specifies the target trajectory position. Note that nothing will happen if this value is too small. The default value is 500.



Sequence number, sample rate and version number identifiers in the combo box.

Number of Samples: is the number of times to repeat a loop that reads the position values. The amount of time spent in polling the position will vary with the speed of the computer. Currently, the minimum period between readings is about 3.3 milliseconds, sufficient for accurate rendering of the shaft position. The maximum limit is 1,000 samples, where the position polling will last more than 3 seconds at that value.

Max Motors: specifies the maximum number of motors to try and address before giving up. The larger the value, the longer the delay before the Tuner detects that the chain is not communicating or that there are no motors attached. It is not advisable to run the tuner in a daisy chain, because each motor adds a small amount of propagation delay (about 1-2ms). It also takes time to initialize a long chain.

Serial Port: where the motors are connected (COM1 or COM2).

Automatically initialize daisy chain on startup: This setting is only for convenience and generally should not be checked. It will cause the **Tuning Utility** to try to initialize a chain before loading its main window, which will introduce a small delay at startup. Use this option only if a motor will be connected before executing the **Tuner** application.

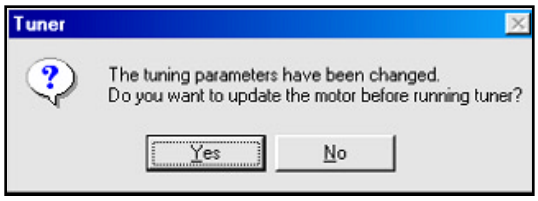
Window confirming tuning values have been successfully retrieved.

Automatically get tuning values: If this box is checked, the Tuner will retrieve the tuning values of the default motor immediately after initializing a daisy chain as well as every time a different Default Motor is selected in the pull-down box.

Automatically set tuning values: Enabling this setting will cause the tuning values entered in the dialog box to be sent to the Default Motor before running the tuner. Otherwise, if any value is modified before running the tuner and without setting those values in the motor, there will be prompt to update the tuning

Window confirming tuning values have been successfully set.

Window confirming motor update.



values in the motor with the newly edited values.

Auto-scale Y axis: The default scaling of the Y (position) axis is so that the target trajectory position offset is in the center of the axis. If there are any values that fall outside this preset range, turn this option on and the chart will auto-scale to fit the entire lower and upper bounds of the chart.

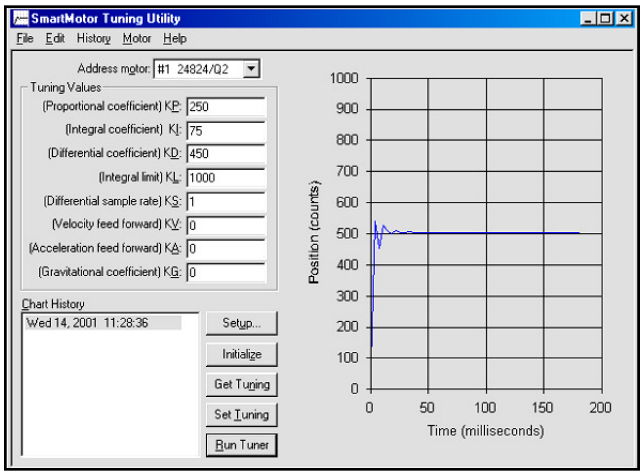
Initialize Daisy Chain: In order to use the Tuning Utility with more than one SmartMotor in a daisy chain, the daisy chain must first be initialized by selecting this menu item or pressing the "Initialize" button. With one motor, there's no need to initialize a daisy chain. The Address Motor combo box clears for a moment and then becomes filled with the sequence number, sample rate and version number of each motor in the daisy chain as shown.

Get Tuning Values: To examine the **PID** values stored in the motor, select this menu item or click on the **Get Tuning** button. This will display the tuning values stored in the currently addressed motor. A dialog box pops up to confirm that the tuning values have been successfully retrieved.

Set Tuning Values: Set the tuning values of the currently addressed motor by typing in new values for the **PID** terms and selecting this menu item or pressing the **Set Tuning** button. A confirmation message box pops up to indicate that the tuning values have been successfully set and the **F** command has been issued to load the values into active registers.

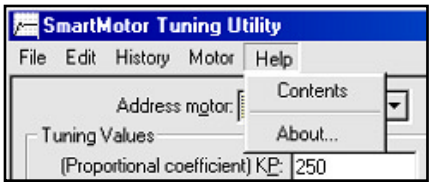
The resulting chart of running the tuner.

Run Tuner: Select this menu item or click on the **Run Tuner** button to run the motor and see the results. Depending on the tuning parameters, with no load attached to the motor, it may simply vibrate a little before settling down. A chart will plot, originating from the bottom left corner if the window and then oscillating up and down as shown here.



Help Menu:

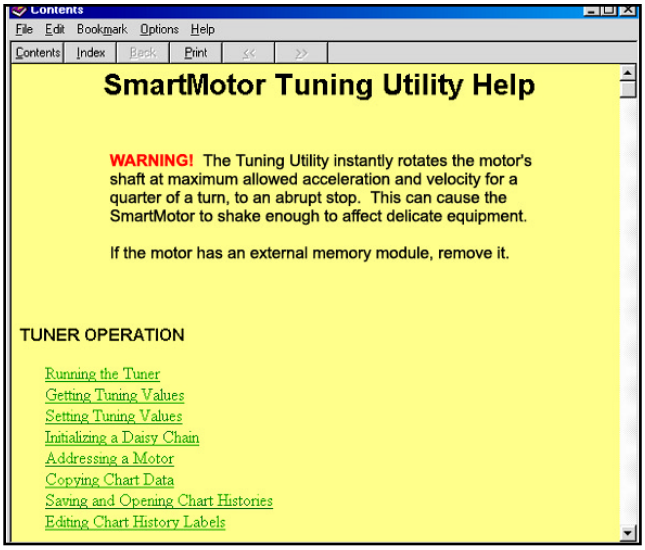
The help menu has two sub-items.



Help menu.

Contents: This brings a dialog box showing the contents for the Smart Motor Tuning Utility help. All features of this program are described in this help system.

About…: This window gives information about the version and copyright.



The contents of SmartMotor Tuning Utility help.

PROGRAMMING TABLE OF CONTENTS

Enter the four commands below in the **SmartMotor Terminal** window, following each command with a return, and the SmartMotor will start to move:

Commands	Comments
A=100	'Set Maximum Acceleration
V=1000000	'Set Maximum Velocity
P=1000000	'Set Absolute Position
G	'Start move (Go)

A complete move requires the user to set a Position, a Velocity and an Acceleration, followed by a Go.

On power-up the motor defaults to position mode. Once **Acceleration** (**A**) and **Velocity** (**V**) are set, simply issue new **Position** (**P**) commands, followed by a **G** (**Go**) command to execute moves to new absolute locations. The motor does not instantly go to the programmed position, but follows a trajectory to get there. The trajectory is bound by the maximum **Velocity** and **Acceleration** parameters. The result is a trapezoidal velocity profile, or a triangular profile if the maximum velocity is never met.

Position, **Velocity** and **Acceleration** can be changed at any time during or between moves. The new parameters will only apply when a new **G** command is sent.

All SmartMotor commands are grouped by function, with the following notations:

#	Integer number
exp	Expression or signed integer
var	Variable
COM	Communication channel

A=exp Set absolute acceleration

Acceleration must be a positive integer within the range of 0 to 2,147,483,648. The default is zero forcing something to be entered to get motion. A typical value is 100. If left unchanged, while the motor is moving, this value will not only determine acceleration but also deceleration which will form a triangular or trapezoidal velocity motion profile. This value can be changed at any time. The value set does not get acted upon until the next **G** command is sent.

For SM17 & SM23
A=rev/sec^2 * 7.91

For SM34, 42 & 56
A=rev/sec^2 * 15.82

If the motor has a 2000 count encoder (sizes 17 and 23), multiply the desired acceleration, in rev/sec^2, by 7.91 to arrive at the number to set **A** to. With a 4000 count encoder (sizes 34, 42 and 56) the multiplier is 15.82. These constants are a function of the motors **PID** rate. If the **PID** rate is lowered, these constants must raised proportionally.

V=exp Set maximum permitted velocity

For SM17 & SM23
*V=rev/sec * 32212*

For SM34, 42 & 56
*V=rev/sec * 64424*

Use the **V** command to set a limit on the velocity the motor can accelerate to. That limit becomes the slew rate for all trajectory based motion whether in position mode or velocity mode. The value defaults to zero so it must be set before any motion can take place. The new value does not take effect until the next **G** command is issued. If the motor has a 2000 count encoder (sizes 17 and 23), multiply the desired velocity in rev/sec by 32212 to arrive at the number to set **V** to. With a 4000 count encoder (sizes 34, 42 & 56) the multiplier is 64424. These constants are a function of the motors **PID** rate. If the **PID** rate is lowered, these constants will need to be raised.

P=exp Set absolute position for move

For SM17 & SM23
*P=rev * 2000*

For SM34, 42 & 56
*P=rev * 4000*

The **P=** command sets an absolute end position. The units are encoder counts and can be positive or negative. The end position can be set or changed at any time during or at the end of previous moves. SmartMotor sizes 17 and 23 resolve 2000 increments per revolution while SmartMotor sizes 34, 42 and 56 resolve 4000 increments per revolution.

The following program illustrates how variables can be used to set motion values to real-world units and have the working values scaled for motor units for a size 17 or 23 SmartMotor.

```
a=100           'Acceleration in rev/sec*sec
v=1             'Velocity in rev/sec
p=100           'Position in revs
GOSUB10         'Initiate motion
END             'End program
C10             'Motion routine
   A=a*8        'Set Acceleration
   V=v*32212    'Set Velocity
   P=p*2000     'Set Position
   G            'Start move
RETURN          'Return to call
```

D=exp Set relative distance for position move

The **D=** command allows a relative distance to be specified, instead of an absolute position. The number following is encoder counts and can be positive or negative.

The relative distance will be added to the current position, either during or after a move. It is added to the desired position rather than the actual position so as to avoid the accumulation of small errors due to the fact that any servo motor is seldom exactly where it should be at any instant in time.

G Go, start motion

G also resets several system state flags

The **G** command does more than just start motion. It can be used dynamically during motion to create elaborate profiles. Since the SmartMotor allows position, velocity and acceleration to change during motion, “on-the-fly”, the **G** command can be used to trigger the next profile at any time.

S Abruptly stop motion in progress

If the **S** command is issued while a move is in progress it will cause an immediate and abrupt stop with all the force the motor has to offer. After the stop, assuming there is no position error, the motor will still be servoing. The **S** command works in both Position and Velocity modes.

X Decelerate to stop

If the **X** command is issued while a move is in progress it will cause the motor to decelerate to a stop at the last entered **A=** value. When the motor comes to rest it will servo in place until commanded to move again. The **X** command works in both Position and Velocity modes.

O=exp Set/Reset origin to any position

The **O=** command (using the letter **O**, not the number zero) allows the host or program not just to declare the current position zero, but to declare it to be any position, positive or negative. The exact position to be re-declared is the ideal position, not the actual position which may be changing slightly due to hunting or shaft loading. The **O=** command directly changes the motor's position register and can be used as a tool to avoid +/- 31 bit roll over position mode problems. If the SmartMotor runs in one direction for a very long time it will reach position +/-2,147,483,648 which will cause the position counter to change sign. While that is not an issue with Velocity Mode, it can create problems in position mode.

OFF Turn motor servo off

The **OFF** command will stop the motor from servoing, much as a position error or limit fault would. When the servo is turned off, one of the status LEDs will revert from Green to Red.

MP Position Mode

Position mode is the default mode of operation for the SmartMotor. If the mode were to be changed, the **MP** command would put it back into position mode. In position mode, the **P#** and **D#** commands will govern motion.

BINARY POSITION DATA TRANSFER

The **ASCII** based command string format, while convenient, is not the fastest way to communicate data. It can be burdensome when trajectory commands are sent to the motor. For that reason a special binary format has been established for the communication of trajectory critical data such as **Position**, **Velocity** and **Acceleration**. Using the binary format, these 32 bit parameters are sent as four bytes following a code byte that flags the data for a particular purpose. The code bytes are 252 for acceleration, 253 for velocity and 254 for position. As an example, the following byte values communicate A=53, V=-1 & P=2137483648.

A=53	252 000 000 000 053 032
V=-1	253 255 255 255 254 032
P=2137483648	254 127 255 255 255 032

For further expediency, the commands can be appended with the **G** command to start motion immediately. Two examples are as follows (the ASCII value for **G** is 71):

P=0 G	254 000 000 000 000 071 032
V=512 G	253 000 000 002 000 071 032

MV Velocity Mode

Velocity mode will allow continuous rotation of the motor shaft. In Velocity mode, the programmed position using the **P** or the **D** commands is ignored. Acceleration and velocity need to be specified using the **A=** and the **V=** commands. After a **G** command is issued, the motor will accelerate up to the programmed velocity and continue at that velocity indefinitely. In velocity mode as in Position mode, Velocity and Acceleration are changeable on-the-fly, at any time. Simply specify new values and enter another **G** command to trigger the change. In Velocity mode the velocity can be entered as a negative number, unlike in Position mode where the location of the target position determines velocity direction or sign. If the 32 bit register that holds position rolls over in velocity mode it will have no effect on the motion.

MT Torque Mode

In torque mode the motor shaft will simply apply a torque independent of position. The internal encoder tracking will still take place, and can be read by a host or program, but the value will be ignored for motion because the **PID** loop is inactive. To specify the amount of torque, use the **T=** command, followed by a number between -1023 and 1023.

T=exp Set torque value, -1023 to 1023

In torque mode, activated by the **MT** command, the drive duty cycle can be

set with the **T=** command. The following number or variable must fall in the range between -1023 and 1023. The full scale value relates to full scale or maximum duty cycle. At a given speed there will be reasonable correlation between drive duty cycle and torque. With nothing loading the shaft, the **T=** command will dictate open-loop speed.

MD Contouring Mode (requires host)

Contouring Mode is the foundation of Animatics' G-Code interface that enables a P.C. and multiple SmartMotors to interpret G-Code files and do multiple axis contouring.

SmartMotors with version 4.15 or greater firmware have the added ability to do multiple axis contouring. This firmware version became standard roughly mid-year 2001. The **Contouring Mode** is the foundation of the Animatics' G-Code interface that enables a P.C. and multiple SmartMotors to interpret G-Code files and do linear, circular and helical interpolation as well as unlimited multi-axis contouring.

The basic principle of operation takes advantage of the fact that each SmartMotor has a very accurate time base. Absolute position-time pairs of data get sent to the SmartMotor to fill buffers that facilitate continuous motion. The SmartMotor will adjust its own Velocity and Acceleration to be certain to arrive at the specified position at the exact specified time without slowing to a stop. As new position-time pairs arrive, the motor transitions smoothly from one profile to the next producing smooth, continuous motion. In a multiple axis configuration, different positions can be sent to different motors, with the same time intervals resulting in smooth, continuous multiple axis motion. The key is for the host to regulate the volume of data in each of the different motor's buffers. The position-time pairs of data are preceded with an identification byte and then four bytes for position and four for time. Time is in units of servo samples and is limited to 23 bits. Time is further constrained to be even powers of 2 (i.e. 1, 2, 4, 8, ..., 32768).

The coordinating host can send the **Q** command to solicit status information on the coordination process. Upon receiving the **Q** command, the SmartMotor will return status, clock and space available in the dedicated circular buffer. The response to **Q** takes two forms, one while the mode is running with trajectory in progress and no errors having occurred and another when the mode is not running. Both responses conform to the overall byte format of:

Q Response: 249 byte1 byte2 byte3 byte4

If the mode is running:

byte1 bit 7 is set
byte1 bits 6 through 0 return data slots available
bytes 2, 3 & 4 return the 24 bit clock of the SmartMotor

If the mode is not running:

byte1 bit 7 is clear
byte1 bits 6 through 0 return status
byte2 returns space available
bytes 3 & 4 return the 12 lower bits of the 24 bit clock of the SmartMotor

As absolute position and time data is sent to the SmartMotor, differences are calculated that we call "deltas". A delta is the difference between the latest value and the one just prior. Time deltas are limited to 16 bits while Position deltas are limited to 23 bits in size.

The Status Byte is constructed as follows:

bit0=1	MD mode pending a G
bit1=1	MD mode actually running
bit2=1	Invalid time delta > 16 bit received
bit3=1	Invalid position delta > 23 bits received
bit4=1	Internal program data space error
bit5=1	Host sent too much data (data buffer overflow)
bit6=1	Host sent too little data (data buffer underflow)

A trajectory terminates if an unacceptable position error occurs, if invalid data is received, if there is a data overflow or if there is a data underflow.

The host should send data pairs only when at least 3 empty data slots are available. **MD** responds to limit switches with an aborted trajectory. The **MD** mode uses KV feed forward for improved performance.

The byte flag that precedes and marks a position is of decimal value 250. The byte flag that precedes and marks a time is of decimal value 251.

The following is an example of the decimal byte values for a series of constant speed motion segments. Firmware versions 4.16 and higher do not need time values after the first two if the time delta is not changing. The byte transfers terminate with a carriage return (13).

Position	250 000 000 000 000 013	Position = 0
Time	251 000 000 000 000 013	Time = 0
Position	250 000 000 016 000 013	
Time	251 000 000 001 000 013	Time delta = 256
Position	250 000 000 032 000 013	
Position	250 000 000 048 000 013	
Position	250 000 000 052 000 013	Reduce position delta
Time	251 000 000 003 064 013	Reduce time delta
Position	250 000 000 056 000 013	
Position	250 000 000 060 000 013	
Position	250 000 000 064 000 013	

What is not shown in the these codes are the addressing bytes that would be used to differentiate multiple motors on a network. As described ahead in this manual (see the **SADDR** command), a network of SmartMotors can

be sorted out by sending a single address byte. When communicating to a particular motor, the address byte need only be sent once, until all of the communications to that particular motor are complete and another motor needs to be addressed. The byte patterns in the previous example would need to be preceded with an address byte (to a properly addressed motor) for multiple axis contouring. In the addressing scheme, there is a global address provision for sending data to all motors at once. By zeroing out the clocks before starting the contouring, the motors will by synchronized and single time values can then be sent to all motors at once, increasing overall bandwidth. Also, as mentioned earlier, SmartMotors with version 4.16 or higher do not need time data past the first two, if there is no change in the time delta.

The basis for contouring using this format is to keep the rate at which data is sent to each motor constant (and as fast as possible). That means that in order to accelerate axes, absolute positions need to be sent that invoke progressively larger position deltas, and to keep constant velocity, absolute positions need to be sent that are equidistant.

With all of the communications to send data and receive status, it would be outstanding to have a bandwidth on a two axis system of 64 samples, or 16ms. Typically, with a three or four axis system a bandwidth of 128 servo samples or 32ms is achievable. This would be at a baud rate of 38.4k. Keep in mind that during this time the SmartMotor is micro interpolating. The motion will be very smooth and continuous.

In contouring mode, all of the binary contouring data goes into the motor's buffers. While this is true, regular commands will still be recognized and they will operate normally. This will take some time, however, and it is up to the programmer to assure that the buffers never underflow due to neglect.

BRAKE COMMANDS
(where optional brake exists)

BRKRLS Brake release

BRKENG Brake engage

BRKSRV Release brake when servo active, engage break when inactive.

BRKTRJ Release brake when running a trajectory, engage under all other conditions. Turns servo off when the brake is engaged.

Many SmartMotors™ are available with power safe brakes. These brakes will apply a force to keep the shaft from rotating should the SmartMotor lose power. Issuing the **BRKRLS** command will release the brake and **BRKENG** will engage it. There are two other commands that initiate automated operating modes for the brake. The command **BRKSRV** engages the brake automatically, should the motor stop servoing and holding position for any reason. This might be due to loss of power or just a position error, limit fault, over-temperature fault.

Finally, the **BRKTRJ** command will engage the brake in response to all of the previously mentioned events, plus any time the motor is not performing a trajectory. In this mode the motor will be off, and the brake will be holding it in position, perfectly still, rather than the motor servoing when it is at rest. As soon as another trajectory is started, the brake will release. The time it takes for the brake to engage and release is on the order of only a few milliseconds.

The brakes used in SmartMotors™ are zero-backlash devices with extremely long life spans. It is well within their capabilities to operate interactively within an application. Care should be taken not to create a situation where the brake will be set repeatedly during motion. That will reduce the brake's life.

Program commands are like chores, whether it is to turn on an output, set a velocity or start a move. A program is a list of these chores. When a programmed SmartMotor is powered-up or its program is reset with the **Z** command, it will execute its program from top to bottom, with or without a host P.C. Connected. This section covers the commands that control the program itself.

RUN Execute stored user program

If the SmartMotor is reset with a **Z** command, all previous variables and mode changes will be erased for a fresh start and the program will begin to execute from the top. Alternatively the **RUN** command can be used to start the program, in which case the state of the motor is unchanged and its program will be invoked.

RUN? Halt program if no RUN issued

To keep a downloaded program from executing at power-up start the program with the **RUN?** Command. It will prevent the program from starting when power is applied, but it will not prevent the program from running when the SmartMotor sees a **RUN** command from a host over the RS-232 port.

Once the program is running, there are a variety of commands that can redirect program flow and most of those can do so based on certain conditions. How these conditional decisions are setup determines what the programmed SmartMotor will do, and exactly how “smart” it will actually be.

GOTO# Redirect program flow

C# Subroutine label, C0-C999

The most basic commands for redirecting program flow, without inherent conditions, are **GOTO#** in conjunction with **C#**. Labels are the letter **C** followed by a number (**#**) between 0 and 999 and are inserted in the program as place markers. If a label, **C1** for example, is placed in a program and that same number is placed at the end of a **GOTO** command, **GOTO1**, the program flow will be redirected to label **C1** and the program will proceed from there.

As many as a thousand labels can be used in a program (0 - 999), but, the more **GOTO** commands used, the harder the code will be to debug or read. Try using only one and use it to create the infinite loop necessary to keep the program running indefinitely, as some embedded programs do. Put a **C1** label near the beginning of the program, but after the initialization code and a **GOTO1** at the end and every time the **GOTO1** is reached the program will loop back to label **C1** and start over from that point until the **GOTO1** is reached, again, which will start the process at

C1 again, and so on. This will make the program run continuously without ending. Any program can be written with only one **GOTO**. It might be a little harder, but it will tend to force better program organization, which in turn, will make it easier to be read and changed.

END End program execution

If it's necessary to stop a program, use an **END** command and execution will stop at that point. An **END** command can also be sent by the host to intervene and stop a program running within the motor. The SmartMotor program is never erased until a new program is downloaded. To erase the program in a SmartMotor, download only the **END** command as if it were a new program and that's the only command that will be left on the SmartMotor until a new program is downloaded. To compile properly, every program needs and **END** somewhere, even if it is never reached. If the program needs to run continuously, the **END** statement has to be outside the main loop.

GOSUB# Execute a subroutine

RETURN Return from subroutine

Just like the **GOTO#** command, the **GOSUB#** command, in conjunction with a **C#** label, will redirect program execution to the location of the label. But, unlike the **GOTO#** command, the **C#** label needs a **RETURN** command to return the program execution to the location of the **GOSUB#** command that initiated the redirection. There may be many sections of a program that need to perform the same basic group of commands. By encapsulating these commands between a **C#** label and a **RETURN**, they may be called any time from anywhere with a **GOSUB#**, rather than being repeated in their totality over and over again. There can be as many as one thousand different subroutines (0 - 999) and they can be accessed as many times as the application requires.

Calling subroutines from the host can crash the stack

By pulling sections of code out of a main loop and encapsulating them into subroutines, the main code can also be easier to read. Organizing code into multiple subroutines is a good practice.

The commands that can conditionally direct program flow to different areas use a constant **[#]** like 1 or 25, a variable like **a** or **al[#]** or a function involving constants and/or variables **a+b** or **a/[#]**. Only one operator can be used in a function. The following is a list of the operators:

- **+** Addition
- **-** Subtraction
- ***** Multiplication
- **/** Division
- **==** Equals (use two =)

!= Not equal

< Less than

> Greater than

<= Less than or equal

>= Greater than or equal

& Bit wise AND (see appendix A)

| Bit wise OR (see appendix A)

WHILE, LOOP

The most basic looping function is a **WHILE** command. The **WHILE** is followed by an expression that determines whether the code between the **WHILE** and the following **LOOP** command will execute or be passed over. While the expression is true, the code will execute. An expression is true when it is non-zero. If the expression results in a "zero" then it is false. The following are valid **WHILE** structures:

```
WHILE 1      '1 is always true
   UA=1      'Set output to 1
   UA=0      'Set output to 0
LOOP         'Will loop forever

a=1          'Initialize variable 'a'
WHILE a      'Starts out true
   a=0       'Set 'a' to 0
LOOP         'This never loops back

a=0          'Initialize variable 'a'
WHILE a<10   'a starts less
   a=a+1     'a grows by 1
LOOP         'Will loop back 10x
```

The task or tasks within the **WHILE** loop will execute as long as the function remains true.

The **BREAK** command can be used to break out of a **WHILE** loop, although that somewhat compromises the elegance of a **WHILE** statement's single test point, making the code a little harder to follow. The **BREAK** command should be used sparingly or preferably not at all in the context of a **WHILE**.

If it's necessary for a portion of code to execute only once based on a certain condition then use the **IF** command.

IF, ENDIF

Once the execution of the code reaches the **IF** command, the code between that **IF** and the following **ENDIF** will execute only when the condition directly following the **IF** command is true. For example:

```
a=UAI          'Variable 'a' set 0,1
a=a+UBI        'Variable 'a' 0,1,2
IF  a==1       'Use double = test
    b=1        'Set 'b' to one
ENDIF          'End IF
```

Variable **b** will only get set to one if variable **a** is equal to one. If **a** is not equal to one, then the program will continue to execute using the command following the **ENDIF** command.

Notice also that the SmartMotor language uses a single equal sign (**=**) to make an assignment, such as where variable **a** is set to equal the logical state of input **A**. Alternatively, a double equal (**==**) is used as a test, to query whether **a** is equal to 1 without making any change to **a**. These are two different functions. Having two different syntaxes has farther reaching benefits.

ELSE, ELSEIF

The **ELSE** and **ELSEIF** commands can be used to add flexibility to the **IF** statement. If it were necessary to execute different code for each possible state of variable **a**, the program could be written as follows:

```
a=UAI          'Variable 'a' set 0,1
a=a+UBI        'Variable 'a' 0,1,2
IF a==0        'Use double '=' test
   b=1         'Set 'b' to one
ELSEIF a==1
   c=1         'Set 'c' to one
ELSEIF a==2
   c=2         'Set 'c' to two
ELSE           'If not 0 or 1
   d=1         'Set 'd' to one
ENDIF          'End IF
```

There can be many **ELSEIF** statements, but at most one **ELSE**. If the **ELSE** is used, it needs to be the last statement in the structure before the **ENDIF**. There can also be **IF** structures inside **IF** structures. That's called “nesting” and there is no practical limit to the number of structures that can nest within one another.

SWITCH, CASE, DEFAULT, BREAK, ENDS

Long, drawn out **IF** structures can be cumbersome, however, and burden the program visually. In these instances it can be better to use the **SWITCH** structure. The following code would accomplish the same thing as the previous program:

The SWITCH statement makes use of the same memory space as variable "zzz". Do not use this variable or array space when using SWITCH

```
a=UAI          'Variable 'a' set 0,1
a=a+UBI        'Variable 'a' 0,1,2
SWITCH a       'Begin SWITCH
  CASE 0
    b=1        'Set 'b' to one
    BREAK
  CASE 1
    c=1        'Set 'c' to one
    BREAK
  CASE 2
    c=2        'Set 'c' to two
    BREAK
  DEFAULT        'If not 0 or 1
    d=1        'Set 'd' to one
    BREAK
ENDS           'End SWITCH
```

Just as a rotary switch directs electricity, the **SWITCH** structure directs the flow of the program. The **BREAK** statement then jumps the code execution to the code following the associated **ENDS** command. The **DEFAULT** command covers every condition other than those listed. It is optional.

TWAIT Wait during trajectory

The **TWAIT** command pauses program execution while the motor is moving. Either the controlled end of a trajectory, or the abrupt end of a trajectory due to an error, will terminate the **TWAIT** waiting period. If there were a succession of move commands without this command, or similar waiting code between them, the commands would overtake each other because the program advances, even while moves are taking place. The following program has the same effect as the **TWAIT** command, but has the added virtue of allowing other things to be programmed during the wait, instead of just waiting. Such things would be inserted between the two commands.

```
WHILE Bt       'While trajectory
LOOP           'Loop back
```

WAIT=exp Wait (exp) sample periods

*For the exact sample period, use the **RSP** command*

There will probably be circumstances where the program execution needs to be paused for a specific period of time. Time, within the SmartMotor, is tracked in terms of servo sample periods. Unless otherwise programmed with the **PID#** command, the sample rate is about 4KHz. **WAIT=**4000 would wait about one second. **WAIT=**1000 would wait for about one quarter of a second. The following code would be the same as **WAIT=**1000, only it will allow code to execute during the wait if it is placed between the **WHILE** and the **LOOP**.

```
CLK=0              'Reset CLK to 0
WHILE CLK<1000     'CLK will grow
  IF UAI==0        'Monitor input A
    GOSUB911       'If input low
  ENDIF            'End the IF
LOOP               'Loop back
```

The above code example will check if port **A** ever goes low, while it is waiting for the **CLK** variable to count up to 1000.

STACK Reset the GOSUB return stack

The **STACK** is where information is held with regard to the nesting of subroutines (nesting is when one or more subroutines exist within others). In the event program flow is directed out of one or more nested subroutines, without executing the included **RETURN** commands, the stack will be corrupted. The **STACK** command resets the stack with zero recorded nesting. Use it with care and try to build the program without requiring the **STACK** command.

One possible use of the **STACK** command might be if the program used one or more nested subroutines and an emergency occurred, the program or operator could issue the **STACK** command and then a **GOTO** command which would send the program back to a label at the beginning. Using this method instead of a **RESET** command would retain the states of the variables and allow further specific action to resolve the emergency.

Variables are data holders that can be set and changed within the program or over the communication channel.

The first 26 variables are long integers (32 bits) and are accessed with the lower case letters of the alphabet, **a, b**, **c**, . . . **x**, **y**, **z**.

a=# Set variable **a** to a numerical value

a=exp Set variable **a** to value of an expression

A variable can be set to an expression with only one operator and two operands. The operators can be any of the following:

+ Addition

- Subtraction

***** Multiplication

/ Division

& Bit wise AND (see appendix A)

| Bit wise OR (see appendix A)

See Appendix C for a table describing User Assigned Variables.

The following are legal:

a=b+c, a=b+3 a=5+8

a=b-c a=5-c a=b-10

a=b*c a=3*5 a=c*3

a=b/c a=b/2 a=5/b

a=b&c a=b&8

a=b|c a=b|15

ARRAYS

In addition to the first 26, there are 52 more long integer variables accessible with double and triple lower case letters: **aa, bb, cc, . . . xxx, yyy, zzz**. The memory space that holds these 52 variables is more flexible, however. This same variable space can be accessed with an array variable type. An array variable is one that has a numeric index component that allows the numeric selection of which variable a program is to access. This memory space is further made flexible by the fact that it can hold 51 thirty two bit integers, or 101 sixteen bit integers, or 201 eight bit integers (all signed).

The array variables take the following form:

ab[i]=exp Set variable to a signed 8 bit value where index i = 0...200

aw[i]=exp Set variable to a signed 16 bit value where index i = 0...100

al[i]=exp Set variable to a signed 32 bit value where index i = 0...50

The index **i** may be a number, a variable **a** thorough **z**, or the sum or difference of any two variables **a** thorough **z** (variables only).

The same array space can be accessed with any combination of variable types. Just keep in mind how much space each variable takes. We can even go so far as to say that one type of variable can be written and another read from the same space. For example, if the first four eight bit integers are assigned as follows:

ab[0]=0

ab[1]=0

ab[2]=1

ab[3]=0

They would occupy the same space as the first single 32 bit number, and due to the way binary numbers work, would make the thirty two bit variable equal to 256. The order is most significant to least with ab[0] being the most.

A common use of the array variable type is to set up what is called a buffer. In many applications, the SmartMotor will be tasked with inputting data about an array of objects and to do processing on that data in the same order, but not necessarily at the same time. Under those circumstances it may be necessary to “buffer” or “store” that data while the SmartMotor processes it at the proper times.

To set up a buffer the programmer would allocate a block of memory to it, assign a variable to an input pointer and another to an output pointer. Both pointers would start out as zero and every time data was put into the buffer the input pointer would increment. Every time the data was used, the output buffer would likewise increment. Every time one of the pointers is incriminated, it would be checked for exceeding the allocated memory space and rolled back to zero in that event, where it would continue to increment as data came in. This is a first-in, first-out or “FIFO” circular buffer. Be sure there is enough memory allocated so that the input pointer never overruns the output pointer.

STORAGE OF VARIABLES

(Not available in SMXXX5 SmartMotors™)

Newer SmartMotors have 32K of non-volatile EEPROM memory to store variables when they need to survive the motor powering down.

EPTR=expression Set EEPROM pointer, 0-7999

To read or write into this memory space it is necessary to properly locate the pointer. This is accomplished by setting **EPTR** equal to the offset.

VST(variable,index) Store variables

To store a series of variables, use the **VST** command. In the "variable" space of the command put the name of the variable and in the "index" space put the total number of sequential variables that need to be stored. Enter a one if just the variable specified needs to be stored. The actual sizes of the variables will be recognized automatically.

VLD(variable,index) Load variables

To load variables, starting at the pointer, use the **VLD** command. In the "variable" space of the command put the name of the variable and in the "index" space put the number of sequential variables to be loaded.

*See **Appendix C** for a table describing **User Assigned Variables.***

FIXED OR PRE-ASSIGNED FUNCTIONS

In addition to the general purpose variables there are variables that are gateways into the different functions of the motor itself.

@P	Current position
@PE	Current position error
@V	Current velocity
ADDR	Motor's self address
CHN0	RS-232 com error flags
CHN1	RS-485 com error flags
CLK	Read/Write sample rate counter
CTR	External encoder count variable
I	Last recorded index position
LEN	# of characters in RS-232 buffer
LEN1	# of characters in RS-485 buffer

REPORT TO HOST COMMANDS

Ra...**Rzzz**	Report variables **a** ... **zzz,** 78 in all
Rab[i]	Report 8 bit variable value Rab[i]
Raw[i]	Report 16 bit variable value Raw[i]
Ral[i]	Report 32 bit variable value Ral[i]
RA	Report buffered acceleration
RAIN{port}{ch}	Report 8 bit analog input port=A-H, ch= 1-4
RAMPS	Report assigned maximum current
RBa	Report over current status bit
RBb	Report parity error status bit
RBc	Report communications error bit
RBd	Report user math overflow status bit
RBe	Report position error status bit
RBf	Report communications framing error status bit
RBk	Report EEPROM read/write status bit
RBl	Report historical left limit status bit
RBi	Report index status bit
RBh	Report overheat status bit
RBm	Report negative limit status bit
RBo	Report motor off status bit
RBp	Report positive limit status bit
RBr	Report historical right limit status bit
RBs	Report program scan status bit
RBt	Report trajectory status bit
RBu	Report user array index status bit
RBw	Report wrap around status bit
RBx	Report hardware index input level
RCHN	Report combined communications status bits
RCHN0	Report RS-232 communications status bits
RCHN1	Report RS-485 communications status bits
RCLK	Report clock value
RCTR	Report secondary counter
RCS	Report RS-232 communications check sum
RCS1	Report RS-485 communications check sum
RD	Report buffered move distance value

RDIN{port}{ch}	Report 8 bit digital input byte, port=A-H, and ch=0-63
RE	Report buffered maximum position error
RI	Report last stored index position
RKA	Report buffered acceleration feed forward coefficient
RKD	Report buffered derivative coefficient
RKG	Report buffered gravity coefficient
RKI	Report buffered integral coefficient
RKL	Report buffered integral limit value
RKP	Report buffered proportional coefficient
RKS	Report buffered sampling interval
RKV	Report buffered velocity feed forward coefficient
RP	Report measured position
RPE	Report present position error
RMODE	Report present positioning mode: **P** Absolute position move **R** Relative position move **V** Velocity move **T** Torque mode **F** Follow mode **S** Step and Direction mode **C** Cam Table mode **W** Drive mode **X** Follow mode with multiplier **E** Position error **O** Motor off **H** Contouring mode
RS2	Restore RS-232 mode
RS4	Assign **UG** to RS-485 control
RS	Report status byte (8 system states)
RSP	Report sample period and version number
RT	Report current requested torque
RV	Report velocity
RW	Report status word (16 system states)

*See **Appendix C** for a table describing **User Assigned Variables.***

Through the two pins, A and B of the I/O connector, quadrature or step and direction signals can be fed into the SmartMotor at high speeds and be followed by the motor itself. This feature brings about the following capabilities:

1 Mode Follow

2 Mode Step and Direction

3 Mode Follow with ratio

4 Mode Step and Direction with ratio

5 Mode Cam

SmartMotors SMXXX5 do not have Quadrature Encoder following capability built-in, but can be adapted to accept quadrature.

In addition to the above embedded modes of operation, the internal counter can be set to either count encoder signals or step signals and be accessible to the internal program or a host through the **CTR** variable.

When the SmartMotor is in one of the above five modes it may also run internal programs and communicate with a host, all at the same time.

MF1, MF2 and MF4 Mode Follow

Mode Follow allows the SmartMotor™ to follow an external encoder. Three resolutions can be selected through hardware, and a virtually infinite number of resolutions can be set in firmware using the MFR command described ahead. Set the hardware for maximum resolution with the **MF4** command. The **MF2** The **MF1** commands set the hardware to lesser resolutions, but are obsolete with the advent of the newer MFR capability.

MF0, MS0

The **MF0** and **MS0** commands must not be issued during one of the other follow modes. They are used for an entirely different purpose. If it is not desired to directly follow an incoming encoder or step signal, but rather, just to track them and use the counter value within a program or from a host, then issuing **MF0** or **MS0** utilizes the maximum resolution available and makes the value available through the **CTR** variable. Issuing **MF0** or **MS0** will zero that variable and incoming encoder or step signals will increment or decrement the signed, 32-bit **CTR** variable value.

MFDIV=expression **Set Ratio divisor**

MFMUL=expression **Set Ratio multiplier**
where -256.0000 < Ratio < 256.0000

After the appropriate **MF#** command is issued, or the **MS** command has been issued, a floating point ratio can further be applied by the firmware. Since the SmartMotor is an integer machine, that floating point ratio is accomplished by dividing one number by another.

MFR Calculate Mode Follow Ratio

MSR Calculate Mode Step Ratio

Once a numerator and denominator have been specified, and the appropriate hardware mode is selected, the motor can be put into ratio mode with the **MFR** or **MSR** commands (**MSR** for ratioing incoming step and direction signals). The following example sets up a 10.5:1 relationship:

```
MF4          'Read in full quadrature decode
MFMUL=2      '10.5:1=21:2
MFDIV=21
D=0          'be sure D is zero
MFR          'Invoke calculation
G            'Start
```

Once in a ratio mode the **V=#** and **D=#** commands will still work. They will invoke a phase shift of length **D** at a relative rate determined by **V**. For that reason, **D** must be zeroed out before issuing an **MFR** or **MSR** command or unexpected shifting could be taking place. In applications such as a Web Press, this ability to phase shift can be very useful.

MC Mode Cam

A cam is a basically round but irregular shape that rotates and causes a follower to move up and down in a profile determined by the shape of the cam's exterior.

Since the beginning of industrialization, cams have been used to create complex, reciprocating motion. Cams are most often carved out of steel and changing them, or having them invoke motion a great distance away are impractical. The SmartMotor provides an electronic alternative. Putting an encoder on the rotating part of a machine, sending the signals to a SmartMotor and programming the cam profile into the SmartMotor allows for the same complex, repeating motions to be accomplished without any of the typical mechanical limitations.

BASE=expression Base length

Part of defining a Cam relationship is specifying how many incoming encoder counts there are for one full cam rotation. Simply set **BASE** equal to this number.

SIZE=expression Number of Cam data entries

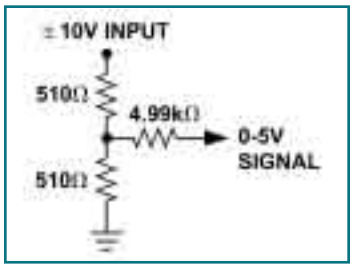
*Do not use variable **aa** through **zzz** while camming.*

The upper variable array space holds the cam profile data. To instruct the SmartMotor as to how many data points have been specified, set **SIZE** equal to that number. The cam firmware looks at words (16 bit numbers). The maximum number of words that can be used is 100. The cam firmware will perform linear interpolation between those entries, as well as between the last and the first as the cam progresses through the end of the table and back to the beginning. The cam table entries occupy the same space as variables **aa** through **yyy** which is the same space as the array variables. Invoking Cam Mode is done as follows:

```
BASE=2000          'Cam period
SIZE=25            'Data segments, this  defines the data
                      table size.
                   'CTR data, note the period at the end
aw[0] 0 10 20 30 40 50 60 70 80 90 100 110 120 120
   110 100 90 80 70 60 50 40 30 20 10 0.
MF0                'Reset external encoder to zero
O=0                'Reset internal encoder position
MC                 'Buffer CAM Mode
G                  'Start following the external encoder
                      using cam data
```

MD50 Drive Mode

The **MD50** command causes the SmartMotor to emulate a traditional servo and amplifier. In this mode, it can be used with yet another controller sending a standard +/-10Volt analog command signal. A small voltage divider is necessary to convert the +/-10 volts into the 0 to 5 volt signal the motor takes as input. The circuit considers the SmartMotor's input impedance. An additional device may be desired to take the single ended encoder signals coming out of the SmartMotor and make them differential for more noise immunity during their travel back to the external controller. An additional measure of optically isolating the encoder signals should be taken to avoid ground loops back to the control.



Voltage divider converting the +/-10 volt signal into a 0-5 volt signal

```
MD50               'Set Drive Mode
```

ENC0, ENC1 Encoder Select

The **ENC1** command causes the SmartMotor to servo off of an external encoder connected to inputs A and B. This can be useful if the external encoder has the advantage of being more accurate. The **ENC0** command restores the default mode of servoing off of the internal encoder.

```
ENC1            'Servo off of external encoder
ENC0            'Servo from internal encoder (default)
```

The following binary values can be tested by **IF** and **WHILE** control flow expressions, or assigned to any variable. They may all be reported using **RB**{bit} commands. Some may be reset using **Z**{bit} commands and some are reset when accessed. The first 8 states are reported in combination by the RS command. RW reports sixteen of these flags in combination.

By writing programs to periodically test these bits, a SmartMotor application can be very “smart” about its own inner-workings and doings.

Bo	Motor off	status bit 7
Bh	Excessive temperature	status bit 6
Be	Excessive position error	status bit 5
Bw	Wraparound occurred	status bit 4
Bi	Index report available	status bit 3
Bm	Real time negative limit	status bit 2
Bp	Real time positive limit	status bit 1
Bt	Trajectory in progress	status bit 0
Ba	Over current state occurred	
Bb	Parity error occurred	
Bc	Communication overflow occurred	
Bd	User math overflow occurred	
Bf	Communications framing error occurred	
Bk	Program check sum/EEPROM failure	
Bl	Historical left limit	
Br	Historical right limit	
Bs	Syntax error occurred	
Bu	User array index error occurred	
Bx	Hardware index input level	

If action is taken based on some of the error flags, the flag will need to be reset in order to look out for the next occurrence, or in some cases depending on how the code is written, in order to keep from acting over and over again on the same occurrence. The flags that need to be reset are listed. Their letter designator is preceded by the letter **Z** in the following table:

G also resets several system state flags

RESET SYSTEM STATE FLAGS

Za	Reset over current violation occurred
Zb	Reset parity error occurred
Zc	Reset com overflow error occurred
Zd	Reset user math overflow occurred
Zf	Reset communications framing error occurred
Zl	Reset historical left limit occurred
Zr	Reset historical right limit occurred
Zs	Reset syntax error occurred
Zu	Reset user array index error occurred
Zw	Reset wraparound occurred
ZS	Reset all **Z**{bit} state flags

The **TWAIT** command pauses program execution until motion is complete. Instead of using **TWAIT**, a routine could be written that does much more. To start with, the following code example would perform the same function as **TWAIT**:

```
WHILE Bt    'While trajectory
LOOP        'Loop back
```

Alternatively, the above routine could be augmented with code that took specific action in the event of an index signal as is shown in the following example

```
WHILE Bt    'While trajectory
  IF Bi           'Check index
    GOSUB500      'take care of it
  ENDIF           'end checking
LOOP        'Loop back
```

The following is a list of all of the commands used to relate to the SmartMotor's many I/O ports, grouped by port.

THE MAIN RS-232 PORT

ECHO — **ECHO** back all received characters

SADDR# — **S**et **ADDR**ess (0 to 120)

SILENT — Suppress print messages

TALK — Re-activate print message

SLEEP — Ignore all commands except **WAKE**

WAKE — Consider all following commands

BAUD19200 — Set baud rate to 19200 bps

OCHN (RS2,0,N,38400,1,8,D)
OpenChnl - RS-232, Channel 0, No parity, 38.4k bps, 1 stop, 8 data, as Data

OCHN (RS4,0,N,38400,1,8,C)
OpenChnl - RS-485 (w/adapter), Channel 0, No parity, 38.4k bps, 1 stop, 8 data, as Control

IF LEN>0 — Check to see if any (or how much) data is in the 16 byte input buffer, Data mode

c=GETCHR — Get byte from buffer into variable **c** for Data mode

PRINT (“Char Rcd:”,c,#13)
Print text, data and ASCII code for carriage return

THE G PORT

UGI — Redefine as general input

UGO — Redefine as general output (Open collector, pulled to 5V)

UG — Return pin to default start function, when low motor starts motion

UG=0 — Set G port Low (**UG=a** to set to variable **a**)

UG=1 — Set G port High (Open collector, weakly pulled to 5V internally)

a=UGI — Set variable **a** to digital input

a=UGA — Set **a** to analog input, 0 to 1023 = 0 to 5V

THE LIMIT PORTS C AND D

UCI	Redefine Right Limit as general input (**UDI** for Left Limit)
UCO	Redefine Right Limit as general output (**UDO** for Left Limit)
UCP	Return pin to limit function (**UDM** for Left Limit)
UC=0	Set Right Limit Low (**UD=0** for Left, or **UD=a** to set to variable **a**)
UC=1	Set Right Limit High **(UD=1** for Left Limit)
a=UCI	Set variable **a** to digital input (**UDI** for Left Limit)
a=UCA	Set **a** to analog input, 0 to 1023 = 0 to 5V (**UDA** for Left Limit)

COUNTER FUNCTIONS OF PORTS A AND B

MF4	Set **M**ode **F**ollow with full quadrature
MFR	Set **M**ode **F**ollow with ratio for gearing
MS	**M**ode **S**tep and Direction
MC	**M**ode **C**am
MF0	Set follow mode to zero and increment counter only
MS0	Set step mode to zero and increment counter only
a=CTR	Set variable **a** to counter value

GENERAL I/O FUNCTIONS OF PORTS A AND B

UAI	Set port A to input (**UBI** for port B)
UAO	Set port A to output (**UBO** for port B)
UA=0	Set port A Low (**UB=0** for port B, or **UB=a** to set to variable **a**)
UA=1	Set port A High (**UB=1** for port B)
a=UAI	Set variable **a** to digital input (**UBI** for port B)
a=UAA	Set **a** to analog input, 0 to 1023 = 0 to 5V (**UBA** for port B)

THE ANILINK PORT (USING I²C PROTOCOL)

AOUTB,c Send variable **c** out to Analog I/O board addressed as **B**

DOUTB0,c Send variable **c** out to Digital I/O board addressed as **B0**

c=AINB2 Set variable **c** to input 2 from Analog I/O board addressed as **B**

c=DINB0 Set variable **c** to input from Digital I/O board addressed as **B0**

PRINTB("Temp:",c,#32)
Print to LCD on network - text, data and ASCII code

THE ANILINK PORT (USING RS-485 PROTOCOL)

OCHN(RS4,1,N,38400,1,8,D)
OpenChnl - RS-485, Channel 1, No parity, 38.4k bps, 1 stop, 8 data, as Data

IF LEN1>0 Check to see if data is in the 16 byte input buffer

c=GETCHR1 Get byte from buffer into variable **c**

PRINT1("Char Rcd:",c,#13) Print text, data and ASCII code

ECHO1 **ECHO** back all received characters

SILENT1 Suppress print messages

SLEEP1 Ignore all commands except **WAKE**

WAKE1 Consider all following commands

THE ANILINK PORT AS GENERAL I/O

UEI Set port E to input (**UFI** for port F)

UEO Set port E to output (**UFO** for port F)

UE=0 Set port E Low (**UF=0** for port F, or **UF=c** to set to variable **c**)

UE=1 Set port E High (**UF=1** for port F)

c=UEI Set variable **c** to digital input (**UFI** for port F)

c=UEA Set **c** to analog input, 0 to 1023 = 0 to 5V (**UFA** for port F)

Ports A through G have internal 5k Ohm resistive pull-ups.

The standard SmartMotor brings out 5 volt power and ground, as well as seven I/O points. Each one has multiple functions. They are **UA, UB, UC, UD, UE, UF** and **UG** and have the following functions:

UA
Digital Input, TTL, 0 to 5 volts
Digital Output, TTL, 0 to 5 volts
Analog Input, 10 bit, 0 to 1023
External Encoder **A** Input
Step and Direction, Step Input

UB
Digital Input, TTL, 0 to 5 volts
Digital Output, TTL, 0 to 5 volts
Analog Input, 10 bit, 0 to 1023
External Encoder **B** Input
Step and Direction, Direction Input

UC
Digital Input, TTL, 0 to 5 volts
Digital Output, TTL, 0 to 5 volts
Analog Input, 10 bit, 0 to 1023
Positive Limit Input

UD
Digital Input, TTL, 0 to 5 volts
Digital Output, TTL, 0 to 5 volts
Analog Input, 10 bit, 0 to 1023
Negative Limit Input

UE
Digital Input, TTL, 0 to 5 volts
Digital Output, TTL, 0 to 5 volts
Analog Input, 10 bit, 0 to 1023
AniLink Data I/O
AniLink RS-485 Signal **A**

UF
Digital Input, TTL, 0 to 5 volts
Digital Output, TTL, 0 to 5 volts
Analog Input, 10 bit, 0 to 1023
AniLink Clock Output
AniLink RS-485 Signal **B**

UG
Digital Input, TTL, 0 to 5 volts
Digital Output, TTL, 0 to 5 volts
Analog Input, 10 bit, 0 to 1023
Start Motion (or **GO**) Input

When used as general I/O, these commands will select the port function.

UAI	Assign pin A to input state
UAO	Assign pin A to output state
UA=0	Set **UA** to zero volts
UA=1	Set **UA** to 5 volts
a=UAI	Read **UA** digital value, 0 or 1
a=UAA	Read **UA** analog voltage
UBI	Assign pin B to input state
UBO	Assign pin B to output state
UB=0	Set **UB** to zero volts
UB=1	Set **UB** to 5 volts
a=UBI	Read **UB** digital value, 0 or 1
a=UBA	Read **UB** analog voltage
UCI	Assign pin C to input state
UCO	Assign pin C to output state
UCP	Re-assign pin C to +limit action
UC=0	Set **UC** to zero volts
UC=1	Set **UC** to 5 volts
a=UCI	Read **UC** digital value, 0 or 1
a=UCA	Read **UC** analog voltage
UDI	Assign pin D to input state
UDO	Assign pin D to output state
UDM	Re-assign pin D to -limit action
UD=0	Set **UD** to zero volts
UD=1	Set **UD** to 5 volts
a=UDI	Read **UD** digital value, 0 or 1
a=UDA	Read **UD** analog voltage
UEI	Assign pin E to input state
UEO	Assign pin E to output state
UE=0	Set **UE** to zero volts
UE=1	Set **UE** to 5 volts
a=UEI	Read **UE** digital value, 0 or 1
a=UEA	Read **UE** analog voltage
UFI	Assign pin F to input state
UFO	Assign pin F to output state
UF=0	Set **UF** to zero volts
UF=1	Set **UF** to 5 volts
a=UFI	Read **UF** digital value, 0 or 1
a=UFA	Read **UF** analog voltage

UGI	Assign pin G to input state
UGO	Assign pin G to output state
UG	Re-assign pin G to "**GO**"
UG=0	Set **UG** to zero volts
UG=1	Set **UG** to 5 volts
a=UGI	Read **UG** digital value, 0 or 1
a=UGA	Read **UG** analog voltage

The **UAA**, **UBA**, **UCA**, **UDA**, **UEA**, **UFA** and **UGA** variables reflect the analog voltages at the port pins regardless of how the pins are configured. The analog voltage of any pin can be read without effecting it's current mode of operation in any way. For example, a pin could be used as an output and then the analog input value could be read to see if it happened to be shorted, or RS-485* signal bias could be monitored at ports E and F.

**RS-485 is not available as standard on SMXXX5 SmartMotors*

The encoder and step counting capabilities of ports A and B are described in the section on External Encoder Modes. The serial data capabilities of ports E and F are described in the section on communications.

ANILINK I/O MODULES

In the event the on-board I/O is not enough, additional I/O can be connected via the AniLink port. A variety of Analog and Digital I/O cards are available, as well as peripheral devices like LCD and LED displays, push-wheel input devices, pendants and more. These products communicate with the SmartMotor through the AniLink port using I^2C protocol.

OUTPUT ASSIGNMENTS

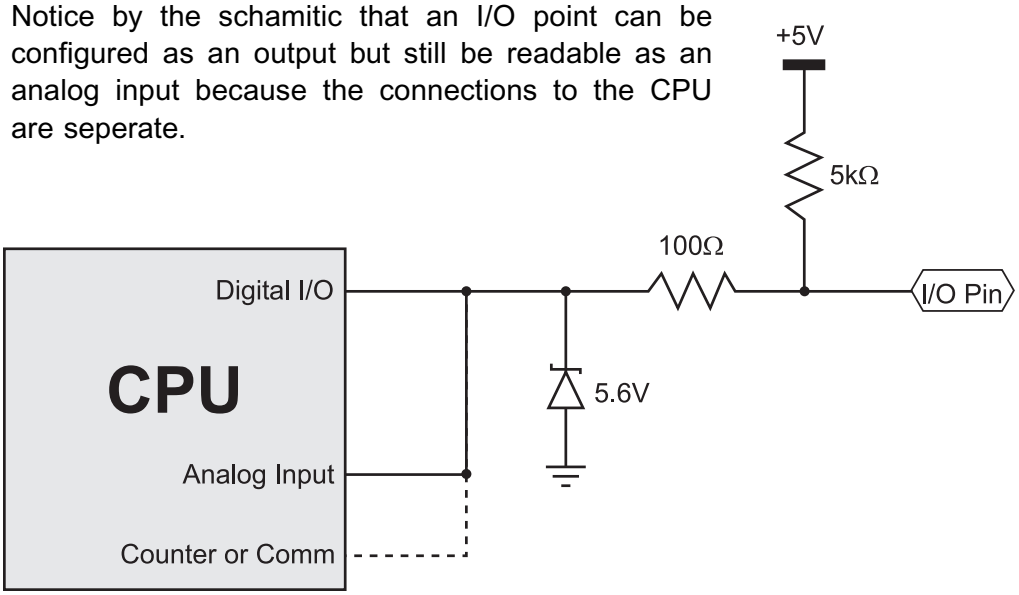
AOUT{address},exp	Output byte to analog address=A-H
DOUT{address}{ch},exp	Output byte to network, address=A-H, ch=0-63

INPUT ASSIGNMENTS

var=AIN{address}{input}	8 bit analog input from network, address=A-H, and input=1-4
var=DIN{address}{ch}	8 bit digital in from network, address=A-H, and ch=0-63

While all SmartMotor I/O is confined to operate between 0 and 5VDC, some circuitry exists to accomodate spikes above and below the operational range as long as those spikes are moderate and short lived.

Notice by the schamitic that an I/O point can be configured as an output but still be readable as an analog input because the connections to the CPU are seperate.



Knowing the SmartMotor's internal schematic can be useful when designing external interfaces.

All SmartMotor I/O points default to inputs when power is applied to the motor. It is the User Program that takes control after that. Because of the pull-up resistor, the voltage read at each port will be about 5VDC. When used as outputs to turn on external devices, it is highly recommended to design the system such that +5V is OFF and 0V is ON. This will prevent external equipment from being turned on immediately after power-up, before the User Program has a chance to take over.

Motor Connectors Pin Identification

4 Pin, 3 Phase Power

A Phase A
B Phase B
C Phase C
D Chassia

24 Pin Data I/O

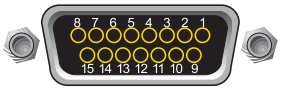
A RS-232 Ground
B RS-485 Ground
C Ground
D Ground
E RS-232 Transmit
F RS-232 Receive
G RS-485 A
H RS-485 B
J +5V Out, Fused
K Encoder A Out
L Encoder B Out
M Encoder I Out
N Ext Encoder A/Step/User I/O A
P Ext Encoder B/Direct/User I/O B
Q Reserved
R User I/O G
S Left Limit/User I/O D
T Right Limit/User I/O C
U AniLink Data/User I/O E
V AniLink Clock/User I/O F
W Reserved
X Reserved
Y Reserved
Z Reserved

7 Pin Combo D-Sub Power and I/O

A1 +20V to +48V DC
A2 Power Ground

1 Sync or I/O G
2 +5V Out
3 RS-232 Transmit
4 RS-232 Receive
5 RS-232 Ground

15 Pin D-Sub I/O



1 I/O A
2 I/O B
3 I/O C
4 I/O D
5 I/O E
6 I/O F
7 I/O G
8 Encoder A Out
9 Encoder B Out
10 SM RS-232 Transmit
11 SM RS-232 Receive
12 +5V Out
13 Ground
14 Power Ground
15 Power

Encoder I/O

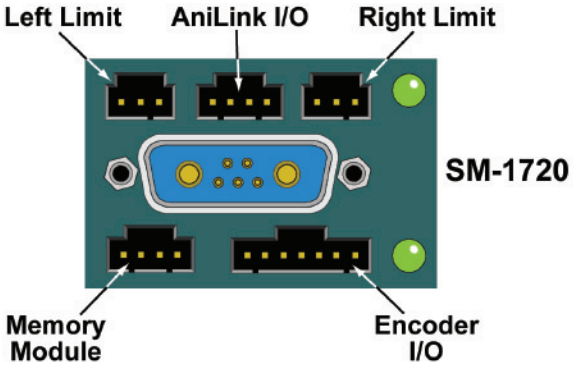
7 Ground
6 +5V Out
5 Encoder Index Out
4 Encoder B Out
3 Encoder A Out
2 Ext Encoder B/Step/User I/O B
1 Ext Encoder A/Direction/User I/O A

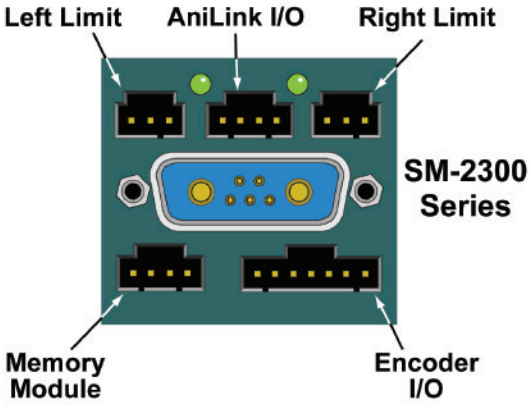
AniLink I/O

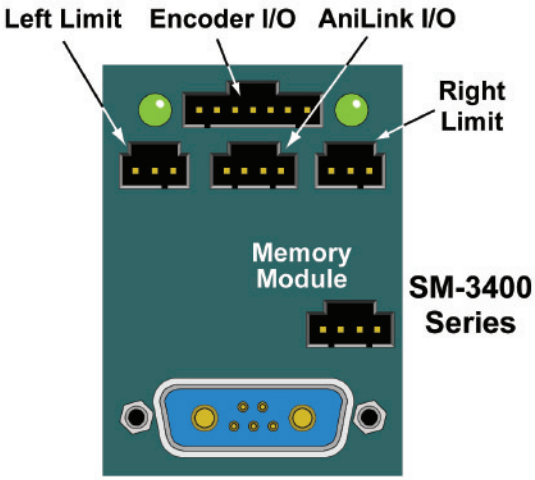
4 Clock/User I/O F
3 Data/User I/O E
2 Ground
1 +5V

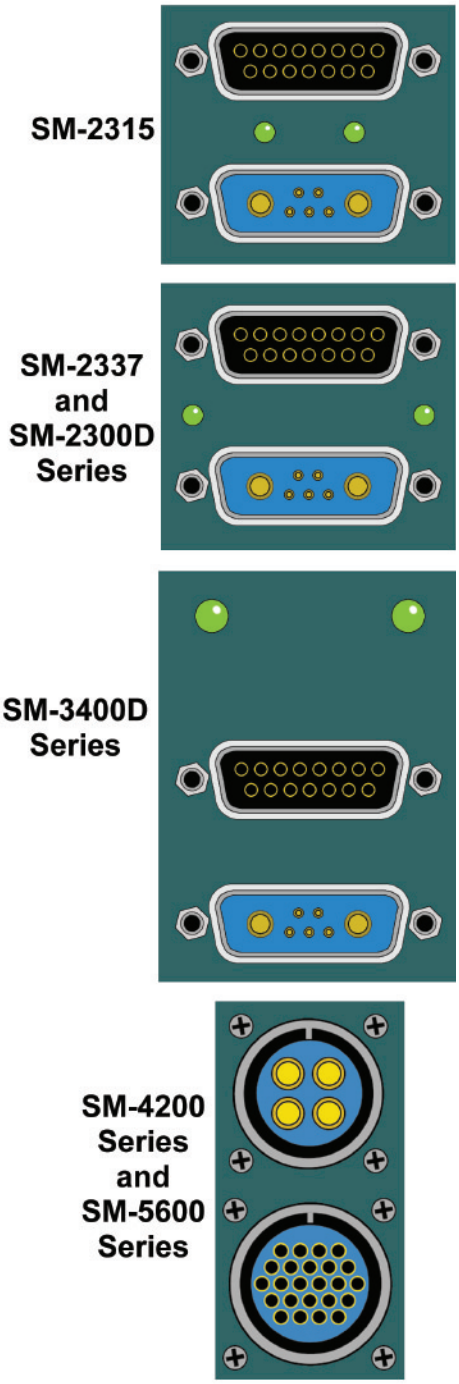
Left/Right Limit I/O

3 Limit Input
2 Ground
1 +5V Out





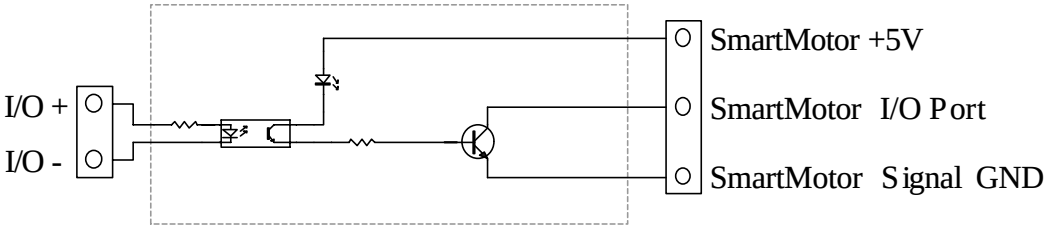




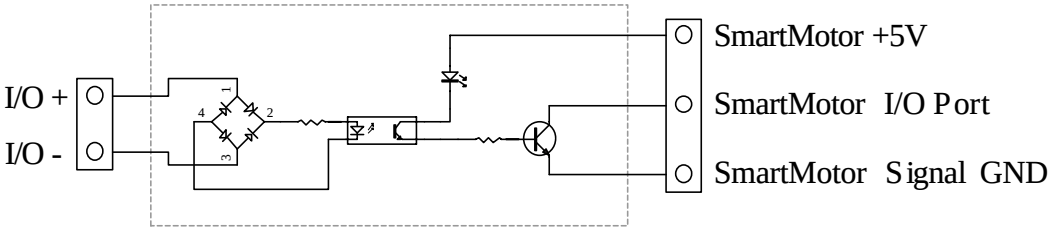
Motor Connector Locator

Please note the Memory Module location. This module uses the same type of connector as the AniLink I/O. If a Memory Module is plugged into the AniLink I/O, it won't break, but it won't work either.

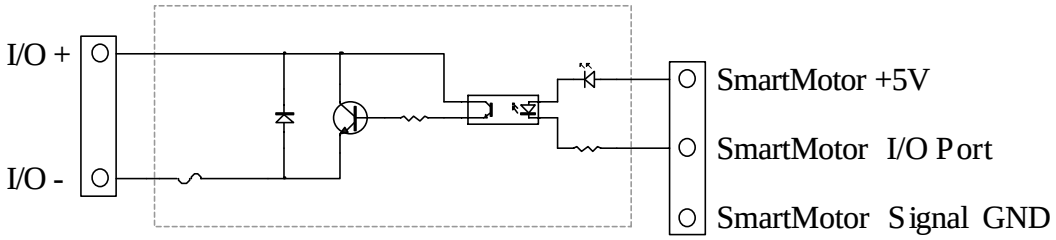
The 5 Volt Logic of the SmartMotor can interface to 24 Volt devices through the use of standard interface modules.



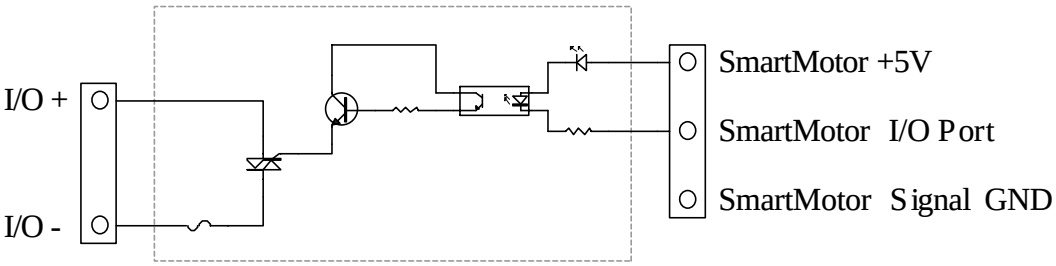
Typical DC Input Module (-IDC5 part numbers)



Typical AC Input Module (-IAC5 part numbers)



Typical DC Output Module (-ODC5 part numbers)



Typical AC Output Module (-OAC5 part numbers)

While there are a variety of options, the default mode for communicating with a SmartMotor is serial RS-232 for the main port. Each SmartMotor is equipped with a secondary serial port called81 the **AniLink** port. The **AniLink** port on a SmartMotor can be configured to communicate with either RS-485 or I^2C. The I^2C connects SmartMotor peripherals like LCD displays, I/O cards, etc., while the RS-485 will interface bar code readers, light curtains, and other "intelligent" peripherals including other SmartMotors if desired. SmartMotor models SMXXX5 do not have RS-485 capability in their **AniLink** ports.

When using I^2C, the SmartMotor is always the bus master. You cannot communicate between SmartMotors via I^2C.

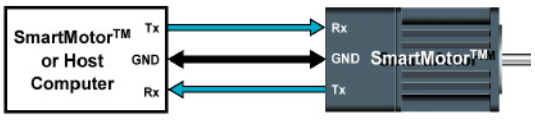
To maximize the flexibility of the SmartMotor, these serial communications ports are fully programmable with regard to bit-rate and protocol.

There is a sixteen-byte input buffer for the primary RS-232 port and another for the secondary RS-485 port. These buffers ensure that no arriving information is ever lost, although when either port is in data mode, it is the responsibility of the user program within the motor to keep up with the incoming data.

By default, the primary RS-232 channel, which shares a connector with the incoming power, is set up as a command port with the following default characteristics:

	Default:	**Other Options:**
Type:	RS-232	RS-485 (w/adapter)
Parity:	None	Odd or Even
Bit Rate:	9600	2400 to 38400
Stop Bits:	1	0 or 2
Data Bits:	8	7
Mode:	Command	Data
Echo:	Off	On

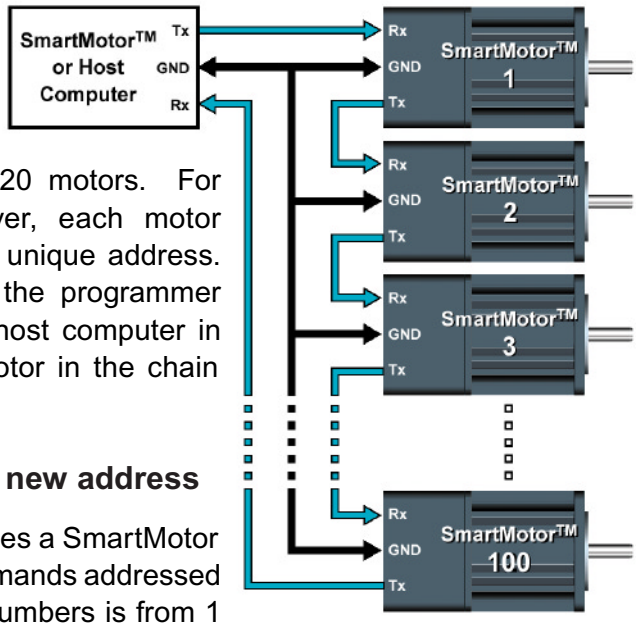
If the cable used is not provided by Animatics, make sure the SmartMotor's power and RS-232 connections are correct.



Because of the buffers on both sides there is no need for any hand shaking protocol when commanding the SmartMotor. Most commands execute in less time than it would take to receive the next one. Be careful to allow processes time to complete, particularly relatively slow processes like printing to a connected LCD display or executing a full subroutine. Since the **EEPROM** long term memory is slow to write, the terminal software does employ two way communication to regulate the download of a new program.

DAISY CHAINING RS-232

Multiple SmartMotors can be connected to a single RS-232 port as shown.



This diagram could be expanded to as many as 120 motors. For independent motion, however, each motor must be programmed with a unique address. In a multiple motor system the programmer has the choice of putting a host computer in control or having the first motor in the chain be in control of the rest.

SADDR# Set motor to new address

The **SADDR#** command causes a SmartMotor to respond exclusively to commands addressed to it. The range of address numbers is from 1 to 120. Once each motor in a chain has a unique address, each individual motor will communicate normally after its address is sent at least once over the chain. To send an address, add 128 to its value and output the binary result over the communication link. This puts the value above the ASCII character set, quickly and easily differentiating it from all other commands or data. The address needs to be sent only once until the host computer, or motor, wants to change it to something else. Sending out an address zero (128) will cause all motors to listen and is a great way to send global data such as a **G** for starting simultaneous motion in a chain. Once set, the address features work the same for RS-232 and RS-485 communications.

Unlike the RS-485 star topology, the consecutive nature of the RS-232 daisy-chain creates the opportunity for the chain to be independently addressed entirely from the host, rather than by having a uniquely addressed program in each motor. Setting up a system this way can add simplicity because the program in each motor can be exactly the same. If the **RUN?** Command is the first in each of the motor's programs, the programs will not start upon power up. Addressing can be worked out by the host prior to the programs being started later by the host sending the **RUN** command globally.

SLEEP, SLEEP1 Assert sleep mode

WAKE, WAKE1 De-assert SLEEP

Telling a motor to sleep causes it to ignore all commands except the **WAKE** command. This feature can often be useful, particularly when establishing unique addresses in a chain of motors. A **1** at the end of the command specifies the AniLink RS-485 port.

ECHO, ECHO1 **ECHO input**

ECHO_OFF, ECHO_OFF1 **De-assert ECHO**

The **ECHO** and **ECHO_OFF** commands toggle the echoing of data input. Because the motors do not echo character input by default, consecutive commands can be presented, configuring them with unique addresses, one at a time. If the host computer or controller sent out the following command sequence, each motor would have a unique and consecutive address.

If a daisy chain of SmartMotors have been powered off and back on, the following commands can be entered into the **SmartMotor Interface** to address the motors (0 equals 128, 1 equals 129, etc.). Some delay should be inserted between commands when sending them from a host computer.

```
0SADDR1
1ECHO
1SLEEP
0SADDR2
2ECHO
2SLEEP
0SADDR3
3ECHO
0WAKE
```

Commanded by a user program in the first motor, instead of a host, the same daisy chain could be addressed with the following sequence:

```
SADDR1                          'Address the first motor
ECHO                            'Echo for host data
PRINT(#128,"SADDR2",#13)        '0SADDR2
WAIT=10                         'Allow time
PRINT(#130,"ECHO",#13)          '2ECHO
WAIT=10
PRINT(#130,"SLEEP",#13)         '2SLEEP
WAIT=10
PRINT(#128,"SADDR3",#13)        '0SADDR3
WAIT=10
PRINT(#131,"ECHO",#13)          '3ECHO
WAIT=10
PRINT(#128,"WAKE",#13)          '0WAKE
WAIT=10
```

The two communications ports have enormous flexibility. To select from the vast array of options, use the **OCHN** command.

OCHN

	Options:	
Type:	RS2, RS4	RS-232 or RS-485
Channel:	0, 1 or 2	0=Main, 1=AniLink
Parity:	N, O or E	None, Odd or Even
Bit rate:	2400, 4800, 9600, 19200, 38400 baud	
Stop bits:	0, 1 or 2	
Data bits:	7 or 8	
Mode:	C or D	Command or Data

Here is an example of the **OCHN** command:

```
OCHN(RS2,0,N,38400,1,8,D)
```

If the primary communication channel (0) is opened as an RS-485 port, it will assume the RS-485 adapter is connected to it. If that is the case then pin **G** in the same connector is assigned the task of directing the adapter to be in Transmit or Receive mode in accordance with the motor's communication activity and will no longer be useful as an I/O port to the outside world.

CCHN(type,channel) Close a communications channel

Use the **CCHN** command to close a communications port when desired.

BAUD# Set BAUD rate of main port

The **BAUD#** command presents a convenient way of changing only the bit rate of the main channel. The number can be from 2400 to 38400 bps.

PRINT( ), PRINT1( ) Print to RS-232 or AniLink channel

A variety of data formats can exist within the parentheses of the **PRINT( )** command. A text string is marked as such by enclosing it between double quotation marks. Variables can be placed between the parentheses as well as two variables separated by one operator. To send out a specific byte value, prefix the value with the **#** sign and represent the value with as many as three decimal digits ranging from 0 to 255. Multiple types of data can be sent in a single **PRINT( )** statement by separating the entries with commas. Do not use spaces outside of text strings because SmartMotors use spaces as delimiters along with carriage returns and line feeds.

The following are all valid print statements and will transmit data through the main RS-232 channel:

```
PRINT("Hello World")   'text
PRINT(a*b)             'exp.
PRINT(#32)             'data
PRINT("A",a,a*b,#13)   'all
```

PRINT1 prints to the AniLink port with RS-485 protocol while **PRINTA** prints to the AniLink port using I^2C protocol in such a way as to send data to an LCD display or standard parallel input line printer (with a DIO-100 card on the AniLink bus).

SILENT, SILENT1 Suppress PRINT() outputs

TALK, TALK1 De-assert silent mode

The **SILENT** mode causes all **PRINT( )** output to be suppressed. This is useful when talking to a chain of motors from a host, when the chain would otherwise be talking within itself because of programs executing that contain **PRINT( )** commands.

! Wait for RS-232 character to be received

A single exclamation mark will cause program execution to stop until a character is received. This can be handy under certain circumstances like debugging a program in real time.

a=CHN0, a=CHN1 RS-485 communications error flags

The **CHN0** and **CHN1** variables hold binary coded information about the historical errors experienced by the two communications channels. The information is as follows:

Bit	Value	Meaning
0	1	Buffer overflow
1	2	Framing error
2	4	Command scan error
3	8	Parity error

A subroutine that printed the errors to an LCD display would look like the following:

```
C911
   IF CHN0          'If CHN0 != 0
      DOUT0,1       'Home LCD cursor
      IF CHN0&1
         PRINTA("BUFFER OVERFLOW")
      ENDIF
      IF CHN0&2
         PRINTA("FRAMING ERROR")
      ENDIF
      IF CHN0&4
         PRINTA("COMMAND SCAN ERROR")
      ENDIF
      IF CHN0&8
         PRINTA("PARITY ERROR")
```

```
        ENDIF
        CHN0=0          'Reset CHN0
      ENDIF
    RETURN
```

a=ADDR Motor's self address

If the motor's address (**ADDR**) is set by an external source, it may still be useful for the program in the motor to know what address it is set to. When a motor is set to an address, the **ADDR** variable will reflect that address from 1 to 120.

GETTING DATA FROM A COM PORT

If a com port is in Command Mode, then the motor will simply respond to arriving commands it recognizes. If the port is opened in Data Mode, however, then incoming data will start to fill the 16 byte buffer until it is retrieved with the **GETCHR** command.

a=LEN	Number of characters in RS-232 buffer
a=LEN1	Number of characters in RS-485 buffer
a=GETCHR	Get character from RS-232 buffer
a=GETCHR1	Get character from RS-485 buffer

The buffer is a standard **FIFO** (**F**irst **I**n **F**irst **O**ut) buffer. This means that if the letter **A** is the first character the buffer receives, then it will be the first byte offered to the **GETCHR** command. The buffer exists to make sure that no data is lost, even if the program is not retrieving the data at just the right time. Two things are very important when dealing with a data buffer for the protection of the data:

1) Never **GETCHR** if there is no **CHR** to **GET**.
2) Never let the buffer overflow.

The **LEN** variable holds the number of characters in the buffer. A program must see that the **LEN** is greater than zero before issuing a command like: **a=GETCHR**. Likewise, it's necessary to arrange the application so that, overall, data will be pulled out of the buffer faster than it comes in.

The ability to configure the communication ports for any protocol as well as to both transmit and receive data allows the SmartMotor to interface to a vast array of RS-232 and RS-485 devices. Some of the typical devices that would interface with SmartMotors over the communication interface are:

1) Other SmartMotors
2) Bar Code Readers
3) Light Curtains
4) Terminals
5) Printers

The following is an example program that repeatedly transmits a message to an external device (in this case another SmartMotor) and then takes a number back from the device as a series of ASCII letter digits, each ranging from 0 to 9. A carriage return character will mark the end of the received data. The program will use that data as a position to move to.

```
A=500              'Preset Accel.
V=1000000          'Preset Vel.
P=0                'Zero out Pos.
O=0                'Declare origin
G                  'Servo in place
OCHN(RS2,0,N,9600,1,8,D)
PRINT("RP",#13)
C0
   IF LEN          'Check for chars
      a=GETCHR     'Get char
      IF a==13     'If carriage return
         G         'Start motion
         P=0       'Reset buffered P to zero
         PRINT("RP",#13)     'Next
      ELSE
         P=P*10    'Shift buffered P
         a=a-48    'Adjust for ASCII
         P=P+a     'Build buffered P
      ENDIF
   ENDIF
GOTO0              'Loop forever
```

The ASCII code for zero is 48. The other nine digits count up from there so the ASCII code can be converted to a useful number by subtracting the value of 0 (ASCII 48). The example assumes that the most significant digits will be returned first. Any time it sees a new digit, it multiplies the previous quantity by 10 to shift it over and then adds the new digit as the least significant. Once a carriage return is seen (ASCII 13), motion starts. After motion is started, **P** (Position) is reset to zero in preparation for building up again. **P** is buffered so it will not do anything until the **G** command is issued.

PID FILTER CONTROL

The SmartMotor™ includes a very high quality, high performance brushless D.C. servomotor. It has a rotor with extremely powerful rare earth magnets and a stator (the outside, stationary part) that is a densely wound multi-slotted electro-magnet.

Controlling the position of a brushless D.C. servo's rotor with only electromagnetism working as a lever is like pulling a sled with a rubber band. Accurate control would seem impossible.

The parameters that makes it all work are found in the **PID** (**P**roportional, **I**ntegral, **D**erivative) filter section. These are the three fundamental coefficients to a mathematical algorithm that intelligently recalculates and delivers the power needed by the motor about 4,000 times per second. The input to the **PID** filter is the instantaneous actual position minus the desired position, be it at rest, or part of an ongoing trajectory. This difference is called the error.

While the Derivative term usually acts to dampen instability, this is not the true definition of the term. It is possible to cause instability by setting the Derivative term too high.

The **P**roportional parameter of the filter creates a simple spring constant. The further the shaft is rotated away from its target position, the more power is delivered to return it. With this as the only parameter the motor shaft would respond just as the end of a spring would if it was grabbed and twisted.

If the spring is twisted and let go it will vibrate wildly. This sort of vibration is hazardous to most mechanisms. In this scenario a shock absorber is added to cancel the vibrations which is the equivalent of what the **D**erivative parameter does. If a person sat on the fender of a car, it would dip down because of the additional weight based on the constant of the car's spring. It would not be known if the shocks were good or bad. If the bumper was jumped up and down on, however, it would quickly become apparent whether the shock absorbers were working or not. That's because they are not activated by position but rather by speed. The **D**erivative parameter steals power away as a function of the rate of change of the overall filter output. The parameter gets its name from the fact that the derivative of position is speed. Electronically stealing power based on the magnitude of the motor shafts vibration has the same effect as putting a shock absorber in the system, and the algorithm never goes bad.

Even with the two parameters a situation can arise that will cause the servo to leave its target created by "dead weight". If a constant torque is applied to the end of the shaft, the shaft will comply until the deflection causes the **P**roportional parameter to rise to the equivalent torque. There is no speed so the **D**erivative parameter has no effect. As long as the torque is there, the motor's shaft will be off of its target.

That's where the **I**ntegral parameter comes in. The **I**ntegral parameter mounts an opposing force that is a function of time. As time passes and there is a deflection present, the **I**ntegral parameter will add a little force to bring it back on target with each **PID** cycle. There is also a separate parameter (**KL**) used to limit the **I**ntegral parameter's scope of what it

can do so as not to over react.

Each of these parameters have their own scaling factor to tailor the overall performance of the filter to the specific load conditions of any one particular application. The scaling factors are as follows:

KP	**P**roportional
KI	**I**ntegral
KD	**D**erivative
KL	Integral Limit

TUNING THE FILTER

The task of tuning the filter is complicated by the fact that the parameters are so interdependent. A change in one can shift the optimal settings of the others. The automatic utility makes all of the settings easy, but it still may be necessary to know how to tune a servo.

When tuning the motor it is useful to have the status monitor running which will monitor various bits of information that will reflect the motors performance.

KP=exp	Set **KP**, proportional coefficient
KI=exp	Set **KI**, time-error coefficient
KD=exp	Set **KD**, damping coefficient
KL=exp	Set **KL**, time-error term limit
F	Update **PID** filter

The main objective in tuning a servo is to get **KP** as high as possible, while maintaining stability. The higher the **KP** the stiffer the system and the more under control it is. A good start is to simply query what to begin with (**RKP**) and then start increasing it 10% to 20% at a time. It is a good idea to start with **KI** equal to zero. Keep in mind that the new settings do not take effect until the **F** command is issued. Each time **KP** is raised, try physically to destabilize the system, by bumping or twisting it. Or, have a program loop cycling that invokes abrupt motions. As long as the motor always settles to a quiet rest, keep raising **KP**. Of course if the **SMI Tuning Utility** is being used, it will employ a step function and show more precisely what the reaction is.

As soon as the limit is reached, find the appropriate derivative compensation. Move **KD** up and down until the position is found that gives the quickest stability. If **KD** is way too high, there will be a grinding sound. It is not really grinding, but it is a sign to go the other way. A good tune is not only stable, but reasonably quiet. After optimizing **KD**, it may be possible to raise **KP** a little more. Keep going back and forth until there's nothing left to improve the stiffness of the system. After that it's time to take a look at **KI**.

KI, in most cases, is used to compensate for friction. Without it the SmartMotor will never exactly reach the target. Begin with **KI** equal to zero and **KL** equal to 1000. Move the motor off target and start increasing **KI** and **KL**. Keep **KL** at least ten times **KI** during this phase.

Continue to increase **KI** until the motor always reaches its target, and once that happens add about 30% to **KI** and start bringing down **KL** until it hampers the ability for the **KI** term to close the position precisely to target. Once that point is reached, increase **KL** by about 30% as well. The **I**ntegral term needs to be strong enough to overcome friction, but the limit needs to be set so that an unruly amount of power will not be delivered if the mechanism were to jam or simply find itself against one of its ends of travel.

E=expression Set maximum position error

The difference between where the motor shaft is and where it is supposed to be is appropriately called the “error”. The magnitude and sign of the error is delivered to the motor in the form of torque, after it is put through the **PID** filter. The higher the error, the more out of control the motor is. Therefore, it is often useful to put a limit on the allowable error, after which time the motor will be turned off. That is what the **E** command is for. It defaults to 1000 encoder counts, but can be set from 1 to 32,000.

There are still more parameters that can be utilized to reduce the position error of a dynamic application. Most of the forces that aggravate a **PID** loop through the execution of a motion trajectory are unpredictable, but there are some that can be predicted and further eliminated preemptively.

KG=expression Set KG, Gravity offset term

The simplest of these is gravity. Why burden the **PID** loop with the effects of gravity in a vertical load application, if it can simply be weeded out. If in a particular application, motion would occur with the power off due to gravity, a constant offset can be incorporated into the filter to balance the system. **KG** is the term. **KG** can range from -8388608 to 8388607. To tune **KG,** simply make changes to **KG** until the load equally favors upward and downward motion.

KV=expression Set KVff, velocity feed forward

Another predictable cause of position error is the natural latency of the **PID** loop itself. At higher speeds, because the calculation takes a finite amount of time, the result is somewhat “old news”. The higher the speed, the more the actual motor position will slightly lag the trajectory calculated position. This can be programmed out with the **KV** term. KV can range from zero to 65,535. Typical values range in the low hundreds. To tune **KV** simply run the motor at a constant speed, if the application will allow, and increase **KV** until the error gets reduced to near zero and stays there. The error can be seen in real time by activating the **Monitor Status** window in the **SMI** program.

KA=expression **Set KAff, acceleration feed forward**

Force equals mass times acceleration. If the SmartMotor is accelerating a mass, it will be exerting a force during that acceleration. This force will disappear immediately upon reaching the cruising speed. This momentary torque during acceleration is also predictable and need not aggravate the **PID** filter. It's effects can be programmed out with the **KA** term. It is a little more difficult to tune **KA**, especially with hardware attached. The objective is to arrive at a value that will close the position error during the acceleration and deceleration phases. It is better to tune **KA** with **KI** set to zero because **KI** will address this constant force in another way. It is best to have **KA** address 100% of the forces due to acceleration, and leave the **KI** term to adjust for friction.

KS=expression **Set KS, dampening sample rate**

Reduce the sampling rate of the derivative term, **KD**, with the **KS** term. This can sometimes add stability to very high inertial loads. Useful values of **KS** range from 1 (the default) to 20. Results will vary from application to application.

The **PID** rate of the SmartMotor can be slowed down.

PID1 Set normal **PID** update rate

PID2 Divide normal **PID** update rate by 2

PID4 Divide normal **PID** update rate by 4

PID8 Divide normal **PID** update rate by 8

The trajectory and **PID** filter calculations occur within the SmartMotor™ 4069 times per second. That is faster than is necessary for very good control, especially with the larger motors. A reduction in the **PID** rate can result in an increase in the SmartMotor™ application program execution rate. The **PID2** command will divide the **PID** rate by two, and the others even more. The most dramatic effect on program execution rate occurs with **PID4**. **PID8** does little more and is encroaching upon poor control. If the **PID** rate is lowered, keep in mind that this is the "sample" rate that is the basis for **V**elocity values, **A**cceleration values, **PID** coefficients and **WAIT** times. If the rate is cut in half, expect to do the following to keep all else the same:

Halve **WAIT** times

Double **V**elocity

Increase **A**cceleration by a factor of 104

KGON **Change Drive Characteristic for Vertical Application**

KGOFF **Restore Drive Characteristic to Default**

Vertical applications can be particularly hard to tune, even with the **KG** term. Often this is seen in an awkward sound or vibration occurring when the motor decelerates on its way down. The filter and the drive electronics are challenged to deal with the situation where the motor wants to go in the

same direction it is told to go, entirely on its own volition. The SmartMotor is equipped with a special drive mode designed to deal with this very situation. This mode of operation is invoked with the **KGON** command. Because this mode is so different from the standard drive mode, it will be necessary to tune the motor once again. The reason this more stable drive mode is not the default is because like most good things, it comes at a cost. The SmartMotor's amplifier is not as efficient at converting current to torque in this mode as it is in the default mode and so it is necessary to verify the motor's reasonably cool operation when **KGON** is in use. Use **KGOFF** to revert back to the standard drive mode.

CURRENT LIMIT CONTROL

AMPS=expression Set current limit, 0 to 1023

In some applications, if the motor misapplied full power, the attached mechanism could be damaged. It can be useful to reduce the maximum amount of current available thus limiting the torque the motor can put out. Use the **AMPS** command with a number, variable or expression within the range of 0 to 1023. The units are tenths of a percent of full scale peak current, and varies in actual torque with the size of the SmartMotor.

APPENDIX A: UNDERSTANDING BINARY DATA

The SmartMotor's[tm] language allows the programmer to access data on the binary level. Understanding binary data is very easy and useful when programming the SmartMotor or any electronic device. What follows is an explanation of how binary data works.

All digital computer data is stored as binary information. A binary element is one that has only two states, commonly described as "on" and "off" or "one" and "zero". A light switch is a binary element. It can either be "on" or "off". A computer's memory is nothing but a vast array of binary switches called "bits".

The power of a computer comes from the speed and sophistication with which it manipulates these bits to accomplish higher tasks. The first step towards these higher goals is to organize these bits in such a way that they can describe things more complicated than "off" or "on".

Different numbers of bits are used to make up different building blocks of data. They are most commonly described as follows:

Four bits	=	Nibble
Eight bits	=	Byte
Sixteen bits	=	Word
Thirty two bits	=	Long

One bit has two possible states, on or off. Every time a bit is added, the possible number of states is doubled. Two bits have four possible states. They are as follows:

00	off-off
01	off-on
10	on-off
11	on-on

A nibble has 16 possible states. A byte has 256 and a Long has billions of possible combinations.

Because a byte of information has 256 possible states, it can reflect a number from zero to 255. This is elegantly done by assigning each bit a value of twice the one before it, starting with one. Each bit value becomes as follows:

Bit	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

If all their values are added together the result is 255. By leaving particular bits out any sum between zero and 255 can be created. Look at the following example bytes and their decimal values:

Byte	Value
0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 1 0	2
0 0 0 0 0 0 1 1	3
0 0 0 1 0 0 0 0	16
1 0 0 0 0 0 0 0	128
1 0 0 0 0 0 0 1	129
1 1 1 1 1 1 1 1	255

Consider the following two bytes of information:

Byte	Value
0 0 1 1 1 1 0 0	60
0 0 0 1 1 1 1 0	30

To make use of the limited memory available with micro controllers that can fit into a SmartMotor, there are occasions where every bit is used. One example is the status byte. A single value can be uploaded from a SmartMotor and have coded into it, in binary, eight or sixteen independent bits of information.

The following is the status byte and its coded information:

Name	Description	Bit	Value
Bo	Motor OFF	7	128
Bh	Excessive temp.	6	64
Be	Excessive pos. err.	5	32
Bw	Wraparound	4	16
Bi	Index reportable	3	8
Bm	Real time neg. lim.	2	4
Bp	Real time pos. lim.	1	2
Bt	Trajectory going	0	1

There are two useful mathematical operators that work on binary data, the "**&**" (and) and the "**|**" (or). The "**&**" compares two bytes, words or longs and looks for what they have in common. The resulting data has ones only where there were ones in both the first byte and the second. The "**|**" looks for a one in the same location of either the first data field or the second. Both functions are illustrated in the following example:

A	B	A&B	A|B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Knowing how the binary data works will enable shorter and faster code to be written. The following are two code examples that are looking to see if both limit inputs are high. One does this without taking advantage of the binary operator while the second shows how using the binary operator makes the code shorter, and therefor faster.

Example 1:

```
IF Bm            'look for - lim high
   IF Bp         'loof for + lim high
      GOSUB100   'handle it
   ENDIF
ENDIF
```

Example 2:

```
IF S&6           'look at both lim
   GOSUB100      'handle it
ENDIF
```

Both examples will execute subroutine 100 if both limit inputs are high. By "anding" the status byte (S) by six, the second routine filters out all of the other status information. If either limit is high, then the result will be non-zero and subroutine 100 will execute. Example two uses much less code than example one and will run much faster as a part of a larger program loop.

The next two examples show how the use of the "|" operator can improve program size and execution speed:

Example 3:

```
IF UAI           'look for input A
   GOSUB200      'handle it
ENDIF
IF UBI           'look for input B
   GOSUB200      'handle it
ENDIF
```

Example 4:

```
IF UAI|UBI       'look at both A,B
   GOSUB200      'handle it
ENDIF
```

Both examples 3 and 4 accomplish the same task with different levels of efficiency.

ASCIIIS an acronym for American Standard Code for Information Interchange. It refers to the convention established to relate characters, symbols and functions to binary data. If a SmartMotor is asked its position over the RS-232 link, and it is at position 1, it will not return a byte of value one, but instead will return the ASCII code for 1 which is binary value 49. That is why it appears on a terminal screen as the numeral 1.

The ASCII character set is as follows:

0	NUL	32	SP	64	@	96	’
1	SOH	33	!	65	A	97	a
2	STX	34	“	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	‘	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FC	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	Del

APPENDIX C: USER ASSIGNED VARIABLES MEMORY MAP

aa	al[0]	aw[0]	ab[0]
			ab[1]
		aw[1]	ab[2]
			ab[3]
bb	al[1]	aw[2]	ab[4]
			ab[5]
		aw[3]	ab[6]
			ab[7]
cc	al[2]	aw[4]	ab[8]
			ab[9]
		aw[5]	ab[10]
			ab[11]
dd	al[3]	aw[6]	ab[12]
			ab[13]
		aw[7]	ab[14]
			ab[15]
ee	al[4]	aw[8]	ab[16]
			ab[17]
		aw[9]	ab[18]
			ab[19]
ff	al[5]	aw[10]	ab[20]
			ab[21]
		aw[11]	ab[22]
			ab[23]
gg	al[6]	aw[12]	ab[24]
			ab[25]
		aw[13]	ab[26]
			ab[27]
hh	al[7]	aw[14]	ab[28]
			ab[29]
		aw[15]	ab[30]
			ab[31]
ii	al[8]	aw[16]	ab[32]
			ab[33]
		aw[17]	ab[34]
			ab[35]
jj	al[9]	aw[18]	ab[36]
			ab[37]
		aw[19]	ab[38]
			ab[39]
kk	al[10]	aw[20]	ab[40]
			ab[41]
		aw[21]	ab[42]
			ab[43]
ll	al[11]	aw[22]	ab[44]
			ab[45]
		aw[23]	ab[46]
			ab[47]
mm	al[12]	aw[24]	ab[48]
			ab[49]
		aw[25]	ab[50]
			ab[51]

nn	al[13]	aw[26]	ab[52]
			ab[53]
		aw[27]	ab[54]
			ab[55]
oo	al[14]	aw[28]	ab[56]
			ab[57]
		aw[29]	ab[58]
			ab[59]
pp	al[15]	aw[30]	ab[60]
			ab[61]
		aw[31]	ab[62]
			ab[63]
qq	al[16]	aw[32]	ab[64]
			ab[65]
		aw[33]	ab[66]
			ab[67]
rr	al[17]	aw[34]	ab[68]
			ab[69]
		aw[35]	ab[70]
			ab[71]
ss	al[18]	aw[36]	ab[72]
			ab[73]
		aw[37]	ab[74]
			ab[75]
tt	al[19]	aw[38]	ab[76]
			ab[77]
		aw[39]	ab[78]
			ab[79]
uu	al[20]	aw[40]	ab[80]
			ab[81]
		aw[41]	ab[82]
			ab[83]
vv	al[21]	aw[42]	ab[84]
			ab[85]
		aw[43]	ab[86]
			ab[87]
ww	al[22]	aw[44]	ab[88]
			ab[89]
		aw[45]	ab[90]
			ab[91]
xx	al[23]	aw[46]	ab[92]
			ab[93]
		aw[47]	ab[94]
			ab[95]
yy	al[24]	aw[48]	ab[96]
			ab[97]
		aw[49]	ab[98]
			ab[99]
zz	al[25]	aw[50]	ab[100]
			ab[101]
		aw[51]	ab[102
			ab[103]

APPENDIX C: USER ASSIGNED VARIABLES MEMORY MAP

aaa	al[26]	aw[52]	ab[104]
			ab[105]
		aw[53]	ab[106]
			ab[107]
bbb	al[27]	aw[54]	ab[108]
			ab[109]
		aw[55]	ab[110]
			ab[111]
ccc	al[28]	aw[56]	ab[112]
			ab[113]
		aw[57]	ab[114]
			ab[115]
ddd	al[29]	aw[58]	ab[116]
			ab[117]
		aw[59]	ab[118]
			ab[119]
eee	al[30]	aw[60]	ab[120]
			ab[121]
		aw[61]	ab[122]
			ab[123]
fff	al[31]	aw[62]	ab[124]
			ab[125]
		aw[63]	ab[126]
			ab[127]
ggg	al[32]	aw[64]	ab[128]
			ab[129]
		aw[65]	ab[130]
			ab[131]
hhh	al[33]	aw[66]	ab[132]
			ab[133]
		aw[67]	ab[134]
			ab[135]
iii	al[34]	aw[68]	ab[136]
			ab[137]
		aw[69]	ab[138]
			ab[139]
jjj	al[35]	aw[70]	ab[140]
			ab[141]
		aw[71]	ab[142]
			ab[143]
kkk	al[36]	aw[72]	ab[144]
			ab[145]
		aw[73]	ab[146]
			ab[147]
lll	al[37]	aw[74]	ab[148]
			ab[149]
		aw[75]	ab[150]
			ab[151]
mmm	al[38]	aw[76]	ab[152]
			ab[153]
		aw[77]	ab[154]
			ab[155]

nnn	al[39]	aw[78]	ab[156]
			ab[157]
		aw[79]	ab[158]
			ab[159]
ooo	al[40]	aw[80]	ab[160]
			ab[161]
		aw[81]	ab[162]
			ab[163]
ppp	al[41]	aw[82]	ab[164]
			ab[165]
		aw[83]	ab[166]
			ab[167]
qqq	al[42]	aw[84]	ab[168]
			ab[169]
		aw[85]	ab[170]
			ab[171]
rrr	al[43]	aw[86]	ab[172]
			ab[173]
		aw[87]	ab[174]
			ab[175]
sss	al[44]	aw[88]	ab[176]
			ab[177]
		aw[89]	ab[178]
			ab[179]
ttt	al[45]	aw[90]	ab[180]
			ab[181]
		aw[91]	ab[182]
			ab[183]
uuu	al[46]	aw[92]	ab[184]
			ab[185]
		aw[93]	ab[186]
			ab[187]
vvv	al[47]	aw[94]	ab[188]
			ab[189]
		aw[95]	ab[190]
			ab[191]
www	al[48]	aw[96]	ab[192]
			ab[193]
		aw[97]	ab[194]
			ab[195]
xxx	al[49]	aw[98]	ab[196]
			ab[197]
		aw[99]	ab[198]
			ab[199]
yyy	al[50]	aw[100]	ab[200]
			ab[201]*
		aw[101]*	ab[202]*
			ab[203]*

* For versions 4.16 and above

! (exclamation point)	
(space)	Single space between user variables
@P	Current position
@PE	Current position error
@V	Current velocity
a . . . z	User variables
aa . . . zzz	More user variables
al[index]	Array variable 32 bit
aw[index]	Array variable 16 bit
ab[index]	Array variable 8 bit
A=exp	Set acceleration
ADDR	Motor's self address variable
AIN{port}{channel}	Assign input byte from module
AMPS=expression	Set PWM drive signal limit
AOUT{port}{expression}	Output analog byte to module
Ba	Over current status bit
Bb	Parity error status bit
Bc	Communication overflow status bit
Bd	Math overflow status bit
Be	Excessive position error status bit
Bf	Communications framing error status bit
Bh	Excessive temperature status bit
Bi	Index captured status bit
Bk	EEPROM data integrity status bit
Bl	Historical left limit status bit
Bm	Real time left limit status bit
Bo	Motor off status bit
Bp	Real time right limit status bit
Br	Historical right limit status bit

Bs	Syntax error status bit
Bt	Trajectory in progress status bit
Bu	Array index error status bit
Bv	EEPROM locked state (obsolete)
Bw	Encoder wrap around status bit
Bx	Real time index inptut status bit
BASE	Cam encoder count cycle length
BAUD	Host communications control
BRKENG	Brake engage
BRKRLS	Brake release
BRKSRV	Brake without servo
BRKTRJ	Brake without trajectory
BREAK	Program execution flow control
C#	Program subroutine label
CCHN{type}{channel}	Close communications channel
CHN0	RS-232 communications error flags
CHN1	RS-485 communications error flags
CLK	Hardware clock variable
CTR	Second encoder/step and direction counter
D=exp	Set relative distance
DEFAULT	Switch-case structure element
DIN{port}{channel}	Input byte from module
DOUT{port}{channel}{expression}	Output byte to module
E=expression	Set allowable position error
ECHO	Echo input data back out main channel
ECHO_OFF	Stop echo main channel
ECHO1	Echo input data back out second channel
ECHO1_OFF	Stop echo second channel
ELSE	If structure element
ENC0	Select internal encoder for servo

ENC1	Select external encoder for servo
END	End program
ENDIF	End **IF** statement
EPTR=expression	Set data EEPROM pointer
ES400	Slow data EEPROM read/write speed
ES1000	Increase data EEPROM read/write speed
F	Load filter
F=expression	Special functions control
G	Start motion (GO)
GETCHR	Get character from main comm channel
GETCHR1	Get character from second comm channel
GOSUB#	Call a subroutine
GOTO#	Branch program execution to a label
I (capital i)	Hardware index position variable
IF expression	Conditional test
KA=expression	PID acceleration feed-forward
KD=expression	PID derivative compensation
KG=expression	PID gravity compensation
KGOFF	PID gravity mode off
KGON	PID gravity mode on
KI=expression	PID integral compensation
KL=expression	PID integral limit
KP=expression	PID proportional compensation
KS=expression	PID derivative term sample rate
KV=expression	PID velocity feed forward
LEN	Main comm chnl buffer fill level, data mode
LEN1	Second comm chnl buffer fill level, data mode
LIMD	Enable directional constraints on limit inputs
LIMH	Limit active high
LIML	Limit active low

LIMN	Restore non-directional limits
LOAD	Initiate program download to motor
LOOP	While structure element
MC	Enable cam mode
MC2	Enable cam mode with position scaled x2
MC4	Enable cam mode with position scaled x4
MC8	Enable cam mode with position scaled x8
MD	Enable contouring mode
MD50	Enable drive mode
MF0	Set mode follow for variable only
MF1	Configure follow hardware for x1 scaling
MF2	Configure follow hardware for x2 scaling
MF4	Configure follow hardware for x4 scaling
MFDIV	Mode follow with ratio divisor
MFMUL	Mode follow with ratio multiplier
MFR	Initiate mode follow ratio calculation
MP	Enable position mode
MS	Enable step and direction input mode
MS0	Configure step and direction for variable only
MSR	Initiate mode step ratio calculation
MT	Enable torque mode
MV	Enable velocity mode
O=expression	Set origin
OCHN	Open main communications channel
OFF	Stop servoing the motor
P=expression	Set position
PID1	Restore PID sample rate to default
PID2	Divide PID sample rate by two
PID4	Divide PID sample rate by four
PID8	Divide PID sample rate by eight

PRINT{expression}	Print data to main comm channel
PRINT1{expression}	Print data to second comm channel
PRINT{port}{expression}	Print data to AniLink peripheral
Q	Report status in contouring mode
Ra . . . Rz	Report variables
Raa . . . Rzz	Report variables
Raaa . . . Rzzz	Report variables
Rab[index]	Report byte array variables (8-bit)
Ral[index]	Report long array variables (32-bit)
Raw[index]	Report word array variables (16-bit)
RA	Report acceleration
RAIN{expression}{input}	Report value from analog AniLink card
RAMPS	Report assigned max. drive PWM limit
RBa	Report over current status
RBb	Report parity error status
RBc	Report communications error status
RBd	Report user math overflow status
RBe	Report position error status
RBf	Report communications framing error status
RBh	Report overheat status
RBi	Report index status
RBk	Report EEPROM read/write status
RBl	Report historical left limit status
RBm	Report negative limit status
RBo	Report motor off status
RBp	Report positive limit status
RBr	Report historical right limit status
RBs	Report program scan status
RBt	Report trajectory status
RBu	Report user array index status

RBw	Report wrap around status
RBx	Report hardware indexinput level
RCHN	Report combined communications status
RCHN0	Report RS-232 communications status
RCHN1	Report RS-485 communications status
RCS	Report RS-232 communications check sum
RCS1	Report RS-485 communications check sum
RCTR	Report secondary counter
RD	Return buffered move distance value
RDIN{port}{channel}	Report value from digital AniLink card
RE	Report buffered maximum position error
RES=expression	High resolution encoder control
RETURN	Return from subroutine
RI	Report last stored index position
RKA	Report buffered acceleration feed forward coef.
RKD	Report buffered derivative coefficient
RKG	Report buffered gravity coefficient
RKI	Report buffered integral coefficient
RKL	Report buffered integral limit
RKP	Report buffered proportional coefficient
RKS	Report buffered sampling interval
RKV	Report buffered velocity feed forward coefficient
RMODE	Report current mode of operation
RP	Report present position
RPE	Report present position error
RPW	Report position and status
RS	Report status byte
RT	Report current requested torque
RUN	Execute stored program
RUN?	Override automatic program execution

RV	Report velocity
RW	Report status word
S (as command)	Stop move in progress abruptly
SADDR#	Set motor to new address
SILENT	Suppress PRINT messages main channel
SILENT1	Suppress PRINT messages second channel
SIZE=expression	Number of data entries in cam table
SLEEP	Initiate sleep mode main channel
SLEEP1	Initiate sleep mode second channel
STACK	Reset nesting stack tracking
SWITCH expression	Program execution control
T=expression	Assign torque value in torque mode
TALK	Enable PRINT messages on main channel
TALK1	Enable PRINT messages on main channel
TEMP	Temperature variable
TH	Sets high temperature set point
THD	Sets temperature fault delay
TWAIT	Pause program during a move
UA=expression	Set I/O A output
UAA	I/O A analog input value (0 to 1024)
UAI (as command)	Set I/O A to input
UAI (as input value)	I/O A input value variable
UAO (as command)	Set I/O A to output
UB=expression	Set I/O B output
UBA	I/O B analog input value (0 to 1024)
UBI (as command)	Set I/O B to input
UBI (as input value)	I/O B input value variable
UBO (as command)	Set I/O B to output
UC=expression	Set I/O C output
UCA	I/O C analog input value (0 to 1024)

UCI (as command)	Set I/O C to input
UCI (as input value)	I/O C input value variable
UCO (as command)	Set I/O C to output
UCP (as command)	Set I/O C to be a right limit input
UD=expression	Set I/O D output
UDA	I/O D analog input value (0 to 1024)
UDI (as command)	Set I/O D to input
UDI (as input value)	I/O D input value variable
UDM (as command)	Set I/O D to be a left limit input
UDO (as command)	Set I/O D to output
UE=expression	Set I/O E output
UEA	I/O E analog input value (0 to 1024)
UEI (as command)	Set I/O E to input
UEI (as input value)	I/O E input value variable
UEO (as command)	Set I/O E to output
UF=expression	Set I/O F output
UFA	I/O F analog input value (0 to 1024)
UFI (as command)	Set I/O F to input
UFI (as input value)	I/O F input value variable
UFO (as command)	Set I/O F to output
UG=expression	Set I/O G output
UGA	I/O G analog input value (0 to 1024)
UGA (as command)	Set I/O G to G synchronous function
UGI (as command)	Set I/O G to input
UGI (as input value)	I/O G input value variable
UGO (as command)	Set I/O G to output
UIA	Read Current (Amps = UIA/100)
UJA	Read Voltage (Volts = UJA/10)
UP	Upload user EEPROM program contents
UPLOAD	Upload user EEPROM readable program

V=expression	Set maximum permitted velocity
VLD	Sequentially load variables from data EEPROM
VST	Sequentially store variables to data EEPROM
WAIT=expression	Suspends program for number of PID samples
WAKE	Terminate sleep mode main channel
WAKE1	Terminate sleep mode second channel
WHILE expression	Conditional program flow command
X	Slow motor motion to stop
Z	Total system reset
Za	Reset current limit violation latch bit
Zb	Reset serial data parity violation latch bit
Zc	Reset communications buffer overflow latch bit
Zd	Reset math overflow violation latch bit
Zf	Reset serial comm framing error latch bit
Zl	Reset historical left limit latch bit
Zr	Reset historical right limit latch bit
Zs	Reset command scan error latch bit
Zu	Reset user array index access latch bit
Zw	Reset encoder wrap around event latch bit
ZS	Reset system latches to power-up state

The **SMI** software is a free download from Animatics' web site. The user still must have the proper cable(s) and power source for the SmartMotor being tested. Every SmartMotor has an ASCII interpreter built in so technically, it is possible to talk to the motor without the **SMI** software.

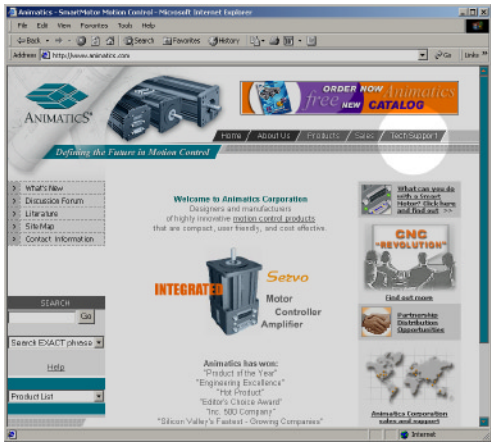
To download the software go to our web site (www.smartmotor.com, see right) and click on the ***Technical Support*** option on the top right side of the Animatics screen.

In the middle of the second screen ***Technical Support***, click on **SMI downloads.**

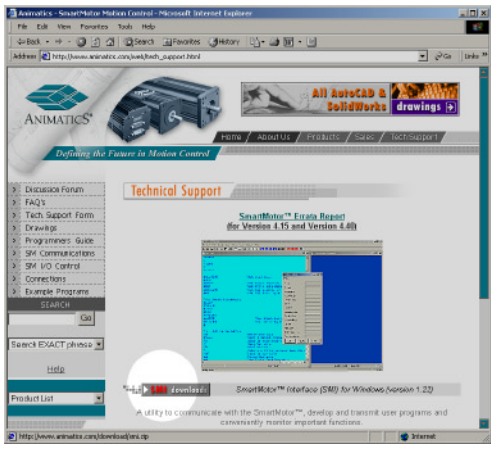
The third screen is legalese concerning how the software is used. **I Agree** must be selected before the software can be downloaded.

The next screen (below) should be **File Download**. Make sure **Save file to disk** is selected and click **OK**.

The **Save As** screen (below) should now be on screen. The small data window at the top of the screen, **Save In:**, should have a default folder already selected (possibly **TEMP** as it is here). Use this window to change drives. In the middle of the **Save As** screen is a large field with a list of files and folders. From here the **Save In:** folder can be changed by double clicking on a different folder or right clicking in an empty portion of the field and selecting **New** then **Folder** from the new menus and entering a different name. After the new folder has been created, double click on it to select it and continue on.

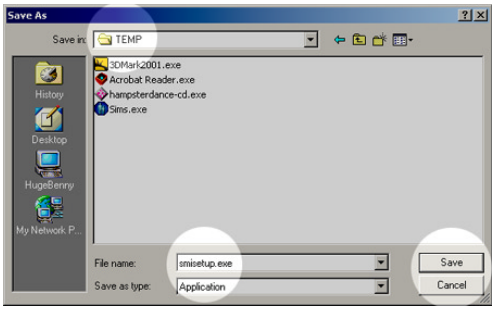


SmartMotor web site opening screen



SmartMotor web site Technical Support screen

Windows' File Download screen



Windows' Save As screen

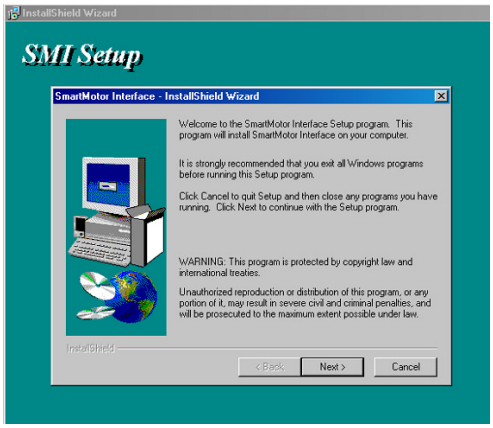
At the bottom of the **Save As** screen are two smaller data windows. The first window (**File name:**) is where the file can be renamed (don't change the **.exe** extension or the file won't work). Make sure the bottom window (**Save as type:**) has **Application** entered and then click on **Save**.

The last screen is the file download progress window. When it's finished the software can be installed. If the **Close this dialog box when download complete** check box is checked the window will close when the file has finished downloading.

The latest version of the SMI software is available at www.smartmotor.com.

If there are any Window's programs running, close them before installing the **SMI** software.

If the **SMI** software is being installed from the SmartMotor CD, insert the CD into the drive. The **SMI** installation program should automatically start. If not, run "**setup.exe**" from the root directory of the CD.



*The **SMI** installation setup window*

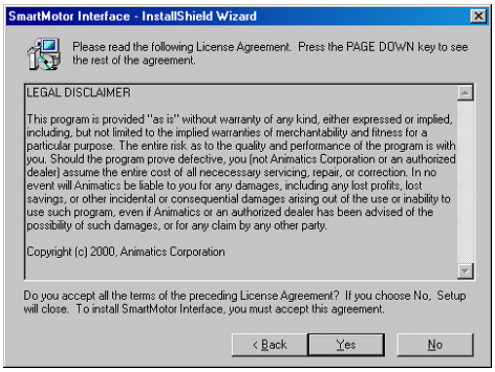
If the **SMI** software is being installed from the downloaded program run "**SMIsetup.exe**" from the directory the software was downloaded to.

Above, is the installation program's opening screen. Press the **Next** button to go to the next step.

License Agreement

This screen is the license agreement.

Read it carefully and if the terms are agreeable, click the **Yes** button to go on. If the **No** button is selected, the installation program will close and the software will not be installed. The terms must be agreed to before the softwate can be installed.



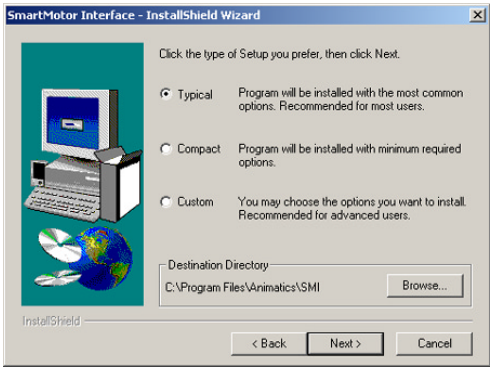
License agreement window

Type of installation

There are three types of installation:

Typical: This is the recommended option for most users. It installs all the common features of the **SMI** software.

Compact: This option installs the minimum required features. Since the typical installation uses very little hard disk space, this option isn't recommended.



Type of installation dialog window.

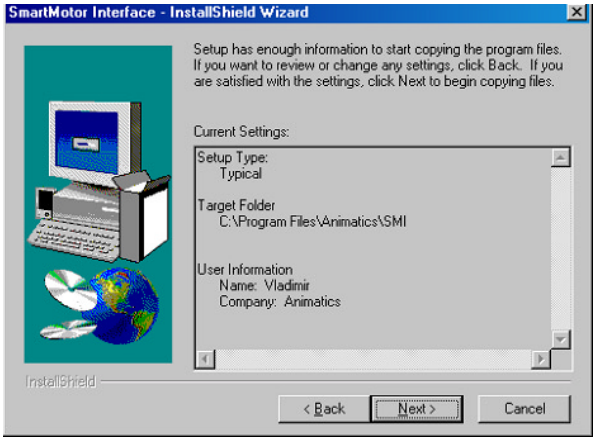
Custom: Select this option to choose only the features to be installed. If the **SMI** software needs to be installed in a directory other than the default directory shown in **Destination Directory** window, click the **Browse** button and select a different directory.

Click the **Next** button to go to the next step.

The confirmation window

The confirmation window

This window gives the user an opportunity to review the settings before copying the files. If anything needs to be changed click on the **Back** button to go to the desired window and change the settings. Click the **Next** button to begin copying files.

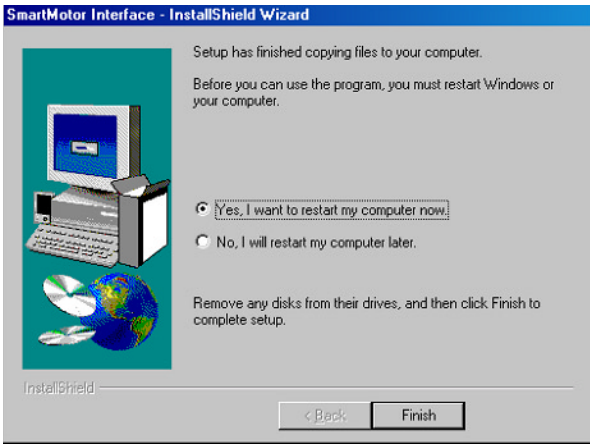


The Finish window

The Finish dialog window

When the files are done copying, the final window (right) will be onscreen.

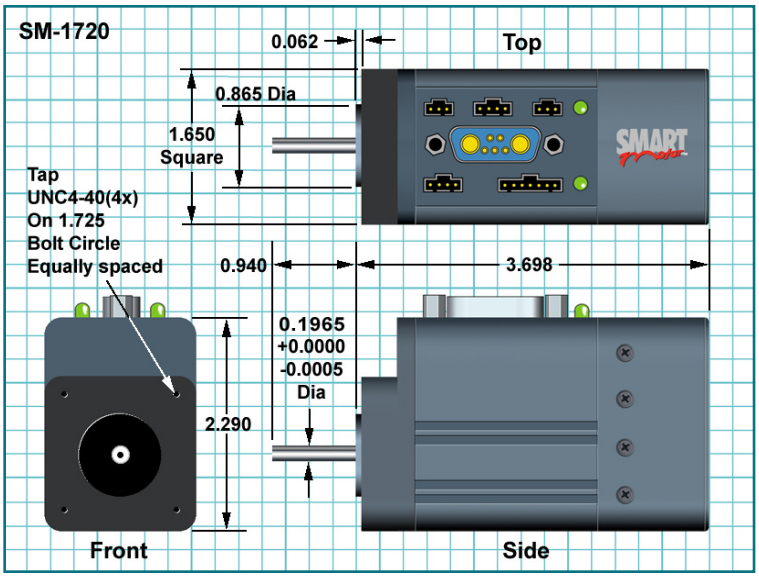
There are two choices in this window, **Yes, I want to restart my computer now** and **No, I will restart my computer later**. The computer must be restarted before the **SMI** software can be used. After the choice has been made, click the **Finish** button

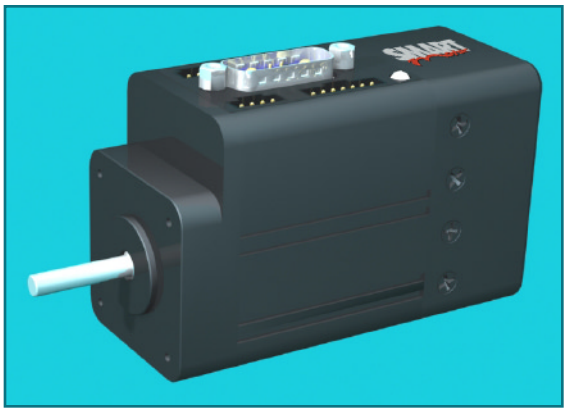


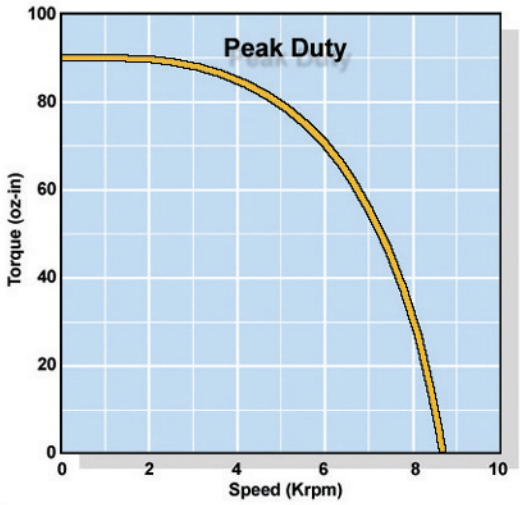
SmartMotor Connections

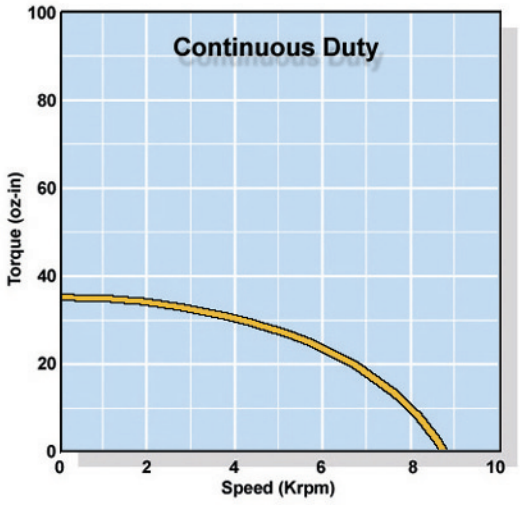
Before starting the **SMI** software make sure that the SmartMotor power and communication cables are properly connected.

SM1720M





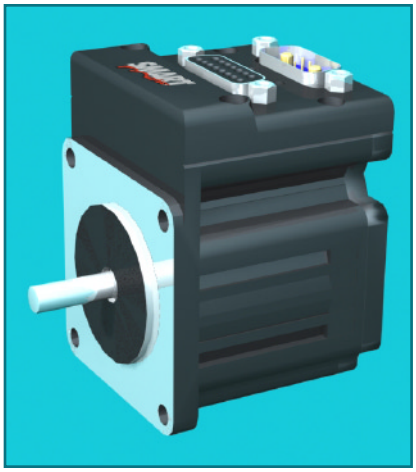


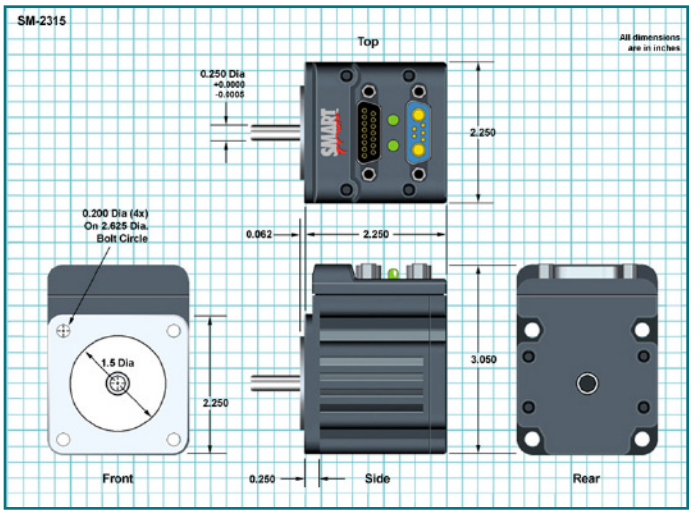


Continuous Torque	in-lb oz-in	2.188 35
Peak Torque	in-lb oz-in	5.625 90
Torque Constant	oz-in/amp	7.57
Nominal Continuous Power	hp KW	0.15 0.11
Top Speed	rpm	8,700
Voltage Constant	V/krpm	5.50
Winding Resistance	ohms	1.8
Encoder Resolution		2,000
Rotor Inertia	oz-in-sec^2 kg-m^2	0.00026 1.80E-06
Poles		4
Slots		15
Shaft Diameter	inches	0.1965
Weight	lbs	1.21
Length	inches	3.70
Width	inches	1.65
Optional 256,000 counts/Rev		
Encoder Index Output		◆
Encoder Follow/Gearing		◆
Infinite Ratio Camming		◆
Removable Memory		◆
Non Volatile Data Memory		◆
AniLink Port With RS-485		◆
AniLink with Separate RS-485		
DC Input to 48V		◆
AC Input to 208V 3 Phase		
Std Military Style Connectors		
CE Mark		◆
Optional Built-in Brake		◆
Rear Shaft Receptacle		
Expansion Bay		

Many free-wheeling vertical applications require a SHUNT to protect the SmartMotor from over voltage resulting from back-drive to the supply, especially if the supply is already very close to the SmartMotor's 48VDC input voltage limit.

Multiple axis applications enduring high current spikes should consider using Isolated RS-485 adapters. Order RS485-ISO for each motor and an RS232485 for the host end. Smaller motors such as the SM1720M with reasonably smooth or soft accelerations work very well in a simple RS-232 daisy chain.

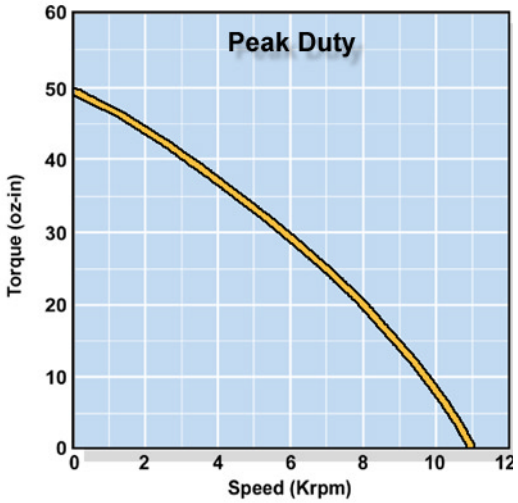


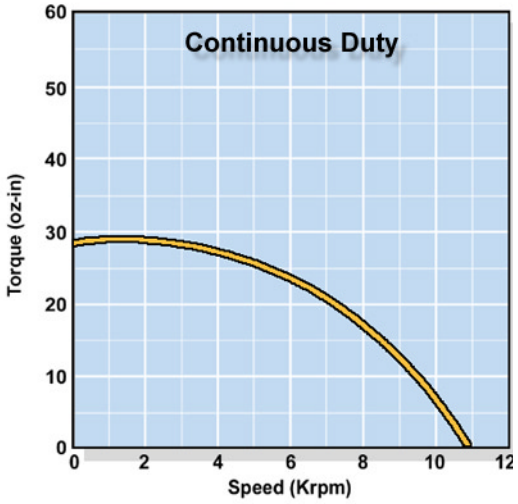


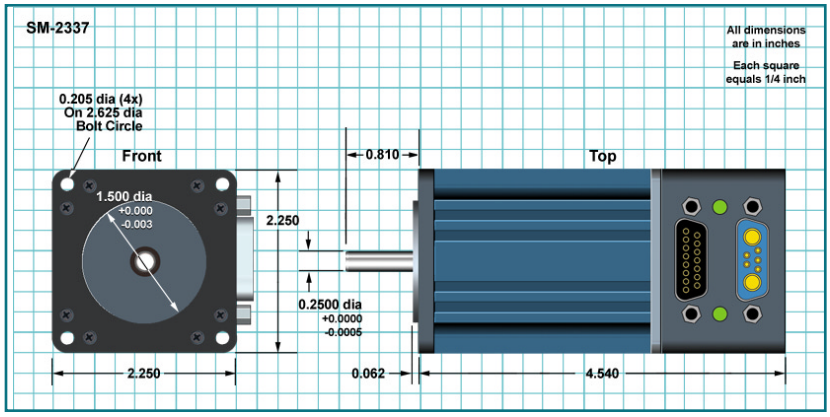
Continuous Torque	in-lb oz-in	1.750 28
Peak Torque	in-lb oz-in	3.125 50
Torque Constant	oz-in/amp	4.17
Nominal Continuous Power	hp KW	0.18 0.13
Top Speed	rpm	10,786
Voltage Constant	V/krpm	4.45
Winding Resistance	ohms	1.0
Encoder Resolution		2,000
Rotor Inertia	oz-in-sec^2 kg-m^2	0.00099 7.00E-06
Poles		8
Slots		12
Shaft Diameter	inches	0.250
Weight	lbs	1.00
Length	inches	2.30
Width	inches	2.25
Optional 256,000 counts/Rev		
Encoder Index Output		
Encoder Follow/Gearing		
Infinite Ratio Camming		
Removable Memory		
Non Volatile Data Memory		
AniLink Port With RS-485		
AniLink with Separate RS-485		
DC Input to 48V		◆
AC Input to 208V 3 Phase		
Std Military Style Connectors		
CE Mark		◆
Optional Built-in Brake		◆
Rear Shaft Receptacle		◆
Expansion Bay		◆

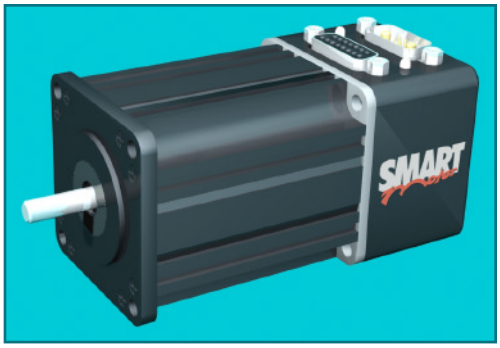
The SM2315 has two features that set apart from other SmartMotors. One, is the 0.250" hole in the back of the shaft. This hole is designed for a press fit with standard, under size 1/4 inch shafting and greatly facilitates customer rear shaft additions and modifications (PLEASE BE SURE NOT TO PRESS AGAINST THE BEARINGS).

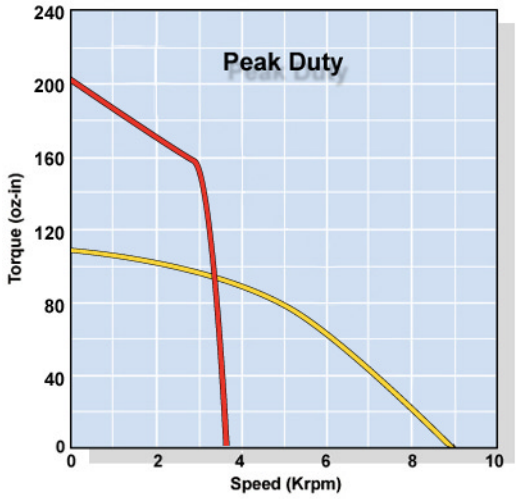
The second unique feature is an interior electronics expansion bay located just inside the back cover. This additional and internally connectorized space is designed to facilitate the addition of custom electronics. Consult the factory for more information on how to take advantage of this unique feature in your application.

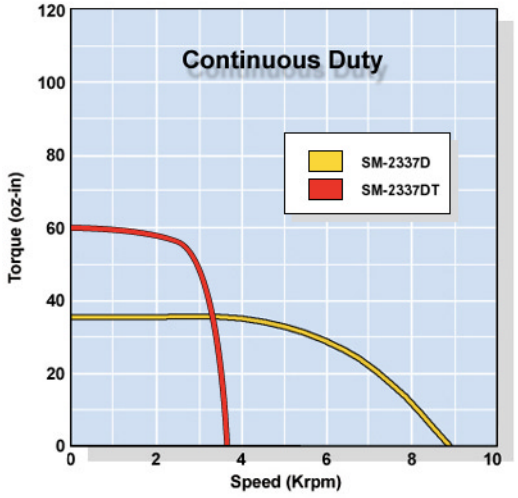










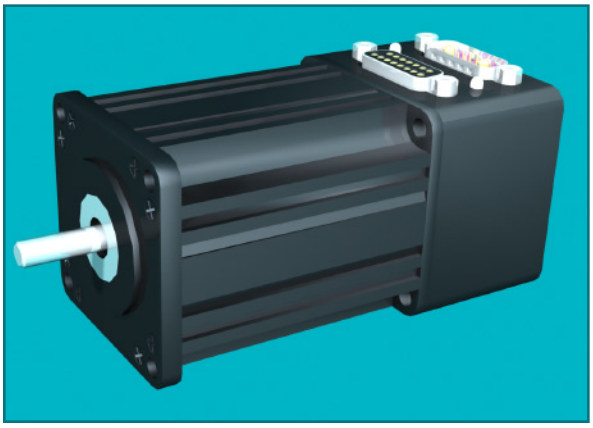


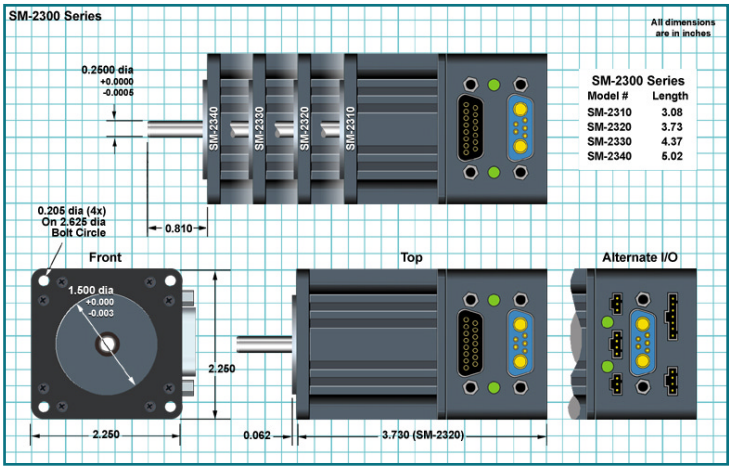
Many free-wheeling vertical applications require a SHUNT to protect the SmartMotor from over voltage resulting from back-drive to the supply, especially if the supply is already very close to the SmartMotor's 48VDC input voltage limit.

Continuous Torque	in-lb oz-in	2.188 35	3.688 59
Peak Torque	in-lb oz-in	6.875 110	12.50 200
Torque Constant	oz-in/amp	5.60	14.80
Nominal Continuous Power	hp KW	0.18 0.13	0.18 0.14
Top Speed	rpm	8,630	3,800
Voltage Constant	V/krpm	5.62	10.95
Winding Resistance	ohms	0.6	0.9
Encoder Resolution		2,000	2,000
Rotor Inertia	oz-in-sec^2 kg-m^2	0.00190 1.34E-05	0.00190 1.34E-05
Poles		8	8
Slots		12	12
Shaft Diameter	inches	0.250	0.250
Weight	lbs	2.10	2.10
Length	inches	4.54	4.54
Width	inches	2.25	2.25
Optional 256,000 counts/Rev Encoder Index Output			
Encoder Follow/Gearing		◆	◆
Infinite Ratio Camming		◆	◆
Removable Memory			
Non Volatile Data Memory		◆	◆
AniLink Port With RS-485		◆	◆
AniLink with Separate RS-485			
DC Input to 48V		◆	◆
AC Input to 208V 3 Phase			
Std Military Style Connectors			
CE Mark		◆	◆
Optional Built-in Brake		◆	
Rear Shaft Receptacle			
Expansion Bay			

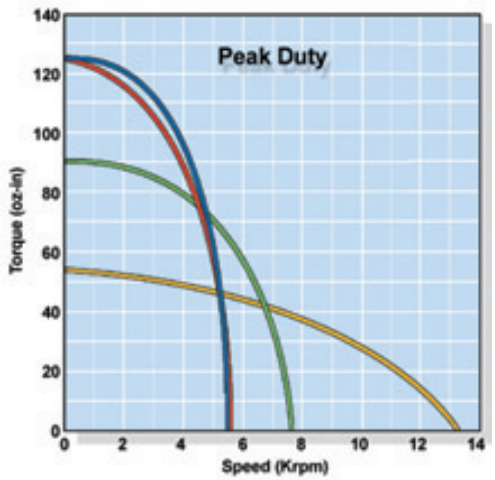
Multiple axis applications enduring high current spikes should consider using Isolated RS-485 adapters. Order RS485-ISO for each motor and an RS232485 for the host end. Smaller motors such as the SM23XX Series with reasonably smooth or soft accelerations work very well in a simple RS-232 daisy chain.

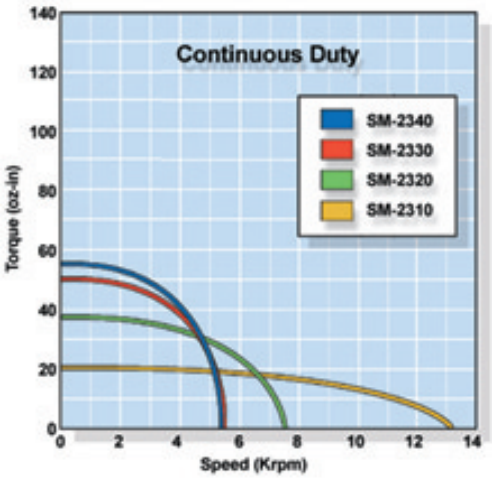
APPENDIX G: MOTOR SPECIFICATIONS
SM2310, SM2320, SM2330, SM2340





Continuous Torque	in-lb oz-in	1.250 20	2.375 38	3.125 50	3.438 55
Peak Torque	in-lb oz-in	3.313 53	5.625 90	7.813 125	7.813 125
Torque Constant	oz-in/amp	5.55	8.92	12.60	13.90
Nominal Continuous Power	hp KW	0.18 0.13	0.19 0.14	0.20 0.15	0.22 0.16
Top Speed	rpm	13,220	7,820	5,590	5,310
Voltage Constant	V/krpm	4.11	6.60	9.32	10.26
Winding Resistance	ohms	1.5	1.1	1.2	1.0
Encoder Resolution		2,000	2,000	2,000	2,000
Rotor Inertia	oz-in-sec^2 kg-m^2	0.00095 6.71E-06	0.00184 1.30E-05	0.00273 1.93E-05	0.00362 2.56E-05
Poles		4	4	4	4
Slots		15	15	15	15
Shaft Diameter	inches	0.250	0.250	0.250	0.250
Weight	lbs	1.20	1.74	2.27	2.79
Length	inches	3.08	3.73	4.38	5.03
Width	inches	2.25	2.25	2.25	2.25
Optional 256,000 counts/Rev		◆	◆	◆	◆
Encoder Index Output		◆			
Encoder Follow/Gearing		◆	◆	◆	◆
Infinite Ratio Camming		◆	◆	◆	◆
Removable Memory					
Non Volatile Data Memory		◆	◆	◆	◆
AniLink Port With RS-485		◆	◆	◆	◆
AniLink with Separate RS-485					
DC Input to 48V		◆	◆	◆	◆
AC Input to 208V 3 Phase					
Std Military Style Connectors					
CE Mark		◆	◆	◆	◆
Optional Built-in Brake		◆	◆	◆	◆
Rear Shaft Receptacle					
Expansion Bay					

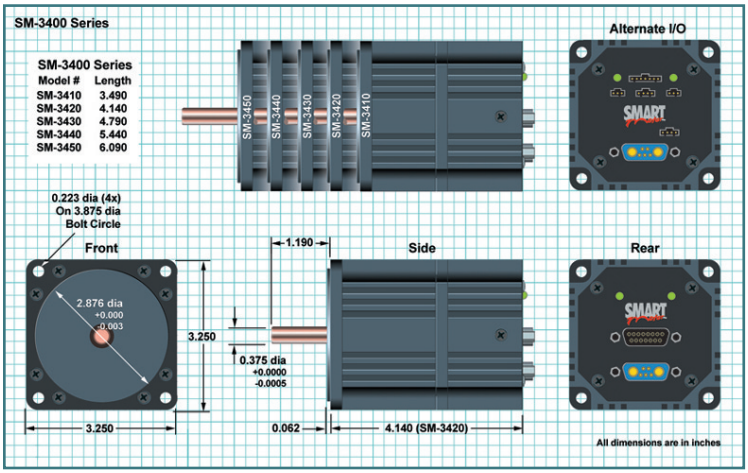


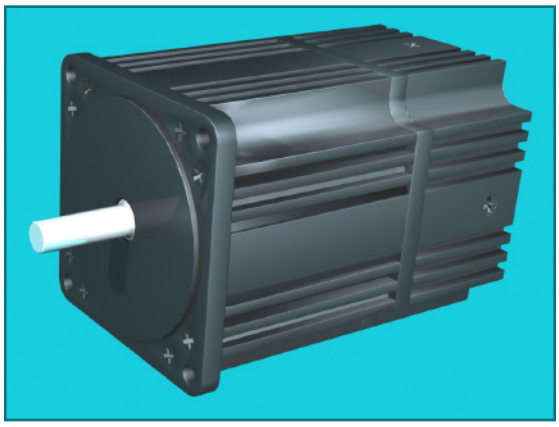


Many vertical applications require a SHUNT to protect the SmartMotor from over voltage resulting from back-drive to the supply, especially if the supply is already very close to the SmartMotor's 48VDC input voltage limit.

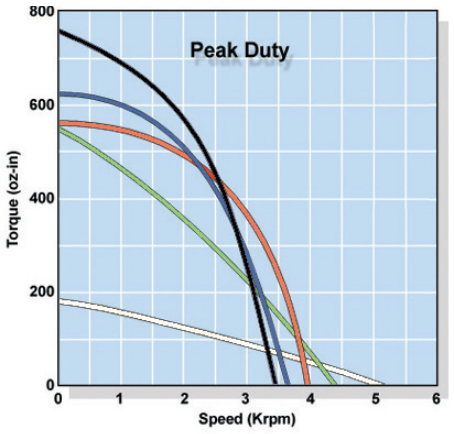
Multiple axis applications enduring high current spikes should consider using Isolated RS-485 adapters. Order RS485-ISO for each motor and an RS232485 for the host end.

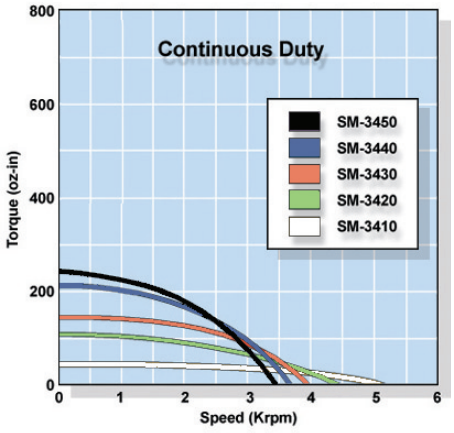
APPENDIX G: MOTOR SPECIFICATIONS
SM3410, SM3420, SM3430, SM3440, SM3450



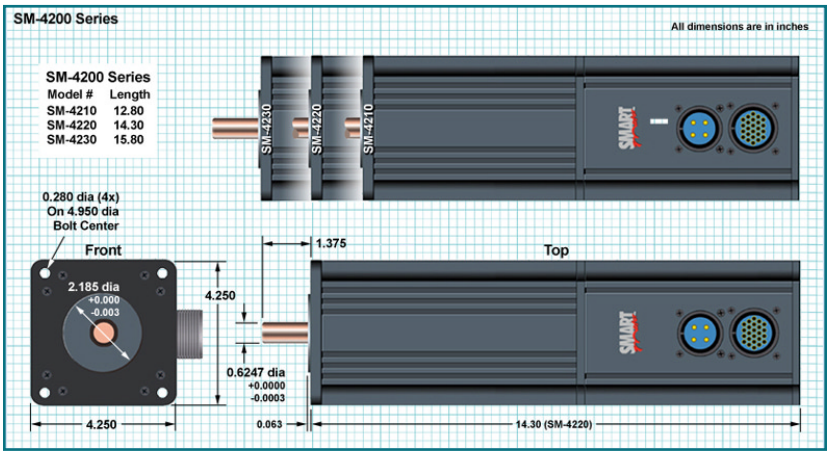


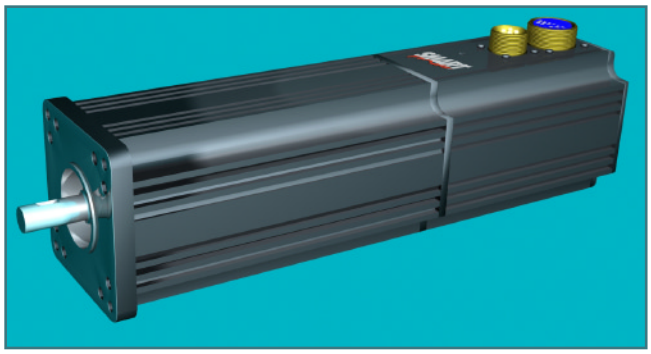
Continuous Torque	in-lb oz-in	2.813 45	6.250 100	9.688 155	13.125 210	15.600 250
Peak Torque	in-lb oz-in	11.250 180	33.750 540	35.978 575	39.063 625	46.875 750
Torque Constant	oz-in/amp	12.50	14.60	16.40	17.40	18.50
Nominal Continuous Power	hp KW	0.16 0.12	0.24 0.18	0.30 0.22	0.34 0.25	0.36 0.27
Top Speed	rpm	5,060	4,310	4,850	3,609	3,398
Voltage Constant	V/krpm	9.20	10.80	12.10	12.90	13.70
Winding Resistance	ohms	2.3	1.2	0.9	0.6	0.6
Encoder Resolution		4,000	4,000	4,000	4,000	4,000
Rotor Inertia	oz-in-sec^2 kg-m^2	0.0060 4.24E-05	0.0130 9.19E-05	0.0190 1.34E-04	0.0250 1.77E-04	0.0300 2.12E-04
Poles		4	4	4	4	4
Slots		24	24	24	24	24
Shaft Diameter	inches	0.375	0.375	0.375	0.375	0.375
Weight	lbs	2.50	3.50	4.50	5.50	6.50
Length	inches	3.49	4.14	4.79	5.44	6.09
Width	inches	3.25	3.25	3.25	3.25	3.25
Optional 256,000 counts/Rev		◆	◆	◆	◆	◆
Encoder Index Output						
Encoder Follow/Gearing		◆	◆	◆	◆	◆
Infinite Ratio Camming		◆	◆	◆	◆	◆
Removable Memory						
Non Volatile Data Memory		◆	◆	◆	◆	◆
AniLink Port With RS-485		◆	◆	◆	◆	◆
AniLink with Separate RS-485						
DC Input to 48V		◆	◆	◆	◆	◆
AC Input to 208V 3 Phase						
Std Military Style Connectors						
CE Mark		◆	◆	◆	◆	◆
Optional Built-in Brake		◆	◆	◆	◆	◆
Rear Shaft Receptacle						
Expansion Bay						





APPENDIX G: MOTOR SPECIFICATIONS
SM4210, SM4220, SM4230



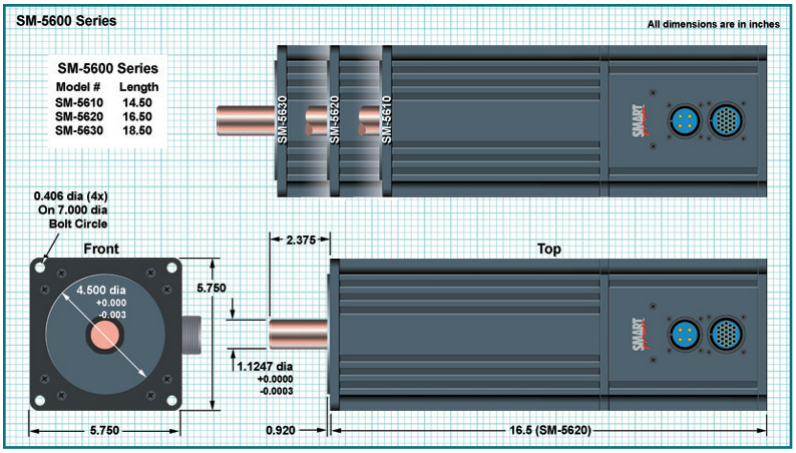


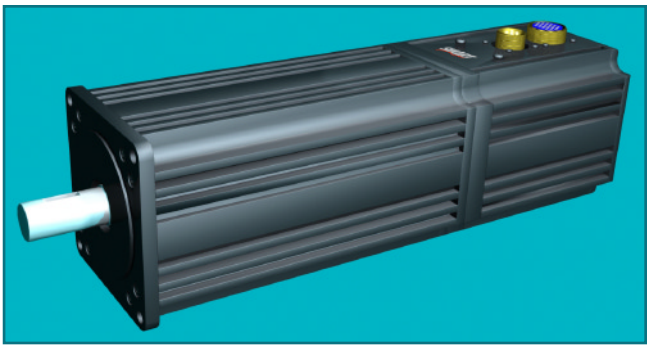
SmartMotors are designed to operate within ambient temperatures ranging between zero and 70 degrees Centegrade. The ratings are for standard room temperature. Continuous torque output derates to zero as the ambient temperature spans from room temperature to 70C.

If the SmartMotor is working hard it will heat up, sometimes so much so as to be too hot to touch. That is normal. SmartMotor MTBF figures are calculated assuming the electronics are at the maximum temperature of 70C (worst case). These theoretical calculations put the SmartMotor MTBF beyond 100,000 hours. Actual field data, now normalized over thousands of installations, shows real SmartMotor average life to exceed the calculated estimates.

Continuous Torque	in-lb oz-in	13.000 208	20.125 322	26.750 428
Peak Torque	in-lb oz-in	132.000 2,112	99.000 1,584	132.000 2,112
Torque Constant	oz-in/amp	172.80	129.60	172.80
Nominal Continuous Power	hp KW	0.44 0.33	0.92 0.69	0.91 0.68
Top Speed	rpm	2,225	2,950	2,200
Voltage Constant	V/krpm	128.00	96.00	128.00
Winding Resistance	ohms	10.0	3.2	3.7
Encoder Resolution		4,000	4,000	4,000
Rotor Inertia	oz-in-sec^2 kg-m^2	0.0290 2.05E-04	0.0420 2.97E-04	0.0540 3.81E-04
Poles		4	4	4
Slots		24	24	24
Shaft Diameter	inches	0.625	0.625	0.625
Weight	lbs	22.00	27.00	32.00
Length	inches	12.80	14.30	15.80
Width	inches	4.25	4.25	4.25
Optional 256,000 counts/Rev				
Encoder Index Output		◆	◆	◆
Encoder Follow/Gearing		◆	◆	◆
Infinite Ratio Camming		◆	◆	◆
Removable Memory				
Non Volatile Data Memory		◆	◆	◆
AniLink Port With RS-485				
AniLink with Separate RS-485		◆	◆	◆
DC Input to 48V				
AC Input to 208V 3 Phase		◆	◆	◆
Std Military Style Connectors		◆	◆	◆
CE Mark				
Optional Built-in Brake				
Rear Shaft Receptacle				
Expansion Bay				

APPENDIX G: MOTOR SPECIFICATIONS
SM5610, SM5620, SM5630





Continuous Torque	in-lb oz-in	28.500 456	46.813 749	65.625 1,050
Peak Torque	in-lb oz-in	241.875 3,870	409.000 6,494	565.750 9,052
Torque Constant	oz-in/amp	94.40	158.40	220.80
Nominal Continuous Power	hp KW	1.81 1.35	1.78 1.33	2.60 1.94
Top Speed	rpm	4,075	2,425	1,743
Voltage Constant	V/krpm	70.00	117.00	164.00
Winding Resistance	ohms	0.9	1.2	1.5
Encoder Resolution		4,000	4,000	4,000
Rotor Inertia	oz-in-sec^2 kg-m^2	0.110 7.77E-04	0.180 1.27E-03	0.240 1.70E-03
Poles		4	4	4
Slots		24	24	24
Shaft Diameter	inches	1.125	1.125	1.125
Weight	lbs	36.00	46.00	56.00
Length	inches	14.50	16.50	18.50
Width	inches	5.75	5.75	5.75
Optional 256,000 counts/Rev				
Encoder Index Output		◆	◆	◆
Encoder Follow/Gearing		◆	◆	◆
Infinite Ratio Camming		◆	◆	◆
Removable Memory				
Non Volatile Data Memory				
AniLink Port With RS-485				
AniLink with Separate RS-485		◆	◆	◆
DC Input to 48V				
AC Input to 208V 3 Phase		◆	◆	◆
Std Military Style Connectors		◆	◆	◆
CE Mark				
Optional Built-in Brake				
Rear Shaft Receptacle				
Expansion Bay				

B&B ELECTRONICS PRODUCT INFORMATION

Integrate – Expand – Simplify

Wireless Radio Modems

Models: X09-009PKC-RA X09-009PKI-RA
X09-019PKC-RA X09-019PKI-RA
X24-009PKC-RA X24-009PKI-RA
X24-019PKC-RA X24-019PKI-RA

Overview

Stand-alone radio modems were designed to provide end users with an easy-to-install wireless serial communication link. Systems integrators love the functionality provided by the serial connection, power switch, indicator lights, high quality antenna, mounting plates and power source inputs.

The 900 MHz radio modem has the best sensitivity in the industry, making it one of the longest range, low-cost radio modems available to date. The 2.4GHz radio modem can provide high performance and dependable operation for deployment throughout the world.

These radio modems are perfect for applications in automatic meter reading (AMR), supervisory control and data acquisition (SCADA), home automation, security, instrument monitoring and point-of-sale (POS) systems among many others.

These radio modems require no configuration to operate, simply plug-and-communicate. Output serial data from any microcontroller or RS-232 port into the radio modem to send FCC & ETSI approved, frequency hopping spread spectrum data and capture it on all receivers within range on the network. Stand-alone radio modems can also be configured to integrate quickly and seamlessly into any new or existing design.

Features

- Plug-and-communicate (default mode - no configuration required).
- True peer-to-peer network (no need to configure a "Master" radio).
- Transparent mode supports existing software applications and legacy systems.
- Addressing capabilities provide for point-to-point and point-to-multipoint networks.
- Uses Standard AT commands and/or fast binary commands for simple configuration, changing parameters.
- Retry and acknowledgements of packets provides guaranteed delivery of critical packets in difficult environments.
- Networking features allow up to 7 independent pairs (networks) to operate in close proximity.
- Multiple low power modes including shutdown pin, cyclic sleep and serial port sleep for current consumption as low as 26 µA.
- Dip-switchable, RS-232/422/485 protocol and multi-drop bus support
- Host interface baud rates from 1200 to 57600 bps.
- Signal strength register for link quality monitoring and debugging.
- Parity support (None, Even, Odd, Mark, Space)
- Source & destination addressing
- Modbus support
- DCD support
- 9-bit support
- Field upgradeable firmware

International Headquarters:
B&B Electronics Mfg. Co. 707 Dayton Road P.O. Box 1040 Ottawa, IL 61350 USA
815-433-5100 Fax 433-5104 *www.bb-elec.com* *orders@bb-elec.com* *support@bb-elec.com*

B&B Electronics Ltd Westlink Commercial Park Oranmore Co. Galway Ireland
+353 91 792444 Fax +353 91 792445 *www.bb-europe.com* *orders@bb-elec.com* *support@bb-europe.com*

Applications

- Monitoring of remote systems
- Production reporting of active systems
- Home automation and building control
- Supervisory control and data acquisition
- Vehicle management and asset tracking

Options

- 900 MHz or 2.4GHz license-free ISM bands
- 9,600 bps or 19,200 bps over-the-air data rates
- Commercial (0º to +70ºC) or industrial (-40º to +85ºC) grades

Specifications

General	900 MHz		2.4 GHz	
Radio Frequency:	902 – 928 MHz		2.4000 – 2.4835 GHz	
Spreading Spectrum Type:	Frequency hopping transceiver		Frequency hopping transceiver	
Frequency Control:	Direct FM		Direct FM	
Network Topology:	Point-Multipoint and Point-to-Point Multi-Drop Transparent Networking		Point-Multipoint and Point-to-Point Multi-Drop Transparent Networking	
Channel Capacity:	7 hop sequences share 25 frequencies		7 hop sequences share 25 frequencies	
Serial Data Interface:	Switch selectable RS-232/422/485		Switch selectable RS-232/422/485	
I/O Data Rate:	Software selectable 1200-57,600kbps		Software selectable 1200-57,600kbps	
Power Requirements				
Supply Voltage:	7-18 VDC		7-18 VDC	
Transmit Voltage:	200 mA		200 mA	
Receive Current:	70 mA		70 mA	
Power Down Current:	< 1 mA		< 1 mA	
Physical Properties				
Enclosure Type:	Extruded aluminum, black anodized		Extruded aluminum, black anodized	
Enclosure Size:	2.75 x 5.50 x 1.124 in (7.90 x 13.90 x 3.80 cm)		2.75 x 5.50 x 1.124 in (7.90 x 13.90 x 3.80 cm)	
Weight:	7.1 oz (200 g)		7.1 oz (200 g)	
Operating Temperature:	0º to +70ºC (commercial) -40º to +85ºC (industrial, wide-temperature)		0º to +70ºC (commercial) -40º to +85ºC (industrial, wide-temperature)	
Antenna				
Type:	½ wave dipole whip		½ wave dipole whip	
Gain:	2.1 dBi		2.1 dBi	
Length:	6.75 in (17.1 cm)		5.25 in (13.3 cm)	
Connector:	Reverse-polarity SMA		Reverse-polarity SMA	
Impedance:	50 Ohms unbalanced		50 Ohms unbalanced	
Certifications				
FCC Part 15.247:	OUR9XSTREAM		OUR24XSTREAM	
Industry Canada:	4214A-9XSTREAM		4214A 12008	
Europe:	n/a		ETSI	
Performance				
Indoor/urban Range*:	Up to 1500 ft. (457m)		Up to 600 ft. (183 m)	
Outdoor LOS Range*:	Up to 7 mi. (11 km) w/dipole Up to 20 mi. (32 km) w/high-gain		Up to 3 mi. (5 km) w/dipole Up to 10 mi. (16 km) w/high-gain	
Serial Data Throughput:	**9600 bps**	**19.2 kbps**	**9600 bps**	**19.2 kbps**
RF Baud Rate:	10,000 bps	20,000 bps	10,000 bps	20,000 bps
Transmit Power Output:	140 mW	(21.5 dBm)	50mW	(17 dBm)
Receiver Sensitivity:	-110 dBm	-107 dBm	-105 dBm	-102 dBm

**Range is calculated assuming line-of-sight. Actual range will vary based upon specific board integration, antenna selection, and environment.*

International Headquarters:
B&B Electronics Mfg. Co. 707 Dayton Road P.O. Box 1040 Ottawa, IL 61350 USA
815-433-5100 Fax 433-5104 *www.bb-elec.com* *orders@bb-elec.com* *support@bb-elec.com*

B&B Electronics Ltd Westlink Commercial Park Oranmore Co. Galway Ireland
+353 91 792444 Fax +353 91 792445 *www.bb-europe.com* *orders@bb-elec.com* *support@bb-europe.com*

Ordering Information: Model Summary

Stand-Alone Radio Modem Models:	Operating Frequency:	RF Data Rate:	Temperature Rating:	Interface:
X09-009PKC-RA	915 MHz	9600 baud	º0 to 70ºC (Commercial)	RS-232/422/485 (DB9 female)
X09-009PKI-RA	915 MHz	9600 baud	-40º to 80ºC (Industrial, wide-temp)	RS-232/422/485 (DB9 female)
X09-019PKC-RA	915 MHz	19200 baud	0º to 70ºC (Commercial)	RS-232/422/485 (DB9 female)
X09-019PKI-RA	915 MHz	19200 baud	-40º to 80ºC (Industrial, wide-temp)	RS-232/422/485 (DB9 female)
X24-009PKC-RA	2.4 GHz	9600 baud	0º to 70ºC (Commercial)	RS-232/422/485 (DB9 female)
X24-009PKI-RA	2.4 GHz	9600 baud	-40º to 80ºC (Industrial, wide-temp)	RS-232/422/485 (DB9 female)
X24-019PKC-RA	2.4 GHz	19200 baud	0º to 70ºC (Commercial)	RS-232/422/485 (DB9 female)
X24-019PKI-RA	2.4 GHz	19200 baud	-40º to 80ºC (Industrial, wide-temp)	RS-232/422/485 (DB9 female)

International Headquarters:
B&B Electronics Mfg. Co. 707 Dayton Road P.O. Box 1040 Ottawa, IL 61350 USA
815-433-5100 Fax 433-5104 *www.bb-elec.com* *orders@bb-elec.com* *support@bb-elec.com*

B&B Electronics Ltd Westlink Commercial Park Oranmore Co. Galway Ireland
+353 91 792444 Fax +353 91 792445 *www.bb-europe.com* *orders@bb-elec.com* *support@bb-europe.com*

Medium Duty Incremental Encoders

Features

The medium duty encoder offers the greatest flexibility of choice in a very high-quality encoder, all for a very low price. Features:

- Small body with 50mm diameter and 35mm depth
- Splash proof (IP65 rating)
- 8mm standard shaft or 8mm hollow shaft
- Incremental resolution available from 60 pulses per revolution to 2500 pulses per revolution
- Available with open collector or push-pull output
- Up to 100kHz response frequency

Standard shaft (TRD-N) model

Hollow shaft (TRD-NH) model

Note: Yellow shaded part numbers are non-stock. Availability may range from four to six weeks.

Incremental Medium Duty Standard Shaft Encoders (Push-Pull Output, TRD-Nxxx-RZVD)

Part Number	Price	Pulses per Revolution	Input Voltage	Output	Body Dia.
TRD-N3-RZWD	check	3	5-30 VDC	Push-pull	50mm
TRD-N4-RZWD	check	4			
TRD-N5-RZWD	check	5			
TRD-N10-RZWD	check	10			
TRD-N30-RZWD	check	30			
TRD-N40-RZWD	check	40			
TRD-N50-RZWD	check	50			
TRD-N60-RZWD	check	60			
TRD-N100-RZWD	check	100			
TRD-N120-RZWD	check	120			
TRD-N200-RZWD	check	200			
TRD-N240-RZWD	check	240			
TRD-N250-RZWD	check	250			
TRD-N300-RZWD	check	300			
TRD-N360-RZWD	check	360			
TRD-N400-RZWD	check	400			
TRD-N480-RZWD	check	480			
TRD-N500-RZWD	check	500			
TRD-N600-RZWD	check	600			
TRD-N750-RZWD	check	750			
TRD-N1000-RZWD	check	1000			
TRD-N1024-RZWD	check	1024			
TRD-N1200-RZWD	check	1200			
TRD-N2000-RZWD	check	2000			
TRD-N2500-RZWD	check	2500			

Incremental Medium Duty Hollow Shaft Encoders (Push-Pull Output, TRD-NHxxx-RZWD)

Part Number	Price	Pulses per Revolution	Input Voltage	Output	Body Dia.
TRD-NH3-RZWD	check	3	5-30 VDC	Push-pull	50mm
TRD-NH4-RZWD	check	4			
TRD-NH5-RZWD	check	5			
TRD-NH10-RZWD	check	10			
TRD-NH30-RZWD	check	30			
TRD-NH40-RZWD	check	40			
TRD-NH50-RZWD	check	50			
TRD-NH60-RZWD	check	60			
TRD-NH100-RZWD	check	100			
TRD-NH120-RZWD	check	120			
TRD-NH200-RZWD	check	200			
TRD-NH240-RZWD	check	240			
TRD-NH250-RZWD	check	250			
TRD-NH300-RZWD	check	300			
TRD-NH360-RZWD	check	360			
TRD-NH400-RZWD	check	400			
TRD-NH480-RZWD	check	480			
TRD-NH500-RZWD	check	500			
TRD-NH600-RZWD	check	600			
TRD-NH750-RZWD	check	750			
TRD-NH1000-RZWD	check	1000			
TRD-NH1200-RZWD	check	1200			
TRD-NH2000-RZWD	check	2000			
TRD-NH2500-RZWD	check	2500			

Medium Duty Incremental Encoders

Note: Yellow shaded part numbers are non-stock. Availability may range from four to six weeks.

Incremental Medium Duty Standard Shaft Encoders (Line Driver Output, TRD-Nxxx-RZVWD)

Part Number	Price	Pulses per Revolution	Input Voltage	Output	Body Dia.
TRD-N3-RZVWD	check	3	5VDC	Line driver (differential)	50mm
TRD-N4-RZVWD	check	4			
TRD-N5-RZVWD	check	5			
TRD-N10-RZVWD	check	10			
TRD-N30-RZVWD	check	30			
TRD-N40-RZVWD	check	40			
TRD-N50-RZVWD	check	50			
TRD-N60-RZVWD	check	60			
TRD-N100-RZVWD	check	100			
TRD-N120-RZVWD	check	120			
TRD-N200-RZVWD	check	200			
TRD-N240-RZVWD	check	240			
TRD-N250-RZVWD	check	250			
TRD-N300-RZVWD	check	300			
TRD-N360-RZVWD	check	360			
TRD-N400-RZVWD	check	400			
TRD-N480-RZVWD	check	480			
TRD-N500-RZVWD	check	500			
TRD-N600-RZVWD	check	600			
TRD-N750-RZVWD	check	750			
TRD-N1000-RZVWD	check	1000			
TRD-N1024-RZVWD	check	1024			
TRD-N1200-RZVWD	check	1200			
TRD-N2000-RZVWD	check	2000			
TRD-N2500-RZVWD	check	2500			

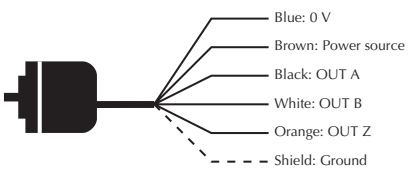
Incremental Medium Duty Hollow Shaft Encoders (Line Driver Output, TRDH-Nxxx-RZVWD)

Part Number	Price	Pulses per Revolution	Input Voltage	Output	Body Dia.
TRD-NH3-RZVWD	check	3	5VDC	Line driver (differential)	50mm
TRD-NH4-RZVWD	check	4			
TRD-NH5-RZVWD	check	5			
TRD-NH10-RZVWD	check	10			
TRD-NH30-RZVWD	check	30			
TRD-NH40-RZVWD	check	40			
TRD-NH50-RZVWD	check	50			
TRD-NH60-RZVWD	check	60			
TRD-NH100-RZVWD	check	100			
TRD-NH120-RZVWD	check	120			
TRD-NH200-RZVWD	check	200			
TRD-NH240-RZVWD	check	240			
TRD-NH250-RZVWD	check	250			
TRD-NH300-RZVWD	check	300			
TRD-NH360-RZVWD	check	360			
TRD-NH400-RZVWD	check	400			
TRD-NH480-RZVWD	check	480			
TRD-NH500-RZVWD	check	500			
TRD-NH600-RZVWD	check	600			
TRD-NH750-RZVWD	check	750			
TRD-NH1000-RZVWD	check	1000			
TRD-NH1024-RZVWD	check	1024			
TRD-NH1200-RZVWD	check	1200			
TRD-NH2000-RZVWD	check	2000			
TRD-NH2500-RZVWD	check	2500			

Wiring diagrams

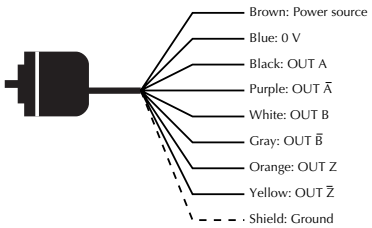
Push-pull connections

Shielded cable is not connected to the encoder body.



Line driver connections

Shielded cable is not connected to the encoder body.



Medium Duty Incremental Encoders

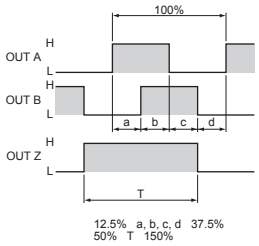
Electrical Specifications				
Model			**TRD-N/NHxxxx-RZWD (Push-Pull)**	**TRD-N/NHxxxx-RZVWD (Line Driver)**
Power Supply	**Operating Voltage**		4.75 - 30VDC*	+4.75 - 5.25VDC*
	Allowable Ripple		3% rms max.	-
	Current Consumption		60 mA max.	
Signal Waveform			Two-phase + home position	
Max. Response Frequency			100KHz max.	
Duty Ratio			50 ± 25% (square wave)	
Signal Width at Home Position			100 ± 50%	
Output	**Rise/Fall Time**		3µs max. (when cable length is 1m)	-
	Output Type		Totem pole	Line driver output (26C31 or equivalent)
	Output Logic		Negative logic (active low)	Negative logic (active high)
	Output Current	**"H"**	10mA max.	-
		"L"	30mA max.	-
	Output Voltage	**"H"**	[(Load power volt) - 2.5V)]	-
		"L"	0.4V max.	-
	Load Power Voltage		35 VDC max.	-

* To be supplied by Class II source

Mechanical Specifications	
Starting Torque	Max. 0.03 Nm (.0022 ft lbs)
Max. Allowable Shaft Load	Radial: 50N (11.24 lbs) Axial: 30N (6.74 lbs)
Max. Allowable Speed	5000 rpm (dust and splash proofed: continuous: 3,000 rpm, instantaneous: 5,000 rpm) (highest speed that can support the mechanical integrity of encoder)
Wire Size	AWG24
Weight	Approx. 250g (8.82 oz) with 2m cable
Environmental Specifications	
Ambient Temperature	10 to 70°C; 14 to 158°F
Storage Temperature	-25 to 85°C; -13 to 185°F
Operating Humidity	35-85% RH
Voltage Withstand	500VAC (50/60Hz) for one minute
Insulation Resistance	50MΩ min. (excluding shield between power supply, signal cable and case)
Vibration Resistance	Durable for one hour along three axes at 10 to 55 Hz with 0.75 mm amplitude (excluding shield between power supply, signal cable and case)
Shock Resistance	11 ms with 490 m/s² applied three times along three axes
Protection	IP50: dust proof; IP65: dust and splash proof

Channel timing chart

Output Signal Timing Chart - Totem Pole Models



The above waveforms apply to normal (clockwise) revolution viewed from the shaft. OUT Z phase is reversed on the RZL and RZWL models.

Accessories

Couplings

If you selected an encoder with a solid shaft, please select a coupling that fits your encoder. All couplings are in stock, ready to ship.

See page 907 for more information.

Mounting bracket

JT-035D metal mounting bracket can be used for all TRD-N/NH/NA encoders.

JT-035D

How to read the timing charts:

Open Collector Models:

Out A and Out B are 90 degrees out of phase. Like any quadrature encoder, four unique logic states are created internal to the encoder. This is based on the rising edge to rising edge (one cycle) on channel A or B that indicates that one set of bars on the internal encoder disk has passed by the optical sensor.

For example, looking at the A and A-not channel, the encoder's internal optical sensor compares the two. If the encoder senses more light at A than at A-not, then OUTPUT A goes high. If more light is sensed at A-not than at A, OUT A goes low. The same applies to channel B. This process is called "push-pull."

OUT Z is the absolute reference added to an incremental encoder and is also known as home position. It signifies a full rotation of the encoder disk.

Line Driver Models:

Channel A (OUT A and A-not) and Channel B (OUT B and B-not) are also 90 degrees out of phase on line driver encoders. The quadrature state of channels A and B creates four unique logic states. When these four unique logic states are decoded, the resolution obtained is 4 times (4X) the resolution of the encoder disk. This means that 250 sets of bars would yield 1000 quadrature states (4 x 250 = 1,000).

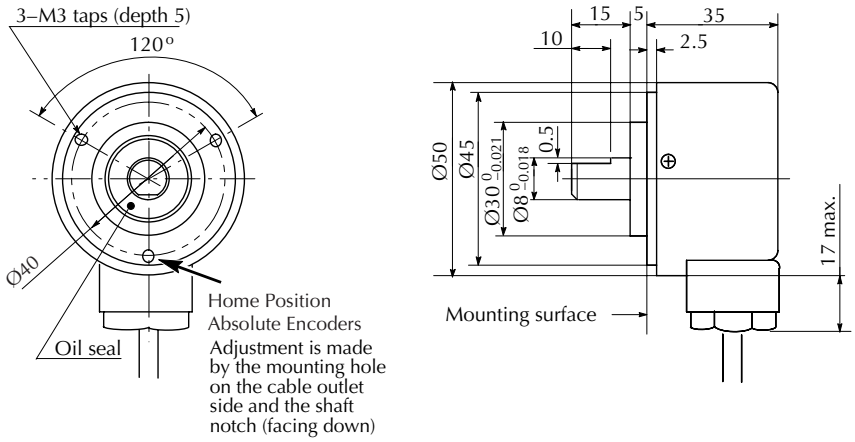
OUT Z is the same as on open collector models, and is the absolute reference (home position). It signifies one full rotation of the encoder.

Medium Duty Absolute and Incremental

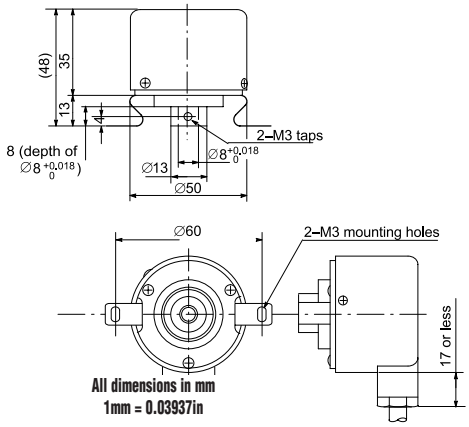
Dimensions

The following are the external dimensions of both incremental and absolute medium duty encoders and the optional mounting bracket.

Standard shaft incremental and absolute encoders (TRD-N, TRD-NA)

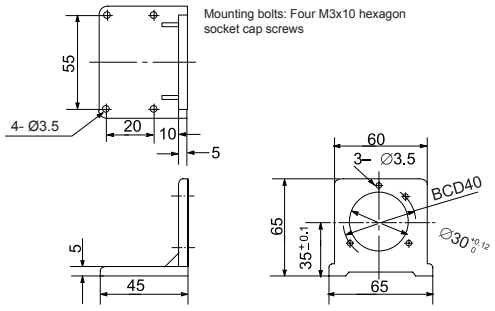


Hollow shaft incremental encoders only (TRD-NH)



Optional mounting bracket for all medium duty encoders

JT-035D



INDICE DE FIGURAS

CAPITULO I

FUNCIONAMIENTO DEL SISTEMA DE PUNTERÍA DIGITAL DEL CAÑÓN AUTOPROPULSADO DE 155mm

CAPITULO II

SOLUCIÓN PROPUESTA Y DESCRIPCIÓN DEL SISTEMA SCADA

CAPITULO III

DISEÑO DEL SOFTWARE DE AUTOMATIZACIÓN

CAPITULO IV

HARDWARE

CAPITULO V

PRUEBAS DE CAMPO Y RESULTADOS

INDICE DE TABLAS

CAPITULO II

SOLUCIÓN PROPUESTA Y DESCRIPCIÓN DEL SISTEMA SCADA

CAPITULO III

DISEÑO DEL SOFTWARE DE AUTOMATIZACIÓN

CAPITULO IV

HARDWARE

CAPITULO V

PRUEBAS DE CAMPO Y RESULTADOS

GLOSARIO

Anteojo panorámico: Instrumento óptico ubicado en el cañón, utilizado para realizar puntería recíproca.

Cuadrante de nivel: Instrumento utilizado en artillería para calibrar el tubo del cañón, con los respectivos indicadores visuales ubicados en el mismo, los cuales permiten el posicionamiento de la pieza.

Deflexión: Curvatura o desviación de un curso o línea horizonta.

Espoleta: aparato o artificio usado para provocar la inflamación y explosión de la carga (explosiva o incendiaria) que **l**evan los proyectiles huecos (bombas, granadas, torpedos, etc.) funcionan de varias maneras: al contacto contra el suelo u obstáculo (instantáneas), después de cierto tiempo de impactar (retardadas) o a cierta distancia del impacto (radio-proximidad)

Exactitud: Grado de acercamiento, aproximación o conformidad al valor verdadero de un determinado parámetro en análisis o medición

Goniómetro: deriva del vocablo *gonio* (ángulo) y *metro* (medida). Por lo que es el instrumento del cual nos valemos para medir y trazar ángulos. Y esta construido con una escuadra unida a un circulo graduado, que concéntricamente a él, gira un disco con un nonius por el que desliza un una regla.

HMI: (Interfase entre el hombre y la máquina). Son paneles de operación, que se encuentran comunicados de forma tal que el operador puede visualizar y alterar las recetas

de producción en diferentes lugares de la planta, teniendo la seguridad que dichos datos llegarán a ser procesados sin errores.

Interpolación: Método por el que se calculan más puntos de muestra o se determinar los valores medios de varios puntos, de acuerdo con un algoritmo.

Lista de concentraciones: es un documento, trabajado en tiempo de paz, en el cual se registran los datos de blanco previamente establecidos sobre los cuales se tiene programado hacer fuego. Para obtener una lista de concentraciones, se necesita la localización exacta de las piezas o del lugar desde el cual se hará fuego en caso de guerra, la dirección general de tiro y las posiciones de los posibles blancos a abatir)

Milésima de grado: Unidad de medida angular definida como el ángulo necesario para que a una distancia de 1000 unidades se de un desplazamiento transversal de una unidad de distancia. Una milésima equivale a 1/6400 de un circulo, siendo igual a 1/17.77778 grados.

Obús autopropulsado: Howitzer. Pieza de artillería de longitud corta (de 20 a 30 veces su calibre). Es de tiro muy curvo y posee motor el cual le permite moverse autónomamente.

Pieza base: Es la pieza que se encuentre mas cercana al centro de batería, y la cual se usa normalmente para efectuar los registros. Esta pieza es aquella que normalmente tiene la velocidad inicial más cercana a la velocidad media de la batería.

Plan de fuego de la unidad: es la planificación de una determinada unidad de combate en el que se coordina e integra el empleo de todo el apoyo de fuego disponible por medio de todas sus armas: morteros, fusilería, armas anti-tanque, etc. así mismo detalla sus objetivos, munición a consumir, etc.

Plancheta: Aparato topográfico que sirve para la determinación gráfica de ángulo

Precisión: Grado de concordancia dentro de un grupo de mediciones o instrumentos.

Puertos COM: Es un interfaz de comunicaciones llamado también puerto serial, a través de la cual se podrá acceder más allá del entorno de la computadora.

Puntería directa: Cuando se usa el procedimiento de puntería directa, la pieza es apuntada en dirección y elevación, visando directamente el blanco. Este tipo de puntería no es muy empleado.

Puntería reciproca o indirecta: Cuando se usa el procedimiento de puntería indirecta, la pieza es apuntada en dirección, graduando una deflexión dada en la escala del anteojo panorámico, y girando la pieza lateral hasta que la línea de mira del anteojo panorámico coincida con el punto de puntería**.**

Reglaje: Galicismo aceptado por algunos ejércitos sudamericanos, por corrección del tiro. Método de ajuste del tiro, por lo cual se establece las correcciones del tiro mediante impactos cortos y largos con relación a la línea de observación

Standard NMEA: Es un interfaz eléctrico y un protocolo de datos para comunicación entre instrumentación Marina.

UTM, cuadrícula: Retícula trazada en proyección transversa de Mercator entre los 80º de latitud norte y los 80º de latitud sur. El elipsoide de referencia terrestre se divide en 60 husos iguales de 6º de longitud; asimismo cada huso queda dividido en 20 áreas de 6º de longitud por 8º de latitud, que se denominan *zonas.* Cada zona se denota con letras mayúsculas desde la C hasta la X inclusive (excluidas las letras I, Ñ y O), empezando en el paralelo 80º sur y terminando en el paralelo 80º norte. La superficie cubierta por la cuadrícula se divide en cuadrados de 100 Km. de lado. Estos cuadrados se designan por dos letras, que indican la columna y la fila, de manera que, dentro de un área de 18º de longitud, por 17º de latitud, no se repita la denominación de un cuadrado. El tercer grado de referencia lo proporciona la cuadrícula de 1 Km., trazada dentro de cada cuadrado de 100 Km. El origen para cada huso está a 500 Km. al oeste del meridiano central del huso, y en ordenadas se le da al ecuador un valor de 10.000 Km. para los puntos situados en el hemisferio sur y 0 para lo puntos situados en el hemisferio norte.Coordenadas globales o geodésicas

WGS-84: Designa el Sistema Coordenado materializado y diseminado por la agencia norteamericana *National Imagery and Mapping Agency* (NIMA). El origen de este Sistema de Referencia se remonta a la era Doppler, aunque en la actualidad está basado prácticamente en observaciones GPS.

Zoom: Capacidad de aumentar o reducir el tamaño de la figura visualizada en la pantalla.